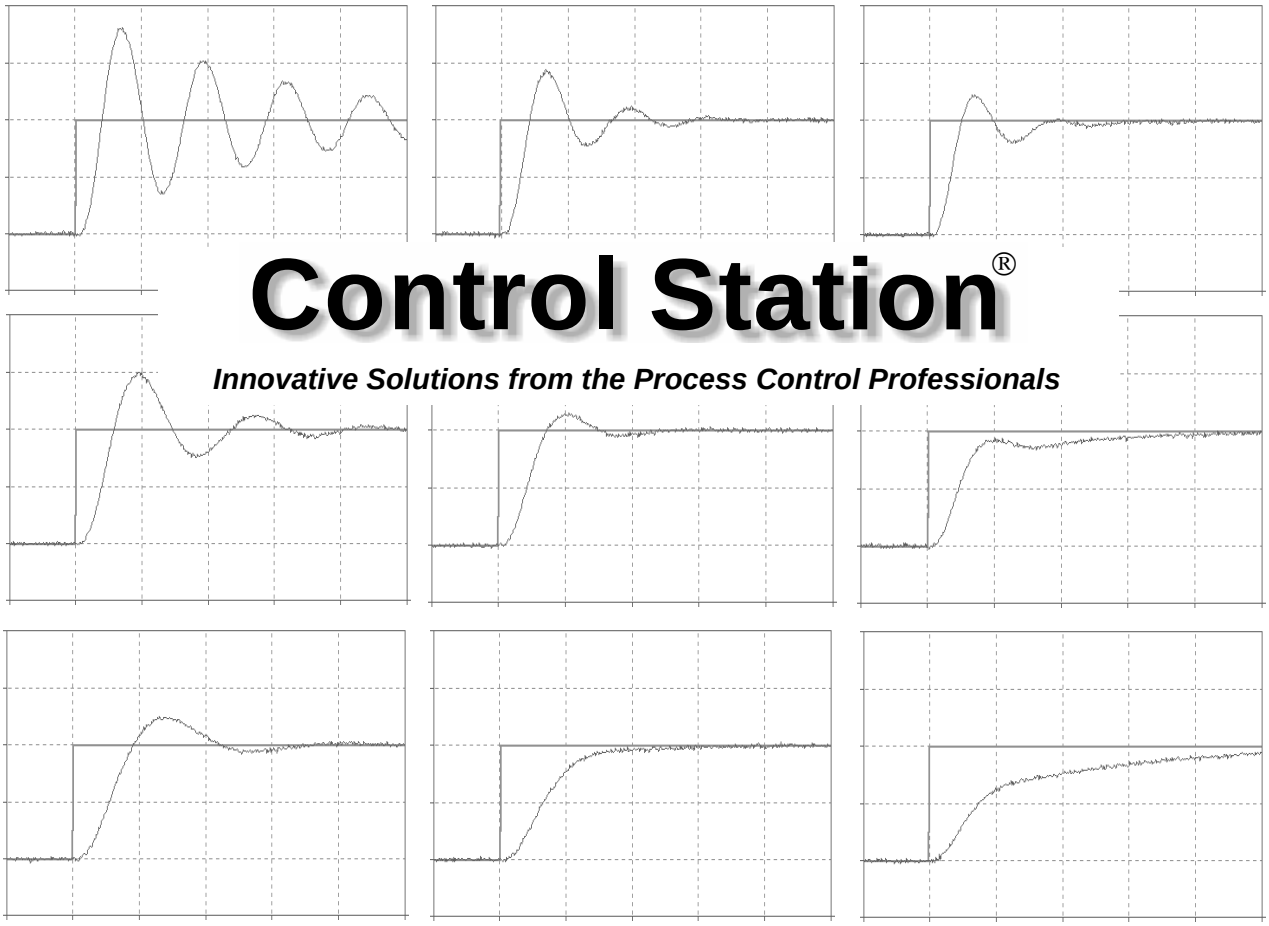


Control Station®

Innovative Solutions from the Process Control Professionals

Practical Process Control

using LOOP-PRO Software

Douglas J. Cooper

Practical Process Control Using LOOP-PRO®

Doug Cooper, professor of chemical engineering at the University of Connecticut, has been teaching and directing research in process control since 1985. Doug's research focuses on the development of advanced control strategies that are reliable, and perhaps most important, easy for practitioners to use. He strives to teach process control from a practical perspective. Thus, the focus of this book is on proven control methods and practices that practitioners and new graduates can use on the job.

Author

Prof. Douglas J. Cooper
Chemical Engineering Dept.
University of Connecticut, Unit 3222
Storrs, CT 06269-3222

Email: cooper@engr.uconn.edu

Publisher

Control Station, Inc.
One Technology Drive
Tolland, CT 06084

Email: doug.cooper@controlstation.com
Web: www.controlstation.com

Table Of Contents

1. Fundamental Principles of Process Control

1.1 Motivation for Automatic Process Control

Safety First

Automatic control systems enable a process to be operated in a safe and profitable manner. They achieve this by continually measuring process operating parameters such as temperatures, pressures, levels, flows and concentrations, and then making decisions to, for example, open valves, slow down pumps and turn up heaters so that selected process measurements are maintained at desired values.

The overriding motivation for modern control systems is safety, which encompasses the safety of people, the environment and equipment. The safety of plant personal and people in the community is the highest priority in any plant operation. The design of a process and associated control system must always make human safety the prime objective.

The tradeoff between safety of the environment and safety of equipment is considered on a case by case basis. At the extremes, a nuclear power plant will be operated to permit as much as the entire plant to be ruined rather than allowing significant radiation to be leaked to the environment. On the other hand, a fossil fuel power plant may be operated to permit an occasional cloud of smoke to be released to the environment rather than permitting damage to a multimillion dollar process unit. Whatever the priorities for a particular plant, safety of both the environment and the equipment must be specifically addressed when defining control objectives.

The Profit Motive

When people, the environment and plant equipment are properly protected, control objectives can focus on the profit motive. Automatic control systems offer strong benefits in this regard. Plant-wide control objectives motivated by profit include meeting final product specifications, minimizing waste production, minimizing environmental impact, minimizing energy use and maximizing overall production rate.

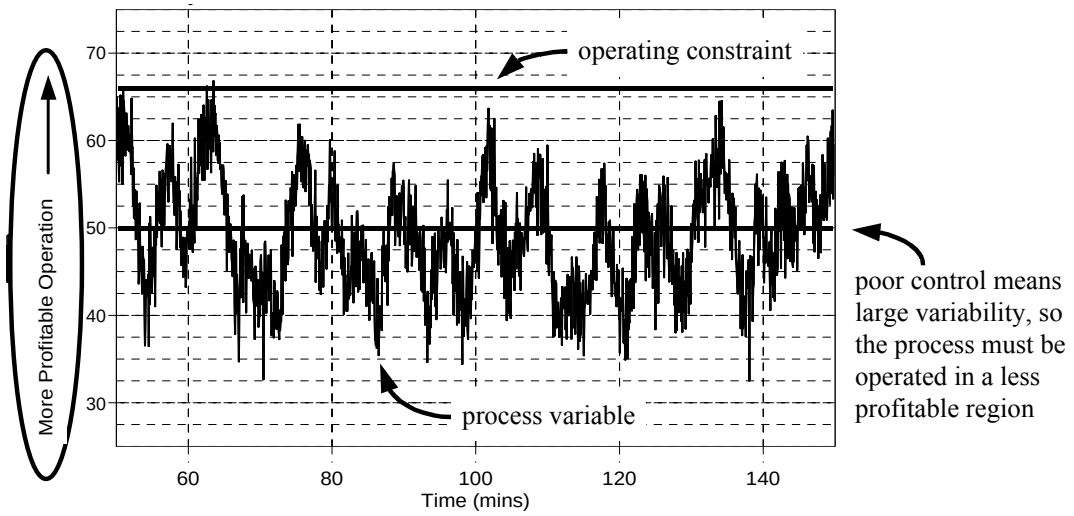


Figure 1.1 - Process variability from poor control means lost profits

Product specifications set by the marketplace (your customers) are an essential priority if deviating from these specifications lessens a product's market value. Example product specifications range from maximum or minimum values for density, viscosity or component concentration, to specifications on thickness or even color.

A common control challenge is to operate close to the minimum or maximum of a product specification, such as a minimum thickness or a maximum impurities concentration. It takes more raw material to make a product thicker than the minimum specification. Consequently, the closer an operation can come to the minimum permitted thickness constraint without going under, the greater the profit. It takes more processing effort to remove impurities, so the closer an operation can come to the maximum permitted impurities constraint without going over, the greater the profit.

All of these plant-wide objectives ultimately translate into operating the individual process units within the plant as close as possible to predetermined values of temperature, pressure, level, flow, concentration or other of the host of possible measured process variables. As shown in Fig. 1.1, a poorly controlled process can exhibit large variability in a process measurement over time. To ensure a constraint limit is not exceeded, the baseline (set point) operation of the process must be set far from the constraint, thus sacrificing profit.

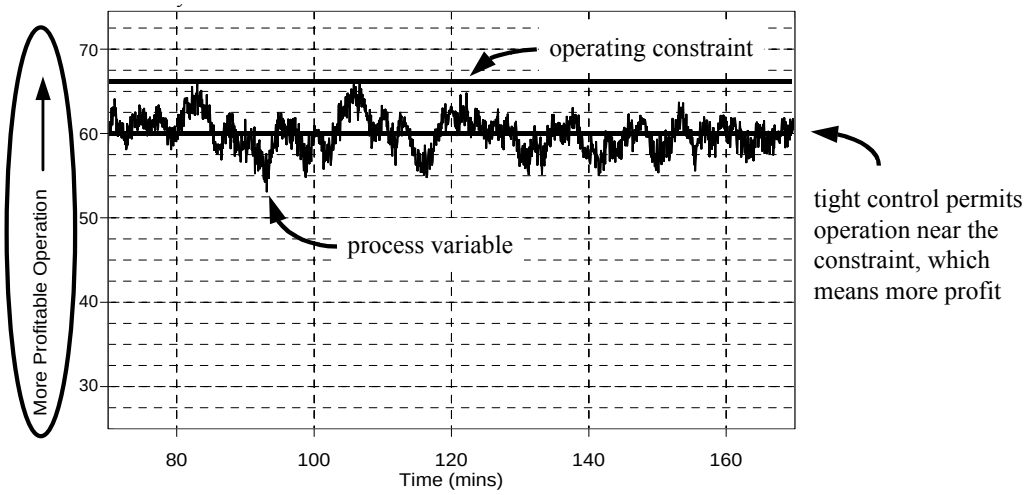


Figure 1.2 - Well controlled process has less variability in process measurements

Figure 1.2 shows that a well controlled process will have much less variability in the measured process variable. The result is improved profitability because the process can be operated closer to the operating constraint.

Automatic Process Control

Because implementation of plant-wide objectives translates into controlling a host of individual process parameters within the plant, the remainder for this text focuses on proven methods for the automatic control of individual process variables. Examples used to illustrate concepts are drawn from the LOOP-PRO® software package.

The *Case Studies* module presents industrially relevant process control challenges including level control in a tank, temperature control of a heat exchanger, purity control of a distillation column and concentration control of a jacketed reactor. These real-world challenges will provide hands-on experience as you explore and learn the concepts of process dynamics and automatic process control presented in the remainder of this book.

1.2 Terminology of Control

The first step in learning automatic process control is to learn the jargon. We introduce some basic jargon here by discussing a control system for heating a home as illustrated in Fig. 1.3. This is a rather simple automatic control example because a home furnace can only be either on or off.

As we will explore later, the challenges of control system design increase greatly when process variable adjustments can assume a complete range of values between full on and full off. In any event, a home heating system is easily understood and thus provides a convenient platform for introducing the relevant terminology.

The *control objective* for the process illustrated in Fig. 1.3 is to keep the *measured process variable* (house temperature) at the *set point* value (the desired temperature set on the thermostat by the home owner) in spite of unmeasured *disturbances* (heat loss from doors and windows opening; heat being transmitted through the walls of the house).

To achieve this control objective, the measured process variable is compared to the thermostat set point. The difference between the two is the *controller error,* which is used in a computation by the controller to compute a *controller output* adjustment (an electrical or pneumatic signal).

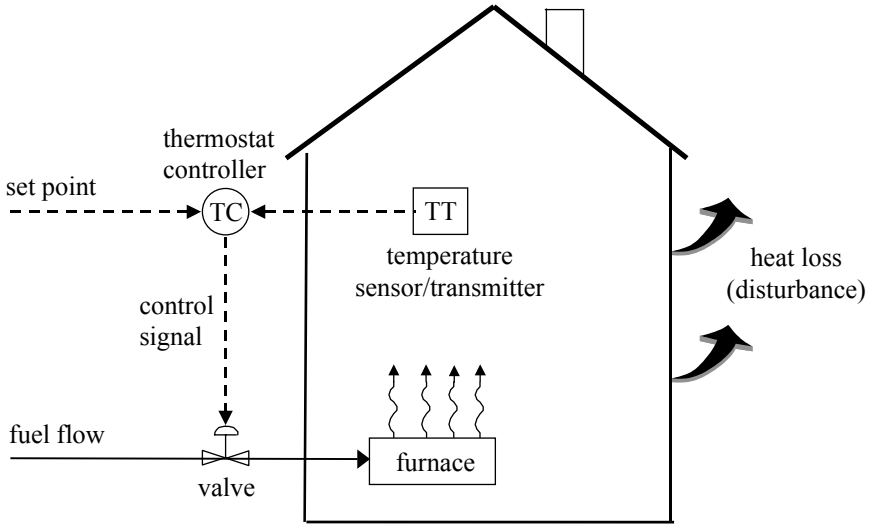


Figure 1.3 - Home heating control system

The change in the controller output signal causes a response in the *final control element* (fuel flow valve), which subsequently causes a change in the *manipulated process variable* (flow of fuel to the furnace). If the manipulated process variable is moved in the right direction and by the right amount, the measured process variable will be maintained at set point, thus satisfying the control objective. This example, like all in process control, involves a measurement, computation and action:

Measurement	*Computation*	*Action*
house temperature, T_{House}	is it colder than set point ($T_{Setpoint} - T_{House} > 0$)?	open fuel valve
	is it hotter than set point ($T_{Setpoint} - T_{House} < 0$)?	close fuel valve

Note that computing the necessary controller action is based on controller error, or the difference between the set point and the measured process variable.

1.3 Components of a Control Loop

The home heating control system of Fig. 1.3 can be organized in the form of a traditional feedback control loop block diagram as shown in Fig. 1.4. Such block diagrams provide a general organization applicable to most all feedback control systems and permit the development of more advanced analysis and design methods.

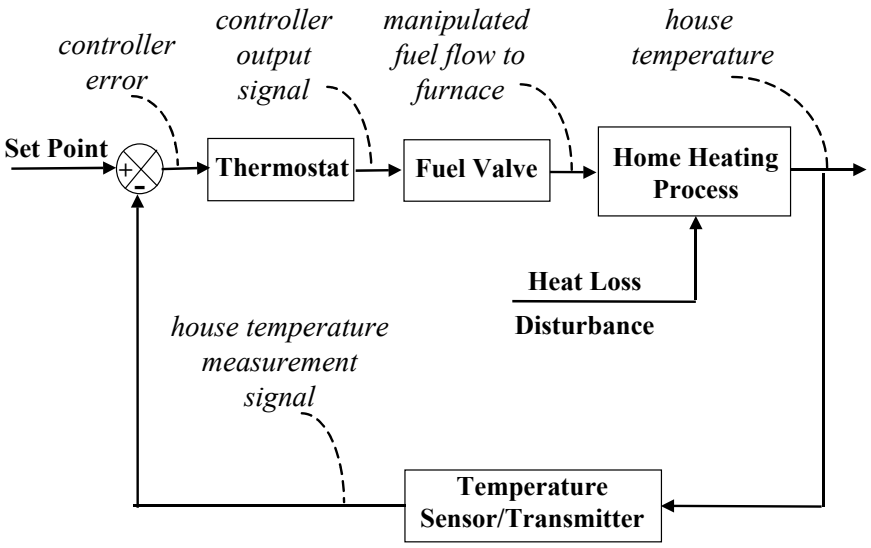


Figure 1.4 - Home heating control loop block diagram

Following the diagram of Fig. 1.4, a *sensor* measures the *measured process variable* and transmits, or feeds back, the signal to the controller. This measurement feedback signal is subtracted from the *set point* to obtain the *controller error*. The error is used by the controller to compute a *controller output* signal. The signal causes a change in the mechanical *final control element*, which in turn causes a change in the *manipulated process variable*. An appropriate change in the manipulated variable works to keep the measured process variable at set point regardless of unplanned changes in the disturbance variables.

The home heating control system of Fig. 1.4 can be further generalized into a block diagram pertinent to all *control loops* as shown in Fig. 1.5. Both these figures depict a *closed loop* system based on *negative feedback*, because the controller works to automatically counteract or oppose any drift in the measured process variable.

Suppose the measurement signal was disconnected, or opened, in the control loop so that the signal no longer feeds back to the controller. With the controller no longer in *automatic*, a person must manually adjust the controller output signal sent to the final control element if the measured process variable is to be affected.

It is good practice to adjust controller *tuning parameters* while in this *manual,* or *open loop* mode. Switching from automatic to manual, or from closed to open loop, is also a common emergency procedure when the controller is perceived to be causing problems with process operation, ranging from an annoying cycling of the measured process variable to a dangerous trend toward unstable behavior.

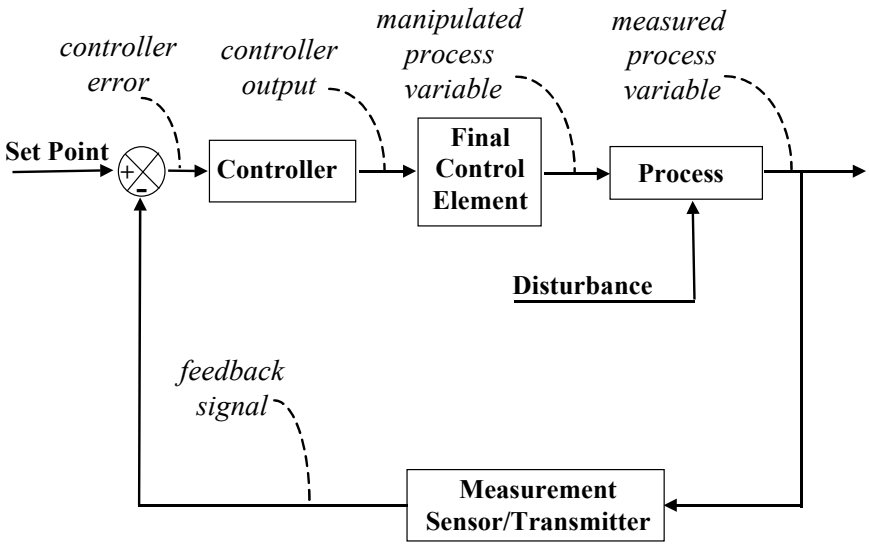


Figure 1.5 - General control loop block diagram

1.4 The Focus of This Book

Although an automatic control loop is comprised of a measurement, computation and action, details about the commercial devices available for measuring process variables and for implementing final control element actions are beyond the scope of this text. For the kinds of process control applications discussed in this book, example categories of such equipment include:

Sensors to Measure: temperature, pressure, pressure drop, level, flow, density, concentration

Final Control Elements: solenoid, valve, variable speed pump or compressor, heater or cooler

The best place to learn about the current technology for such devices is from commercial vendors, who are always happy to educate you on the items they sell. Contact several vendors and learn how their particular merchandise works. Ask about the physical principles employed, the kinds of applications the device is designed for, the accuracy and range of operation, the options available, and of course, the cost of purchase. Keep talking with different vendors, study vendor literature, visit websites and participate in sales demonstrations until you feel educated on the subject and have gained confidence in a purchase decision. Don't forget that installation and maintenance are important variables in the final cost equation.

The third piece of instrumentation in the loop is the controller itself. The automatic controllers explored in some detail in this book include:

Automatic Controllers: on/off, PID, cascade, feed forward, model-based Smith predictor, multivariable, sampled data, parameter scheduled adaptive control

Although details about the many commercial products are beyond the scope of this text, fortunately, the basic computational methods employed by most all vendors for the controllers listed above are remarkably similar. Thus, the focus of this book is to help you:

- learn how to collect and analyze process data to determine the essential dynamic behavior of a process,
- learn what "good" or "best" control performance means for a particular process,
- understand the computational methods behind each of the control algorithms listed above and learn when and how to use each one to achieve this best performance,
- learn how the different adjustable or tuning parameters required for control algorithm implementation impact closed loop performance and how to determine values for these parameters,
- become aware of the limitations and pitfalls of each control algorithm and learn how to turn this knowledge to your advantage.

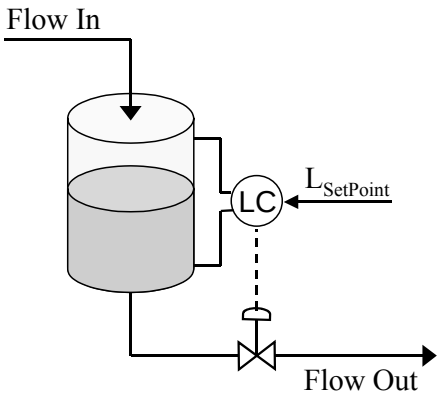
1.5 Exercises

Q-1.1 New cars often come with a feature called cruise control. To activate cruise control, the driver presses a button while traveling at a desired velocity and removes his or her foot from the gas pedal. The control system then automatically maintains whatever speed the car was traveling when the button was pressed in spite of disturbances. For example, when the car starts going up (or down) a hill, the controller automatically increases (or decreases) fuel flow rate to the engine by a proper amount to maintain the set point velocity.

a) For cruise control in an automobile, identify the:
   - control objective
   - measured process variable
   - manipulated variable
   - set point
   - two different disturbances
   - measurement sensor
   - final control element

b) Draw and properly label a closed loop block diagram for the cruise control process.

Q-1.2 The figure below shows a tank into which a liquid freely flows. The flow of liquid out of the tank is regulated by a valve in the drain line. The control objective is to maintain liquid level in the tank at a fixed or set point value. Liquid level is inferred by measuring the pressure differential across the liquid from the bottom to the top of the tank. The level sensor/controller, represented in the diagram as the LC in the circle, continually computes how much to open or close the valve in the drain line to increase or decrease the flow out as needed to maintain level at set point.



Draw and label a closed loop block diagram for this level control process.

2. *Case Studies* for Hands-On and Real-World Experience

2.1 Learning With a Training Simulator

Hands-on challenges that demonstrate and reinforce important concepts are crucial to learning the often abstract and mathematical subject of process dynamics and control. The *Case Studies* module, a training simulator that challenges you with real-world scenarios, provides this reinforcement. Use *Case Studies* to explore dozens of challenges, brought to life in color-graphic animation, to safely and inexpensively gain hands-on experience.

The *Case Studies* module contains several simulations for study. You can manipulate process variables to obtain step, pulse, ramp, sinusoidal or PRBS (pseudo-random binary sequence) test data. Process data can be recorded as printer plots and as disk files for process modeling and controller design studies. The *Design Tools* module in LOOP-PRO is well suited for this modeling and design task. After designing a controller, return to the *Case Studies* simulation to immediately evaluate and improve upon the design for both set point tracking and disturbance rejection.

The processes and controllers available in *Case Studies* enable exploration and study of increasingly challenging concepts in an orderly fashion. Early concepts to explore include basic process dynamic behaviors such as process gain, time constant and dead time. Intermediate concepts include the tuning and performance capabilities of P-Only through PID controllers and all combinations in between. Advanced concepts include cascade, decoupling, feed forward, dead time compensating and discrete sampled data control.

2.2 Simulating Noise and Valve Dynamics

For all processes, changes in the controller output signal pass through a first order dynamic response element before impacting the manipulated or disturbance variables. This simulates the lag associated with the mechanical movement of a process valve.

Normally distributed random error is added to the measured process variable for all processes to simulate measurement noise. The value entered can be user-adjusted, is in the units of the measured process variable, and represents ±3 standard deviations of the normal distribution of the random error.

2.3 Gravity Drained Tanks

The gravity drained tanks process, shown in Fig. 2.1, is two non-interacting tanks stacked one above the other. Liquid drains freely through a hole in the bottom of each tank. As shown, the measured process variable is liquid level in the lower tank. To maintain level, the controller manipulates the flow rate of liquid entering the top tank. The disturbance variable is a secondary flow out of the lower tank from a positive displacement pump. Thus, the disturbance flow is independent of liquid level except when the tank is empty.

This is a good process to start your studies because its dynamic behavior is reasonably intuitive. Increase the liquid flow rate into the top tank and the liquid level rises. Decrease the flow rate and the level falls. One challenge this process presents is that its dynamic behavior is modestly nonlinear. That is, the dynamic behavior changes as operating level changes. This is because gravity driven flows are proportional to the square root of the hydrostatic head, or height of liquid in a tank.

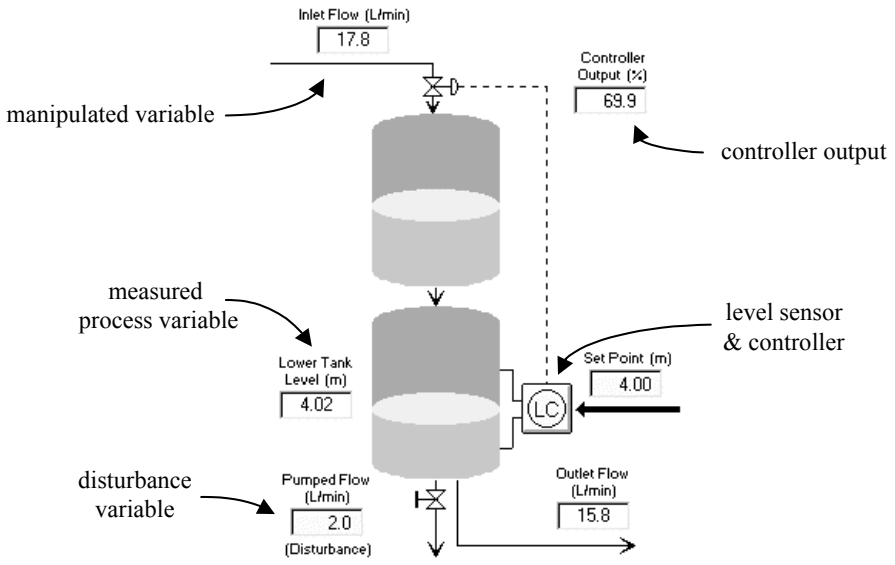


Figure 2.1 - Gravity drained tanks case study

2.4 Heat Exchanger

The heat exchanger, shown in Fig. 2.2, is a counter-current, shell and tube, lube oil cooler. The measured process variable is lube oil temperature exiting the exchanger on the tube side. To maintain temperature, the controller manipulates the flow rate of cooling liquid on the shell side. The nonlinear character of the heat exchanger, or change in dynamic process behavior evident as operating level changes, is more pronounced compared to that of the gravity drained tanks. The significance of nonlinear dynamic behavior on controller design will become apparent as we work through this book.

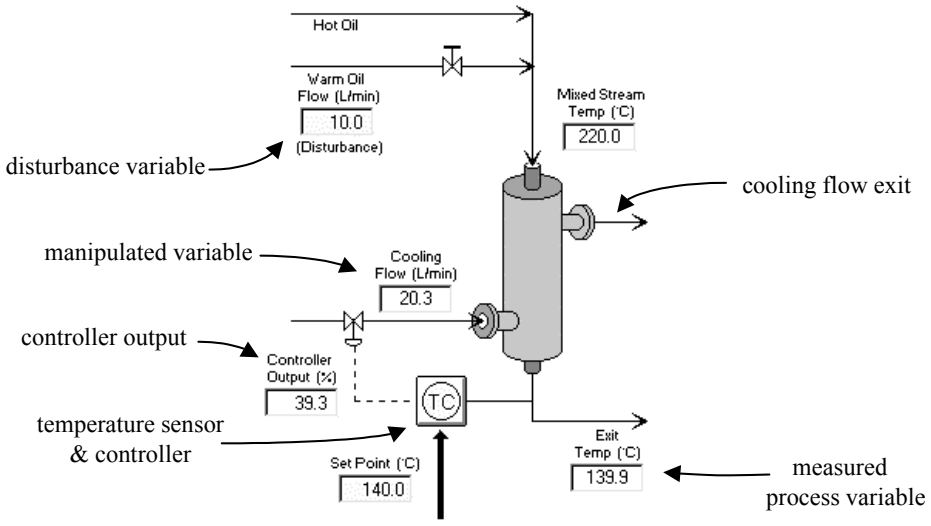


Figure 2.2 - Heat exchanger case study

This process has a negative steady state process gain. Thus, as the controller output (and thus flow rate of cooling liquid) increases, the exit temperature (measured process variable) decreases. Another interesting characteristic is that disturbances, which result from changes in the flow rate of warm oil that mixes with the hot oil entering the exchanger, cause an inverse or nonminimum phase open loop response in the measured exit temperature.

To understand this inverse response, consider that an increase in the warm oil disturbance flow increases the total flow rate of liquid passing through the exchanger. Liquid already in the exchanger when the disturbance first occurs is forced through faster than normal, reducing the time it is exposed to cooling. Hence, the exit temperature initially begins to rise. Now, the warm oil disturbance stream is cooler than the hot oil, so an increase in the disturbance flow lowers the mixed stream temperature entering the exchanger.

Once the new cooler mixed liquid works its way through the exchanger and begins to exit, it will steady out at a colder exit temperature than prior to the disturbance. Thus, an increase in the disturbance flow rate causes the measured exit temperature to first rise (from faster flow) and then decrease (from cooler mixed liquid entering) to a new lower steady state temperature.

2.5 Pumped Tank

The pumped tank process, shown in Fig. 2.3, is a pickle brine surge tank. The measured process variable is liquid brine level. To maintain level, the controller manipulates the brine flow rate out of the bottom of the tank by adjusting a throttling valve at the discharge of a constant pressure pump. This approximates the behavior of a centrifugal pump operating at relatively low throughput. The disturbance variable is the flow rate of a secondary feed to the tank.

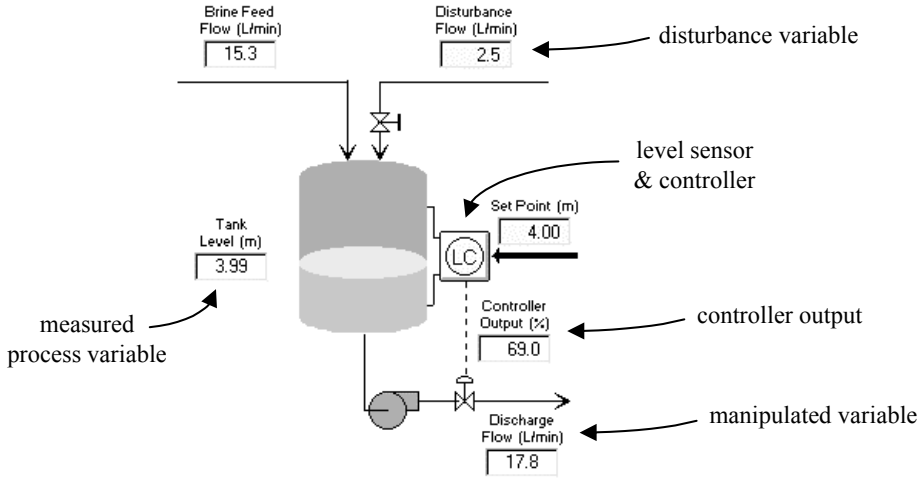


Figure 2.3 - Pumped tank case study

Unlike the gravity drained tanks, the pumped tank is not a *self regulating* process (it does not reach a natural steady state level of operation). Consider that the discharge flow rate is regulated mechanically and changes only when the controller output changes. The height of liquid in the tank does not impact the discharge flow rate. As a result, when the total flow rate into the tank is greater than the discharge flow rate, tank level will continue to rise until the tank is full, and when the total flow rate into the tank is less than the discharge flow rate, the tank level will fall until empty.

This non-self-regulating dynamic behavior is associated with integrating processes. The pumped tank appears almost trivial in its simplicity. Its integrating nature presents a remarkably difficult control challenge.

2.6 Jacketed Reactor

The jacketed reactor, shown in Fig. 2.4 for the single loop case, is a continuously stirred tank reactor (CSTR) in which the first order irreversible exothermic reaction A→B occurs. Residence time is constant in this perfectly mixed reactor, so the steady state conversion of reactant A from the reactor can be directly inferred from the temperature of the reactor product stream. To control reactor temperature, the vessel is enclosed with a jacket through which a coolant passes.

The measured process variable is the product exit stream temperature. To maintain temperature, the controller manipulates the coolant flow rate through the jacket. The disturbance variable is the inlet temperature of coolant flowing through the jacket.

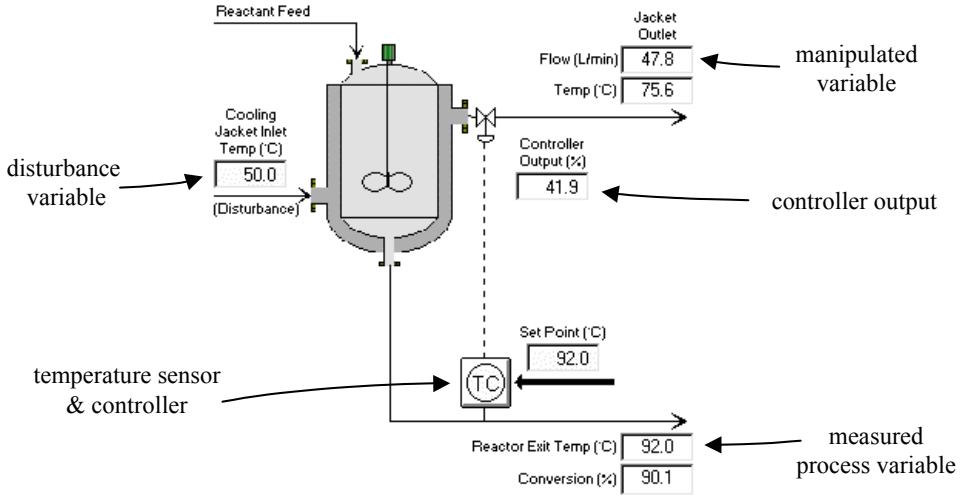


Figure 2.4 – Single loop jacketed reactor case study

This process has an upper and lower operating steady state. The process initializes at the upper steady state, indicated by high values for percent conversion in the exit stream. You can move the process to its lower steady state by dropping the cooling jacket inlet temperature to low values (for example, with the controller output at the startup value of 42%, change the jacket inlet temperature to 30°C and the reactor will fall to the lower operating steady state).

The process is modeled following developments similar to those presented in many popular chemical engineering texts. Assuming an irreversible first order reaction (A→B); perfect mixing in the reactor and jacket; constant volumes and physical properties; and negligible heat loss, the model is expressed:

Mass balance on Reactor A:

$$\frac{dC_A}{dt} = \frac{F}{V}\left(C_{A0} - C_A\right) - kC_A$$

Energy balance on reactor contents:

$$\frac{dT}{dt} = \frac{F}{V}\left(T_0 - T\right) - \frac{\Delta H_R}{\rho C_P} kC_A - \frac{UA}{V\rho C_P}\left(T - T_J\right)$$

Energy balance on reactor jacket: $$\frac{dT_J}{dt} = \frac{UA}{V_J \rho_J C_{PJ}}(T - T_J) + \frac{F_J}{V_J}(T_{J0} - T_J)$$

Reaction rate coefficient: $$k = k_0 e^{-E/RT}$$

2.7 Cascade Jacketed Reactor

The open loop process behavior of the cascade jacketed reactor, shown in Fig. 2.5, is identical to the single loop case. The only difference between the two is the controller architecture. A control cascade consists of two process measurements, two controllers, but only one final control element - the same final control element as in the single loop case study. The benefit of a cascade architecture, as explored later in Chapter 18, is improved disturbance rejection.

For the cascade case, the secondary measured process variable is the coolant temperature out of the jacket. The secondary or inner loop controller manipulates the cooling jacket flow rate. As in the single loop case, the primary or outer measured process variable is still the product stream temperature. As with all cascade architectures, the output of the primary controller is the set point of the secondary controller.

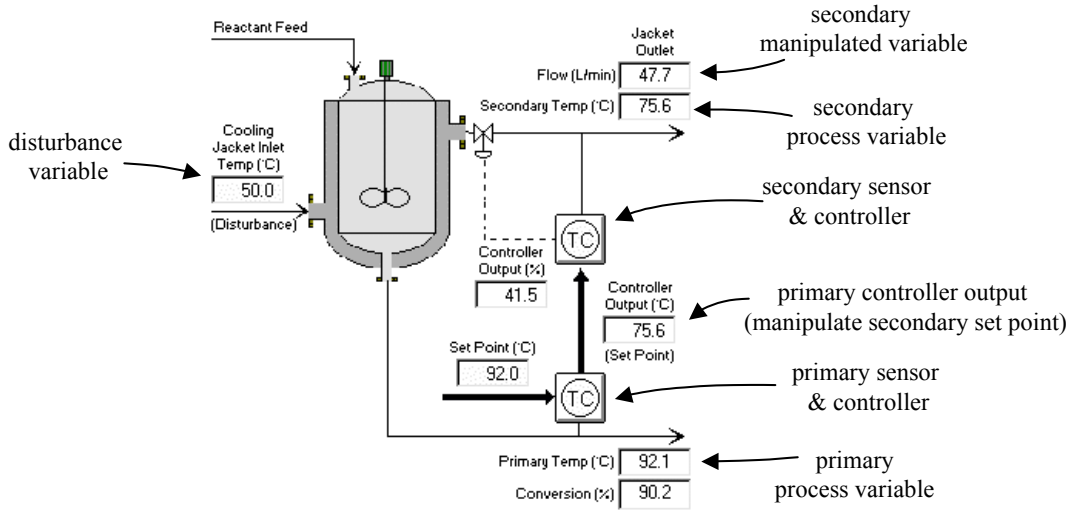


Figure 2.5 - Cascade jacketed reactor case study

2.8 Furnace Air/Fuel Ratio

The furnace air/fuel ratio case study is a process in which a furnace burns natural gas to heat a process liquid flowing through tubes at the top of the fire box. As shown in Fig. 2.6, the measured process variable is the temperature of the process liquid as it exits the furnace. To maintain temperature, controllers adjust the feed rate of combustion air and fuel to the fire box using a ratio control strategy. The flow rate of the process liquid acts as a load disturbance to the process.

A ratio controller measures the flow rate of an independent or "wild" stream. It then adjusts the flow rate of a second dependent stream to maintain a user specified ratio between the two. Blending operations are a typical application for ratio control, and here we are blending air and fuel in a desired fashion.

For the furnace, the independent stream is the combustion air flow rate and the dependent stream is the fuel flow rate. Note that while air flow rate is considered the independent stream for ratio control, its flow rate is specified by the temperature controller on the process liquid exiting the furnace.

Air/fuel ratio control provides important environmental, economic and safety benefits. Running in a fuel-rich environment (too little air to complete the combustion reaction) permits the evolution of carbon monoxide and unburned hydrocarbons in the stack gases. These components are not only pollutants, but unburned hydrocarbon represents wasted energy. A fuel-lean environment (air in excess of that needed to complete combustion) results in economic loss because it takes extra fuel to heat extra air that is then just lost up the stack. Also, excess air can lead to the increased production of nitrogen oxides that promote the formation of smog. In general, it is desirable to maintain the ratio of air to fuel at a level that provides a small excess of oxygen, say 2-5%, compared to the stoichiometric requirement.

The ratio controller also provides an important safety benefit. As shown in Fig. 2.6, it is the air flow rate that is adjusted by the temperature controller on the liquid exiting the furnace. When the liquid temperature falls below set point, the temperature controller raises the set point flow rate of the combustion air. As the air flow ramps up, the air flow transmitter detects the change and sends the new flow rate to the ratio controller. The ratio controller responds by increasing the set point to the fuel flow controller in order to maintain the specified ratio.

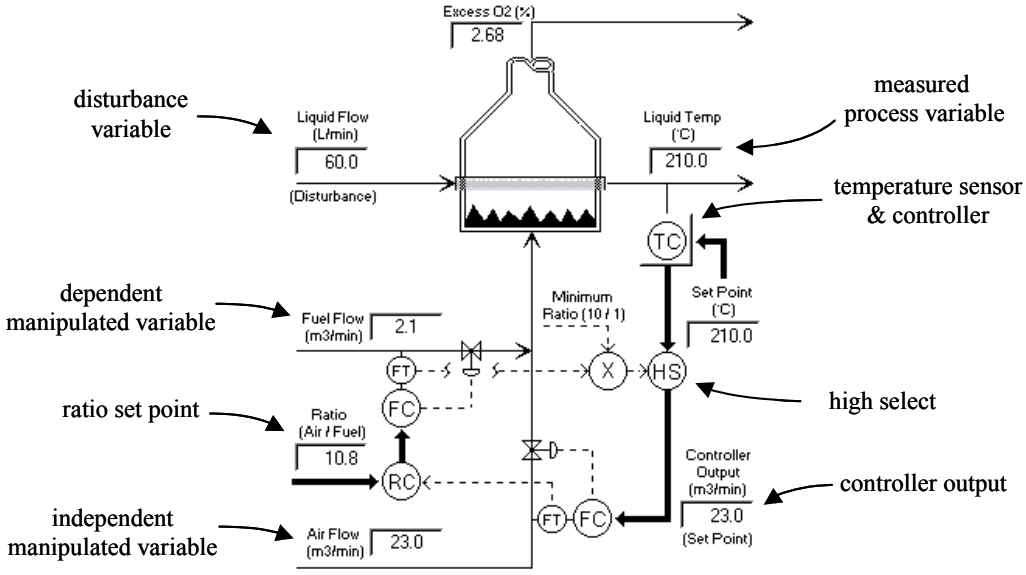


Figure 2.6 -Furnace Air/Fuel Ratio case study

To understand why this is the safer alternative, consider what would happen if the plant were to lose combustion air (e.g. the air compressor dies). The air flow transmitter will immediately detect the falling flow rate and send the information to the ratio controller. The ratio controller will respond by cutting the fuel flow. As a result, the hazardous situation of fuel without sufficient combustion air in the furnace fire box will be averted.

Now suppose we make fuel the independent stream and control air flow rate in ratio to the fuel. If we were to lose air as the liquid temperature controller calls for more fuel, the ratio controller will not be aware that a problem exists because it will only be receiving information from the fuel flow transmitter. The ratio controller will continue to send fuel to the furnace in an attempt to raise temperature, and an explosive environment will develop.

An additional piece of the safety control strategy lies with the high select component (the HS with the circle around it on the process graphic that appears when the temperature controller is in automatic). As shown in Fig. 2.6, the high select receives two candidate set points. One is the air flow rate set point requested by the temperature controller. The other is the minimum air flow rate permitted given the current flow rate of fuel. The minimum air/fuel ratio permitted is 10/1, and to go below this value means there will be more fuel in the fire box than air needed to combust it. The high select sends the highest of the two candidate set points to the air flow controller, thus ensuring that a fuel rich environment is not created. This intervention in the control loop creates interesting control challenges.

2.9 Multi-Tank Process

The multi-tank process is a multivariable version of the single loop gravity drained tanks process. As shown in the process graphic of Fig. 2.7, there are two sets of freely draining tanks positioned side by side. The two measured process variables are the liquid levels in the lower tanks. To maintain level, the two level controllers manipulate the flow rate of liquid entering their respective top tanks.

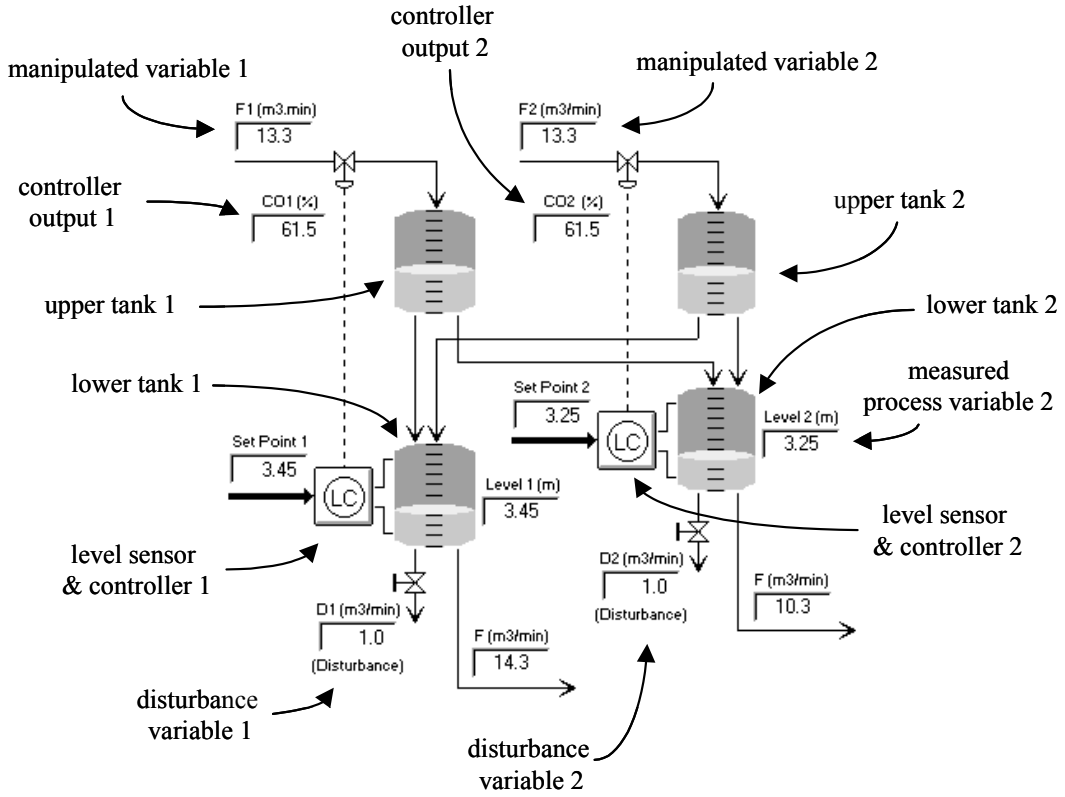


Figure 2.7 – Multi-Tank case study

Similar to the single loop case, the gravity driven flows are proportional to the square root of the height of liquid in the tank (hydrostatic head), so this process displays moderately nonlinear behavior. Each lower tank has a secondary flow out of the lower tank that is independent of liquid level and acts as an operating disturbance.

An important feature of this process is that each of the upper tanks drains into *both* lower tanks. This creates a multivariable interaction because actions by one controller affect both measured process variables. Note that the characteristics of each drain stream are all different so there is no symmetry between feed flow rate and steady state tank level.

To gain a better understanding of the multivariable loop interactions, suppose the measured level in lower tank 1 is below set point. The tank 1 level controller responds by increasing F1, the flow rate entering upper tank 1. While this action will cause the level to rise in lower tank 1, the side drain out of upper tank 1 will increase, causing the level in lower tank 2 to rise also.

As the level in lower tank 2 is forced from set point, the tank 2 level controller compensates by decreasing the flow rate of F2, the flow rate entering upper tank 2. This decreases the level in tank 2, but the decrease in the side drain rate out of upper tank 2 causes the level in lower tank 1 to decrease. The tank 1 level controller "fights back" with more corrective actions, causing challenging multivariable loop interactions.

Decouplers are simple models of the process that can be designed into a controller architecture to minimize such multivariable interaction. As we will learn in Section 20.3, decouplers are feed forward elements that treat the actions of another controller as a measured disturbance.

2.10 Multivariable Distillation Column

The distillation column case study, shown in Fig. 2.8, consists of a binary distillation column that separates benzene and toluene. The objective is to send a high percentage of benzene (and thus low percentage of toluene) out the top distillate stream and a low percentage of benzene (and thus high percentage of toluene) out the bottoms stream. The column dynamic model employs tray-by-tray mass and energy calculations similar to that proposed by McCune and Gallier [*ISA Transactions*, **12**, 193, (1973)].

To separate benzene from toluene, the top controller manipulates the reflux rate to control the distillate composition. The bottom controller adjusts the rate of steam to the reboiler to control the bottoms composition. Any change in feed rate to the column acts as a disturbance to the process.

To understand the loop interaction in this multivariable process, suppose the composition (or purity) of benzene in the top distillate stream is below set point. The top controller will respond by increasing the flow rate of cold reflux into the column. This increased reflux flow will indeed increase the benzene purity of the distillate stream. However, the additional cold liquid will work its way down the column, begin to cool the bottom, and as a result permit more benzene to flow out the bottoms stream.

As the bottoms composition moves off set point and produces a controller error, the bottom controller will compensate by increasing the flow of steam into the reboiler to heat up the bottom of the column. While this works to restore the desired purity to the bottoms stream, unfortunately, it also results in an increase of hot vapors traveling up the column that eventually will cause the top of the column to begin to heat up.

As the top of the column heats up, the purity of benzene in the distillate stream again becomes too low. In response, the top controller compensates by further increasing the flow of cold reflux into the top of the column. The controller "fight," or multivariable interaction, begins. Like the multi tank process, decouplers can minimize such multivariable loop interaction.

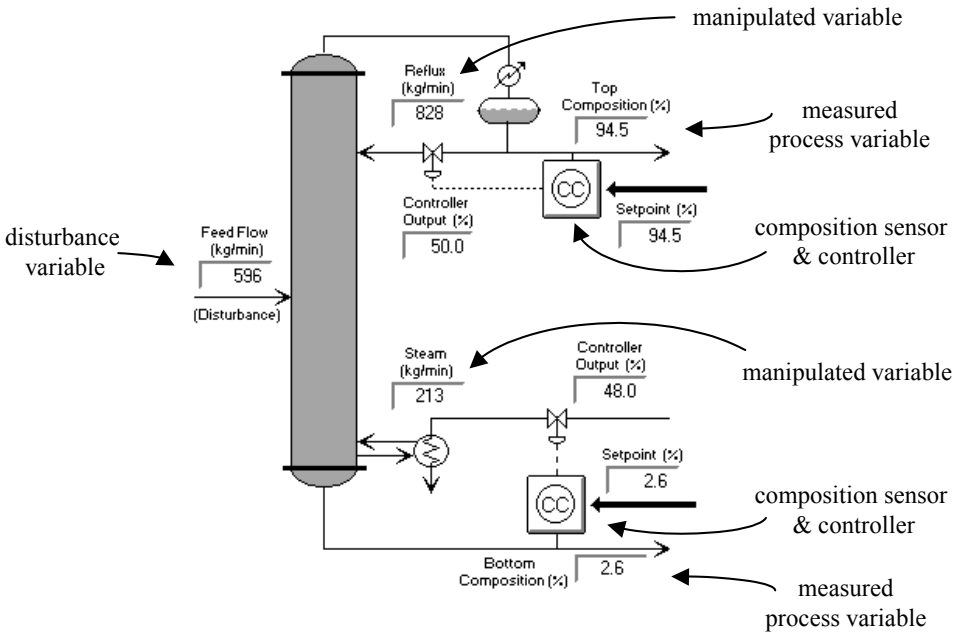


Figure 2.8 - Distillation column case study

2.11 Exercises

Q-2.1 Measurement sensors are not discussed in this book but are a critical part of any control loop. Among other features, consider that a sensor should:

- generate a continuous electrical signal that can be transmitted to the controller,
- be reasonably priced and easy to install and maintain,
- be capable of withstanding the rigors of the environment in which they are placed,
- respond quickly to changes in the measured process variable,
- be sufficiently accurate and properly calibrated to the application.

Using only information you obtain from the world wide web, research and then specify such details for a:

a) liquid level sensor suitable for both the gravity drained tanks *and* the pumped tank process

b) temperature sensor for the heat exchanger process

c) temperature sensor for the single loop jacketed reactor process

3. Modeling Process Dynamics - A Graphical Analysis of Step Test Data

3.1 Dynamic Process Modeling for Control Tuning

Consider the design of a cruise control system for a car versus that for an eighteen wheel truck. For the two vehicles, think about:

- how quickly each vehicle can accelerate or decelerate using the gas pedal, and
- the effect of disturbances on each vehicle such as wind, hills and other vehicles passing by.

The actions that a properly designed controller must take to control a car versus a truck, that is, how the controller output signal should manipulate gas flow to maintain a constant set point velocity in spite of disturbances such as wind and hills, will differ because the *dynamic behavior* of each vehicle "process" is different.

The dynamic behavior of a process describes how the measured process variable changes with time when forced by the controller output and any disturbance variables. The manner in which a measured process variable responds over time is fundamental to the design and tuning of an automatic controller.

The best way to learn about the dynamic behavior of a process is to perform experiments. Whether the process to be controlled is found in LOOP-PRO, a lab or a production facility, this experimental procedure entails the following steps:

1) The controller output is stepped, pulsed or otherwise perturbed, usually in manual mode, and as near as practical to the design level of operation,
2) The controller output and measured process variable data are recorded as the process responds,
3) A first order plus dead time (FOPDT) dynamic model is fit to this process data,
4) The resulting FOPDT dynamic model parameters are used in a correlation to obtain initial estimates of the controller tuning parameters,
5) The tuning parameters are entered into the controller, the controller is put in automatic and controller performance is evaluated in tracking set points and/or rejecting disturbances,
6) Final tuning is performed on-line and by trial and error until desired controller performance is obtained.

Generating experimental data for step 1 above is best done in manual mode (open loop). The goal is to move the controller output far enough and fast enough so that the dynamic character of the process is revealed as the measured process variable responds. Because dynamic process behavior usually differs as operating level changes (real processes display this nonlinear behavior), these experiments should be performed at the design level of operation (where the set point will be set during normal operation).

The response of the measured process variable during a test must clearly be the result of the change in the controller output. To obtain a reliable model from test data, the measured process variable should be forced to move at least 10 times the size of the noise band, or 10 times farther than the random changes resulting from noise in the measurement signal that are evident prior to the start of the test. If process disturbances occurring during a dynamic test influence the response of the measured process variable, the test data is suspect and the experiment should be repeated.

It is becoming increasingly common for dynamic studies to be performed with the controller in automatic (closed loop). For closed loop studies, the dynamic data is generated by stepping, pulsing or otherwise perturbing the set point. Closed loop testing can be problematic because, in theory, the information contained in the data will reflect the character of the controller as well as that of the process. In practice, however, the controller character rarely produces significant corruption in the final FOPDT model parameter values. In the unhappy event that you are not permitted to perform either open or closed loop dynamic tests on a process, a final alternative is to search data logs for useful data that have been recorded in the recent past.

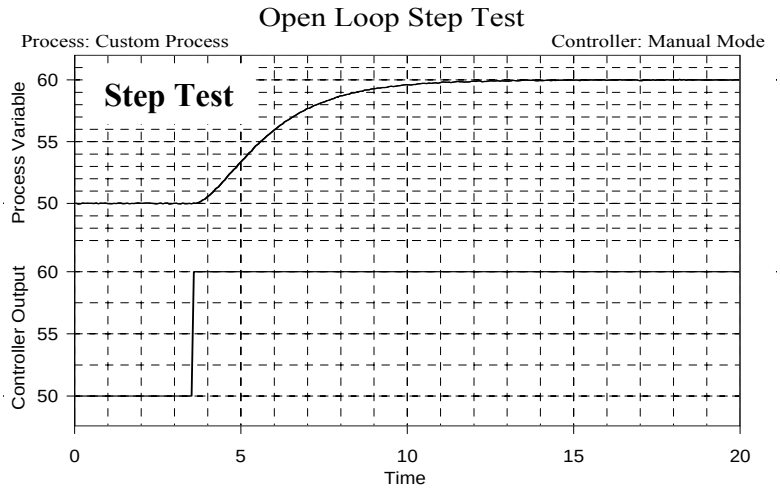


Figure 3.1 - Controller output step with measured process variable response

The dynamic process data set is then analyzed in step 3 to yield the three parameters of a linear first order plus dead time (FOPDT) dynamic process model:

$$\tau_P \frac{dy(t)}{dt} + y(t) = K_P\, u(t - \theta_P) \tag{3.1}$$

where $y(t)$ is the measured process variable and $u(t)$ is the controller output signal. When Eq. 3.1 is fit to the test data, the all-important parameters that describe the dynamic behavior of the process result:

- Steady State Process Gain, K_P
- Overall Process Time Constant, τ_P
- Apparent Dead Time, θ_P

The importance of these three model parameters is revealed in step 4, where they are used in correlations to compute initial tuning values for a variety of controllers. We will also learn later that this model is important because, for example:

- the sign of K_P indicates the sense of the controller ($+K_P \rightarrow$ reverse acting; $-K_P \rightarrow$ direct acting)
- the size of τ_P indicates the maximum desirable loop sample time (be sure sample time $T \leq 0.1\tau_P$)
- the ratio θ_P/τ_P indicates whether a Smith predictor would show benefit (useful if $\theta_P > \tau_P$)
- the dynamic model itself can be employed within the architecture of feed forward, Smith predictor, decoupling and other model-based controller strategies.

Thus, the collection and modeling of dynamic process data are indeed critical steps in controller design and tuning.

3.2 Generating Step Test Data for Dynamic Process Modeling

The popular experiments to generate dynamic process data include the step test, pulse test, doublet test and pseudo-random binary sequence (PRBS) test. Data generated by a pulse, doublet or PRBS require a computer tool for fitting the dynamic model and are discussed later in Section 6.3.

Here we explore the fitting of a FOPDT dynamic model to step test data using a hand-analysis of the process response plot. Our goal is to obtain values for process gain, K_P, overall time constant, τ_P, and apparent dead time, θ_P, that reasonably describe the dynamic behavior of the process and thus can be used for controller design and tuning.

Figure 3.1 shows a typical step test response plot. As prescribed in the experimental test procedure described earlier, the controller is in manual mode and the controller output and measured process variable are initially at steady state. The test entails stepping the controller output to a new constant value and collecting data as the measured process variable responds completely to its final steady state.

3.3 Process Gain, K_P, From Step Test Data

The steady state process gain describes how much the measured process variable, $y(t)$, changes in response to changes in the controller output, $u(t)$. A step test starts and ends at steady state, so K_P can be determined directly from the plot axes. Specifically, K_P is computed as the steady state change in the measured process variable divided by the change in the controller output signal that forced that change, or:

$$K_P = \frac{\text{Steady State Change in the Measured Process Variable, } \Delta y(t)}{\text{Steady State Change in the Controller Output, } \Delta u(t)} \tag{3.2}$$

where $\Delta u(t)$ and $\Delta y(t)$ represent the total change from initial to final steady state.

Example: Gravity Drained Tanks

Figure 3.2 shows open-loop step response data from the gravity drained tanks. The process is initially at steady state with the controller output at 50%, while the pumped flow disturbance (not shown on plot) remains constant at 2.0 L/min. The controller output is stepped from 50% up to a new steady state value of 60%. This step increase causes the valve to open, increasing the liquid flow rate into the top tank.

In response, the measured variable for this process, the liquid level in the lower tank, rises from its initial steady state value of 1.93 m up to a new steady state level of 2.88 m. Note that the response plot of Fig. 3.2 displays data collected on either side of an average measured liquid level of about 2.4 m.

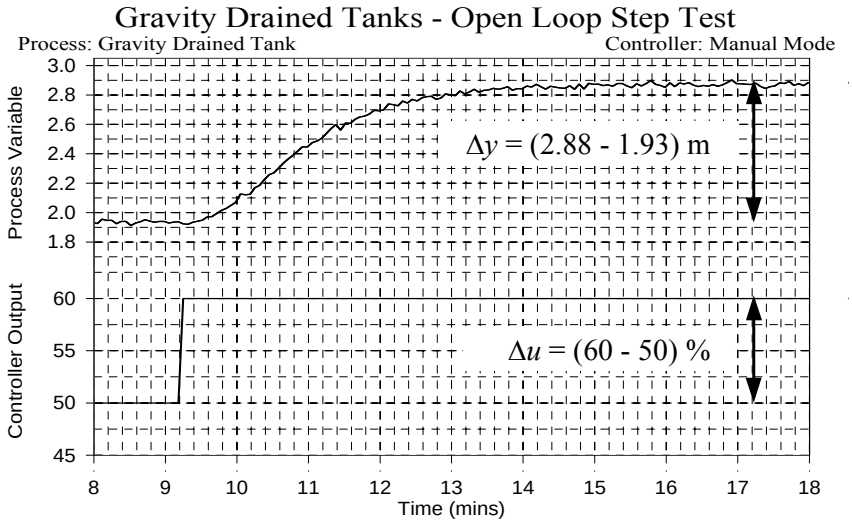


Figure 3.2 - K_P computed from gravity drained tanks step test plot

Using Eq. 3.2, the steady state process gain for this process is:

$$K_P = \frac{\Delta y}{\Delta u} = \frac{2.88 - 1.93 \text{ m}}{60 - 50\,\%} = 0.095 \frac{\text{m}}{\%}$$

Note that K_P has a size (0.095), a sign (positive, or +0.095 in this case), and units (m/%).

Example: Heat Exchanger

Figure 3.3 shows open loop step response data from the heat exchanger. The controller output is initially constant at 25% while the warm oil disturbance flow rate (not shown on plot) remains constant at 10 L/min. The controller output is stepped from 25% up to a new steady state value of 35%. This step increase causes the valve to open, increasing the flow rate of cooling liquid into the shell side of the exchanger.

The additional cooling liquid causes the measured variable for this process, liquid exit temperature on the tube side, to decrease from its initial steady state value of 151.2°C down to a new steady state value of 142.6°C. Note that the response plot of Fig. 3.3 displays data collected on either side of an average measured exit temperature of about 147°C.

Using Eq. 3.2, the steady state process gain is:

$$K_P = \frac{\Delta y}{\Delta u} = \frac{142.6 - 151.2 \text{ °C}}{35 - 25\,\%} = -0.86 \frac{\text{°C}}{\%}$$

Again, K_P has a size (–0.86), a sign (negative, or –0.86 in this case), and units (°C/%).

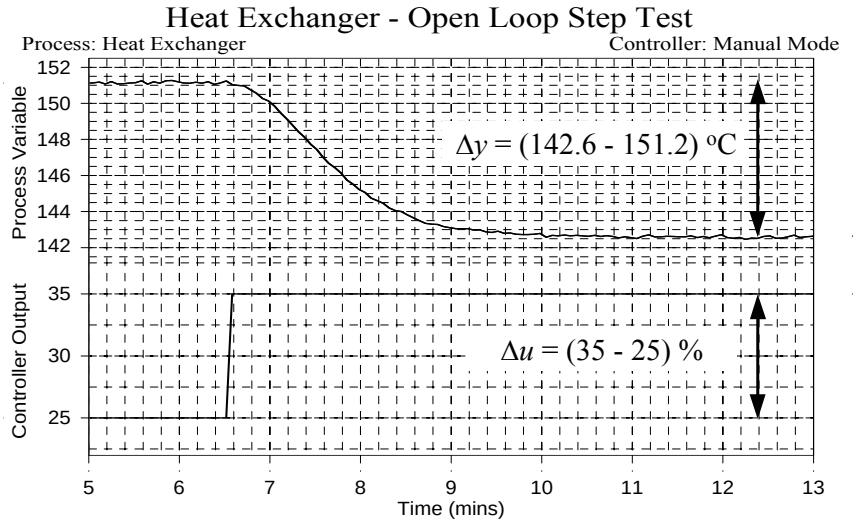


Figure 3.3 - K_P computed from heat exchanger step test plot

☆ ☆ ☆

3.4 Overall Time Constant, τ_P, From Step Test Data

The overall process time constant describes how fast a measured process variable responds when forced by a change in the controller output. The clock that measures "how fast" does not start until the measured process variable shows a clear and visible response to the controller output step. We will soon see in Section 3.5 that the delay that occurs from when the controller output is stepped until the measured process variable actually starts to respond is the process dead time. To estimate τ_P from step test data:

1) Locate on the plot where the measured process variable first shows a clear initial response to the step change in controller output. The time where this response starts is $t_{Y\text{start}}$.
2) Locate on the plot the time when the measured variable process reaches $y_{63.2}$, which is the point where $y(t)$ has traveled 63.2% of the total change it is going to experience in response to the controller output step. Label $t_{63.2}$ as the point in time where $y_{63.2}$ occurs.
3) The time constant, τ_P, is then estimated as the time difference between $t_{Y\text{start}}$ and $t_{63.2}$. Since the time constant marks the passage of time, it must have a positive value.

The value 63.2% is derived assuming the process dynamic character is exactly described by the linear FOPDT model of Eq. 3.1. Details can be found in Section 13.3 (Deriving the τ_p = 63.2% of Process Step Response Rule). In reality, no real process is exactly described by this model form. Nevertheless, the procedure most always produces a time constant value sufficiently accurate for controller design and tuning procedures.

Example: Gravity Drained Tanks

Figure 3.4 shows the same step response data from the gravity drained tanks used in the previous K_P calculation. With the pumped flow disturbance constant at 2.0 L/min, the controller output is stepped from 50% up to 60% and the measured liquid level responds from an initial steady state of 1.93 m up to a new steady state of 2.88 m.

Applying the time constant analysis procedure to this plot:

1) The time when the measured process variable starts showing a clear initial response to the step change in controller output is t_{Ystart}. From the plot:

$$t_{Ystart} = 9.6 \text{ min}$$

2) The measured process variable starts at steady state at 1.93 m and exhibits a total response of $\Delta y = 0.95$ m. Thus, $y_{63.2}$ is computed:

$$y_{63.2} = 1.93 \text{ m} + 0.632(\Delta y) = 1.93 \text{ m} + 0.632(0.95 \text{ m}) = 2.53 \text{ m}$$

From the plot, $y(t)$ passes through 2.53 m at:

$$t_{63.2} = 11.2 \text{ min}$$

3) The overall time constant for this process response is then:

$$\tau_P = t_{63.2} - t_{Ystart} = 11.2 \text{ min} - 9.6 \text{ min} = 1.6 \text{ min}$$

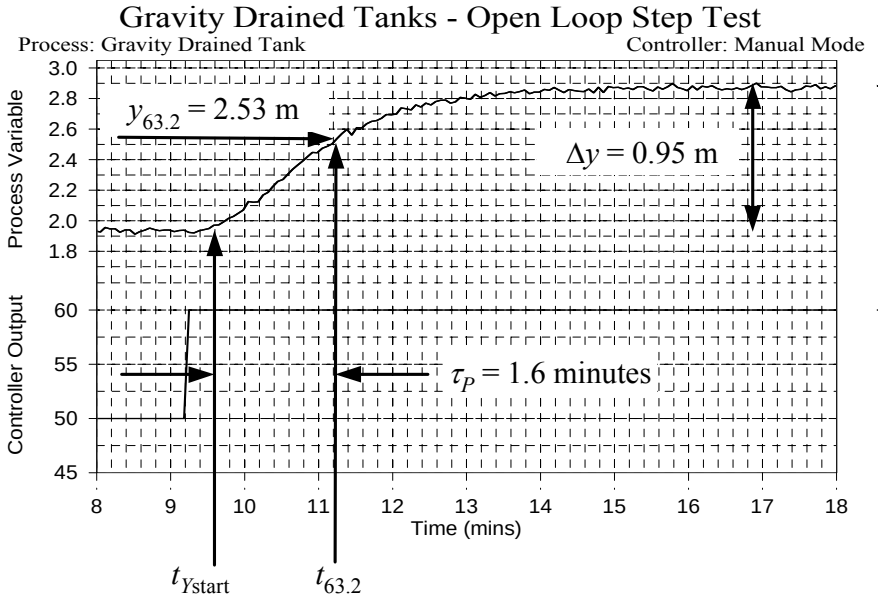


Figure 3.4 - τ_P computed from gravity drained tanks step test plot

☆ ☆ ☆

Example: Heat Exchanger
Figure 3.5 shows the same step test data used in the previous K_P calculation. With the warm oil disturbance flow rate constant at 10 L/min, the controller output is stepped from 25% up to 35% and the measured exit temperature displays a negative response as it moves from an original steady state of 151.2°C down to a final steady state of 142.6°C.

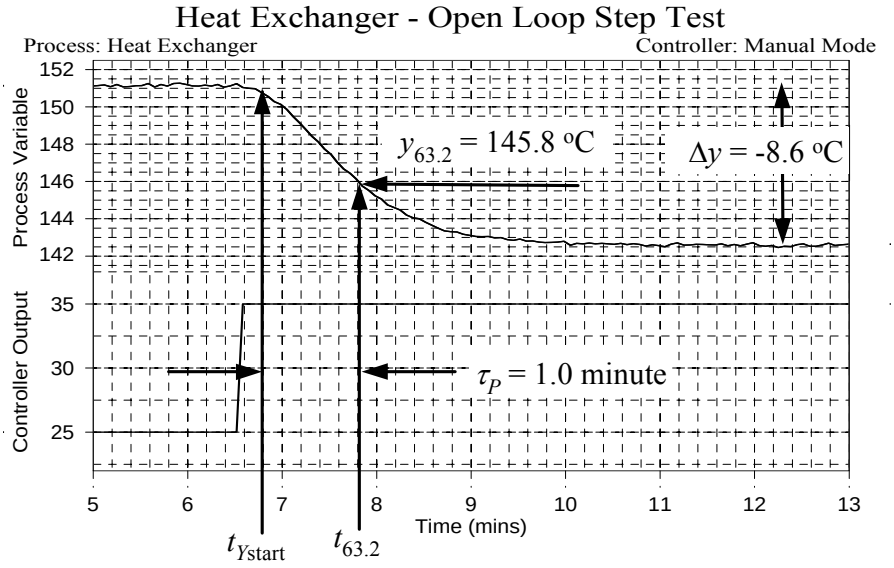


Figure 3.5 - τ_P computed from heat exchanger step test plot

Applying the time constant analysis procedure to this plot:

1) The time when the measured process variable starts showing a clear initial response to the step change in controller output is t_{Ystart}. From the plot:

$$t_{Ystart} = 6.8 \text{ min}$$

2) The measured process variable starts at steady state at 151.2°C and exhibits a total response of $\Delta y = -8.6$°C. Thus, $y_{63.2}$ is computed:

$$y_{63.2} = 151.2°C + 0.632(\Delta y) = 151.2°C + 0.632(-8.6°C) = 145.8°C$$

From the plot, $y(t)$ passes through 145.8°C at:

$$t_{63.2} = 7.8 \text{ min}$$

3) The overall time constant for this process response is then:

$$\tau_P = t_{63.2} - t_{Ystart} = 7.8 \text{ min} - 6.8 \text{ min} = 1.0 \text{ min}$$

3.5 Apparent Dead Time, θ_P, From Step Test Data

Dead time is the time that passes from the moment the step change in the controller output is made until the moment when the measured process variable shows a clear initial response to that change. Dead time arises because of transportation lag (the time it takes for material to travel from one point to another) and sample or instrument lag (the time it takes to collect, analyze or process a measured variable sample).

Dead time can also appear to exist in higher order processes simply because such processes are slow to respond to a change in the controller output signal. Overall or *apparent* dead time, θ_P, refers to the sum of dead times evident from all sources.

If $\theta_P > \tau_P$, tight control of a process becomes challenging, and the larger dead time is relative to the process time constant, the worse the problem. For important loops, every effort should be made to avoid introducing unnecessary dead time. This effort should start at the early design stages, continue through the selection and location of sensors and final control elements, and persist up through final installation and testing.

The procedure for estimating dead time from step response data is:

1) Locate on the plot $t_{U\text{step}}$, the time when the controller output step, $\Delta u(t)$, is made.

2) As in the previous τ_P calculation, locate $t_{Y\text{start}}$, the time when the measured process variable starts showing a clear initial response to this controller output step.

3) The apparent dead time, θ_P, is then estimated as the difference between $t_{U\text{step}}$ and $t_{Y\text{start}}$. Since dead time marks the passage of time, it must have a positive value.

Example: Gravity Drained Tanks
Figure 3.6 shows the same step response data from the gravity drained tanks used in the previous calculations. Applying the dead time analysis procedure to the step test plot of Fig 3.6:

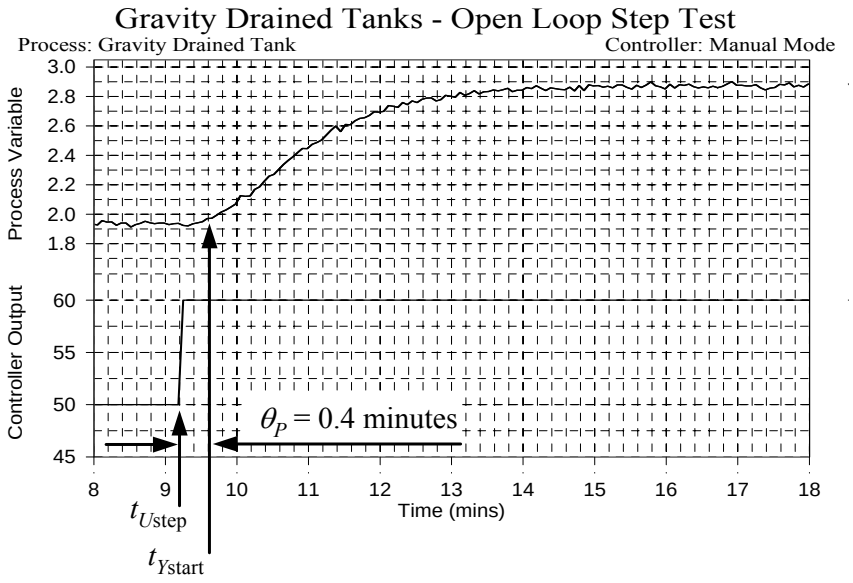


Figure 3.6 - θ_P computed from gravity drained tanks step test plot

1) The step change in the controller output occurs at time t_{Ustep} (9.2 min).

2) As in the time constant calculation, the time when the measured process variable starts showing a clear initial response to the step change in controller output is $t_{Y\text{start}}$ (9.6 min).

3) Thus, the apparent dead time for this process response is

$$\theta_P = t_{Ystart} - t_{Ustep} = 9.6 \text{ min} - 9.2 \text{ min} = 0.4 \text{ min}$$

Example: Heat Exchanger
Figure 3.7 shows the same step response data from the heat exchanger process used in the previous calculations.

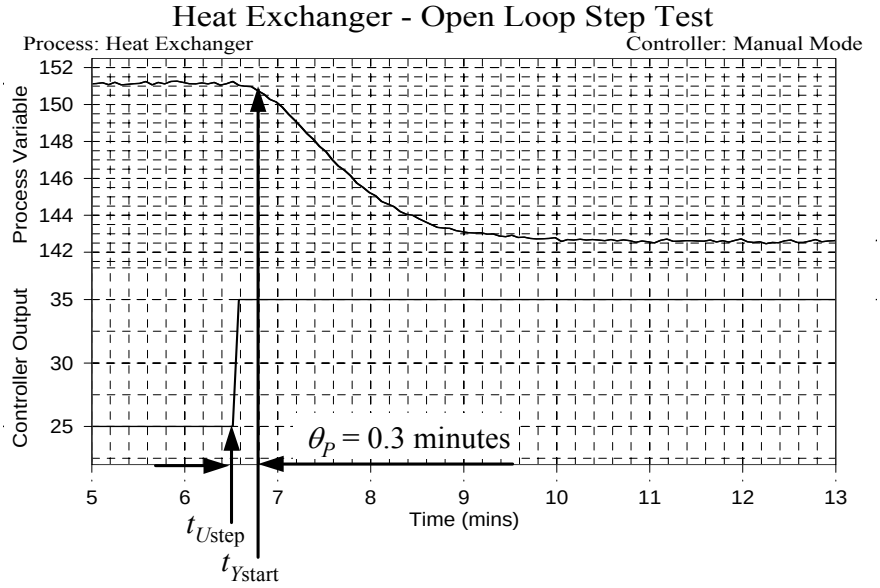


Figure 3.7 - θ_P computed from heat exchanger step test plot

Applying the dead time analysis procedure to the step test plot of Fig 3.7:

1) The step change in the controller output occurs at time t_{Ustep} (6.5 min).

2) As in the time constant calculation, the time when the measured process variable, y_{start}, starts showing a clear initial response to the step change in controller output is t_{Ystart} (6.8 min).

3) Thus, the apparent dead time for this process response is:

$$\theta_P = t_{Ystart} - t_{Ustep} = 6.8 \text{ min} - 6.5 \text{ min} = 0.3 \text{ min}$$

3.6 FOPDT Limitations - Nonlinear and Time Varying Behaviors

As detailed in this chapter, a critical step in controller design and tuning is the fitting of the linear FOPDT model of Eq. 3.1 to dynamic process data. It may come as a surprise to learn, then, that no real process has its dynamics exactly described by this model form. A linear FOPDT model is just too simple to describe the higher order, time varying and nonlinear behaviors of the real world.

Though only an approximation, and for some processes a very rough approximation, the value of the FOPDT model is that it captures those essential features of dynamic process behavior that are fundamental to control. When forced by a change in the controller output, a FOPDT model reasonably describes the direction, how far, how fast and with how much delay the measured process variable will respond.

The FOPDT model of Eq. 3.1 is called "first order" because it has only one time derivative. The dynamics of real processes are more accurately described by models that are second, third or higher order time derivatives. This notwithstanding, the simplifying assumption of using a first order plus dead time model to describe dynamic process behavior is usually reasonable and appropriate for controller tuning procedures. And often, a FOPDT model is sufficient for use as the model in model-based strategies such as of feed forward, Smith predictor and multivariable decoupling control.

Time Varying Behaviors

The dynamic response of the FOPDT model does not change with time, though the dynamics of real processes do. For a given K_P, τ_P, θ_P and steady state value of $u(t)$, the FOPDT model will compute the same steady state value of $y(t)$ tomorrow, next week and next year. If $u(t)$ is stepped from this steady state value, the FOPDT model will predict the same $y(t)$ response with equal consistency.

Real processes, on the other hand, have surfaces that foul or corrode, mechanical elements like seals or bearings that wear, feedstock quality or catalyst activity that drifts, environmental conditions such as heat and humidity that changes, and a host of other phenomena that impact dynamic behavior. The result is that real processes behave a little differently with every passing day.

The time varying (also called *nonstationary*) nature of real processes is important to recognize because, for a particular operating level, the K_P, τ_P, and θ_P that best describe the dynamics will change over time. As we will see later, this means that the performance of a well tuned controller will eventually degrade. It may take weeks or it may take months, but this result is difficult to avoid with conventional controllers.

From a practical perspective, even if loop performance is only fair, in many instances that is good enough. Hence, a slow degradation in performance is not always a pressing concern. For important loops where tight control has a significant impact on safety or profitability, periodic performance evaluation and retuning by a skilled practitioner is a justifiable expense.

Nonlinear Behaviors

Though a mathematician would classify time varying behaviors as nonlinear, here the term is reserved to describe processes that behave differently at different operating levels. Figure 3.8 shows process test data at two operating levels. First, a step test moves the process from steady state *A* to steady state *B*. Then, a second step moves the process from steady state *B* to steady state *C*.

If individual FOPDT models were fit to these two data sets and the process was linear, then the K_P, τ_P, and θ_P computed for both sets would be identical. As shown in Fig 3.8, like most real processes, the response is different at the different operating levels. If individual FOPDT models are fit to these two data sets, one or more of the model parameters will be different. Hence, *this process is considered to be nonlinear.*

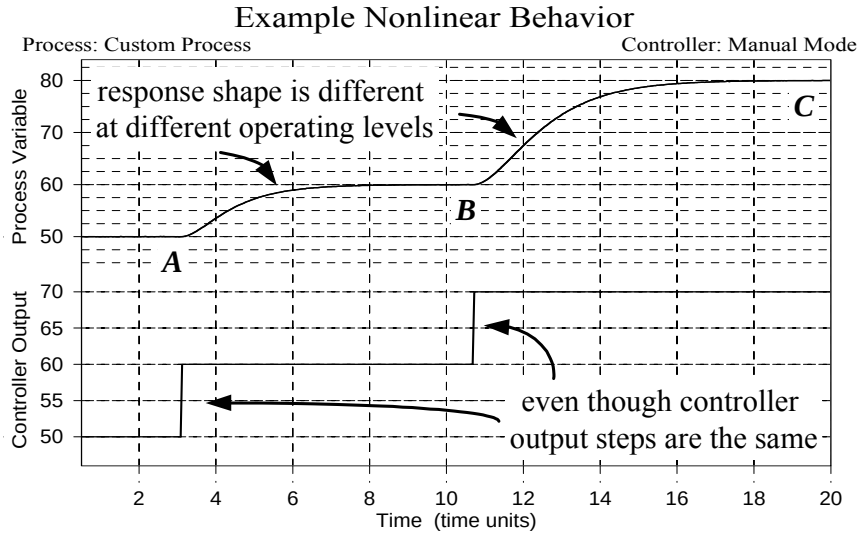


Figure 3.8 - A nonlinear process behaves differently at different operating levels

Figure 3.9 explores the nonlinear nature of the gravity drained tanks process. The bottom half of the plot shows the controller output stepping in five uniform Δu's from 40% up to 90%. The top half of the plot shows the actual response of the measured process variable, liquid level in the lower tank, along with the response of a linear FOPDT model.

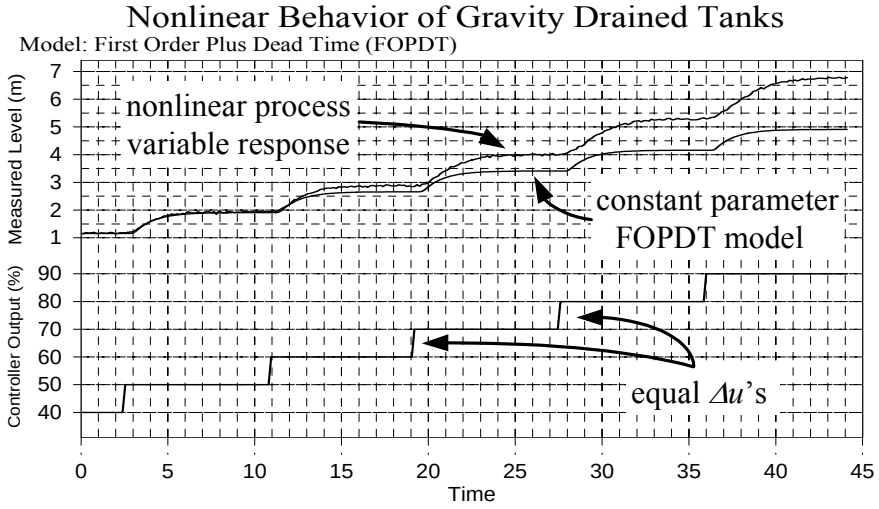


Figure 3.9 - Linear FOPDT model does not reasonably describe nonlinear behavior of gravity drained tanks process across a range of operating levels

Each of the controller output steps are the same, so the linear FOPDT model exhibits five identical responses across the entire range of operation. However, the model accurately describes the behavior of the process only as it responds to the first controller output step from 40% up to 50%. By the time the final steps are reached, the linear model and process behavior become quite different. Thus, the gravity drained tanks process is nonlinear, and this is evident because the response of the measured process variable changes as operating level changes.

The implication of nonlinear behavior is that a controller designed to give desirable performance at the lower operating levels may not give desirable performance at the upper operating levels. *When designing and tuning controllers for nonlinear processes, it is important that the test data be collected at the same level of operation where the process will operate once the control loop is closed (where the set point is expected to be set).*

3.7 Exercises

For all exercises, leave the controller in manual mode and use default values for noise level, disturbance value and all other parameters of the case study.

Q-3.1 Click the *Case Studies* button on the LOOP-PRO main screen and from the pop-up list of processes click on "Gravity Drained Tanks" to start the tanks simulation.

a) To generate dynamic process data, we need to change the controller output signal. This moves the valve position, causing a manipulation in the flow rate of liquid into the top tank. In this exercise we are interested in a step change in the controller output.

   At the upper right of the draining tanks graphic on your screen, locate the white number box below the Controller Output label. The most convenient way to step the controller output value is to click once on this white number box. Click once on the controller output box and it will turn blue. For the first step test, type 55 into the box and press Enter. This will cause the controller output to change from its current value of 70% down to the new value of 55%, decreasing the flow of liquid into the top tank and causing the liquid level to fall.

b) Watch as the process responds. When the measured process variable (liquid level) reaches its new steady state, click on the “pause” icon on the toolbar above the graphic to stop the simulation and then click on the "view and print plot" icon to create a fixed plot of the response. Use the plot options as needed to refine your plot so it is well suited for graphical calculations and then print it.

c) Using the methodology described in this chapter, use a graphical analysis to fit a first order plus dead time (FOPDT) dynamic model to the process response curve. That is, compute from the response plot the FOPDT model parameters: steady state process gain, K_P, overall time constant, τ_P, and apparent dead time, θ_P.

d) Repeat the above procedure for a second step in the controller output from 55% down to 40%.

e) The two steps in the controller output (70→55% and 55→40%) were both of the same size. Are the model parameters the same for these two steps? How/why are they different?

Q-3.2 Click the *Case Studies* button on the LOOP-PRO main screen and from the pop-up list of processes click on "Heat Exchanger" to start the simulation.

a) To generate dynamic process data, we will step the controller output signal. At the lower left of the exchanger graphic, locate the white number box below the Controller Output label. Click once on this box, type 49 and press Enter to change the controller output from its current value of 39% up to the new value of 49%. This increases the flow of cooling liquid into the shell side of the exchanger and causes the exit temperature of liquid on the tube side to fall.

b) Watch as the process responds. When the measured process variable (exit temperature) reaches its new steady state, click on the “pause” icon on the toolbar above the graphic to stop the simulation and then click on the "view and print plot" icon to create a fixed plot of the response. Use the plot options as needed to refine your plot so it is well suited for graphical calculations and then print it.

c) Using the methodology described in this chapter, use a graphical analysis to fit a first order plus dead time (FOPDT) dynamic model to the process response curve. That is, compute from the response plot the FOPDT model parameters: steady state process gain, K_P, overall time constant, τ_P, and apparent dead time, θ_P.

d) Repeat the above procedure for a second step in the controller output from 49% up to 59%.

e) The two steps in the controller output (39→49% and 49→59%) were both of the same size. Are the model parameters the same for these two steps? How/why are they different?

Q-3.3 Start the jacketed reactor simulation (<u>not</u> the cascade case) by clicking on the *Case Studies* button on the LOOP-PRO main screen and then clicking on "Jacketed Reactor."

a) To the right of the graphic, locate the white number box below the Controller Output label. Click once on this box change the controller output from 42% up to a new value of 52%. This increases the flow of cooling liquid through the cooling jacket side of the reactor and causes the measured exit temperature of liquid from the reactor to fall.

b) Watch as the process responds. When the measured process variable (exit temperature) reaches its new steady state, click on the “pause” icon on the toolbar above the graphic to stop the simulation and then click on the "view and print plot" icon to create a fixed plot of the response. Use the plot options as needed to refine your plot so it is well suited for graphical calculations and then print it.

c) Using the methodology described in this chapter, use a graphical analysis to fit a first order plus dead time (FOPDT) dynamic model to the process response curve. That is, compute from the response plot the FOPDT model parameters: steady state process gain, K_P, overall time constant, τ_P, and apparent dead time, θ_P.

d) Repeat the above procedure for a second step in the controller output from 52% up to 62%.

e) The two steps in the controller output (42→52% and 52→62%) were both of the same size. Are the model parameters the same for these two steps? How/why are they different?

4. Process Control Preliminaries

4.1 Redefining "Process" for Controller Design

As shown in Fig. 4.1, the feedback control loop consists of individual pieces: a controller, a final control element, a process, and a sensor/transmitter. The sensor measures the process variable and feeds back the signal to the controller. This measured value is subtracted from the set point to determine the controller error. The controller uses this error in an algorithm to compute an adjustment signal to the final control element. The final control element reacts to the controller output signal and changes the manipulated variable in an effort to make the measured process variable equal the set point.

The final control element, process, sensor/transmitter and even controller all have individual dynamic behaviors. That is, they all have their own gain, overall time constant and apparent dead time. Awareness of this fact is important when specifying instrumentation for a new process. Also, if undesirable dynamic behaviors of an existing control loop, such as a large dead time, can be traced to the final control element or measurement sensor, then the purchase of new equipment or relocation of existing equipment should be explored.

In general, a final control element and measurement sensor should be specified and installed to respond immediately (add little dead time) and complete the response quickly (have a small time constant). Note that qualifiers such as "little" and "quickly" are relative to the overall time constant of the entire loop. A dead time of 9 minutes is large relative to a time constant of 10 minutes and is small relative to a time constant of 1000 minutes.

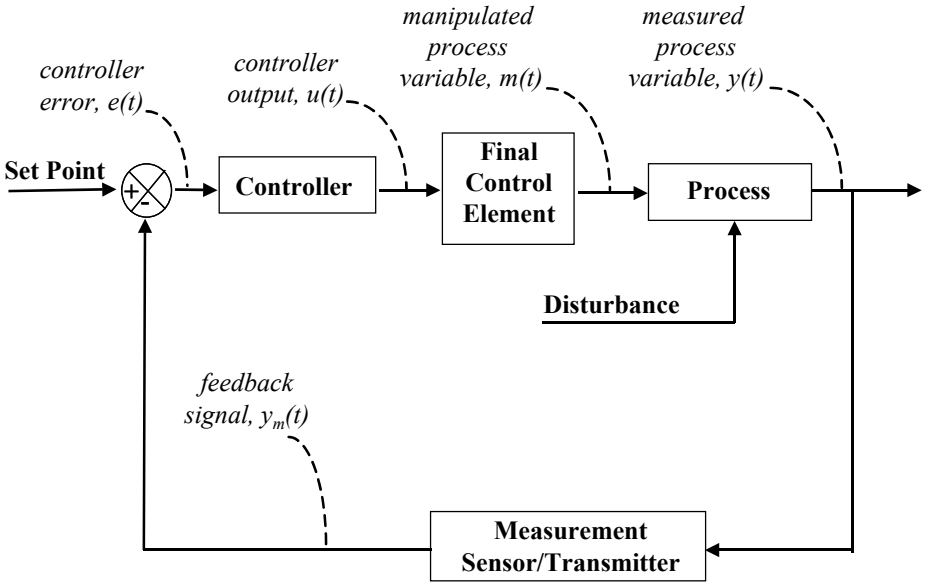


Figure 4.1 - The control loop from a designer's view point

While the final control element, process and sensor/transmitter all have individual dynamic behaviors, from a controller's viewpoint it is impossible to separate out these different behaviors. Consider that a controller sends a signal out on one wire and sees the result of this action as a change in the process variable when the measurement returns on another wire. From the controller's viewpoint, the individual gains, time constants and dead times all lump together into a single overall dynamic behavior.

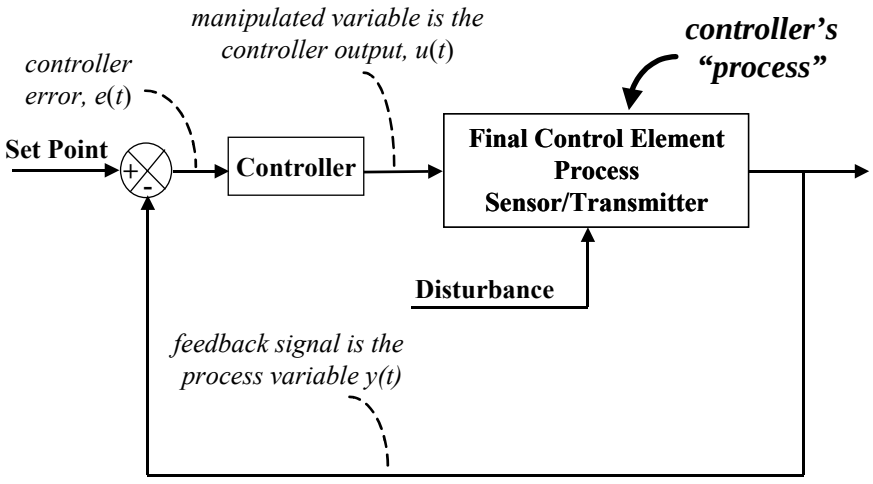


Figure 4.2 - The "process" of a control loop when tuning a controller

As we will learn, controller design and tuning proceeds in the same fashion regardless of whether this lumped behavior exhibits a large or small gain, time constant or dead time. The procedure is unaffected by whether a particular behavior is due to the process or sensor or final control element. No matter which piece contributes a dominant influence, the combined or overall loop behavior must be addressed.

Since controller design and tuning is the focus of the remainder of this manuscript, it is the controller's viewpoint that is taken. Consequently, for the remainder of this manuscript, "process dynamics" refers to the combined behaviors of the final control element, process and sensor/transmitter as illustrated in Fig. 4.2.

4.2 On/Off Control – The Simplest Control Algorithm

The simplest control law is *on/off* control. For on/off control, the final control element is either completely open/on/maximum or completely closed/off/minimum. There are no intermediate values or positions for the final control element in on/off control. One big disadvantage with pure on/off control is that mechanical final control elements can experience significant wear as they continually and rapidly switch from open to closed and back again.

To protect the final control element, a popular modification is the use of a *dead band,* which is a zone bounded by an upper and a lower set point, or dead band limit. As long as the measured variable remains between these limits, no changes in control action are made. Thus, if a valve is closed, it stays closed until the measured process variable passes below the lower dead band limit. The valve will then open completely and will remain open until the measured process variable eventually passes above the upper dead band limit.

On/off with dead band is found in many places in our daily lives. A home heating system uses this control law. So does an oven, a refrigerator and an air conditioner. All of these appliances cycle the temperature under their control between an upper and low limit around a set point specified by the home owner. Figure 4.3 illustrates the actions of a controller output and measured process variable in an on/off with dead band control system.

As shown in the figure, the controller output is either completely on or completely off. The change in the manipulated variable from one position to the other occurs whenever the measured process variable crosses one of the dead band limits. Note that by using a dead band, the wear and tear on the final control element is reduced but the size of the oscillations in the measured variable increases.

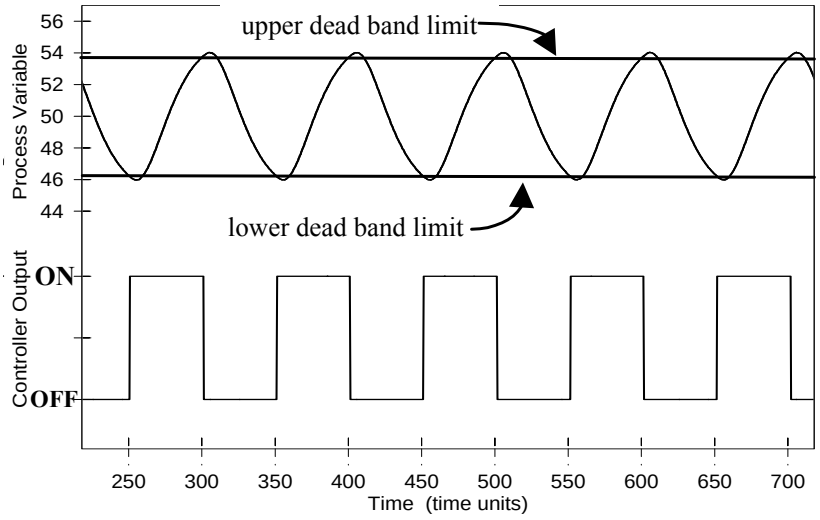


Figure 4.3 - Process under on/off control with dead band

Usefulness of On/Off Control

On/off controllers have the advantage of being easy to design. The biggest algorithm design issue is specification of the dead band limits. Unfortunately, for many applications, such a control law is just too limiting. Think about riding in a car that has on/off cruise control. This means the gas pedal can be either fully depressed to the car floor or completely disengaged with no intermediate pedal positions possible. It would be quite a ride indeed.

In general, on/off control is not sufficiently capable for most lab and production applications. The remainder of this book explores the design and tuning of intermediate value control laws that offer this necessary sophistication and consequently are widely used in the lab and plant.

4.3 Intermediate Value Control and the PID Algorithm

More powerful control algorithms that permit tighter control with less oscillation in the measured process variable are required for a variety of process applications. These algorithms compute a complete range of control actions between full on and full off. Not surprisingly, one requirement for implementation is a final control element that, when signaled by the controller, can assume intermediate positions between full on and full off. Example final control elements include process valves, variable speed pumps and compressors, and electronic heating elements.

The most popular intermediate algorithm, the proportional-integral-derivative controller, computes an intermediate value signal based on the current value of the control error (error equals set point minus measurement). The basic PID algorithm is expressed as follows:

$$u(t) = u_{\text{bias}} + \underbrace{K_C\, e(t)}_{\text{proportional}} + \underbrace{\frac{K_C}{\tau_I}\int e(t)dt}_{\text{integral}} + \underbrace{K_C \tau_D \frac{de(t)}{dt}}_{\text{derivative}} \tag{4.1}$$

where:

$u(t)$ = controller output signal
u_{bias} = controller bias or null value
$e(t)$ = controller error; $e(t) = y_{\text{setpoint}} - y(t)$
$y(t)$ = measured value of process variable

K_C = controller gain (proportional tuning parameter)
τ_I = controller reset time (integral tuning parameter)
τ_D = controller derivative time (derivative tuning parameter)

The PID algorithm continually computes control actions $u(t)$ in an attempt to drive error $e(t)$ to zero. As labeled in Eq. 4.1, each of the three terms works independently and with a slightly different agenda. The proportional term computes a contribution to the control action based on the current size of $e(t)$ at the moment of measurement. No influence of past measurements or future trends is considered in the proportional computation.

The integral term continually sums or accumulates $e(t)$ over time. The integral continues to grow as long as error is positive and begins shrinking when error becomes negative. Thus, the integral term increases its influence when either positive or negative error persists for some time.

The derivative term looks at the slope or rate of change in error. Thus, its influence grows when error is rapidly changing and seeks to slow down such movement. One result is that derivative action tends to dampen oscillations in the measured process variable.

The PID algorithm can be implemented in P-Only, PI, PD and full PID forms. The design and tuning of such controllers and the strengths and weaknesses of each P, I and D *controller mode* is the focus of the next several chapters.

5. P-Only Control - The Simplest PID Controller

5.1 The P-Only Controller

The PID (proportional-integral-derivative) controllers are by far the most widely used family of intermediate value controllers in the chemical process industry. The simplest in this family is called *proportional* or *P-Only* control.

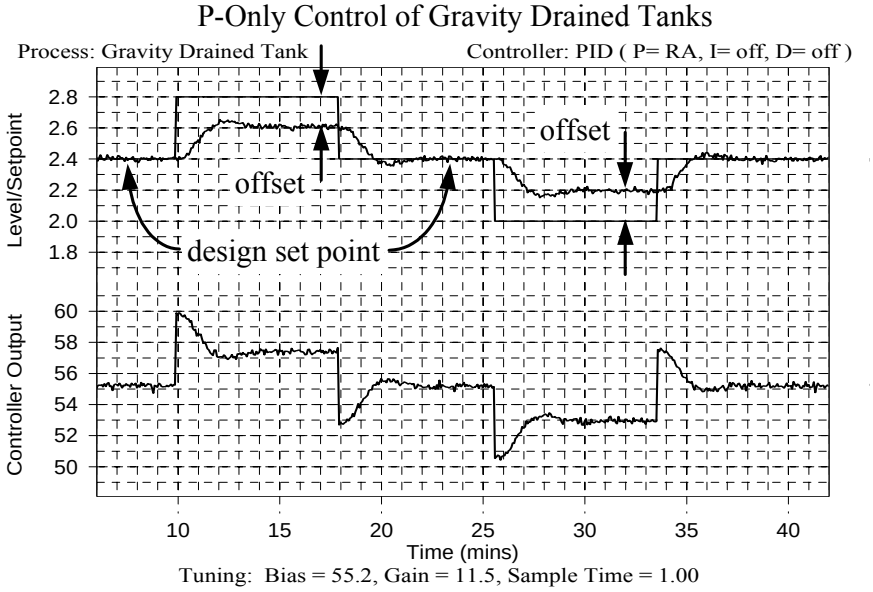


Figure 5.1 - Gravity drained tanks under P-Only control

Figure 5.1 shows the gravity drained tanks under P-Only control. The set point, the square wave in the upper half of the plot, starts at a level of 2.4 m and steps up to 2.8 m followed by a step down to 2.0 m. The measured level in the lower tank, the upper plot line distinguished by the noise or random error, generally moves along with the changing set point though it displays some oscillation with the effort.

The controller output signal is the line in the lower half of the plot. As the term "intermediate value control" implies, the controller output assumes many intermediate values between full open and full closed as the controller works to cause the liquid level in the lower tank to track the changing set point. The pumped disturbance flow (not shown) was constant at 2.0 L/min during the test.

This control system, typical of the family of PID controllers, repeats the measurement, computation and action procedure at every loop sample time:

- a sensor measures the liquid level in the lower tank,
- this measurement is subtracted from the set point level to determine a control error,
- the control algorithm uses this error to compute a new controller output signal, which when transmitted to the valve causes it to move to a new position,
- the change in valve position causes the flow rate of liquid into the top tank to change, which ultimately causes a change in level in the lower tank.

The goal is to eliminate (or at least minimize) the controller error by making the measured level in the lower tank equal the set point level.

One distinguishing feature of a P-Only controller, as shown in Fig. 5.1, is that it is only able to make the measurement equal the set point when the set point is at the design value of 2.4 m. When set point is not at the design value, *offset* occurs in most processes. This phenomenon is discussed later in this chapter.

The P-Only control algorithm computes a controller output signal every loop sample time as:

$$u(t) = u_{bias} + K_C e(t) \tag{5.1}$$

where $u(t)$ is the controller output signal, u_{bias} is the controller bias (also called the null value by some manufacturers) and K_C is the single tuning parameter called controller gain. Controller error, $e(t)$, is computed at every sample time as:

$$e(t) = y_{\text{setpoint}} - y(t) \tag{5.2}$$

where y_{setpoint} is the set point and $y(t)$ is the current value of the measured process variable.

Controller gain, K_C, in Eq. 5.1 is the second "gain" we have discussed. Controller gain should be distinguished from steady state process gain, K_P. Rearrangement of Eq. 5.1 reveals that controller gain, K_C, describes how the controller output signal changes given a change in the controller error. A larger K_C means the controller output will change more for a given error. Similar to K_P, controller gain also has a size, sign and units.

5.2 The Design Level of Operation

Nonlinear processes have a dynamic process behavior that changes as operating level changes. Because real processes are nonlinear, it is important to recognize that a controller should be designed for a specific level of operation. When designing a cruise control system for a car, would it make sense to perform dynamic modeling studies when the car is traveling twice the normal speed limit while going down hill in a hurricane? Of course not.

For a proper controller design, first determine where you expect the set point to be set during normal operation. Second, determine typical values for the important disturbance variables. This is important because as disturbance variables change, so can the values for the FOPDT model parameters K_P, τ_P and θ_P that best describe the dynamic behavior of the process.

With this information in hand, dynamic testing should occur as close as practical to the design value of the measured process variable when the disturbances are quiet and near their typical values. Thus, the design level of operation for a cruise control system is when the car is traveling near the normal speed limit on flat ground on a relatively calm day. The design level of operation for the P-Only control system shown in Fig 5.1 is a level in the lower tank near 2.4 m when the pumped disturbance flow is 2.0 L/min.

5.3 Understanding Controller Bias, u_{bias}

Suppose Eq. 5.1 is the algorithm used for cruise control in an automobile and $u(t)$ is the flow of gas to the engine computed by the controller. Further suppose that the velocity set point for the car is 70 kph and the current measured velocity is also 70 kph. Since $y(t)$ equals y_{setpoint}, then $e(t)$ equals zero and Eq. 5.1 becomes:

$$u(t) = u_{\text{bias}} \tag{5.3}$$

If u_{bias} is zero, then Eq. 5.3 says that when set point equals measurement, the flow of gas to the engine, $u(t)$, is also zero. This makes no sense. Clearly if the car is traveling 70 kph then some baseline flow of gas is going to the engine. This baseline value of the controller output is called the bias or null value. In this example, the bias is the flow of gas that, in open loop, causes the car to travel the design velocity of 70 kph when the disturbance variables are at their normal or expected values.

Consider a second example of the gravity drained tanks. Here suppose the measured level equals the set point value, so $e(t)$ in Eq. 5.1 equals zero. If no liquid is flowing into the top tank, the tanks will empty. Hence, to maintain the liquid level in the lower tank at set point, some baseline flow rate of liquid, or u_{bias}, must always be entering the top tank.

Similarly for the heat exchanger, if no cooling water is flowing through the shell side, then the hot liquid will exit the exchanger at the same temperature that it enters. To achieve any amount of cooling, some baseline volume of cooling water must always be flowing through the exchanger. This requirement of a controller output baseline value holds true for most all processes.

Thus, when a process is under P-Only control and the set point equals the measurement, some baseline or *bias* value of the controller output must exist or the measured process variable will drift from set point. This bias value of the controller output is determined from the design level of operation of the process to be controlled. Specifically, u_{bias} *is the value of the controller output that, in open loop (manual mode), causes the measured process variable to maintain steady state at the design level of operation when the process disturbances are at their design or expected values.*

A P-Only controller bias is assigned a value as part of the controller design and remains fixed once the controller is put in automatic. Some commercial manufacturers call the bias the null value.

5.4 Controller Gain, K_C, From Correlations

The P-Only controller has the advantage of having only one adjustable or *tuning parameter*, K_C, that describes how aggressive the controller output will move in response to changes in controller error, $e(t)$. For a given value of $e(t)$ in the P-Only algorithm of Eq. 5.1, if K_C is small, then the amount added to u_{bias} is small and the controller responds sluggishly. If K_C is large, then the amount added to u_{bias} is large and the controller responds aggressively. Thus, K_C can be adjusted or tuned for each process to make the controller more or less active in its actions when measurement does not equal set point.

Designing any controller from the family of PID algorithms entails the following steps:

- specifying the design level of operation,
- collecting dynamic process data as near as practical to this design level,
- fitting a FOPDT model to the process data to obtain model parameters K_P, τ_P and θ_P,
- using these model parameters in a correlation to obtain initial controller tuning values.

Final tuning may require some trial and error once the controller is online so loop performance will match that desired by the control designer.

The tuning correlations available in *Design Tools* for all controllers of the PID family are the Internal Model Control (IMC) relations. The exception is the P-Only controller, as no IMC correlation can be derived for this simple controller form.

As an alternative, *Design Tools* computes K_C using the integral of time-weighted absolute error (ITAE) tuning correlation for set point tracking (also called *servo* control) as:

$$K_C = \frac{0.20}{K_p}(\tau_p / \theta_p)^{1.22} \tag{5.4}$$

Although not automatically computed by *Design Tools*, the FOPDT model parameters can be used in the ITAE for disturbance rejection (also called *regulatory* control) correlation:

$$K_C = \frac{0.50}{K_p}(\tau_p / \theta_p)^{1.08} \tag{5.5}$$

These correlations provide an initial guess or starting point for final controller tuning. Even with this formal controller design procedure, final tuning requires online trial and error because:

- the control designer may desire performance different from that provided by the correlation,
- the FOPDT model used for tuning may not match the actual dynamic behavior of the plant,
- performance may need to be balanced over a range of operation for nonlinear processes,
- performance may need to be balanced for best set point tracking *and* disturbance rejection.

Ultimately, it is the designer who defines what "best" control performance is for an application.

When two correlations provide quite different initial controller gain estimates, the conservative approach is to start with the smallest K_C value. This will give the least aggressive (most sluggish) performance. If the resulting performance is too sluggish in rejecting disturbances and tracking changes in the set point, K_C should be increased. Conversely, if the process responds quickly and oscillates to an uncomfortable degree, K_C is too large and should be reduced.

5.5 Reverse Acting, Direct Acting and Control Action

Controller gain, K_C, computed from tuning correlations such as Eq. 5.4 and 5.5 will always have the same sign as the process gain, K_P. Time constant and dead time cannot affect the sign of the controller gain in these correlations because parameters of time are always positive. Thus, a process with a positive K_P will have a controller with a positive K_C.

For a process with a positive K_P, the process variable increases when the controller output increases. Thus, in closed loop, if the process variable is too high, the controller has to decrease the controller output signal to correct the error. The controller action is the reverse of the problem. Thus, when a process has a *positive* K_P and thus K_C, the controller must be *reverse acting*. Conversely, when K_P and thus K_C are *negative*, the controller must be *direct acting*, or:

$$K_P \text{ and } K_C \text{ positive} \quad \rightarrow \quad \text{reverse acting}$$

$$K_P \text{ and } K_C \text{ negative} \quad \rightarrow \quad \text{direct acting}$$

In most commercial controllers, a positive value of the controller gain is always entered. The sign (or *action*) of the controller is then assigned by specifying that the controller is either reverse or direct acting to indicate a positive or negative K_C respectively. If the wrong control action is entered, the controller will quickly drive the final control element to full on/open or full off/closed and remain there until a proper control action entry is made.

5.6 Set Point Tracking in Gravity Drained Tanks Using P-Only Control

P-Only controller design requires specifying a design level of operation, a controller bias and controller gain. Here we explain the design of the P-Only controller used in Fig. 5.1. The design level of operation for this example is a measured level in the lower tank of 2.4 m while the pumped flow disturbance during normal operation is expected to be about 2.0 L/min.

The controller bias is determined by searching for the value of the controller output that, in manual mode, causes the measured level to steady at 2.4 m when the disturbance is at its design value. Through trial and error, the bias value of the controller output, or u_{bias}, is determined to be 55.2% (please note that in typical industrial operations, such a three significant digit accuracy for the controller output far exceeds realistic expectations).

To verify that this bias is used, note that y_{setpoint} and $y(t)$ both equal 2.4 m (the design value) for the first few minutes in Fig 5.1. Since $e(t)$ equals zero for these minutes, then Eq. 5.1 says $u(t)$ should equal the u_{bias} value of 55.2%. The controller output trace in Fig. 5.1 reveals that this is true.

To compute K_C for the P-Only controller, we need a FOPDT model fit of dynamic process data collected around the design level of operation. We take advantage of the FOPDT fit detailed in Fig. 3.2, 3.4 and 3.6, which show a step test for the gravity drained tanks. In these figures, the process is initially at steady state with the controller output at 50% while the pumped flow disturbance is constant at 2.0 L/min. The controller output is stepped from 50% up to 60%, causing the measured tank level to rise from its initial steady state value of 1.93 m up to a new steady state level of 2.88 m.

Because the gravity drained tanks is nonlinear, a best design would fit dynamic process data collected from above *and* below the design level of operation so as to average the nonlinear effects. The plots contain dynamic data equally distributed above and below the design level of 2.4 m, making this FOPDT fit well-suited for our design. The analysis presented in Chapter 3 yields the model parameters:

$$\text{Process Gain, } K_P = 0.095 \text{ m/\%}$$

$$\text{Time Constant, } \tau_P = 1.6 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.40 \text{ min}$$

Using these parameters in the ITAE for set point tracking correlation of Eq. 5.4 produces an initial K_C estimate:

$$K_C = \frac{0.20}{0.095}\left(\frac{1.6}{0.40}\right)^{1.22} = 11.5 \text{ \%/m}$$

Because K_C is positive, the controller must be specified as reverse acting. As discussed above, the bias is 55.2%. Thus, the P-Only controller is as follows:

$$u(t) = 55.2\% + 11.5\, e(t) \tag{5.6}$$

Again please recognize that there are more significant digits used in Fig 5.6 than can be realistically obtained from typical plant data.

The performance of this controller in tracking set point changes is pictured in Fig. 5.1. The upper right corner of the plot displays information about the controller: PID (P= RA, I= off, D= off). Since the integral and derivative terms are off, this confirms a P-Only reverse acting controller. As shown in Fig. 5.1, the level in the lower tank does not track the first set point step all the way up to 2.8 m. This sustained error between the measurement and set point is called *offset* and is discussed in the next section. The measured level then responds well and exhibits no offset when the set point returns to the design level of operation of 2.4 m. With the final set point step down to 2.0 m, the controller again exhibits offset.

5.7 Offset - The Big Disadvantage of P-Only Control

The biggest advantage of P-Only control is that there is only one tuning parameter to adjust, so it is relatively easy to achieve a "best" final tuning. The disadvantage is that this control algorithm permits *offset.* Offset occurs in most processes under P-Only control when the set point and/or disturbances are at a value other than that used as the design level of operation (that used to determine u_{bias}).

Consider the P-Only controller of Eq. 5.6. If $y(t)$ is steady at $y_{setpoint}$, then $e(t)$ is steady at zero. If $e(t)$ is steady at zero, then controller output $u(t)$ is steady at the u_{bias} value of 55.2%. And if $u(t)$ is steady at 55.2% (and assuming the disturbance is constant at 2.0 L/min), then the lower tank will steady at the design level of 2.4 m. We know this because the u_{bias} value of 55.2% was determined by trial and error as the controller output in manual mode that causes the lower tank to steady at 2.4 m. Thus, under P-Only control, the measured process variable can be steady at the design set point and no sustained error or offset will be present.

How can Eq. 5.6 produce a value for $u(t)$ that is different from u_{bias} at steady state? The *only* way this can happen is if $e(t)$ is not steady at zero. That is, whenever the desired level of operation is other than the design value of 2.4 m, there must be a steady state error so $u(t)$ can assume a value other than the u_{bias} value of 55.2%. This sustained error is called offset.

As controller gain increases, the offset will decrease. Unfortunately, as controller gain increases, the oscillations in the measured process variable will increase and can even go unstable. The impact of K_C on offset and oscillatory behavior is illustrated in Fig. 5.2, which shows a step set point change for the gravity drained tanks under P-Only control for two different controller gains.

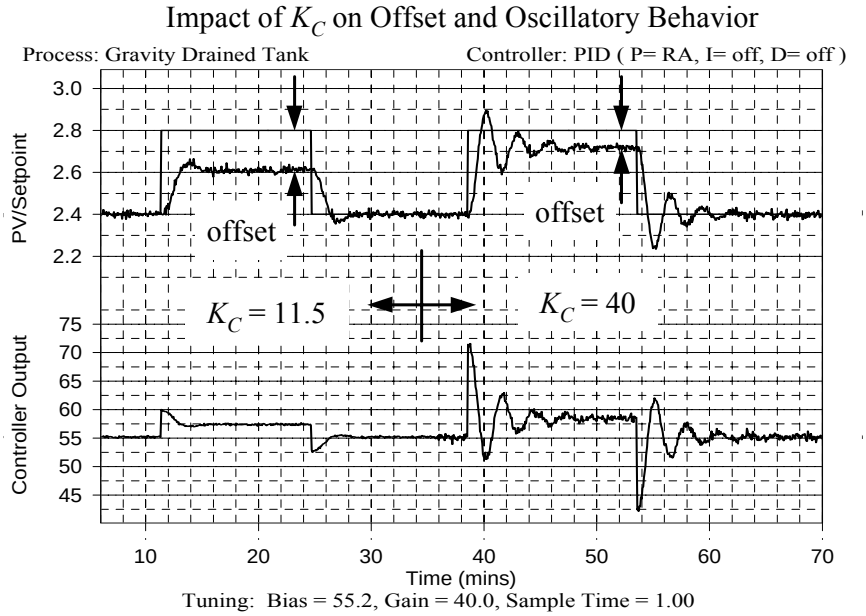


Figure 5.2 – P-Only offset decreases and oscillations increase as controller gain increases

5.8 Disturbance Rejection in Heat Exchanger Using P-Only Control

Whether the control objective is set point tracking or disturbance rejection, controller design always begins by determining the design level of operation. Dynamic process data is then collected as near as practical around this level. For this study, a constant measured exit temperature of 147°C is desired. The control objective is to reject disturbances that occur when the warm oil flow rate changes, causing a change the mixed stream temperature entering the tube side of the exchanger. The warm oil disturbance flow is normally about 10 L/min but spikes as high as 20 L/min on occasion.

The bias is the value of the controller output that, in open loop, causes the measured exit temperature to steady at 147°C when the disturbance flow is 10 L/min. Through trial and error, u_{bias} is found to be 29.2% (again, for industrial operations, this three significant digit accuracy far exceeds realistic expectations). As shown for the first few minutes in Fig 5.3, when $y(t)$ and $y_{setpoint}$ both equal 147°C and the disturbance equals 10 L/min, then $u(t)$ indeed equals the u_{bias} value of 29.2%.

Even though disturbance rejection is the goal, it is still controller output to measured process variable dynamics that must be fit with the FOPDT model. We take advantage of the FOPDT fit detailed in Fig. 3.3, 3.5 and 3.7 for the heat exchanger. In these figures, the process is initially at steady state with the controller output at 25% while the warm oil disturbance flow is constant at 10 L/min. The controller output is stepped from 25% up to 35%, causing the measured exit temperature to fall from its initial steady state value of 151.2°C down to a new steady state level of 142.6°C.

Because the heat exchanger is nonlinear, a best design would fit dynamic process data collected from above *and* below the design level of operation to average out the nonlinear effects. The plots contain dynamic data equally distributed above and below the design level of 147°C, making this FOPDT fit well-suited for our design. The analysis presented in Chapter 3 yields the model parameters:

$$\text{Process Gain, } K_P = -0.86 \text{ °C/\%}$$

$$\text{Time Constant, } \tau_P = 1.0 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.3 \text{ min}$$

The ITAE correlation of Eq. 5.5, intended for disturbance rejection applications, results in the controller gain:

$$K_C = \frac{0.50}{-0.86}\left(\frac{1.0}{0.3}\right)^{1.08} = -2.1 \text{ \%/°C}$$

As expected, K_C carries the same negative sign as K_P. Thus, we choose direct acting as the action of the P-Only controller. Using Eq. 5.1, the P-Only controller is shown below in Eq. 5.7:

$$u(t) = 29.2\% - 2.1\,e(t) \tag{5.7}$$

Figure 5.3 shows the performance of this controller in rejecting step changes in the warm oil disturbance flow rate. The set point is held constant throughout the experiment at the design operating level of 147°C. As expected, offset equals zero whenever the set point and disturbance are at their design values. When the disturbance flow rate steps from 10 L/min up to 20 L/min, however, offset results even though the set point remains at the design value.

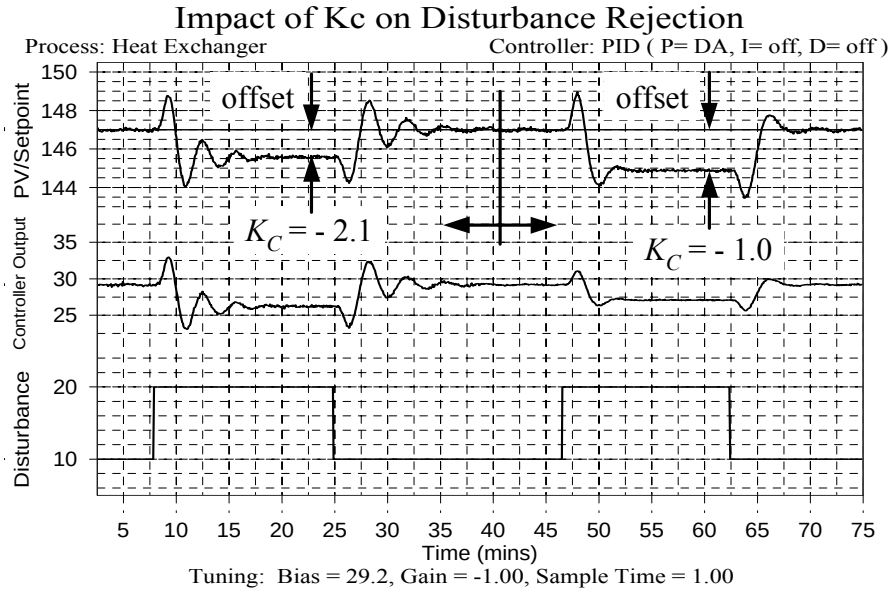


Figure 5.3 - Disturbance rejection performance of heat exchanger under P-Only control for two different values of controller gain while set point is constant

Half way through the experiment, the (absolute value of the) controller gain is decreased from the ITAE design value of −2.1 %/°C to −1.0 %/°C. The disturbance flow rate step is then repeated. Similar to the controller gain investigation shown in Fig. 5.2 for gravity drained tanks set point steps, the offset increases and the oscillatory nature of the response decreases as the (absolute value) controller gain is decreased.

5.9 Proportional Band

Some instrumentation manufacturers use different terminology for the controller gain tuning parameter. The popular alternative found in the marketplace is proportional band, *PB*. If the measured and manipulated variable of a process have units of percent and both can range from 0 to 100%, then the conversion between gain and proportional band is:

$$PB = \frac{100}{K_C} \tag{5.8}$$

While the examples in this book assign engineering units to the measured process variable, most commercial installations use units of % for both controller output and the process variable.

5.10 Bumpless Transfer to Automatic

Commercial controllers with a “bumpless transfer" feature achieve a smooth transition to closed loop by automatically setting values for the controller bias and set point when the controller is put in automatic. Specifically, when the control loop is closed:

1) the bias is set equal to the current value of the controller output, and
2) the set point is set equal to the current value of the measured process variable.

Hence, when the controller is put in automatic, there is no controller error and the bias is properly set to produce no offset. As a result, no immediate control action is necessary that would “bump” the measured process variable.

5.11 Exercises

Unless otherwise directed, use default values for noise level and all other simulation parameters.

Q-5.1 For the gravity drained tanks, the level in the lower tank is to remain constant at 3.0 m. Your process operator tells you that disturbances are a problem because the pumped flow disturbance, normally constant at 2.0 L/min, occasionally spikes to 5.0 L/min. Your objective is to explore a P-Only controller designed to reject these disturbances. As a first step, record the design set point for your P-Only controller.

a) Adjust the controller output signal to move the process to the design level of operation (a measured process variable of 3.0 m when the major disturbance is 2.0 L/min). Record the bias value for your P-Only controller.

b) Perform an appropriate open loop step test and use the dynamic response data to estimate a first order plus dead time (FOPDT) model for this process. (Hint: even though disturbance rejection is the goal, it is still the controller output that is stepped. Because the process is nonlinear, it is good practice to include data from above *and* below the design value of the measured process variable).

c) Use these FOPDT model values in the ITAE for disturbance rejection correlation to compute a P-Only controller gain, K_C. Be sure to specify the sign, magnitude and units of K_C.

d) Using your design set point, bias and controller gain, implement a P-Only controller. Generate plots showing the performance of the controller in rejecting step increases in the pumped disturbance flow rate from 2.0 L/min up to 5.0 L/min and back again, allowing the response to settle after each step.

e) Double your K_C computed in part *d*, step increases in the pumped disturbance flow rate from 2.0 L/min up to 5.0 L/min and back again, and generate a new set of plots. Halve your K_C from part *d* and repeat the experiment again. Using these plots, discuss how the magnitude of K_C impacts offset and the oscillatory nature of the response.

Q-5.2 Design a P-Only controller for the heat exchanger where the design level of operation is a measured exit stream temperature of 133°C when the warm oil disturbance flow is at its design value of 10 L/min. Several times a day, the exchanger must respond to requests to step the set point from 133°C up to 138°C. Begin by recording the design set point.

a) Adjust the controller output signal to move the process to the design level of operation (a measured process variable of 133°C when the major disturbance is 10 L/min). Record the bias value for your P-Only controller.

b) Perform an appropriate open loop step test and use the dynamic response data to estimate a first order plus dead time (FOPDT) model for this process. (Hint: the process is nonlinear so include data from above *and* below the design value of the measured process variable).

c) Use these FOPDT model values in the ITAE set point tracking correlation to compute a P-Only controller gain, K_C. Be sure to specify the sign, magnitude and units of K_C.

d) Using your design set point, bias and controller gain, implement a P-Only controller. Generate a plot showing the performance of the controller in tracking steps in the set point from 133°C up to 138°C and back again, allowing the response to settle after each step.

e) Fine tune your controller gain by trial and error and search for the "best" K_C that balances offset with oscillatory behavior when tracking this set point step. Remember that as the designer, you define what constitutes "best" performance. As well as generating a plot of your final tuning, be sure to explain what criteria you used to define "best."

f) Using your "best" K_C, generate a plot showing a step change in set point from 133 °C up to 138°C and back again, allowing the response to settle after each step. Follow that with a step decrease from 133°C down to 128°C and back again. Using the plot to support your argument, discuss how the nonlinear nature of this process impacts controller performance for set point tracking.

Q-5.3 For the jacketed reactor (not the cascade case), the design level of operation is a measured reactor exit temperature of 92°C when the cooling jacket inlet temperature (a disturbance) is at its design value of 50 L/min. During product change over, which occurs once a shift, the reactor temperature must be changed from 92°C up to 95°C. Your assignment is to design a P-Only controller to track this set point change.

a) Determine and record the design set point and controller bias value for your controller.

b) Perform an appropriate open loop step test and use the dynamic response data to estimate a first order plus dead time (FOPDT) model for this process. Be sure the response plot includes data from above and below the design level of operation.

c) Use these FOPDT model values in the ITAE set point tracking correlation to compute a P-Only controller gain, K_C. Be sure to specify the sign, magnitude and units of K_C.

d) Using your design set point, bias and controller gain, implement a P-Only controller. Generate a plot showing the performance of the controller in tracking steps in the set point from 92°C up to 95°C and back again, allowing the response to settle after each step.

e) Fine tune your controller gain by trial and error and search for the "best" K_C that balances offset with oscillatory behavior when tracking this set point step. Remember that as the designer, you define what constitutes "best" performance. As well as generating a plot of your final tuning, be sure to explain what criteria you used to define "best."

6. Automated Controller Design Using *Design Tools*

6.1 Defining Good Process Test Data

Whether you are fitting a model to process test data for controller tuning, for process simulation using *Custom Process*, or for developing models for advanced strategies such as feed forward or the Smith predictor, the answer to all five of these questions about your data should be "yes," and ultimately, it is your responsibility to consider all of these five steps to ensure success.

1. Was the process at steady state before data collection started?
Suppose a controller output change forces a dynamic response in a process, but the data file only shows the tail end of the response without showing the actual controller output change that caused the dynamics in the first place. *Design Tools* will indeed fit a model to this data, but it will skew the fit in an attempt to account for an unseen "invisible force." This model will not be descriptive of your actual process and hence of little value for control. To avoid this problem, it is important that data collection begin only after the process has settled out. *Design Tools* can then properly account for all process variations when fitting the model.

2. Is the first point in the test data equal to the steady state value of step 1?
Asking *Design Tools* to assume important facts about your data can lead to unfortunate results if the assumptions are wrong. Knowledge of the initial steady state of the process prior to data collection is fundamental to computing process and controller gain for any software tool. Thus, you must assume responsibility for verifying the initial steady state. *Design Tools* helps you by using the first data point in your data file as representative of the initial state of the process. But if noise or random error has caused this first point to shift from a reasonable initial value, edit the data using a *Design Tools* utility and adjust this data point by hand based on your knowledge of the process.

3. Did the test dynamics clearly dominate the process noise?
When generating dynamic process data, it is important that the change in controller output cause a response in the process that clearly dominates the measurement noise. We suggest defining a noise band as ±3 standard deviations of the random error around the process variable during steady operation. Then, when during data collection, the change in controller output should force the process variable to move at least ten times this noise band (the signal to noise ratio should be greater than ten). If you meet or exceed this requirement, your data will be rich in the dynamic information needed for controller design.

4. Were the disturbances quiet during the dynamic test?
It is essential that the test data contain process variable dynamics that have been clearly (and in the ideal world exclusively) forced by changes in the controller output as discussed in step 3. Dynamics caused by unmeasured disturbances can seriously degrade the accuracy of an analysis because *Design Tools* will model those behaviors as if they were the result of changes in the controller output signal. In fact, a model fit can look perfect, yet a disturbance that occurred during data collection can cause the model fit to be nonsense. If you suspect that a disturbance event has corrupted test data, it is conservative to rerun the test.

5. Did the model fit appear to visually approximate the data plot?
This is perhaps the easiest of the steps because *Design Tools* will display a plot that shows the model fit on top of the data. If the two lines don't look similar, then the model fit is suspect. Of course, as discussed in step 4, if the data has been corrupted by unmeasured disturbances, the model fit can look great yet the usefulness of the analysis can be compromised.

6.2 Limitations of the Step Test

Chapter 3 explores generating dynamic process data using a step test, where the controller output is stepped from one constant value to another, causing the measured process variable to move from one steady state to a new steady state. As shown in that chapter, step tests are useful because FOPDT (first order plus dead time) models can be fit to the response plot by direct graphical analysis.

Unfortunately, the step test is simply too limiting to be useful in many practical applications. The drawback is that it takes the process away from the desired operating level for a relatively long period of time. Recall that when collecting data for a nonlinear process using a step test, best practice is to move the measured process variable to one side of the design level of operation and then step the controller output so that response data is centered on average around the design level. As a consequence, step tests in the plant can result in significant off-spec product that may need reprocessing or even disposal.

A second problem common to all open loop tests, including those discussed in the next section, is that in production situations, operating personnel may simply be unwilling to open a loop (put the controller in manual) "just" to generate dynamic process data. In this surprisingly common situation, closed loop testing must be performed as discussed toward the end of this chapter.

6.3 Pulse, Doublet and PRBS Test

Popular open loop experiments used to generate dynamic process data beyond the step test include the pulse, doublet and pseudo-random binary sequence (PRBS) tests. Unlike the step test, model parameters cannot be fit by a simple graphical analysis of the process response plots from these tests. In fact, a computer program such as LOOP-PRO's *Design Tools* is required for data analysis.

Pulse Testing

A pulse test, shown in Fig. 6.1, can be thought of as two step tests performed in rapid succession. The controller output is stepped up, and as soon as the measured process variable shows a clear response, the controller output is returned to its original value. Ordinarily, the process does not reach steady state before the return step is made.

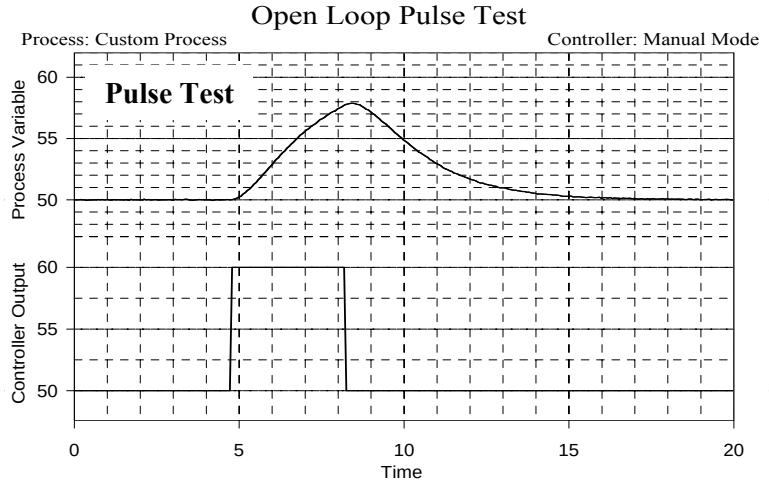


Figure 6.1 - Controller output pulse with measured process variable response

Pulse tests have the desirable feature of starting from and returning to an initial steady state. Unfortunately, they only generate data on one side of this steady state (which presumably is the design level of operation) and this is not best practice when nonlinear processes are being studied. The doublet test solves this problem.

Doublet Testing

A doublet test, shown in Fig. 6.2, is two pulse tests performed in rapid succession and in opposite direction. The second pulse is implemented as soon as the process has shown a clear response to the first pulse. The process does not ordinarily respond to steady state for either pulse.

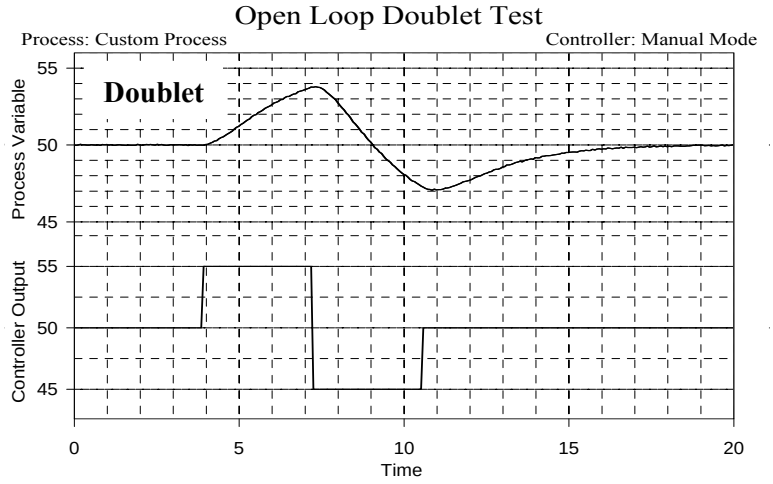


Figure 6.2 - Controller output doublet with measured process variable response (popular with practitioners)

The doublet test offers several benefits, including:

- starting from and returning reasonably quickly to the design level of operation,
- producing data both above and below the design level, and
- having a relatively small maximum deviation in the measured process variable from the initial steady state, thus minimizing off-spec production.

For these reasons, *many industrial practitioners find the doublet to be the preferred method of generating open loop dynamic process data.*

PRBS Testing

A pseudo-random binary sequence (PRBS) test, shown in Fig. 6.3 is characterized by a sequence of controller output pulses that are uniform in amplitude, alternating in direction, and of random duration. The "pseudo" is in the name because true random behavior is a theoretical concept that is unattainable by computer algorithm. Thus, in practice, approximate or pseudo-randomness using a random number generator must suffice for determining the duration of each pulse.

The PRBS test permits generation of useful dynamic process data while causing the smallest maximum deviation in the measured process variable from the initial steady state. Since this implies a minimum of off-spec production, PRBS, in theory anyway, is the most desirable of the open loop methods. Unfortunately, proper PRBS experiment design presents practical difficulties that often counterbalance this benefit.

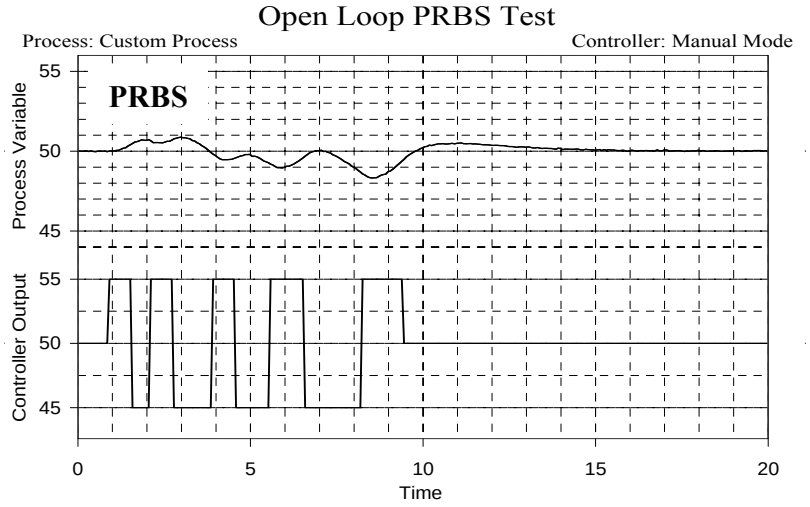


Figure 6.3 - Controller output pseudo-random binary sequence with process variable response

PRBS experiment design requires specification of several characteristics of the controller output signal trace, including the controller output:

- initial value,
- pulse amplitude,
- average pulse duration, and
- standard deviation of the random change in pulse duration around this average.

The length of the experiment itself must also be specified. Since an optimum experiment design requires knowledge of the process gain, time constant and dead time, the very values you are performing the experiment to determine, the design logic becomes somewhat circular. If you ultimately perform the experiment a number of times in a search of a "best" test, you may well have been better off simply performing the quick and practical doublet test

6.4 Noise Band and Signal to Noise Ratio

When generating dynamic process data, it is important that the change in the controller output signal causes a response in the measured process variable that clearly dominates the measurement noise. One way to quantify the amount of noise in the measured process variable is with a *noise band*.

As illustrated in Fig. 6.4, a noise band is based on the standard deviation of the random error in the measurement signal when the controller output is constant and the process is at steady state. Here we define noise band as ±3 standard deviations of the measurement noise around the steady state of the measured process variable (99.7% of the signal trace is contained within the noise band). While other definitions of the noise band have been proposed, this definition is conservative when used for controller design.

Employing this definition for generating dynamic process data, the change in controller output should cause the measured process variable to move at least ten times the size of the noise band. Expressed concisely, the *signal to noise ratio should be greater than ten*. In Fig. 6.4, the noise band is 0.25°C. Hence, the controller output should be moved far and fast enough during a test to cause the measured exit temperature to move at least 2.5°C. This is a minimum specification. In practice it is conservative to exceed this value.

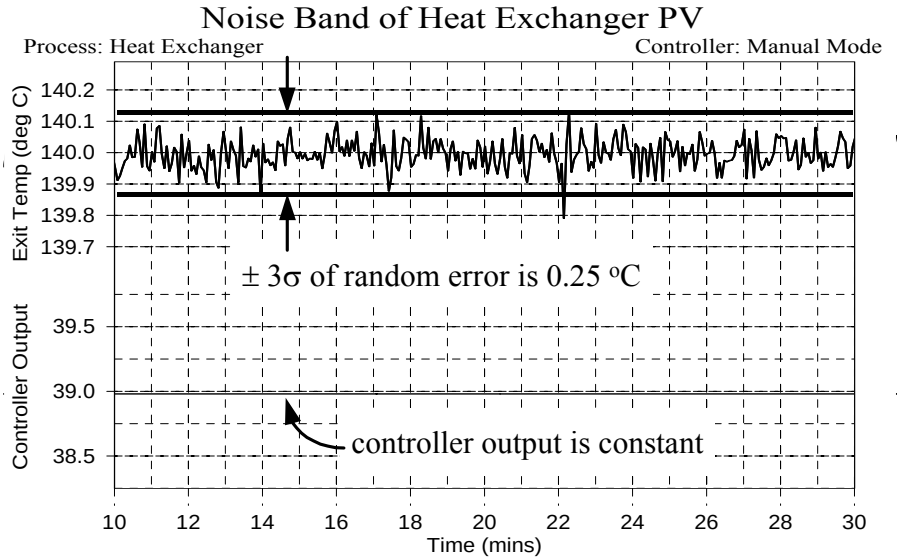


Figure 6.4 - Noise band encompasses ± 3 standard deviations of the measurement noise

6.5 Automated Controller Design Using *Design Tools*

Design Tools offers two tools for controller design and analysis. One is the dynamic model fitting tool that automatically fits dynamic models to process data. Models can be fit to data generated by LOOP-PRO, other software packages, and perhaps most important, to data generated by real processes in the lab or plant.

Figure 6.5 shows a portion of a data file in proper format for use in *Design Tools*. The file must contain at least three columns, one for time stamp data, one for manipulated variable data (which is the controller output for PID algorithm design), and one for the measured process variable data. The data must be ASCII text with entries separated by tabs, commas or spaces. Every space in a column must have an entry; there can be no blank spaces.

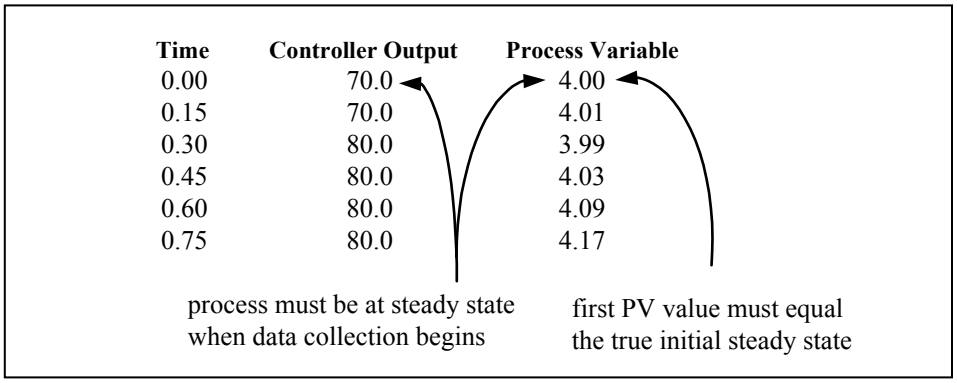


Figure 6.5 – Example process data file used by Design Tools

The linear dynamic models available in the *Design Tools* library include:

First Order Plus Dead Time (FOPDT)
First Order Plus Dead Time Integrating
Second Order Plus Dead Time (SOPDT) Overdamped
Second Order Plus Dead Time Overdamped with Lead Time
Second Order Plus Dead Time (SOPDT) Underdamped

The fit routine systematically searches for the model parameters that minimize the sum of squared errors (SSE) between the response contained in the measured data and the response predicted by the model being fit when it is forced by the manipulated variable data in the file. With i indicating any one of the N total data points in the set, the SSE is expressed:

$$\text{SSE} = \sum_{i=1}^{N} [\text{Measured Data}_i - \text{Model Data}_i]^2 \tag{6.1}$$

In general, the smaller the SSE, the better the model describes the data. To obtain a meaningful fit, it is essential to recognize that:

- the process must be at steady state before collection of dynamic data begins,
- the first data point in the file must equal this initial steady state value.

If these conditions are not met, the model fit will be incorrect and of little use.

When your goal is controller design and tuning, a FOPDT model should be fit so the popular controller tuning correlations and design rules-of-thumb can be exploited. The other dynamic models in the library are useful when constructing advanced controller architectures such as feed forward, Smith predictor and model predictive control (MPC) algorithms. They are also useful when accurate simulation of dynamic process behavior using *Custom Process* is the goal.

The second feature of *Design Tools* is the controller tuning tool. Using the results of a successful FOPDT model fit, tuning values for P-Only, PI and PID controllers are computed. The library of popular tuning correlations is based on the internal model control (IMC) relations. These are an extension of the popular *lambda* tuning correlations and include the added sophistication of directly accounting for dead time in the tuning computations.

Example: Heat Exchanger Doublet Test

Section 5.8 presented a P-Only controller design for the heat exchanger using a graphical fit of step test data. The design level of operation for that study was a measured exit temperature of 147°C. The control objective focused on disturbance rejection because the warm oil disturbance flow rate, normally about 10 L/min, was said to occasionally spike as high as 20 L/min. We learned in that study that in open loop, a controller output of 29.2% causes the measured exit temperature to steady at 147°C when the disturbance is at its design value of 10 L/min. In fact, this is how we determined the controller bias in that example.

In this example we fit a FOPDT model and tune a P-Only controller using a doublet test and *Design Tools*, and compare the results with the graphical step test analysis of Section 5.8. As shown in Fig.6.6, we start with the heat exchanger at steady state at the design level of operation.

While saving data to file, we pulse the controller output first up 3% and then down 3% from the initial value of 29.2% (pulses of this size move the measured process variable across a range of nonlinear operation similar to that of the previous step test). After each pulse, we wait for a clear response in the measured process variable but do not wait for steady state.

The data file is then read into *Design Tools* and the columns containing time, manipulated variable (controller output) and process variable data are correctly labeled. The FOPDT model is selected from the library and the fit routine is executed. The model fit is also shown in Fig.6.6.

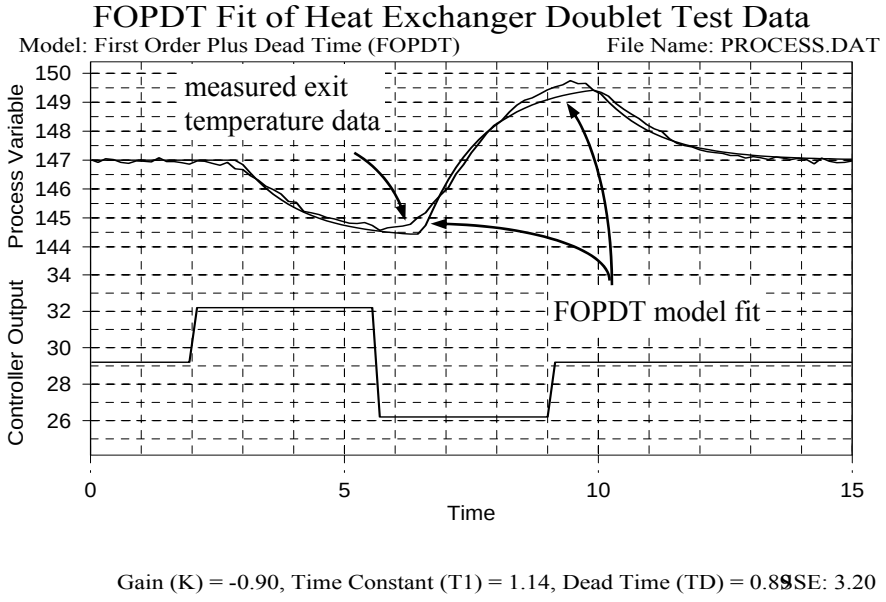


Figure 6.6 - Design Tools fit of heat exchanger doublet test data using a FOPDT model

The model fit may appear to be in error because its trace lies below the data during much of the dynamic portion of the experiment. In reality, however, the model slightly *over*shoots the data on the downward trace and then *under*shoots the data on the upward trace. Thus, the *Design Tools* fit of the linear FOPDT model effectively averages the nonlinear behavior of the heat exchanger.

	← Open Loop Data →		Closed Loop Data
	Graphical Analysis Of Step Test	***Design Tools* Doublet Fit**	***Design Tools* Fit of Set Point Doublet**
Process Gain, K_P (°C/%)	–0.86	–0.90	–0.86
Time Constant, τ_P (min)	1.0	1.1	1.2
Dead Time, θ_P (min)	0.3	0.9	1.0
Sum of Squared Errors (SSE)	44.1	3.2	5.4
ITAE Controller Gain, K_C (%/°C)	–2.1	–0.7	–0.7

Table 6.7 – Comparing FOPDT models for heat exchanger

In Table 6.7, the two columns under the "Open Loop Data" label show the model fit from this doublet test along side the FOPDT fit results from the step test graphical analysis. As shown, the step test analysis yields (what we will learn is) essentially the same values for K_P and τ_P when compared to the *Design Tools* fit of open loop doublet test data. The θ_P for the two methods, on the other hand, are quite different.

So which fit is "better?" And how do we determine "better?" One time-tested method of comparing a model to data is by visual inspection. Figure 6.8 shows the same heat exchanger doublet test data that was used in Fig. 6.6. Also shown is a FOPDT model trace generated using the K_P, τ_P and θ_P from the step test graphical analysis.

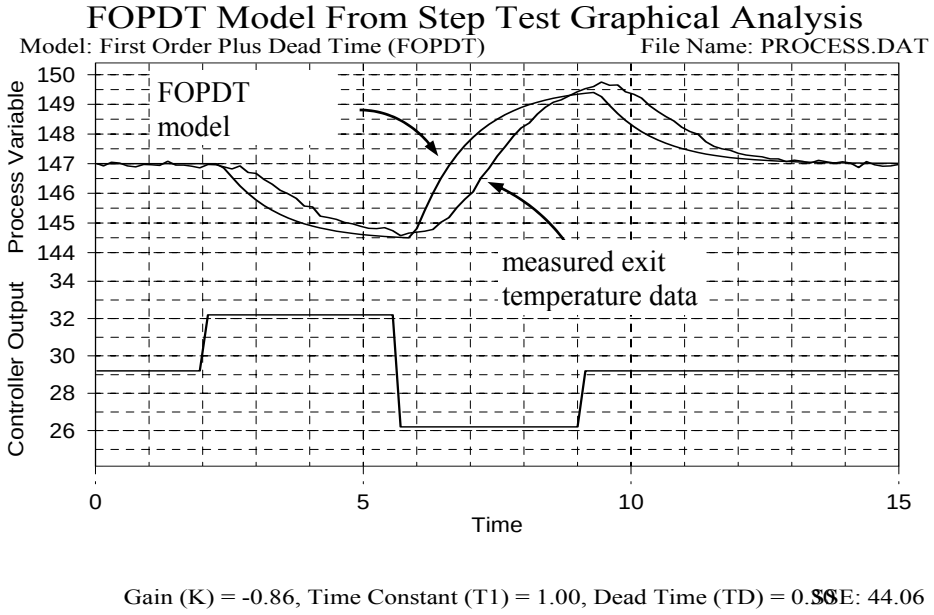


Figure 6.8 - Comparing FOPDT model from step test analysis to doublet test data

It is clear from visual inspection that the *Design Tools* model of Fig. 6.6 more accurately describes the data then does the step test model shown in Fig. 6.8. An alternative to visual inspection is to compare the SSE as defined in Eq. 6.1 for the two fits. As listed in Table 6.7, the *Design Tools* doublet fit has an SSE of 3.2, while the step test graphical analysis model has an SSE of 44.1. The dramatically lower SSE for the *Design Tools* fit confirms our visual conclusion.

Note that the poor model from the step test was a result of the graphical analysis methodology, which makes many simplifying assumptions. The step test itself is capable of producing useful dynamic data, and in general, *Design Tools* can accurately model data regardless of whether a step, pulse, doublet or PRBS is used as the testing method (though to be useful in controller design, the data must have been properly generated).

6.6 Controller Design Using Closed Loop Data

It is increasingly common to have operations personnel in a production facility reject the idea of opening an existing loop so controller design data can be collected. In these situations, you must be prepared to perform dynamic studies when the controller is in automatic. In theory, closed loop testing can be problematic because the information contained in the data will reflect the character of the controller as well as that of the process. In practice, however, this theoretical concern rarely causes real world problems.

For closed loop studies, dynamic data is generated by stepping, pulsing or otherwise perturbing the set point. To generate proper data, the controller must be tuned aggressively enough so that, similar to open loop testing, the changing controller output forces the measured process variable to move more than ten times the noise band. Also, the data set must begin at steady state.

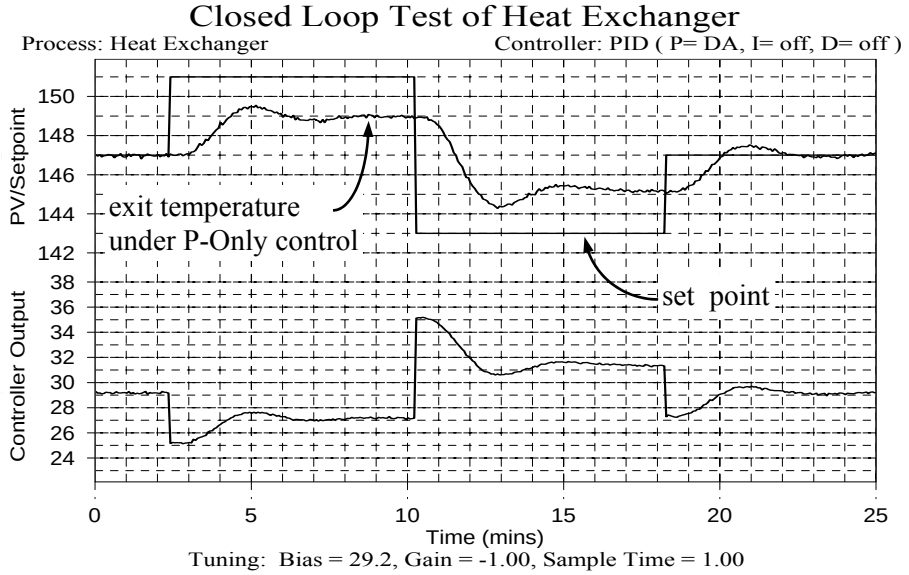


Figure 6.9 - Closed loop test of heat exchanger using a set point doublet

Figure 6.9 shows a closed loop test of the heat exchanger. The P-Only controller gain is $K_C = -1$ for this test (a wide range of controller gains will produce similar FOPDT models). The set point doublet used in the experiment generates a measured process variable response that covers a range of nonlinear operation similar to that evident in the previous open loop step and doublet tests.

As required for a *Design Tools* analysis, the process is at steady state before the first set point step is made. While data is being saved to file, the set point is stepped up and then down 4 ^{o}C from the initial design level of operation. After the experiment, the file is read into *Design Tools*, the data columns are labeled, and a FOPDT model is fit to the data.

Figure 6.10 shows the FOPDT model fit of the heat exchanger data generated in the closed loop test of Fig. 6.9. A visual inspection of Fig. 6.10 and a comparison of SSE's and model parameters as summarized in Table 6.7 establish that it is certainly possible to obtain an accurate dynamic model from closed lop data.

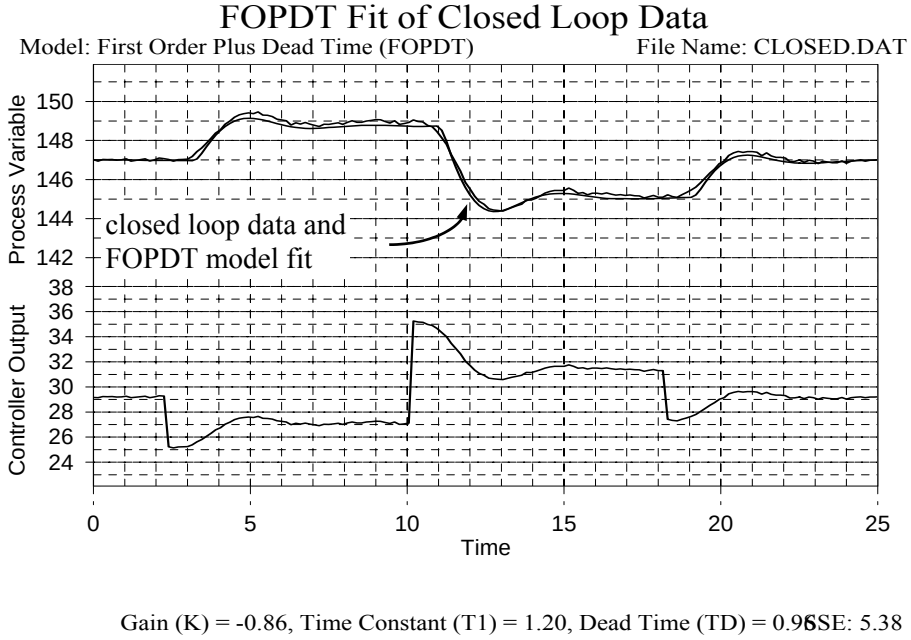


Figure 6.10 - Design Tools fit of closed loop heat exchanger test data

6.7 Do Not Model Disturbance Driven Data!

For a controller to take appropriate action in response to controller error, the algorithm must specifically associate how the controller output affects the measured process variable. The way this association is best developed is through a FOPDT dynamic model. The appropriateness of initial tuning from tuning correlations is directly related to the accuracy of the model employed.

For this reason, it is essential that the process data used for modeling studies contains measured process variable dynamics that have been clearly (and in the best of all worlds exclusively) forced by changes in the controller output. Extraneous events such as dynamics caused by disturbance variables will degrade the accuracy of the FOPDT model, and thus the initial tuning of the controller.

To illustrate, consider the extreme case shown in Fig. 6.11. Similar to Fig. 6.9, the heat exchanger is under P-Only control with $K_C = -1$. Here, however, the set point remains constant and the dynamic event is forced by changes in the warm oil disturbance flow rate. With data being saved to file, the disturbance flow, initially at 10 L/min, is stepped up to 12.5 L/min, down to 7.5 L/min, and back to the initial value of 10 L/min. The experiment data is read into *Design Tools* and fit with a FOPDT model. The resulting model and experiment data are shown in Fig. 6.12. Visual inspection of Fig. 6.12 reveals that the model accurately represents the data.

Table 6.13 compares the results of this disturbance driven FOPDT model with a proper model, assigned here as the *Design Tools* fit of open loop doublet data listed in Table 6.7. The fit of disturbance driven data is quite disturbing. Not only are all of the FOPDT model parameters quite wrong, but the steady state process gain even has the wrong sign!

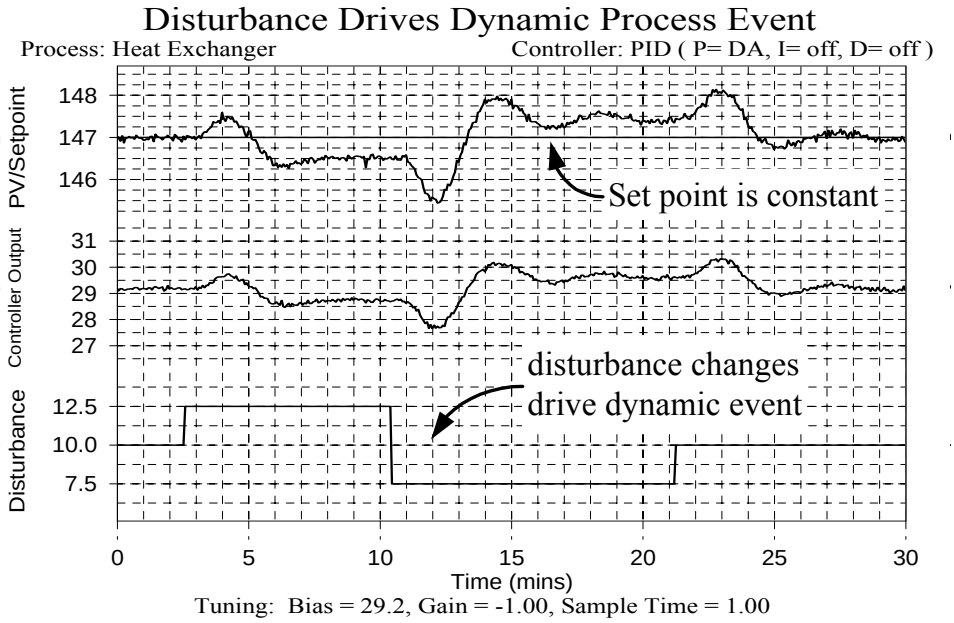


Figure 6.11 – Dynamic process data forced by changes in the disturbance variable

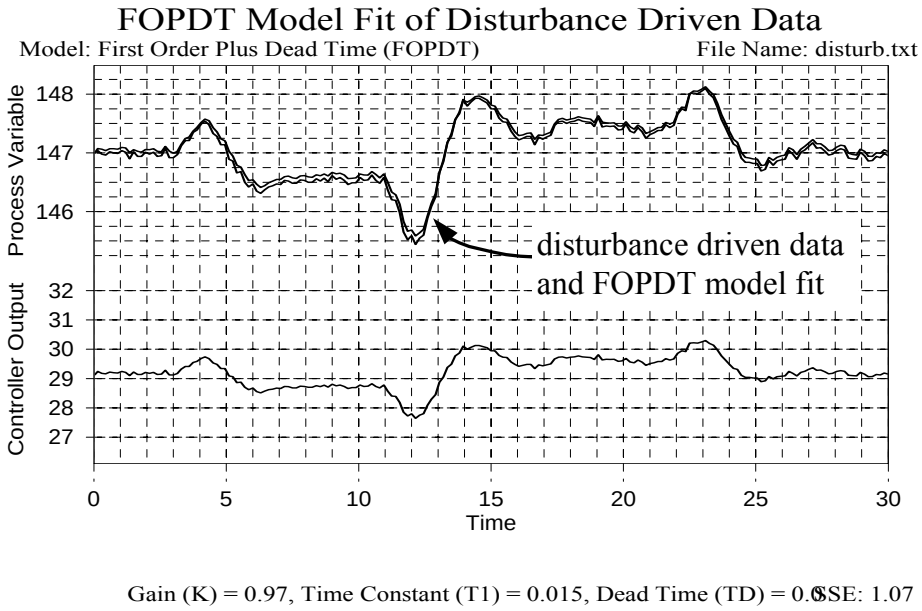


Figure 6.12 – Disturbance driven process data is well described by the FOPDT model

Although not shown by example, be aware that open loop data can also be corrupted by disturbance events. Further, it is not required for the event to be solely disturbance driven for problems to arise. To be conservative, if you suspect that a disturbance significant enough to influence the measured process variable has occurred during a test, you should repeat the test.

	Controller Output Driven Data (Proper FOPDT Model)	Disturbance Driven Data (Nonsense Result)
Process Gain, K_P ($^oC/\%$)	–0.90	0.97
Time Constant, τ_P (min)	1.1	0.02
Dead Time, θ_P (min)	0.9	0.0

Table 6.13 – Comparing controller output driven model to disturbance driven model

6.8 FOPDT Fit of Underdamped and Inverse Behaviors

FOPDT models cannot accurately describe the dynamic behavior of some processes even in narrow operating ranges. Underdamped and inverse processes display dynamic response behavior that is quite different than the classic step response shape discussed to this point.

Underdamped Process

Figure 6.14 shows a FOPDT fit of an underdamped step response. When a process is underdamped, the measured process variable oscillates as it responds to a controller output step.

Interestingly, even though the process response looks quite different from the FOPDT model fit, the popular tuning correlations still yield initial controller tuning parameters that provide reasonable closed loop performance. This is because, in spite of the mismatch between the measurement and model, the FOPDT model still describes the direction, how far, how fast and with how much delay the measured process variable responds to the change in controller output. This, after all, is the information ultimately required for controller design and tuning.

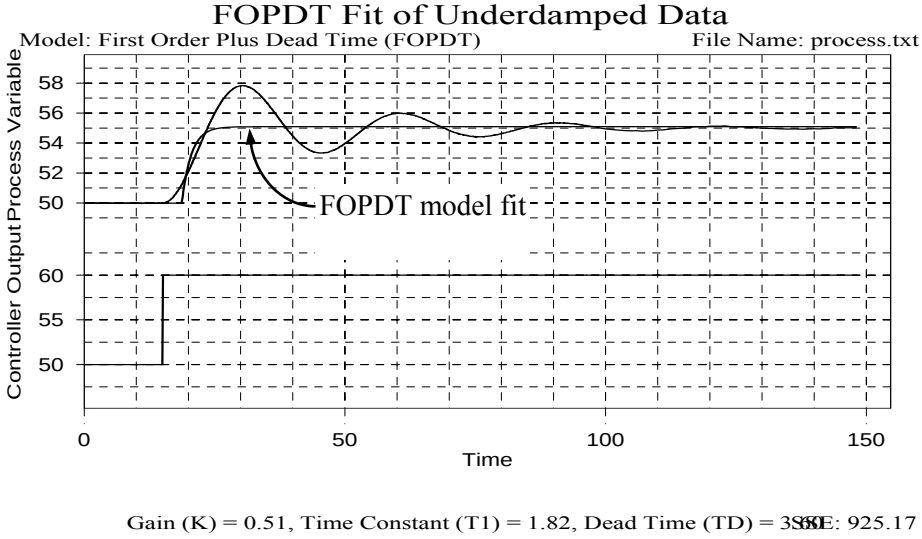


Figure 6.14 - FOPDT model fit of an underdamped process

Inverse Process

Figure 6.15 shows a FOPDT fit of an inverse (also called nonminimum phase) step response. When a process exhibits inverse behavior, the measured process variable first moves in one direction before it ultimately responds to steady state in the opposite direction. An example of this behavior can be found on the heat exchanger, where a step change in the warm oil disturbance flow rate produces an inverse response in the measured exit temperature.

As shown in Fig. 6.15, the FOPDT model approximates the inverse portion of the measured process variable as dead time. By doing so, the tuning correlations will cause the controller to essentially ignore the inverse portion of the response and rely on the final behavior in making control action decisions. Otherwise, the controller will be "chasing its tail" in trying to compensate for the temporary deviations. Thus the FOPDT model again provides appropriate information for the popular tuning correlations.

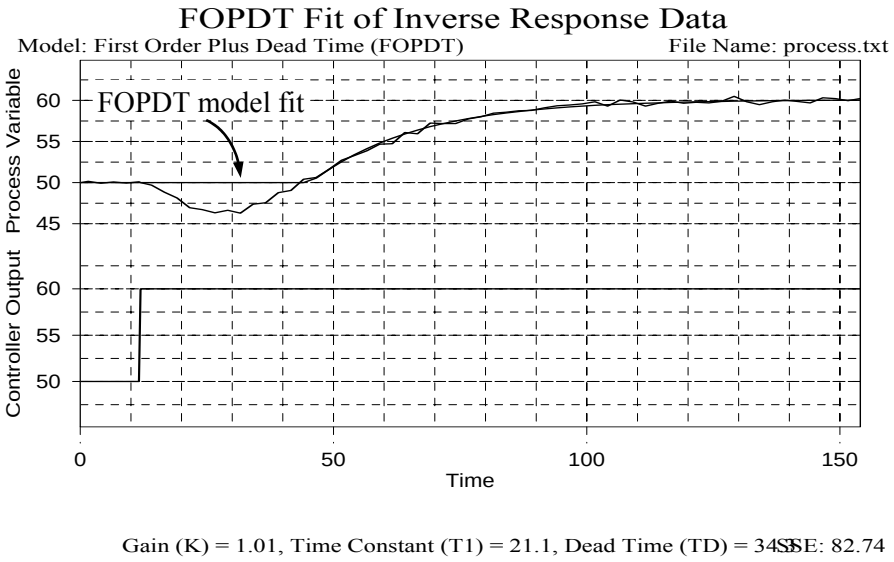


Figure 6.15 - FOPDT model fit of an inverse process

7. Advanced Modeling of Dynamic Process Behavior

7.1 Dynamic Models Have an Important Role Beyond Controller Tuning

As discussed toward the end of Chapter 3, no real process has its dynamic behavior exactly described by a FOPDT (first order plus dead time) model. Yet when forced by a change in the controller output, a FOPDT model reasonably describes the direction, how far, how fast and with how much delay the measured process variable will respond. This simple model thus provides the essential information required for controller tuning.

Dynamic models play an important role in process control beyond tuning. Two uses considered in this chapter are modeling for offline simulation, and modeling for the construction of model based controller architectures.

For offline simulation, LOOP-PRO provides for the creation of stand alone process models using *Custom Process*. Simulations are useful, for example, to investigate how different controller algorithms, architectures and tuning options perform on a particular process. The varied behaviors of processes can also be studied. The more studies that can substitute a simulation for the actual process are, the greater will be the savings in both time and money.

Offline studies become especially important when operations makes a process "off limits" for extended experimentation. It should come as no surprise to hear that the value of conclusions drawn using an offline simulation depend on how well the simulation model describes the true dynamic behavior of the process.

For model based control, LOOP-PRO offers a number of popular architectures including feed forward, Smith predictor and multivariable decouplers. These advanced architectures employ a dynamic model within the controller structure to predict the future behavior of a process. This enables control actions to be taken in advance if the predicted future does not match a desired behavior. As might be expected, the controller model must predict the true dynamics of a process with reasonable accuracy for success with these advanced architectures.

For offline simulation and model based control, a model must describe more than the basic "how far, how fast and with how much delay" features required for tuning. Improved accuracy is achieved by fitting the best model form to an appropriate set of dynamic process data. For these uses, an expanded list of models should be considered.

As summarized in Table 7.1, the two most popular dynamic models other than FOPDT include second order plus dead time (SOPDT) and second order plus dead time with lead time (SOPDT w/L). These models contain additional parameters that permit them to describe certain process behaviors with greater accuracy than can be achieved with the FOPDT form.

In earlier chapters we discussed what constitutes an "appropriate" set of dynamic process data. In particular, we noted that the manipulated variable must be moved far enough and fast enough so that the measured process variable displays a clear response that dominates the noise band of the measurement signal. Also, the data sample rate must be ten times the process time constant or faster, the process must start at steady state before data collection begins, and the first point in the data file must equal this steady state value.

In this chapter we explore the three dynamic models of Table 7.1 to better understand how to select the "best" model form. Like controller tuning, there is not one answer. Ultimately, it is the designer who decides when a dynamic process model is suitably descriptive for the task at hand.

7.2 Overdamped Process Model Forms

This book focuses on the control of process properties like temperature, pressure, level, flow, density, concentration and the like where the process streams are comprised of gases, liquids, slurries and melts. For this class of control challenges, the process variables very rarely display a natural tendency to oscillate (unlike, say, a mass and spring process that will oscillate quite nicely when perturbed). Processes that do not have an inherent (open loop) tendency to oscillate are called *overdamped*.

A large portion of overdamped processes are also *self regulating*. Self regulating processes seek a steady state operating level if all manipulated and disturbance variables are held constant for a sufficient period of time. LOOP-PRO's gravity drained tanks, heat exchanger, jacketed reactor and distillation column are examples of overdamped, self regulating processes. If you perform a step test on the pumped tank, on the other hand, you will see the behavior of a non-self regulating process.

First Order Plus Dead Time

Time Domain $$\tau_P \frac{dy(t)}{dt} + y(t) = K_P u(t - \theta_P)$$

Laplace Domain $$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Where:

- $y(t)$ = measured process variable signal
- $u(t)$ = controller output signal
- K_P = process gain; units of $y(t)/u(t)$
- τ_P = process time constant; units of time
- τ_L = process lead time; units of time
- θ_P = process dead time; units of time

Second Order Plus Dead Time

Time Domain $$\tau_{P1}\tau_{P2} \frac{d^2 y(t)}{dt^2} + (\tau_{P1} + \tau_{P2}) \frac{dy(t)}{dt} + y(t) = K_P u(t - \theta_P)$$

Laplace Domain $$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P e^{-\theta_P s}}{(\tau_{P1} s + 1)(\tau_{P2} s + 1)}$$

Second Order Plus Dead Time with Lead Time

Time Domain $$\tau_{P1}\tau_{P2} \frac{d^2 y(t)}{dt^2} + (\tau_{P1} + \tau_{P2}) \frac{dy(t)}{dt} + y(t) = K_P \left[u(t - \theta_P) + \tau_L \frac{du(t - \theta_P)}{dt} \right]$$

Laplace Domain $$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P (\tau_L s + 1) e^{-\theta_P s}}{(\tau_{P1} s + 1)(\tau_{P2} s + 1)}$$

Table 7.1 – Practical Overdamped Dynamic Model Forms

Table 7.1 lists the three most popular dynamic models used in process control to describe overdamped, self regulating dynamic process behavior. Each model is of higher order than the previous model on the list and this implies additional parameters that must be fit to the data.

An additional adjustable parameter lets a model better track the "bumps and curves" commonly found in process data. Because a dynamic model must reliably interpolate the data it is fit to, this added capability seems desirable.

Yet in control applications a model is also often required to extrapolate the data. During extrapolation beyond the limits of the original data, it is not uncommon for this "bumps and curves" descriptive capability to become exaggerated. The result can be wholly unrealistic predictions of process behavior. In this sense, a higher order model is a detriment.

In general, *always choose the simplest model form that provides an acceptable fit of your data. Simple models will likely provide the most reasonable extrapolation beyond the limits of that data.*

7.3 The Response Shape of First and Second Order Models

Table 7.1 shows that the FOPDT model has one time derivative and a single time constant. In comparison, the SOPDT model has a first and second time derivative and two time constants. The added derivative and associated time constant of the SOPDT model permits a rather subtle but surprisingly valuable benefit in describing dynamic process behavior.

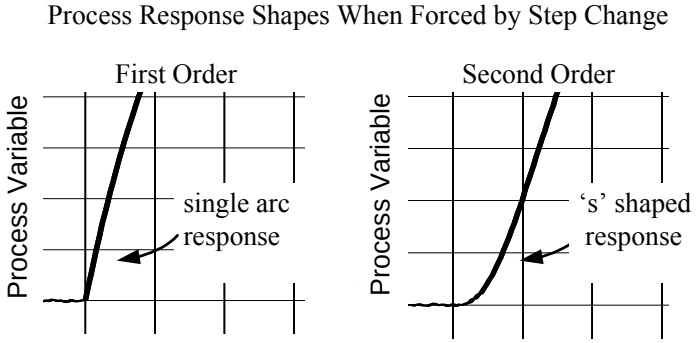


Figure 7.2 – Added time constant of second order model enables a gradual "s" shaped response

As shown in Fig. 7.2, the second time derivative and time constant of the SOPDT model permits the computed process variable, $y(t)$, to respond with a gradual "s" shape when forced by a step change. The single time derivative and time constant of the FOPDT model restricts the computed process variable response to a sudden or disjoint arc shape as shown in the figure.

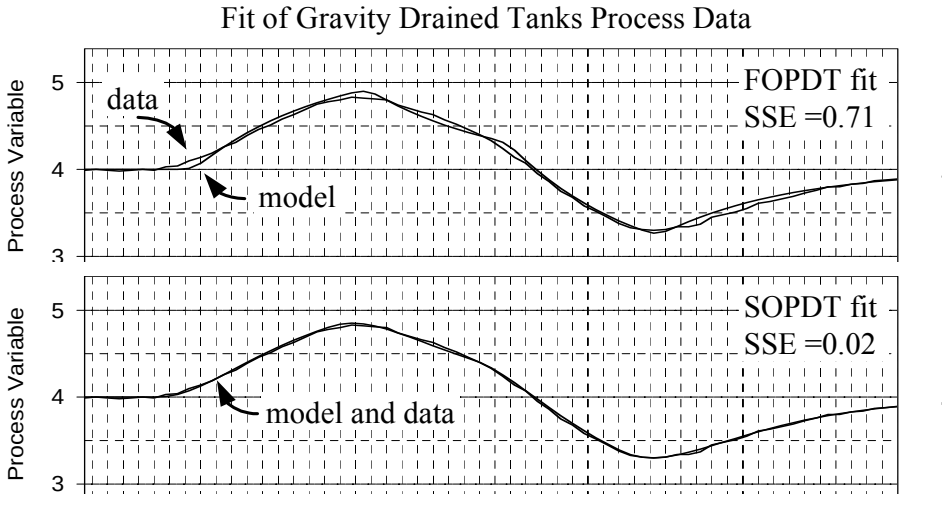


Figure 7.3 – Second order model permits better description of gravity drained tanks data

The ability of the SOPDT model to respond in a more gradual fashion means that the model can describe real process data with greater accuracy. Figure 7.3 shows doublet response data from the gravity drained tanks. A *Design Tools* fit of a FOPDT (upper plot) and SOPDT (lower plot) model are also shown. Not only is the fit better in the lower chart based on visual inspection, but the sum of squared errors (SSE) is significantly lower (recall as discussed in Section 6.5, the smaller the SSE the better the model describes the data).

7.4 The Impact of K_P, τ_P and θ_P on Model Behavior

As the process gain, K_P, time constant, τ_P, and dead time, θ_P, change, so does model behavior. The following study of these parameters is presented to help you refine your intuition. LOOP-PRO's *Custom Process* module is used to implement the models.

Steady State Process Gain, K_P

As shown in Fig 7.4, process gain is the "how far" variable. When the gain doubles in the center portion of the plot, the model computes a response that travels twice as far for the same change in controller output. When gain changes sign in the right portion of the plot, the response reverses directions but the response shape is unaffected. Though the plot is from a FOPDT model with a time constant of 10 time units and no dead time, the behaviors and observations hold for the other models in Table 7.1.

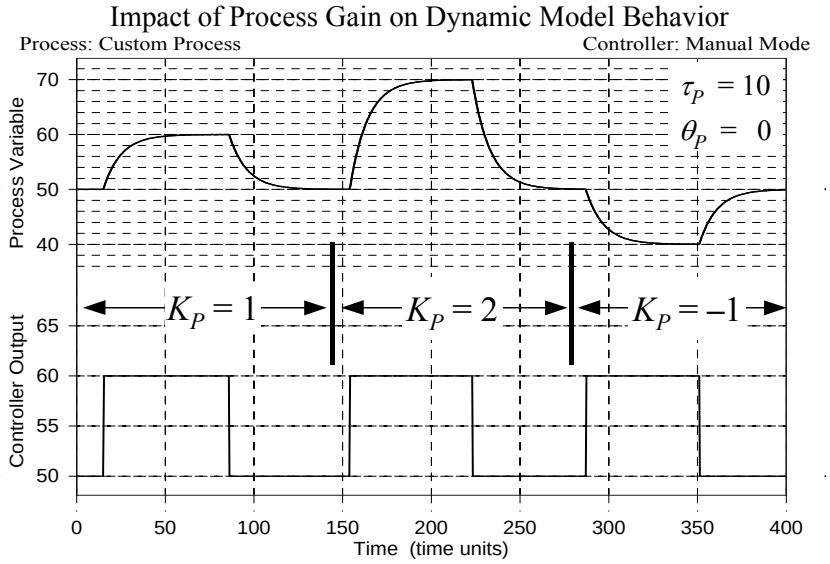


Figure 7.4 – Larger gain means the process variable responds farther for a step in controller output

Time Constant, τ_P

Figure 7.5 illustrates that time constant is the "how fast" variable, describing how quickly a process responds to a change in controller output. The right portion of Fig. 7.5 shows that when the first order time constant increases from 10 to 25 time units while the FOPDT gain and dead time are held constant, the response takes two and half times longer to complete.

Relating this time constant observation to a SOPDT model requires Eq. 7.1, which explains how the two time constants of a second order model can be approximated with a single equivalent first order time constant. To compute the first order equivalent, add the larger of the two second order time constants to one half the smaller value, or:

$$\tau_{FO} \approx \tau_{SO,max} + 0.5\,\tau_{SO,min} \tag{7.1}$$

where: τ_{FO} = equivalent first order time constant
$\tau_{SO,max}$ = larger of the two second order time constants
$\tau_{SO,min}$ = smaller of the two second order time constants

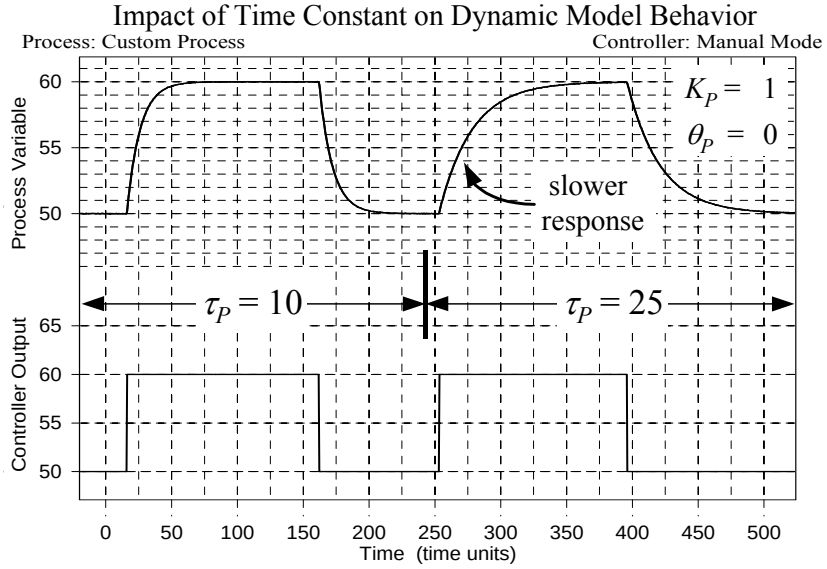


Figure 7.5 – Larger time constant means the process variable responds slower

To demonstrate, consider a SOPDT model with time constants of 7.5 and 5.0 time units. The first order approximation is computed as:

$$\tau_{FO} \approx 7.5 + 0.5(5.0) = 10 \text{ time units}$$

That is, the behavior of the second order model can be approximated as a first order model with a single time constant of 10 time units.

This result is demonstrated in Fig 7.7, which shows the response of a FOPDT model with a single time constant of 10 time units to the left and a SOPDT model with time constants of 7.5 and 5.0 time units to the right. Except for the more gradual "s" shape at the beginning of the second order response, the two plots are very similar as predicted by Eq. 7.1.

The implication of this result is that doubling the response time of a second order model *is not achieved* by doubling the two individual time constants. On the contrary, it requires doubling the equivalent first order time constant approximation. As suggested by Eq. 7.1, this can be achieved a number of ways for a model. Two ways the response time of the previous second order process can be doubled to 20 time units are shown in Table 7.6:

τ_{P1}	τ_{P2}	τ_{FO}
17.5	5.0	$\tau_{FO} \approx 17.5 + 0.5(5.0) = 20$ time units
15.0	10.0	$\tau_{FO} \approx 15.0 + 0.5(10.0) = 20$ time units

Table 7.6 – Two ways to make the response time of a second order model equal 20 time units

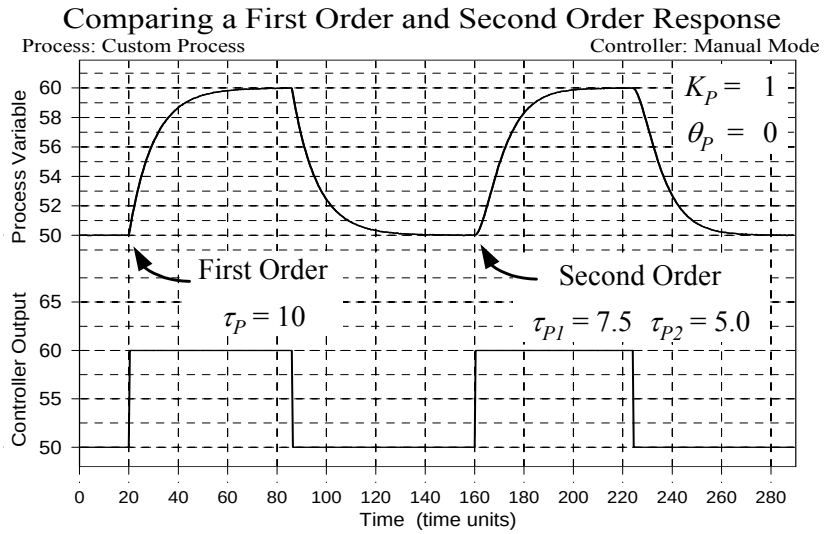


Figure 7.7 – How the time constants of a SOPDT model relate to that of the FOPDT model

Dead Time, θ_P

Dead time is the "with how much delay" variable. A response shape is unaffected by a change in dead time except that as dead time increases, the time delay before the process begins its first detectable response to a change in controller output increases. This is true for all of the models in Table 7.1.

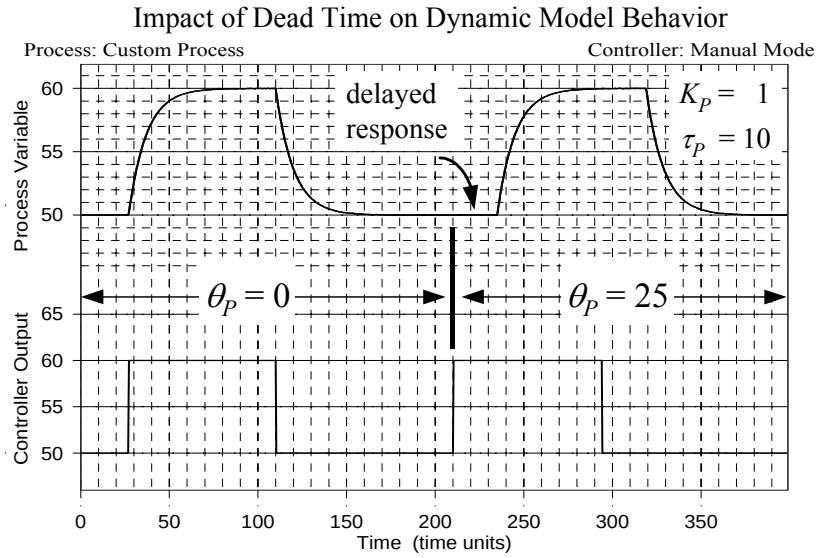


Figure 7.8 – Longer dead time means there is a longer delay before the process responds

Figure 7.8 shows the response of two models with identical gain and time constant when forced by the controller output. The plot to the left is from a model with zero dead time while that to the right is from a model with a dead time of 25 time units. Except for the delayed response indicated in Fig 7.8 due to dead time, the response shapes are identical.

7.5 The Impact of Lead Element τ_L on Model Behavior

As indicated in Table 7.1, lead time, τ_L, is a parameter that weights the rate of change (derivative) of the controller output signal, $u(t)$. It has units of time, but unlike a process time constant, it can be either positive or negative. This is possible because lead time does not describe a time frame in which a process variable evolves (which must proceed forward in time). Rather, it describes the influence that the controller output has on the measured process variable.

Suppose $u(t)$ is rapidly increasing (has a large positive derivative). If lead time, τ_L, is positive, the model says to impart a large positive movement to $y(t)$ on top of that dictated by the process gain and time constants (which describe the "natural" dynamics of the process). If τ_L is negative, the model says to impart a large negative movement to $y(t)$ when $u(t)$ is rapidly increasing.

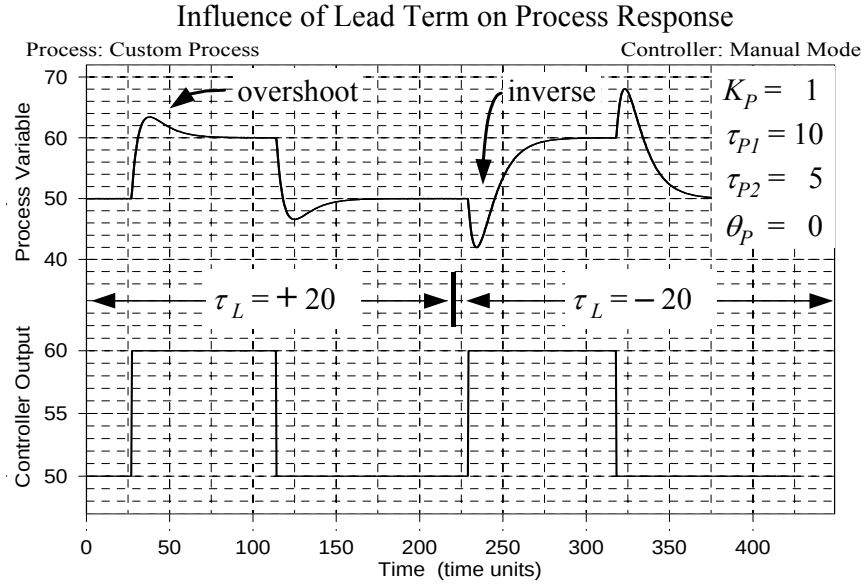


Figure 7.9 – Lead time can be positive or negative and has a profound impact on response shape

The influence of lead time is shown in Fig. 7.9 for a SOPDT w/L (second order plus dead time with lead time) model. When lead time is large and positive as shown to the left in Fig. 7.9, the additional large positive movement imparted to $y(t)$ causes it to overshoot the steady state before it returns to its proper final value. When lead time is large and negative as shown to the right in the figure, the additional large negative movement imparted to $y(t)$ actually causes it to go the wrong direction (inverse response) before correcting itself and moving to its proper steady state.

In spite of the extra dynamic influence of the lead element, the final steady state is still established by the value of the process gain, K_P. As always, the process time constants influence the speed of response, but their impact is well masked in the above example.

When the SOPDT w/L model is viewed in the Laplace domain (shown in Table 7.1), note that the lead term is in the numerator while the time constant terms are in the denominator. Because of this arrangement, it is possible for the lead term to cancel out a process time constant if they are the same value.

$$\frac{Y(s)}{U(s)} = \frac{\cancel{(10\,s+1)}\,e^{-2s}}{(5.0\,s+1)\cancel{(10\,s+1)}} \tag{7.2}$$

Equation 7.2 shows a SOPDT w/L model where $K_P = 1$, $\theta_P = 2$, $\tau_{P1} = 5$, $\tau_{P2} = 10$, and $\tau_L = 10$. Because terms cancel as shown, Eq. 7.2 will behave exactly like a FOPDT model.

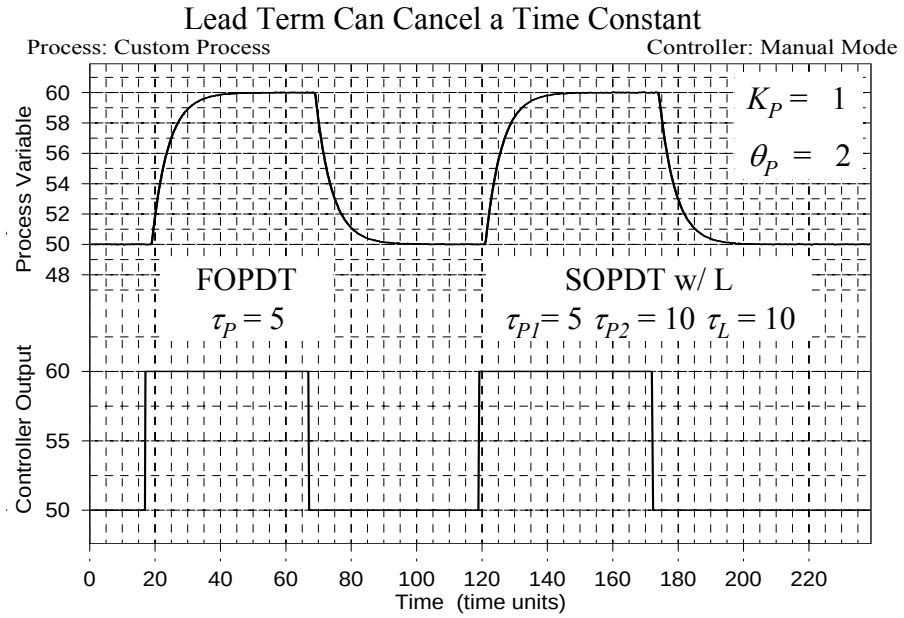


Figure 7.10 – FOPDT response is identical to SOPDT w/ L when terms cancel

Figure 7.10 demonstrates this. Two responses are shown, one for a true FOPDT model response with $K_P = 1$, $\theta_P = 2$ and $\tau_P = 5$ to the left and one for the SOPDT w/L model discussed above to the right. Because terms cancel as shown in Eq. 7.2, the two responses are identical.

8. Integral Action and PI Control

8.1 Form of the PI Controller

Like the P-Only controller, the Proportional-Integral (PI) controller computes an output signal to the final control element based on tuning parameters and the controller error, $e(t)$. Called the ideal, continuous and position form, the PI controller is expressed in Eq. 8.1:

$$u(t) = u_{\text{bias}} + K_C\, e(t) \; + \; \frac{K_C}{\tau_I} \int e(t)\, dt \tag{8.1}$$

As before, $u(t)$ is the controller output, u_{bias} is the controller bias, and K_C is controller gain (a tuning parameter). The additional tuning parameter, τ_I, provides a separate weight to the integral term, has units of time (and thus is always positive), and is commonly referred to as *reset time*. Because τ_I is in the denominator, smaller values of reset time provide a larger weight to (increase the influence of) the integral term.

The first two terms to the right of the equal sign in Eq. 8.1 are identical to the P-Only controller of Eq. 5.1. The integral mode of the PI controller is an additional and separate term added to the equation that integrates or continually sums the controller error over time.

Though not required to understand the actions of a PI controller, we present the Laplace transfer function form for completeness. The Laplace form of the PI controller is:

$$G_C(s) \; = \; \frac{U(s)}{E(s)} \; = \; K_C\left(1 + \frac{1}{\tau_I s}\right) \tag{8.2}$$

8.2 Function of the Proportional and Integral Terms

As with the P-Only controller, the proportional term of the PI controller, $K_C e(t)$, adds or subtracts from u_{bias} based on how far the measured process variable is from the set point at any instant in time. That is, the contribution to $u(t)$ from the proportional term is based on the size of $e(t)$ at time t. As $e(t)$ grows or shrinks, the amount added to u_{bias} grows or shrinks immediately and proportionately. The past history and current trajectory of the controller error have no influence on the proportional term computation.

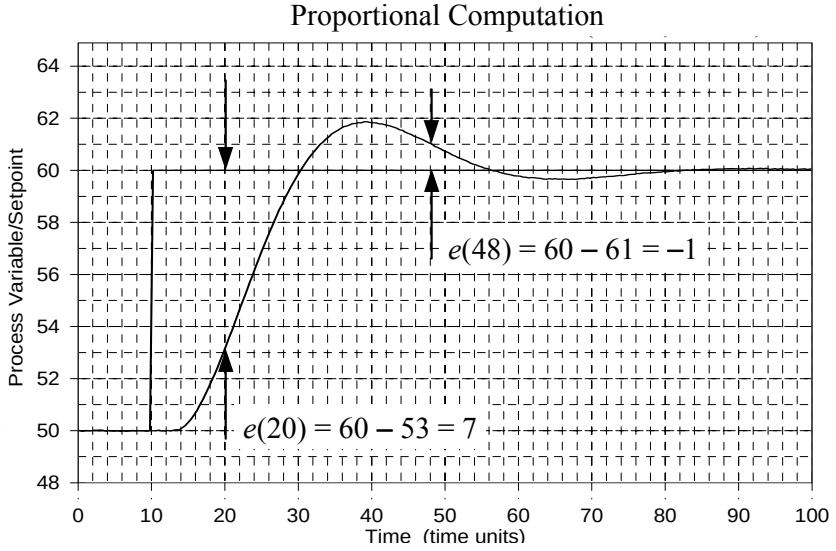


Figure 8.1 – Proportional calculation is based only on current error at time t

Figure 8.1 illustrates this for a set point response with arrows that mark the size of $e(t)$ used in the proportional term computation at time $t = 20$ and at time $t = 48$. Figure 8.2 shows the identical data cast as a plot of error itself, created by subtracting the measured process variable from set point at each point in time. As shown, controller error continually changes size and sign as time passes. Controller error also has units, which in commercial systems is often percent of span.

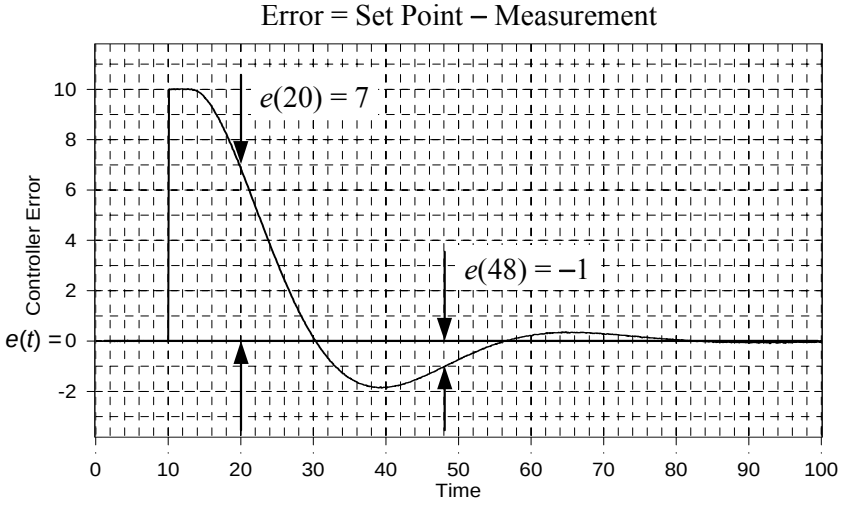


Figure 8.2 – Error, e(t), continually changes size and sign as time passes

In contrast, the integral term of the PI computation considers the history of the error, addressing how long and how far the measured process variable has been from the set point over time. The integral term integrates or continually sums up the error history. Thus, even a small error, if it persists, will have a sum total that grows over time and the integral term contribution added to u_{bias} will similarly grow.

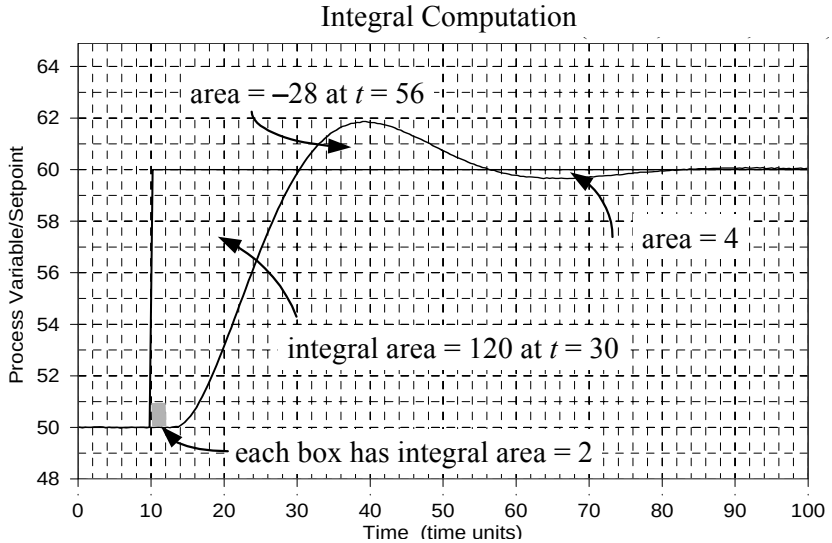


Figure 8.3 - Integral of error is the area between the set point and measurement

As indicated in Fig. 8.3, the result of the continual summing of integration, starting from the moment the controller is put in automatic, is the computation of the area on the graph between the set point and measured process variable. Thus, at time t = 30 min, when the measured process variable first crosses the set point in Fig.8.3, the integral is:

$$\int_{0\,\text{min}}^{30\,\text{min}} e(t)\,dt = 120 \tag{8.3}$$

The contribution this integral ultimately makes to u_{bias} in the PI controller of Eq. 8.1, and thus its impact on $u(t)$, depends on the values of the tuning parameters K_C and τ_I. Because K_C is in the numerator and τ_I is in the denominator, larger values of K_C and smaller values of τ_I increase the influence of the integral term.

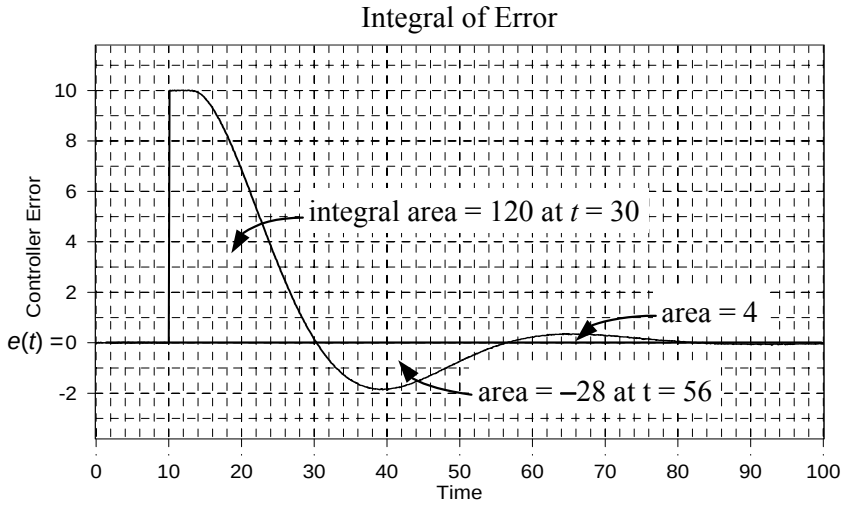


Figure 8.4 - Integral of error continually grows and shrinks as time passes

Integration is continual, growing as long as $e(t)$ is positive and shrinking when the error is negative. At time t = 56 min, when the measured process variable crosses the set point the second time in Fig. 8.3 (or when the error changes sign for the second time in Fig. 8.4), the total value of the integral is (+120 – 28) = 92. When the dynamic event (transient) ends, the total integral is (+120 – 28 + 4) = 96.

Recognize that after the transient is over, the integral term can have a residual value even though $e(t)$ is constant at zero. In Fig. 8.4, the transient has essentially died out yet the integral of the complete transient has a final or residual value of 96. As discussed in the next section, the consequence of this is that integral action enables the PI controller to eliminate offset.

8.3 Advantages and Disadvantages to PI Control

As discussed in Section 5.7, offset occurs in most processes under P-Only control when the set point and/or disturbance are at a value other than that used in the controller design, or more specifically, used to determine u_{bias}. The big advantage of a PI controller is that it eliminates offset.

Offset is eliminated with the PI controller of Eq. 8.1 because as long as there is any error (as long as $e(t)$ is not zero), the integral term will grow or shrink in size, causing the controller output, $u(t)$, to change. Changes in $u(t)$ will only cease when $y(t)$ equals $y_{setpoint}$ ($e(t) = 0$) for a sustained period of time. At that point, the proportional term of Eq. 8.1 equals zero, and the integral term may have a residual value as just discussed. *This residual value of integration, when added to u_{bias}, in effect creates a new overall bias value that corresponds to the new level of operation.*

The ability to eliminate offset is a tremendous advantage. In fact, PI controllers are the most widely used of the PID family. There are disadvantages with the algorithm, however, including that:

- two tuning parameters that interact in their influence must be balanced by the designer,
- the integral term increases the oscillatory or rolling behavior of the closed loop system.

Because the two tuning parameters interact with each other, it can be challenging to arrive at "best" tuning values once the controller is placed in automatic.

8.4 Controller Bias From Bumpless Transfer

As just discussed, if $e(t)$ is not zero, the integral term will cause the controller output, $u(t)$, to change. Changes in $u(t)$ will only cease when $e(t)$ equals zero for a sustained period of time. At that point, the integral term can have a residual value that, when added to u_{bias}, in effect creates a new overall bias value corresponding to the current level of operation.

Although integral action enables the controller to adjust to changes in operating level, the bias should still be properly initialized when the controller is put in automatic. Achieving this is surprisingly simple and requires no work on your part. When put in automatic with integral action enabled, most commercial controllers (including LOOP-PRO) initialize the bias to the current value of the controller output. This "bumpless transfer" feature means that when the control loop is closed, the bias is correct for the current level of operation. From that point forward, integral action enables the controller to track any changes in operating level.

8.5 Controller Tuning From Correlations

Designing any controller from the family of PID algorithms entails the following steps:

- specifying the design level of operation,
- collecting dynamic process data as near as practical to this design level,
- fitting a FOPDT model to the process data, and
- using the resulting model parameters in a correlation to obtain initial controller tuning values.

As always, final tuning is performed by trial and error once the controller is online so loop performance matches that desired by the control designer.

The tuning correlations available in *Design Tools* for the PI controller are the Internal Model Control (IMC) relations. These are an extension of the popular *lambda* tuning correlations and include the added sophistication of directly accounting for dead time in the tuning computations.

The first step in using the IMC (lambda) tuning correlations is to compute, τ_C , the closed loop time constant. All time constants describe the speed or quickness of a response. The closed loop time constant describes the desired speed or quickness of a controller in responding to a set point change. Hence, a small τ_C (a short response time) implies an aggressive or quickly responding controller.

LOOP-PRO's *Design Tools* automatically computes a τ_C for moderate tuning, which can be changed to more aggressive or conservative values using software options.Moderate tuning provides a set point response that, while reasonably energetic, shows little or no overshoot. The closed loop time constants are computed using Eq. 8.4:

Aggressive Tuning: τ_C is the larger of $0.1\,\tau_P$ or $0.8\,\theta_P$ (8.4a)

Moderate Tuning: τ_C is the larger of $1.0\,\tau_P$ or $8.0\,\theta_P$ (8.4b)

Conservative Tuning: τ_C is the larger of $10\,\tau_P$ or $80.0\,\theta_P$ (8.4c)

With τ_C computed, the PI correlations for IMC (lambda) tuning are

$$K_C = \frac{1}{K_P}\frac{\tau_p}{(\theta_P + \tau_C)} \qquad \tau_I = \tau_P \tag{8.5}$$

Another popular set of tuning correlations are the integral of time-weighted absolute error (ITAE) tuning correlations. We do not believe they are as dependable as the IMC correlations so they are not computed by *Design Tools*. They can easily be computed directly from the FOPDT model parameters, however. When *set point tracking* is the control objective, the ITAE tuning correlation is shown in Eq. 8.6:

$$K_C = \frac{0.586}{K_P}(\theta_P/\tau_P)^{-0.916} \qquad \tau_I = \frac{\tau_P}{1.03 - 0.165(\theta_P/\tau_P)} \tag{8.6}$$

When *disturbance rejection* is the control objective, the ITAE tuning correlation is shown in Eq. 8.7:

$$K_C = \frac{0.859}{K_P}(\theta_P/\tau_P)^{-0.977} \qquad \tau_I = \frac{\tau_P}{0.674}(\theta_P/\tau_P)^{0.680} \tag{8.7}$$

The different correlations will yield different tuning values and your judgment is required in selecting which to use. If you are uncertain, remember that it is most conservative to start with the smallest gain and largest reset time as this will give the least aggressive controller.

Final tuning is performed on-line and by trial and error until desired controller performance is obtained. If the process is responding sluggishly to disturbances and changes in the set point, the controller gain is too small and/or the reset time is too large. Conversely, if the process is responding quickly and is oscillating to a degree that makes you uncomfortable, the controller gain is too large and/or the reset time is too small.

8.6 Set Point Tracking in Gravity Drained Tanks Using PI Control

The design level of operation for this study is a measured level in the lower tank of 2.4 m while the pumped flow disturbance is at its expected value of 2.0 L/min. The control objective is to track set point steps in the range of 2.0 to 2.8 m. The process is currently under P-Only control and operations personnel will not open the loop for controller design experiments. Hence, closed loop set point steps are used to generate dynamic process data.

As shown in Fig. 8.5, the P-Only controller being used has a $K_C = 40$ %/m and a bias value of 55.2% (determined as the value of the controller output that, in open loop, causes the measured level in the lower tank to steady at the design value of 2.4 m when the pumped flow disturbance is at its expected value of 2.0 L/min). With data being saved to file, the dynamic testing experiment begins. Specifically, the set point is stepped up to 2.8 m, then down to 2.0 m, and finally back to the design level of 2.4 m (set point sequences of other sizes and durations would be equally reasonable).

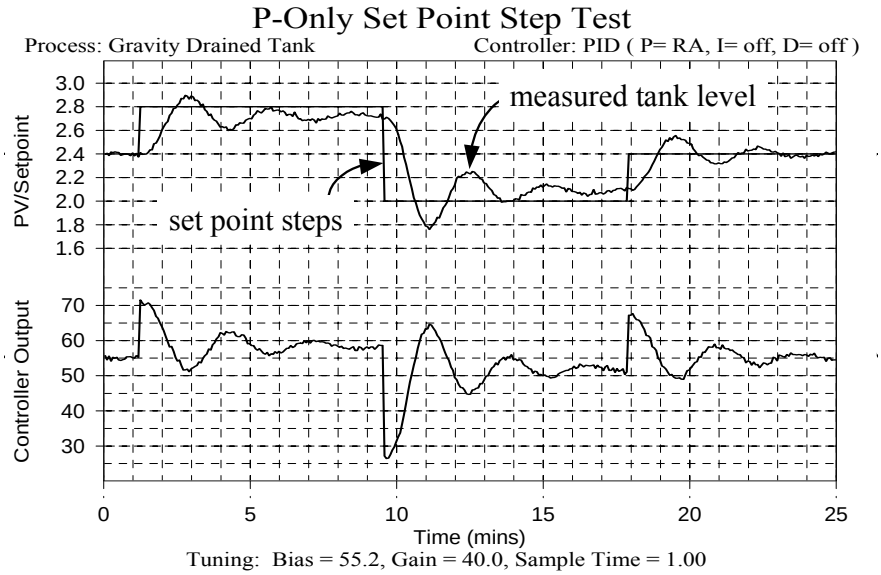


Figure 8.5 – Set point step tests on gravity drained tanks under P-Only control

Visual inspection of Fig. 8.5 confirms that the dynamic event is set point driven (as opposed to disturbance driven). Also, control action appears energetic enough such that the response of the measured process variable clearly dominates the noise.

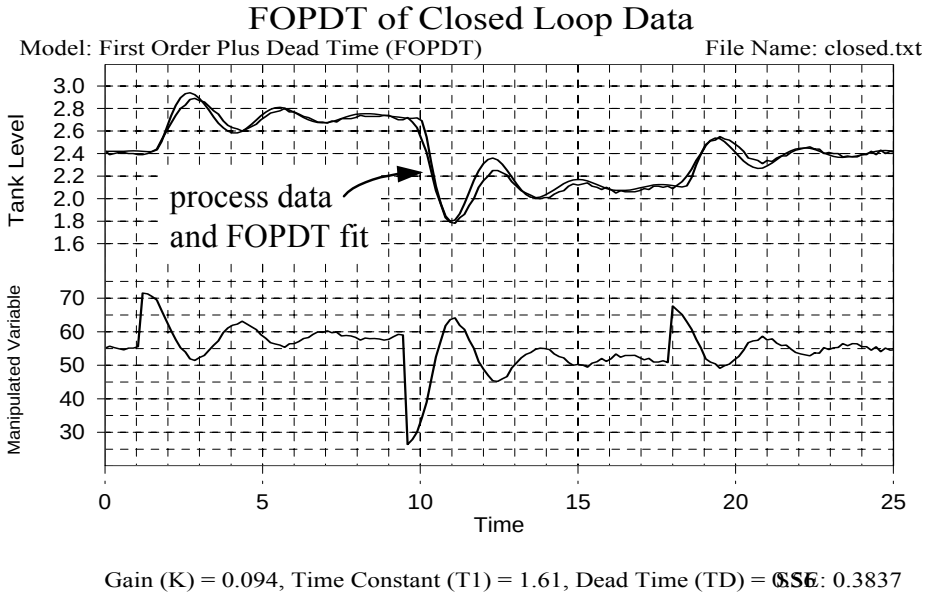


Figure 8.6 – FOPDT fit of closed loop dynamic data generated in Fig.8.5

The dynamic data of Fig. 8.5 is read into *Design Tools* and fit with a FOPDT model. A plot of the model and closed loop process data is shown in Fig. 8.6. The model appears to be reasonable and appropriate based on visual inspection, thus providing the design parameters:

Process Gain, K_P = 0.094 m/%

Time Constant, τ_P = 1.6 min

Dead Time, θ_P = 0.56 min

We first compute the closed loop time constant. Here we choose aggressive tuning, which in the software is computed as:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = \text{larger of } 0.1(1.6) \text{ or } 0.8(0.56) = 0.45 \text{ min.}$$

Substituting this closed loop time constant and the above FOPDT model parameters into the IMC tuning correlations of Eq. 8.5 yields the following tuning values:

$$K_C = \frac{1}{0.094}\left(\frac{1.6}{0.56 + 0.45}\right) = 16.9 \text{ \%/min} \qquad \tau_I = 1.6 \text{ min}$$

A reverse acting controller is required because K_C is positive. Integral with anti-reset windup logic (which is always desired, as discussed in Section 8.11) is enabled to complete the PI algorithm. Because integral action is being used, the bias value is not entered but rather is automatically initialized to the current value of the controller output at the moment the loop is closed.

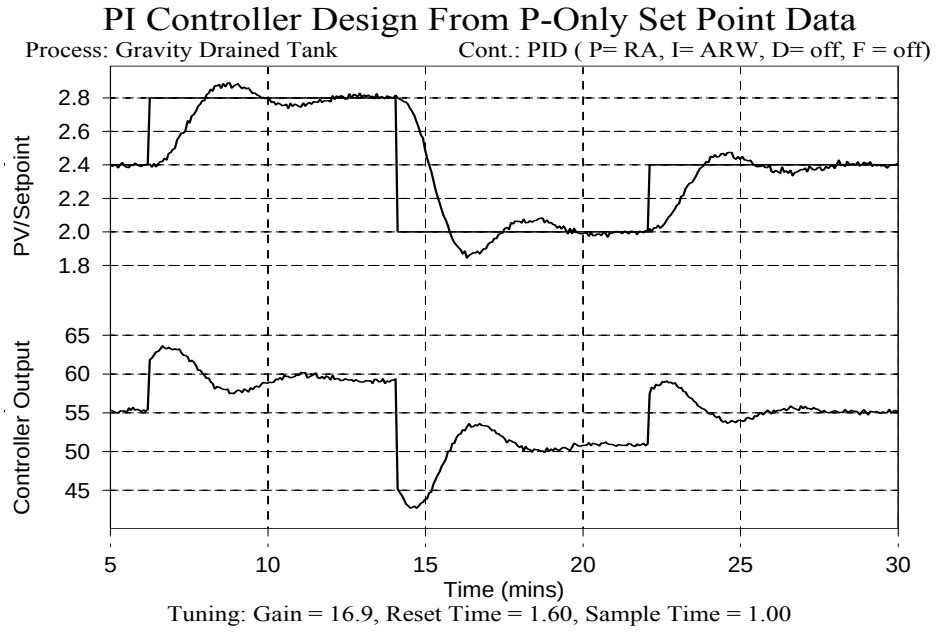


Figure 8.7 – Performance of PI controller in tracking set point steps

The performance of this controller in tracking set point changes is pictured in Fig. 8.7. The upper right corner of the plot shows; PID (P= RA, I= ARW, D= off, F=off), confirming that a reverse acting PI controller with anti-reset windup logic is being used. Although good or best performance is decided based on the capabilities of the process, the goals of production, the impact on downstream units and the desires of management, Fig. 8.7 exhibits generally desirable performance. That is, the process responds quickly, shows modest overshoot, settles quickly, and displays no offset. Compare this to Fig. 8.5, that shows P-Only performance for the same control challenge.

8.7 Disturbance Rejection in Heat Exchanger Using PI Control

For this study, a constant measured exit temperature of 147°C is desired. The control objective is to reject disturbances that occur when the warm oil flow rate, normally about 10 L/min, occasionally spikes as high as 20 L/min. Using the open loop doublet test shown in Fig.6.6, appropriate FOPDT model parameters for this design are as follows:

$$\text{Process Gain, } K_P = -0.90\,°\text{C}/\%$$

$$\text{Time Constant, } \tau_P = 1.1 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.9 \text{ min}$$

The ITAE for disturbance rejection tuning correlations of Eq. 8.7 are demonstrated here, though they are not our first choice for industrial applications. Substituting the above FOPDT model parameters into these correlations yield the tuning values:

$$K_C = \frac{0.859}{-0.90}(0.9/1.1)^{-0.977} = -1.2\ \%/°\text{C} \qquad \tau_I = \frac{1.1}{0.674}(0.9/1.1)^{0.680} = 1.4 \text{ min}$$

A direct acting controller is required because K_C is negative. Integral with anti-reset windup logic is enabled to complete the PI algorithm. Because integral action is being used, the bias value is automatically initialized to the current value of the controller output at the moment the loop is closed.

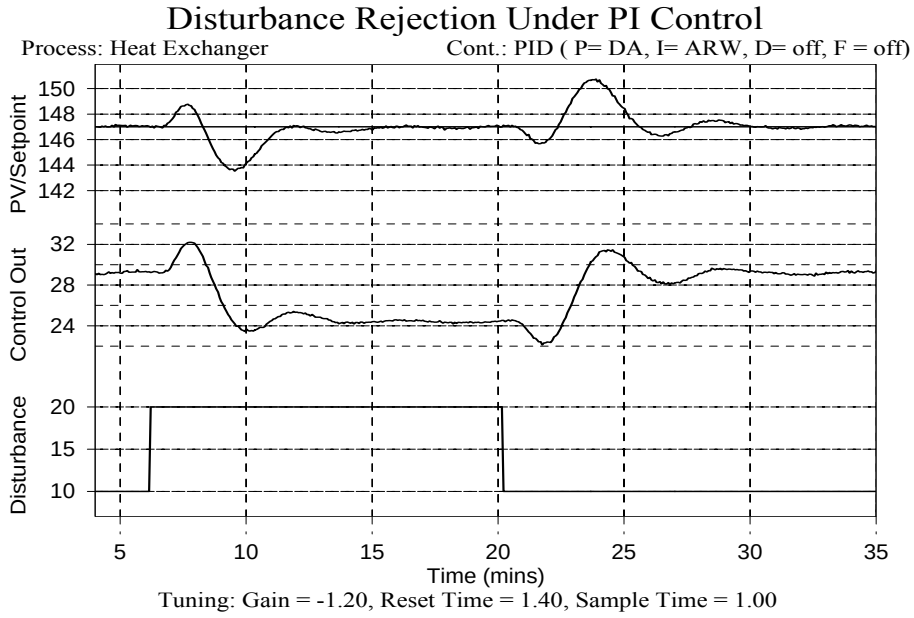


Figure 8.8 - Disturbance rejection performance of heat exchanger under PI control

Figure 8.8 shows the performance of this controller in rejecting step changes in the warm oil disturbance flow rate. The set point is held constant throughout the experiment at the design operating level of 147°C. When the disturbance flow rate steps from 10 L/min up to 20 L/min, the PI controller succeeds in returning the measured process variable to set point, thus eliminating offset.

8.8 Interaction of PI Tuning Parameters

As mentioned, one disadvantage of the PI controller is that there are two tuning parameters to adjust and difficulties can arise because these parameters interact with each other. Fig.8.9 shows a tuning map that illustrates how a typical set point response might vary as the two tuning parameters are changed.

The center of Fig. 8.9 shows a set point step response that is labeled as the base case performance. It is important to recognize that this base case plot will not be considered by some to be the "best" performance. What is best must be determined by the operator or engineer for each implementation. Some require no overshoot while others will tolerate some overshoot in exchange for a faster set point response. In any event, the grid shows how a set point step response changes as the two tuning parameters are doubled and halved from a base (here defined as desired) tuning.

The plot in the upper left of the grid shows that when gain is doubled and reset time is halved, the controller produces large, slowly damping oscillations. Conversely, the plot in the lower right of the grid shows that when controller gain is halved and reset time is doubled, the response becomes sluggish. This chart is called a tuning map because, in general, if a controller is behaving poorly, you can match the performance you observe with the closest picture in Fig. 8.9 and obtain guidance as to the appropriate tuning adjustments required to move toward your desired performance.

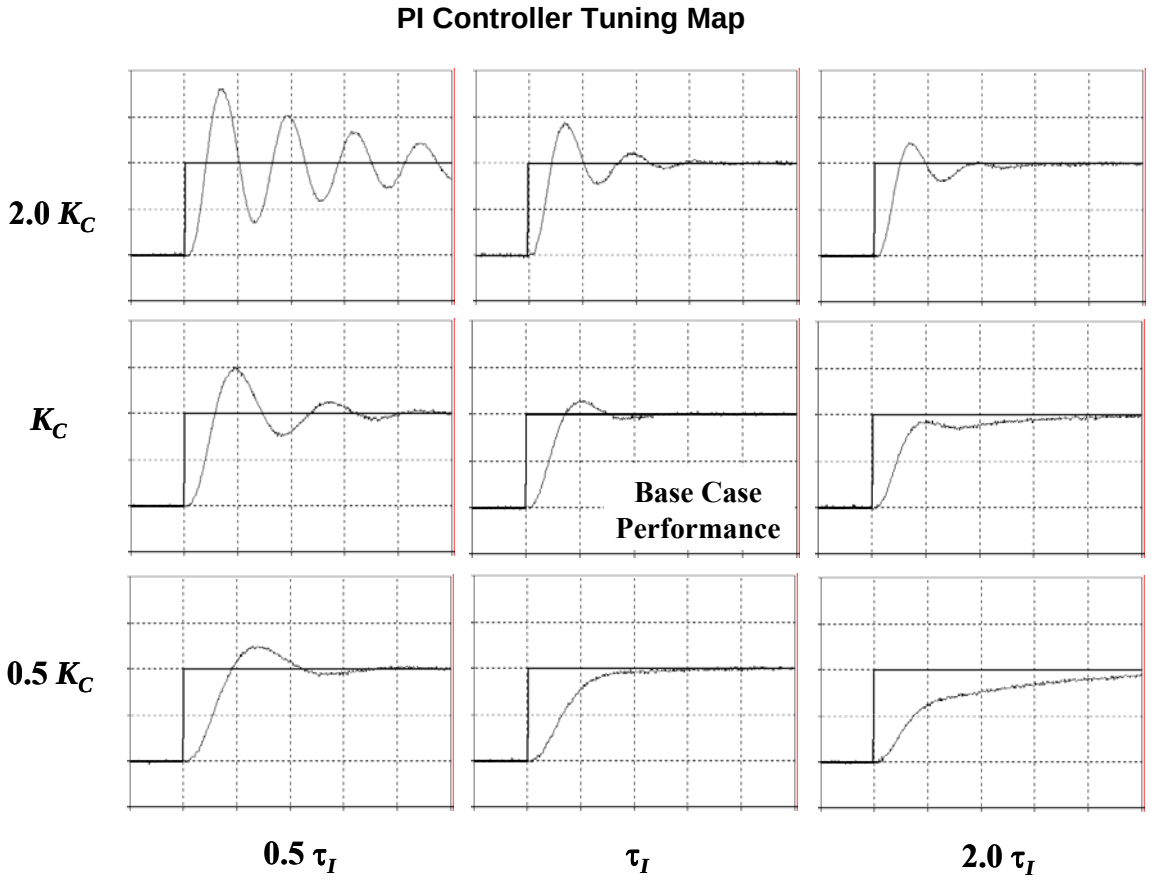


Figure 8.9 – How PI controller tuning parameters impact set point tracking performance

8.9 Reset Time Versus Reset Rate

Different manufacturer of controllers use different names for the tuning parameters. As discussed in Chapter 5, some use *proportional band* rather than controller gain. Also, some use *reset rate* instead of reset time, shown in Eq. 8.8 below:

$$\text{Reset Rate} = \frac{1}{\tau_I} \tag{8.8}$$

Reset rate has units of 1/time or sometimes repeats/minute. In any case, it is *important to know your manufacturer before you start tuning any controllers!*

8.10 Continuous (Position) Versus Discrete (Velocity) Form

As expressed in Eq. 8.1, the continuous form of the PI controller is:

$$u(t) = u_{\text{bias}} + K_C\, e(t) \; + \; \frac{K_C}{\tau_I} \int e(t)\, dt \tag{8.9}$$

This is sometimes called the position form because the computed $u(t)$ is a specific value in the range from 0-100%. When the final control element is a process valve, this controller output in essence is specifying the actual position between opened and closed that the valve should take.

The first step in deriving the discrete form is to take the time derivative of the continuous form (the time derivative, or rate of change, of a position is a velocity. Thus, the discrete form of the PI controller is sometimes called the velocity form). Taking the derivative with respect to time yields the following:

$$\frac{d\,u(t)}{dt} = \frac{d\,u_{\text{bias}}}{dt} + K_C\, \frac{d\,e(t)}{dt} + \frac{K_C}{\tau_{\text{I}}} e(t) \tag{8.10}$$

Since u_{bias} is a constant, then $du_{\text{bias}}/dt = 0$. Assigning finite difference approximations for the continuous derivatives, Eq. 8.10 becomes the following:

$$\frac{\Delta u}{\Delta t} \; = \; K_C \left(\frac{e_i - e_{i-1}}{\Delta t} \right) \; + \; \frac{K_C}{\tau_I} e_i \tag{8.11}$$

where e_i is the current controller error and e_{i-1} is the controller error from the last sample. Assigning loop sample time as $T = \Delta t$, then the *discrete* or velocity PI controller form results in Eq, 8.12, shown below:

$$\Delta u \; = \; K_C \left(1 + \frac{T}{\tau_I} \right) e_i \; - \; K_C\, e_{i-1} \tag{8.12}$$

This discrete form of the PI controller computes a *change* in valve position rather than the absolute position itself. Wherever the valve (final control element) position happens to be, the Δu instructs the valve in what direction and by how much to change.

As long as the controller output never reaches the maximum (100%) or minimum (0%) value, the continuous and discrete forms of the PI controller behave identically (see the next section on reset windup to learn more).

8.11 Reset Windup

As long as an error persists, the integral term in Eq. 8.9 will continue to grow. If an error is large enough and persists long enough, it is mathematically possible for the integral term to grow so large and its contribution to u_{bias} to become so great that the final control element saturates or reaches a physical limit of fully open/on or fully closed/off. If this extreme position is still not sufficient to eliminate the error, the mathematics of Eq. 8.9 permit the integral term to grow yet more.

When the computed $u(t)$ exceeds the physical capabilities of the final control element because the integral term has reached a huge value, the condition is known as windup. Because windup is associated with the integral term, it is often referred to as *reset windup*. Once this condition has occurred, the controller losses the ability to regulate the process. If and when the error eventually changes sign and the integral term "unintegrates" or shrinks sufficiently so that the final control element is no longer saturated, control action can then resume.

The discrete (velocity) form of Eq. 8.12 will not windup because the continuous integral is eliminated with the introduction of the time derivative in Eq. 8.10. Thus, using the discrete form of Eq. 8.12 not only eliminates the bias term, but also solves the windup problem.

Unfortunately, the usefulness of Eq. 8.12 in industrial practice is limited because the form suffers problems when derivative action is included. When taking the derivative of the full PID algorithm as shown in Eq. 8.13, a derivative of the derivative term yields a second derivative. A numerical second derivative applied to data that contains even modest noise produces nonsense results.

$$\frac{d\,u(t)}{dt} = \frac{d\,u_{\text{bias}}}{dt} + K_C\,\frac{d\,e(t)}{dt} + \frac{K_C}{\tau_{\text{I}}}e(t) + K_C\tau_D\,\frac{d^2e(t)}{dt^2} \tag{8.13}$$

Hence, most industrial controllers employ the continuous PID algorithm and include *jacketing logic* to halt integration when the controller output reaches a maximum or minimum value.

Modern controllers will not suffer from reset windup. It is an old problem that has long been solved. Beware if you program your own controller, however. Reset windup is a trap that novices fall into time and again.

9. Evaluating Controller Performance

9.1 Defining "Good" Controller Performance

Consider that a bioreactor might not be able to tolerate sudden changes in operating conditions because the fragile living organisms could die. In such a process, it would be desirable to design the controllers so that the measured process variables respond slowly when counteracting disturbances or working to track changes in set point.

On the other hand, designers of a concentration control application where the stream then flows into a mixing tank may seek a very rapid response to set point changes. They may be willing to tolerate the slowly damping oscillations in the measured concentration that result from such aggressive control action because the averaging effect of the mixing tank minimizes the ultimate impact on product quality. In this situation, an aggressive controller that causes significant oscillation in the measured process variable may be considered acceptable or even good performance.

As these examples illustrate, different applications can have quite different definitions of "good" closed loop performance. Ultimately, it is the operator or engineer who defines what is good or best for a particular application. To make this determination, he/she should consider the following:

- goals of production,
- capabilities of the process,
- impact on down stream units, and
- desires of management.

9.2 Popular Performance Criteria

There are popular measures or criteria used in industrial practice to describe closed loop performance. This permits an orderly comparison of different process response shapes. As shown in Fig. 9.1, certain features of the closed loop response are assigned the definitions:

A = Size of the set point step
B′ = Size of the first peak above the new set point
B = Size of the first peak above the new steady state
C = Size of the second peak above the new steady state

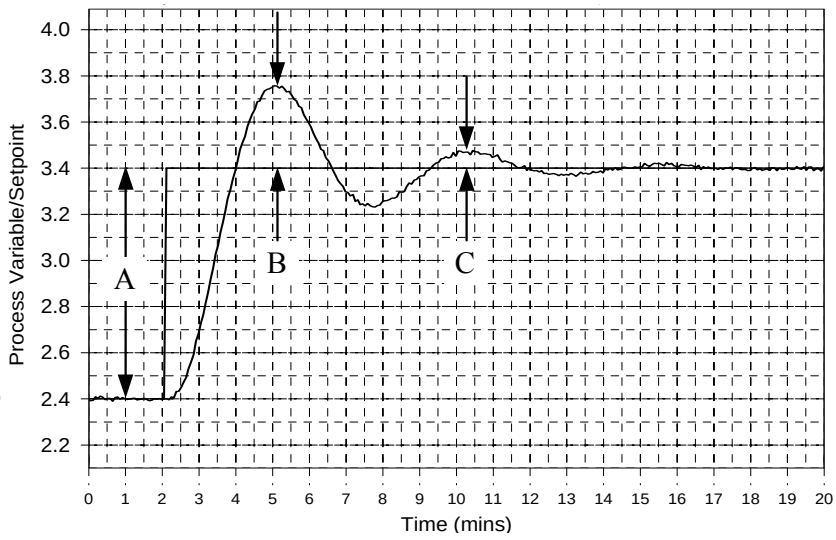


Figure 9.1 - Process response to a set point change with labels indicating response features

And as shown in Fig. 9.2, the time of occurrence of certain events, such as the time when the measured process variable first crosses the new set point or reaches its first peak are also used to describe controller performance.

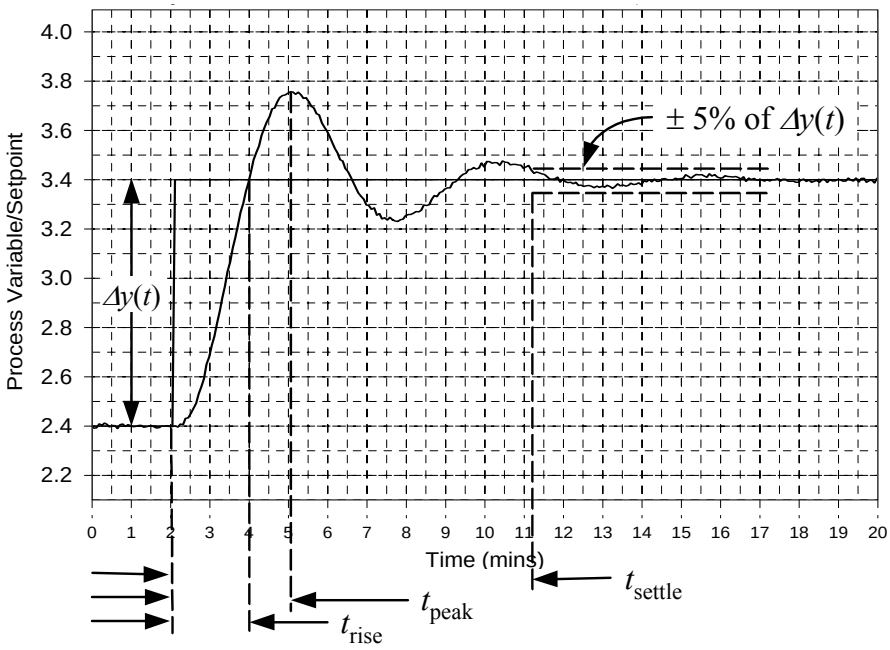


Figure 9.2 - Process response to a set point change with labels indicating response features

Another popular criteria is settling time, or the time required for the measured process variable to first enter and then remain within a band whose width is computed as ± 5% (or 3 % or 10 %) of the total change in $y(t)$, labeled as $\Delta y(t)$ in Fig. 9.2. These popular performance criteria are summarized:

Peak Overshoot Ratio (POR) = B′/A

Decay Ratio = C/B

Rise Time = t_{rise}

Peak Time = t_{peak}

Settling Time = t_{settle}

Popular values for these criteria include a 10% peak overshoot ratio, a 25% decay ratio and a settling time band of ± 5%. Also, these criteria are not independent. A process with a large decay ratio will likely have a long settling time. A process with a long rise time will likely have a long peak time.

Example: Gravity Drained Tanks

As long as a closed loop response looks similar to the classical pattern depicted in Fig. 9.1 and 9.2, then computing the performance criteria is straightforward. Because of offset, the P-Only set point response of the gravity drained tanks shown in Fig. 9.3 does not possess this classic pattern.

Here, the set point is stepped up from 2.4 m to 4.4 m. The P-Only controller gain is $K_C = 40$ m/% and the controller bias is $u_{bias} = 55.2$ %. Because of offset, the measured process variable does not settle at the set point, but rather settles at a final steady state of about 4.0 m.

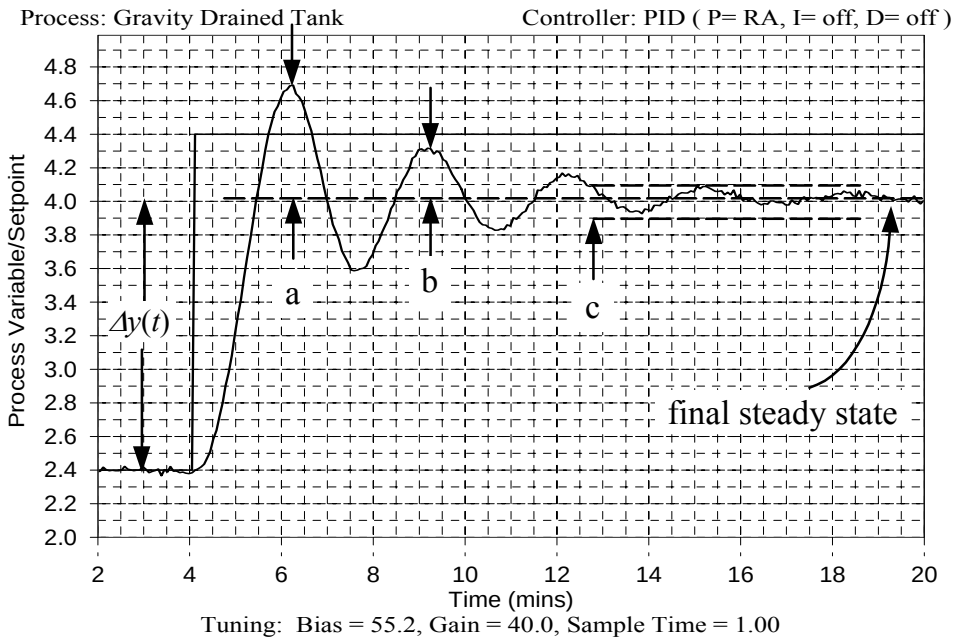


Figure 9.3 - Performance of Gravity Drained Tanks under P-Only control

From Fig. 9.3, we read the following information:

- the set point step occurs at about time t = 4.0 min
- the measured process variable first crosses the new set point at about time t = 5.7 min
- the first process variable peak ("a" on plot) occurs at about time t = 6.2 min

- The steady state change in y(t), or $\Delta y(t)$, is (4.0 – 2.4 m) = 1.6 m
- The 5% settling time band is (2.4 + 0.95(1.6 m)) = 3.9 m, and (2.4 + 1.05(1.6 m)) = 4.1 m
- the process variable first enters and then remains within the settling time band ("c" on plot) at about time t = 12.7 min

- Size of set point step is (4.4 – 2.4 m) = 2.0 m
- Size of first peak over new set point is (4.7 – 4.4 m) = 0.3 m
- Size of first peak ("a" on plot) over new steady state is (4.7 – 4.0 m) = 0.7 m
- Size of second peak ("b" on plot) over new steady state is (4.3 – 4.0) = 0.3 m

Using this data, the performance criteria are then computed as such:

- Peak Overshoot Ratio (POR) = (0.3/2.0)·100% = 15%
- Decay Ratio = (0.3/0.7)·100% = 43%
- Rise Time = (5.7 – 4.0 min) = 1.7 min
- Peak Time = (6.2 – 4.0 min) = 2.2 min
- Settling Time = (12.8 – 4.0 min) = 8.8 min

As discussed at the beginning of this chapter, whether or not these criteria qualify the controller performance as "good" is subjective and is left to the opinion of the designer.

☆ ☆ ☆

10. Derivative Action, Derivative Filtering and PID Control

10.1 Ideal and Non-interacting Forms of the PID Controller

Like the PI controller, PID computes an output signal to the final control element based on tuning parameters and the controller error, $e(t)$. As before, $u(t)$ is the controller output and u_{bias} is the controller bias. Controller gain, K_C, and reset time, τ_I, are tuning parameters. A new tuning parameter, τ_D, provides weight to the derivative term, has units of time (so it is always positive), and is called the *derivative time*. Larger values of τ_D provide a larger weighting to (increase the influence of) the derivative term.

Vendors offer different forms of the PID algorithm. We explore here the two most popular forms and learn that they are identical in capability, but users must use caution because they require slightly different correlations for tuning. Because there are three tuning parameters, these forms are commonly called *three mode* controllers.

Ideal (Non-interacting) PID Form

One form is referred to under a variety of names including the *ideal, non-interacting* and *ISA* algorithm. We will henceforth call this the ideal form. It is expressed in Eq. 10.1:

$$u(t) = u_{\text{bias}} + K_C\, e(t) + \frac{K_C}{\tau_I}\int e(t)dt + K_C\tau_D\,\frac{de(t)}{dt} \tag{10.1}$$

The first three terms to the right of the equal sign in Eq. 10.1 are identical to the PI controller of Eq. 8.1. The derivative mode of the PID controller is an additional and separate term added to the end of the equation that considers the derivative, or rate of change, of the error as it varies over time.

We show the Laplace transfer function form because it helps us understand how the names for the alogrithms evolved. The Laplace form of the ideal, non-interacting PID controller is expressed in Eq. 10.2:

$$G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s} + \tau_D s\right) \tag{10.2}$$

As is evident in Eq. 10.1 and Eq. 10.2, each term sits alone and does not interact with any of the other terms in the algorithm.

Interacting PID Form

The other PID form is referred to interchangeably as the *interacting*, *series* and *industrial* form. We will refer to this as the interacting PID form. It is expressed as follows:

$$u(t) = u_{\text{bias}} + \left(K_C + \frac{K_C\tau_D}{\tau_I}\right)e(t) + \frac{K_C}{\tau_I}\int e(t)dt + K_C\tau_D\,\frac{de(t)}{dt} \tag{10.3}$$

The Laplace form of the interacting PID algorithm reveals why it is so named. As Eq. 10.4 shows, the derivative action is introduced as a separate term that is multiplied across the PI terms.

$$G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s}\right)(\tau_D s + 1) \tag{10.4}$$

The impact of this construction seems to fall on the proportional action when expressed in the time domain of Eq. 10.3 because the proportional term is multiplied by a combination of all three tuning parameters. As we show later in this chapter, however, *all three* tuning parameters, K_C, τ_I, and τ_D, require slightly different values to make the performance of the ideal and noninteracting forms equivalent.

10.2 Function of the Derivative Term

As we have discussed, the *proportional term considers how far* $y(t)$ is from $y_{setpoint}$ at any instant in time. It adds or subtracts from u_{bias} based on the size of $e(t)$ only at time t. As $e(t)$ grows or shrinks, the amount added to u_{bias} grows or shrinks immediately and proportionately.

The *integral term addresses how long* and how far $y(t)$ has been away from $y_{setpoint}$. The integral term integrates or continually sums up $e(t)$ over time. Thus, even a small error, if it persists, will have a sum total that grows over time and the integral term contribution added to u_{bias} will similarly grow.

The *derivative term considers how fast* $e(t)$ is changing at an instant in time. The derivative computation yields a rate of change or slope of the error curve. An error that is changing rapidly yields a large derivative regardless of whether a dynamic event has just begun or if it has been underway for some time. A large derivative (a rapidly changing error) causes the derivative term to have a large impact on the value added to u_{bias}, and thus the controller output, $u(t)$. It is interesting to note that the derivative computation does not consider whether $e(t)$ is positive or negative, just whether it is changing rapidly.

10.3 Derivative on Measurement is Used in Practice

Before exploring the contribution of the derivative term to the action of a PID controller, consider that if $y_{setpoint}$ is constant, then:

$$\frac{de(t)}{dt} = \frac{d\left[y_{setpoint} - y(t)\right]}{dt} = -\frac{dy(t)}{dt} \tag{10.5}$$

That is, except for a negative sign, Eq. 10.5 shows that the derivative (or slope or rate of change) of the controller error is the same as the derivative (or slope or rate of change) of the measured process variable, $y(t)$.

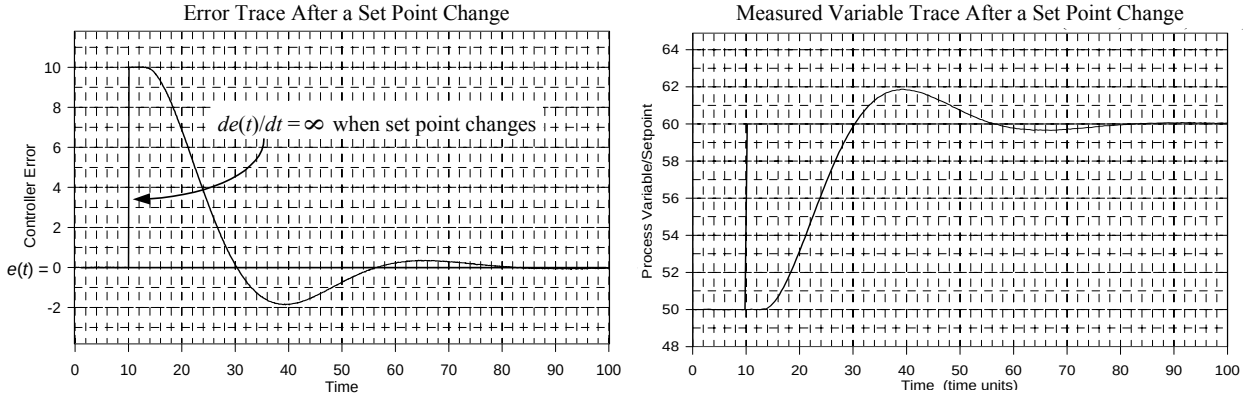


Figure 10.1 – The trace of e(t) and y(t) are reflections except when set point changes

Figure 10.1 provides a visual appreciation of this by showing a set point response curve two ways. The left plot shows the error trace, $e(t)$, after a set point step. The right plot shows the measured process variable trace, $y(t)$, for the same event. After the set point step, the changing slope of $e(t)$ identically tracks the slope of $y(t)$ except that they are reflections of one another (the derivatives have opposite signs).

The big difference is that the left plot in Fig. 10.1 shows a momentary vertical spike in $e(t)$ at the instant that y_{setpoint} changes. The derivative or slope of this spike in error, $de(t)/dt$, in theory approaches infinity and in the real world will be at least a very large value. If τ_D is large enough to provide any meaningful weight to the derivative term, this very large derivative value will cause a large and sudden manipulation in $u(t)$. This large manipulation, referred to as *derivative kick*, is almost always undesirable.

From the $y(t)$ trace to the right in Fig. 10.1, we see that $y(t)$ itself does not go through this dramatic vertical change in slope. Though set points can change instantly causing the computed error to change in kind, physical processes change in a more gradual and continuous fashion. Thus, $y(t)$ will not display dramatic vertical changes in slope when y_{setpoint} changes , but will trace a gradual and continuous dynamic (assuming that loop sample time is $T \leq 0.1\tau_P$ to adequately track the process behavior).

Because derivative on error behaves identically to derivative on measurement at all times except at those moments when y_{setpoint} changes, and when the set point does change, the derivative on error provides information we don't want our controller to use, we substitute Eq. 10.5 into Eq. 10.1 and Eq. 10.3 to obtain the *PID with derivative on measurement* controller.

Ideal (Non-interacting) PID with Derivative on Measurement:

$$u(t) = u_{\text{bias}} + K_C\, e(t) + \frac{K_C}{\tau_I}\int e(t)dt - K_C\tau_D\,\frac{dy(t)}{dt} \tag{10.6}$$

Interacting PID with Derivative on Measurement:

$$u(t) = u_{\text{bias}} + \left(K_C + \frac{K_C\tau_D}{\tau_I}\right)e(t) + \frac{K_C}{\tau_I}\int e(t)dt - K_C\tau_D\,\frac{dy(t)}{dt} \tag{10.7}$$

Because the performance of Eq. 10.6 and Eq.10.7 is the same as Eq. 10.1 and Eq. 10.2 respectively except they do not "kick" when the set point changes, *PID with derivative on measurement is the preferred algorithm in practical applications.*

10.4 Understanding Derivative Action

The derivative $dy(t)/dt$, as illustrated in Fig. 10.2, is computed as the slope of $y(t)$. During a set point response transient, the slope is large and positive when the measured process variable trace is increasing. When $y(t)$ is decreasing, the derivative (slope) is negative. And when the measurement goes through a peak or a trough, there is a moment in time when the derivative is zero.

For discussion purposes, assume that K_C is positive and that τ_D (always positive) is large enough to provide meaningful weight to the derivative term (after all, if τ_D is zero, the derivative term has no influence regardless of the slope of the measured process variable). The negative sign in front of the derivative term in Eq. 10.6 and Eq. 10.7 means that the control action contribution from the derivative will be opposite to the sign of the slope. Thus, when $dy(t)/dt$ is large and positive, the derivative term has a large influence and seeks to decrease the amount added to u_{bias}. Conversely, when $dy(t)/dt$ is negative, the derivative term seeks to increase the amount added to u_{bias}.

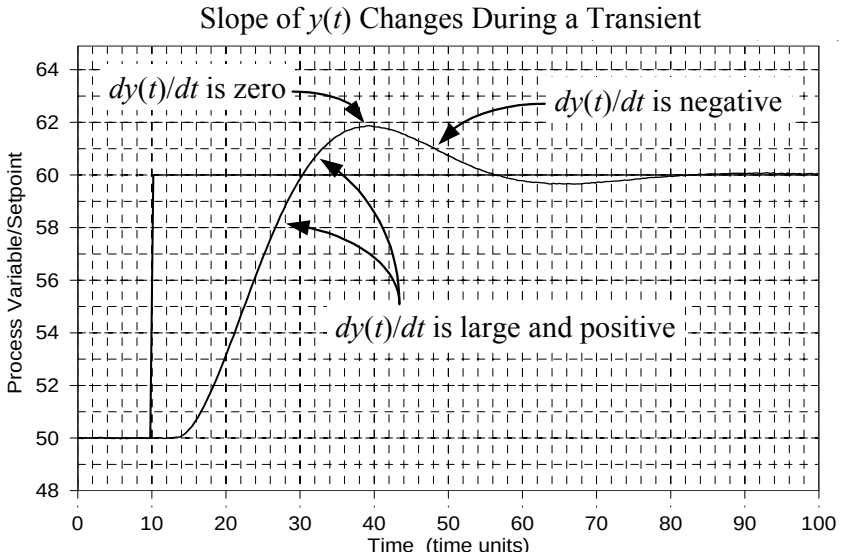


Figure 10.2 - A set point transient produces y(t) slopes that are positive, negative or zero

It is interesting to note that the derivative term does not consider whether the measurement is heading toward or away from the set point (whether $e(t)$ is positive or negative). The only consideration is whether the measurement is heading up or down and how quickly. At the peak of the transient in Fig. 10.2, $dy(t)/dt = 0$ and the derivative term momentarily makes no contribution to control action. The proportional and integral terms definitely influence $u(t)$ at that point in time, however.

10.5 Advantages and Disadvantages of PID Control

As we have discussed in previous chapters, offset occurs in most processes under P-Only control when the set point and/or disturbances are at values other than that used in the controller design, or more specifically, used to determine u_{bias}. The benefit of the PI controller is that integral action works to eliminate offset because as long as there is any error, the integral term will grow or shrink in influence and cause $u(t)$ to change. The integration ceases only when $y(t)$ equals $y_{setpoint}$ for a sustained period of time. A disadvantage of the integral term is that it increases the oscillatory or rolling behavior of $y(t)$. Also, the two tuning parameters of the PI controller interact in their influence and it is sometimes difficult to determine what action to take if controller performance is not as desired.

The big benefit of the derivative term is that it works to decrease oscillations in y(t) because it has its largest influence when $y(t)$ is changing rapidly. Thus, the three terms of a properly tuned PID controller can work together to provide rapid response to error (proportional term), eliminate offset (integral term), and minimize oscillations in the measured process variable (derivative term).

Unfortunately, there are significant disadvantages to including derivative action in your controller. We just mentioned the challenge of balancing two interacting tuning parameters for PI control. A three mode PID algorithm has *three* tuning parameters that interact and must be balanced to achieve desired controller performance. Indeed, tuning a PID controller can be quite challenging as it is often not at all obvious which of the three parameters is most responsible for an undesirable performance.

A second disadvantage relates to the uncertainty in the derivative computation for processes that have noise in the measured process variable. This problem and a solution using a derivative filter is explored in more detail later in Section 10.11.

10.6 Three Mode PID Tuning From Correlations

Designing any controller from the family of PID algorithms entails the following steps:

- specifying the design level of operation,
- collecting dynamic process data as near as practical to this design level,
- fitting a FOPDT model to the process data, and
- using the resulting model parameters in a correlation to obtain initial controller tuning values.

As always, final tuning is performed by trial and error once the controller is online so loop performance matches that desired by the control designer.

The tuning correlations available in *Design Tools* for the PID controller are the Internal Model Control (IMC) relations. These are an extension of the popular *lambda* tuning correlations and include the added sophistication of directly accounting for dead time in the tuning computations.

The first step in using the IMC (lambda) tuning correlations is to compute, τ_C , the closed loop time constant. All time constants describe the speed or quickness of a response. The closed loop time constant describes the desired speed or quickness of a controller in responding to a set point change. Hence, a small τ_C (a short response time) implies an aggressive or quickly responding controller.

LOOP-PRO's *Design Tools* automatically computes a τ_C for moderate tuning, which can be changed to more aggressive or conservative values using software options.Moderate tuning provides a set point response that, while reasonably energetic, shows little or no overshoot. The closed loop time constants are computed using Eq. 10.8:

Aggressive Tuning:	τ_C is the larger of $0.1\,\tau_P$ or $0.8\,\theta_P$	(10.8a)
Moderate Tuning:	τ_C is the larger of $1.0\,\tau_P$ or $8.0\,\theta_P$	(10.8b)
Conservative Tuning:	τ_C is the larger of $10\,\tau_P$ or $80\,\theta_P$	(10.8c)

With τ_C computed, three mode PID correlations for IMC (lambda) tuning are as follows:

	K_C	τ_I	τ_D	
PID Ideal	$\frac{1}{K_P}\left(\frac{\tau_P + 0.5\theta_P}{\tau_C + 0.5\theta_P}\right)$	$\tau_P + 0.5\theta_P$	$\frac{\tau_P \theta_P}{2\tau_P + \theta_P}$	(10.9)
PID Interacting	$\frac{1}{K_P}\left(\frac{\tau_P}{\tau_C + 0.5\theta_P}\right)$	τ_P	$0.5\theta_P$	(10.10)

As mentioned earlier, all three tuning parameters are different for the ideal (non-interacting) and interacting PID forms. Thus, *it is essential that you know the PID form used by your manufacturer* when computing tuning values for a real implementation.

Like with all tuning correlations, your judgment is required for final tuning. Remember that the most conservative (least aggressive) controller uses a smaller gain, larger reset time and smaller derivative time.

10.7 Converting From Interacting PID to Ideal PID

By setting the individual terms of the ideal PID form of Eq. 10.1 (or Eq. 10.6) to those of the interacting PID form of Eq. 10.3 (or Eq. 10.7) and solving for the ideal tuning parameters yields conversion relations between the two forms as such:

$$K_{C,\text{Ideal}} = K_{C,\text{Interact}}\left(1+\frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right) \tag{10.11a}$$

$$\tau_{I,\text{Ideal}} = \tau_{I,\text{Interact}}\left(1+\frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right) \tag{10.11b}$$

$$\tau_{D,\text{Ideal}} = \frac{\tau_{D,\text{Interact}}}{\left(1+\frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right)} \tag{10.11c}$$

These conversion relations indicate that the ideal and non-interacting PID forms will provide identical performance if the values of the individual tuning parameters are properly specified. And thus it is not surprising that by applying Eq. 10.11 to the interacting PID tuning correlations of Eq. 10.10 indeed yields the ideal PID correlations of Eq. 10.9.

10.8 Exploring Set Point Tracking Using PID Control

Building on the PI control set point tracking study for the gravity drained tanks presented in Chapter 8, we recall that the design level of operation is a measured level in the lower tank of 2.4 m while the pumped flow disturbance is at its expected value of 2.0 L/min. The control objective is to track set point steps in the range of 2.0 to 2.8 m. As detailed in Section 8.6, FOPDT dynamic model parameters fit to controller output driven data that was collected at the design level of operation are as follows:

Process Gain, $K_P = 0.094$ m/%

Time Constant, $\tau_P = 1.6$ min

Dead Time, $\theta_P = 0.56$ min

We first compute the closed loop time constant. Here we choose aggressive tuning:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = \text{larger of } 0.1(1.6) \text{ or } 0.8(0.56) = 0.45 \text{ min.}$$

Substituting this closed loop time constant and the above FOPDT model parameters into the correlations of Eq. 10.9 and Eq. 10.10 yields the following tuning values:

PID Ideal	$K_C = 27.5$ %/m	$\tau_I = 1.9$ min	$\tau_D = 0.24$ min
PID Interacting	$K_C = 23.5$ %/m	$\tau_I = 1.6$ min	$\tau_D = 0.28$ min

The performance of these PID controllers are compared in Fig. 10.3 to an IMC tuned PI controller as presented in Chapter 8. Those tuning values were as follows:

PI control	$K_C = 16.9$ %/m	$\tau_I = 1.6$ min

The result of the set point tracking comparison is shown in Fig. 10.3 (note that the "Advanced" box must be checked in LOOP-PRO's controller design menu to access the interacting PID form).

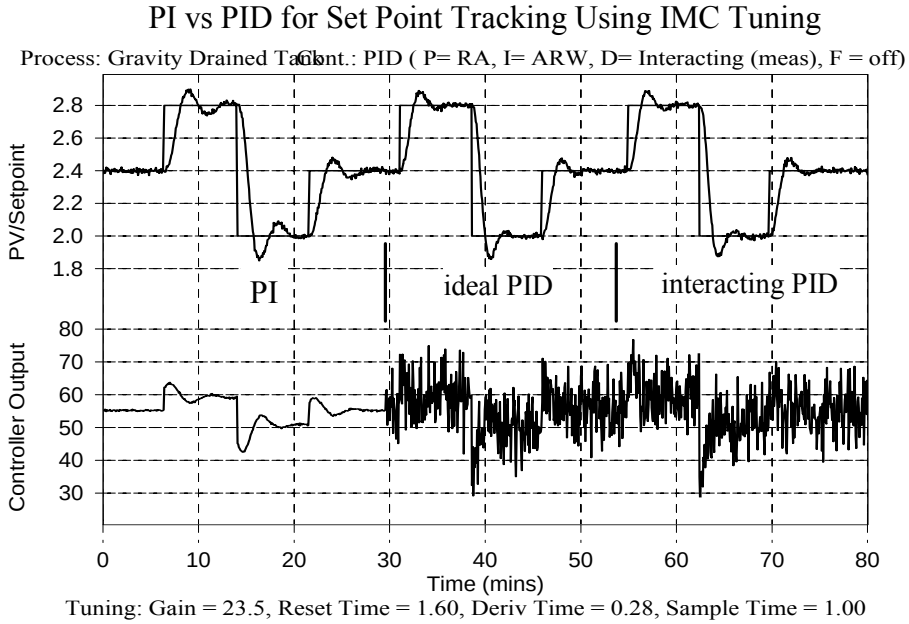


Figure 10.3 – Comparison of PI and PID controller performance in tracking set point steps

The most obvious difference between PI and PID control is that derivative action causes the noise in the measured process variable to be amplified and reflected in the controller output signal. Such extreme control action will wear a mechanical final control element, requiring continued and expensive maintenance.

Closer scrutiny of Fig. 10.3 reveals that derivative action impacts the oscillatory behavior of the measured process variable. Specifically, the PID controller not only achieves a faster rise time (because K_C is bigger), but also a smaller peak overshoot ratio and faster settling time because of derivative action.

Finally, Fig. 10.3 establishes that the ideal PID and interacting PID algorithms can produce identical control performance. Recall that each form was tuned in this example using its own IMC tuning correlations of Eq.10.9 and Eq. 10.10.

10.9 Derivative Action Dampens Oscillations

In Fig. 10.4 we further explore this observation that derivative action dampens oscillations in the measured process variable. To facilitate this study, measurement noise is set to zero to eliminate its influence on the control signal. This lets us isolate derivative time, τ_D, as a tuning parameter and study its impact on set point tracking performance.

The middle plot in Fig. 10.4 shows the set point tracking performance of a PID controller tuned using IMC tuning and the standard τ_C value of Eq. 10.8. For all three plots, K_C and τ_I remain constant. The plot to the left shows how the oscillating nature of the response increases as derivative action is decreased by half. The plot to the right shows that when derivative action is too large, it inhibits rapid movement in the measure process variable, causing the rise time and settling time to lengthen.

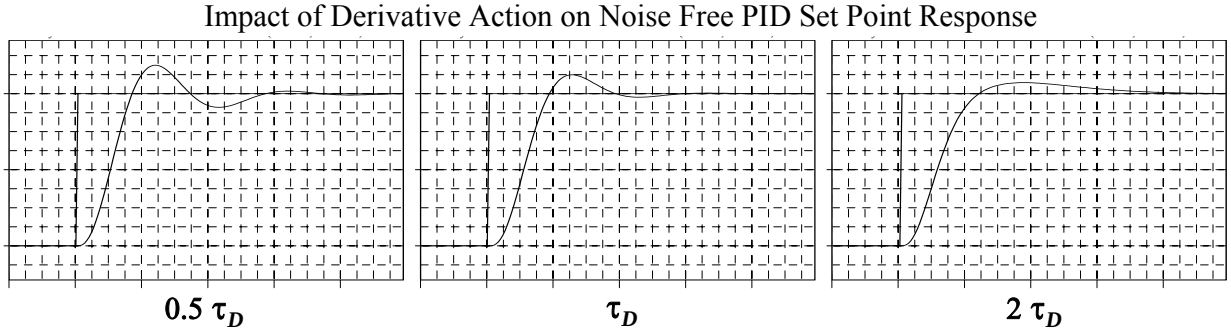


Figure 10.4 - How derivative time, τ_D, impacts PID set point tracking performance

10.10 Measurement Noise Hurts Derivative Action

We noted in Fig. 10.3 that derivative action causes measurement noise to be amplified and reflected in the controller output signal. The reason for this extreme control action, as illustrated in Fig. 10.5, is that a noisy signal produces conflicting derivatives as the slope appears to dramatically alternate direction at every sample.

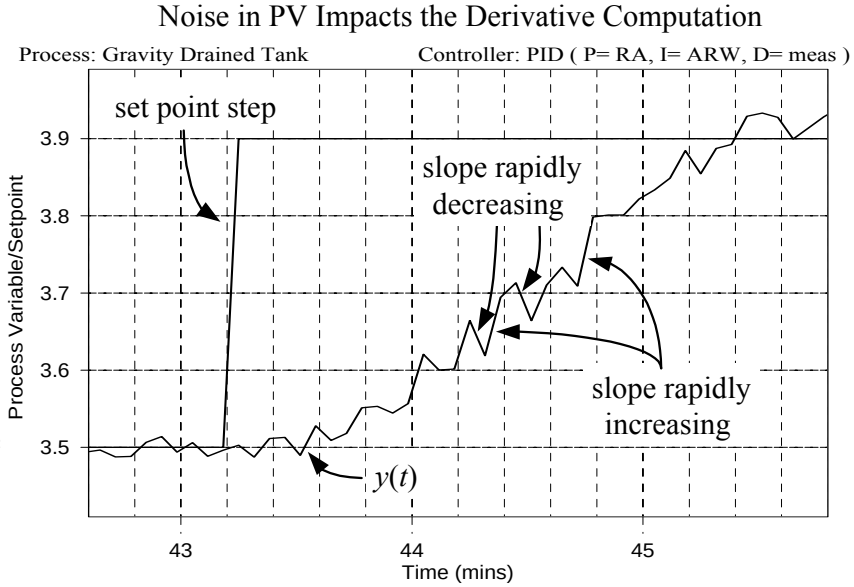


Figure 10.5 – Measured process variable noise causes uncertainty in the derivative computation

As noted in Fig. 10.5, the derivative can alternate between a large increasing slope followed by a large decreasing slope, sample after sample. The derivative mode thus reflects this noise as it computes a series of alternating and compensating controller actions. The degree to which the noise is amplified in the controller output depends on the size of τ_D.

Figure 10.6 illustrates an additional problem that measurement noise can cause with derivative action when the level of operation is near a controller output constraint (either the maximum value, u_{max}, or the minimum value, u_{min}). The PID tuning values in Fig. 10.6 are constant throughout the experiment. As indicated on the plot, measurement noise is increased from small to medium to large in three set point tracking tests. (The terms "small" and "large" are used to indicate relative levels of noise. Please do not use this plot as a standard for determining if noise in your process is small or large.)

As long as measurement noise causes the derivative to alternate equally between suddenly increasing and suddenly decreasing, and the controller output can reflect this "equality in randomness" unimpeded, then controller performance is reasonably consistent in spite of significant noise. A comparison of the small noise and medium noise cases shows the controller output is not hitting a constraint and set point performance is consistent.

If a constraint inhibits the controller output from the "equality in randomness" symmetry, causing it to become skewed or off center, then controller performance degrades. This is demonstrated in Fig. 10.6 in the right most set point steps for the case where measurement noise is large. For this case, the controller output signal becomes so active that it hits the maximum controller output value, u_{max}. By constraining $u(t)$, the controller output loses its symmetry and controller performance is affected as the measured process variable temporarily deviates from set point.

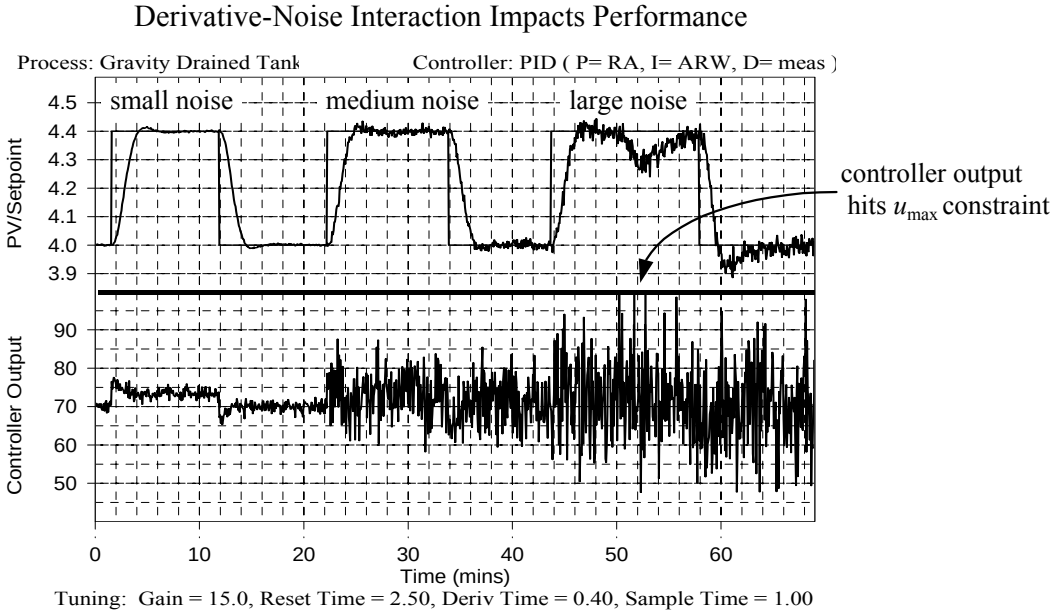


Figure 10.6 – When controller output becomes constrained, controller performance degrades

10.11 PID With Derivative Filter Reduces the Impact of Noise

To improve performance when there is noise or random error in the measured process variable, the PID control algorithm can be modified with the addition of a derivative filter. The filter works to limit the large controller output moves that derivative action causes as a result of measurement noise. Even if noise is not causing performance problems, the derivative filter can help reduce the constant controller output fluctuations that can lead to wear in the final control element.

As with three mode PID, vendors offer different forms of a four mode PID with derivative filter controller. We explore the two most popular forms here. While not shown, the controllers are typically implemented in a "derivative on measurement" form analogous to Eq. 10.6 and Eq. 10.7.

Ideal (Non-interacting) PID with Filter Form

The *ideal* or *non-interacting* PID with derivative filter algorithm is expressed as:

$$u(t) = u_{\text{bias}} + K_C e(t) + \frac{K_C}{\tau_I}\int e(t)dt + K_C\tau_D\frac{de(t)}{dt} - \alpha\tau_D\frac{du(t)}{dt} \qquad (10.12)$$

The first four terms on the right side of Eq. 10.12 are identical to the PID controller of Eq. 10.1. The last term on the right side of the equation is the filter term that subtracts a derivative or rate of change in the controller output, $u(t)$, from the four PID terms.

When the first four PID terms compute large and sudden changes in $u(t)$ as might arise from noise in the process variable (see Fig. 10.5), then the rate of change (derivative) of the controller output will also become large. This controller output derivative is scaled by $\alpha\tau_D$ and subtracted from the computation as shown in Eq. 10.12. If the derivative of $u(t)$ is large, then in effect, the final computed controller output change is moderated (or filtered). The actual change in $u(t)$ sent to the final control element is smaller than it otherwise would be with the traditional three mode PID form. The size of the filter constant, α, dictates how much of each change is filtered out of the final $u(t)$ signal.

The Laplace form of the ideal, non-interacting PID with derivative filter controller is shown in Eq. 10.13:

$$G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)\left(\frac{1}{\alpha\tau_D s + 1}\right) \qquad (10.13)$$

This form shows that the filter term is multiplied across, and thus affects the computations of, all terms of the three mode PID algorithm.

Interacting PID with Filter Form

The *interacting* or *series* form of the four mode PID with derivative filter form is expressed as follows:

$$u(t) = u_{\text{bias}} + \left(K_C + \frac{K_C\tau_D}{\tau_I}\right)e(t) + \frac{K_C}{\tau_I}\int e(t)dt + K_C\tau_D\frac{de(t)}{dt} - \alpha\tau_D\frac{du(t)}{dt} \qquad (10.14)$$

The same filtering computation just discussed for the ideal form applies here. The Laplace form of the interacting PID with derivative filter algorithm is expressed as follows:

$$G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s}\right)\left(\frac{\tau_D s + 1}{\alpha\tau_D s + 1}\right) \qquad (10.15)$$

The interacting form is also commonly referred to as the "*lead/lag*" form.

10.12 Four Mode PID Tuning From Correlations

The derivative filter constant, α, becomes a fourth tuning parameter, creating a yet more challenging tuning problem. The tuning correlations available in *Design Tools* are the IMC (lambda) relations that use the closed loop time constant,τ_C.. Again, as computed in Eq. 10.8, LOOP-PRO offers both "standard" and "conservative" τ_C values. With τ_C computed, four mode PID correlations for IMC (lambda) tuning are as follows:

	K_C	τ_I	τ_D	α	
PID Ideal w/filter	$\frac{1}{K_P}\left(\frac{\tau_P + 0.5\theta_P}{\tau_C + \theta_P}\right)$	$\tau_P + 0.5\theta_P$	$\frac{\tau_P \theta_P}{2\tau_P + \theta_P}$	$\frac{\tau_C(\tau_P + 0.5\theta_P)}{\tau_P(\tau_C + \theta_P)}$	(10.16)
PID Interacting w/filter	$\frac{1}{K_P}\left(\frac{\tau_P}{\tau_C + \theta_P}\right)$	τ_P	$0.5\theta_P$	$\frac{\tau_C}{\tau_C + \theta_P}$	(10.17)

All four tuning parameters are different for the ideal (non-interacting) and interacting PID forms. As stated before, *know the PID form used by your manufacturer* when computing tuning values for a real implementation. And again, as with all tuning correlations, your judgment is required for final tuning.

10.13 Converting From Interacting PID with Filter to Ideal PID with Filter

Setting the individual terms of the ideal four mode PID form of Eq. 10.12 to those of the interacting form of Eq. 10.14 and solving for the ideal tuning parameters yields conversion relations between the two forms, shown in Eq. 10.18:

$$K_{C,\text{Ideal}} = K_{C,\text{Interact}}\left(1 + \frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right) \tag{10.18a}$$

$$\tau_{I,\text{Ideal}} = \tau_{I,\text{Interact}}\left(1 + \frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right) \tag{10.18b}$$

$$\tau_{D,\text{Ideal}} = \frac{\tau_{D,\text{Interact}}}{\left(1 + \frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right)} \tag{10.18c}$$

$$\alpha_{\text{Ideal}} = \alpha_{\text{Interact}}\left(1 + \frac{\tau_{D,\text{Interact}}}{\tau_{I,\text{Interact}}}\right) \tag{10.18d}$$

Based on these conversion relations, we conclude as we did with the three mode forms that the four mode ideal and non-interacting PID controllers will provide identical performance if the values of the individual tuning parameters are properly specified. And applying Eq. 10.18 to the interacting PID tuning correlations of Eq. 10.17 indeed yields the ideal PID correlations of Eq. 10.16.

10.14 Exploring Set Point Tracking Using PID with Derivative Filter Control

We repeat the set point tracking study for the gravity drained tanks presented in section 10.8. Again,

Process Gain, $K_P = 0.094$ m/%

Time Constant, $\tau_P = 1.6$ min

Dead Time, $\theta_P = 0.56$ min

and the closed loop time constant using aggressive tuning:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = 0.45 \text{ min.}$$

Substituting this closed loop time constant and the above FOPDT model parameters into the correlations of Eq. 10.16 and Eq. 10.17 yields the following tuning values:

PID Ideal	K_C = 19.8 %/m	τ_I = 1.9 min	τ_D = 0.24 min	α = 0.52
PID Interacting	K_C = 16.9 %/m	τ_I = 1.6 min	τ_D = 0.28 min	α = 0.44

The performance of these PID controllers are compared in Fig. 10.3 to an IMC tuned PI controller as presented in Chapter 8. Those tuning values were as follows:

PI control	K_C = 16.9 %/m	τ_I = 1.6 min

The result of the set point tracking comparison is shown in Fig. 10.7 (note that the "Advanced" box must be checked in LOOP-PRO's controller design menu to access the interacting PID form and the derivative filter option).

Comparing the filtered PID of Fig. 10.7 to the unfiltered results of Fig. 10.3 reveals the dramatic capability of the filter to temper the actions of the controller output signal. The ability of derivative action to achieve a faster rise time, smaller peak overshoot ratio and faster settling time is similar to that shown in Fig 10.3, implying that the filter does not necessarily impact performance.

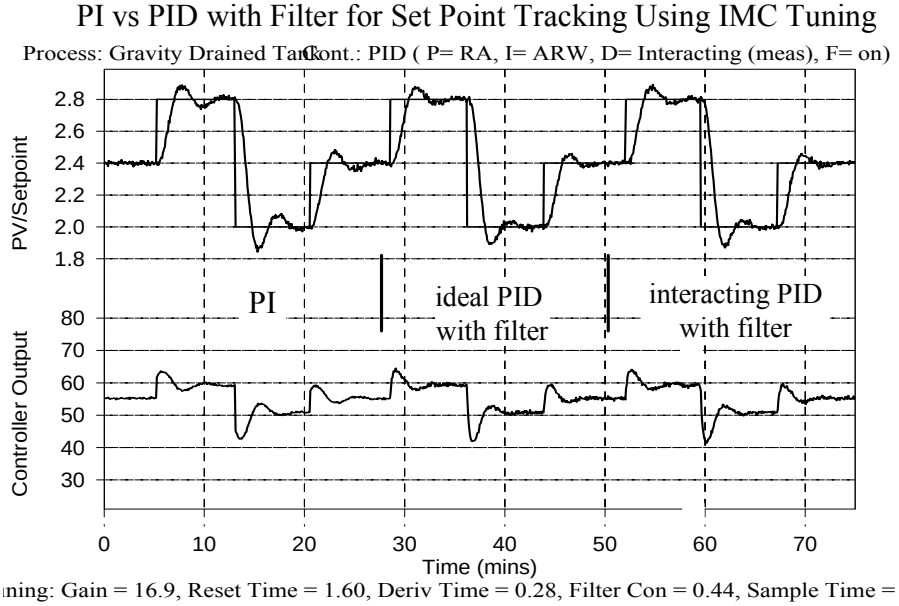


Figure 10.7 – Comparing PI and PID with derivative filter in tracking set point steps

On the other hand, because the derivative filter works to limit sudden changes in the controller output signal, a large filter can reduce the benefits of derivative action. Ideally, the filter constant should be just large enough to contain the erratic fluctuations in the controller output signal, but not so much as to degrade the overall performance of the controller.

Finally, Fig. 10.7 reinforces the observation of Fig. 10.3 that the ideal PID with filter and interacting PID with filter algorithms can produce identical control performance. Recall that each form was tuned in this example using its own IMC tuning correlations of Eq.10.16 and Eq. 10.17.

11. First Principles Modeling of Process Dynamics

11.1 Empirical and Theoretical Dynamic Models

Dynamic models describe how the behavior of a process changes with time. Empirical dynamic models are essentially curve fits of observed process behavior, and as such, they can be developed relatively quickly. To create an empirical model, a general dynamic model form is fit to process data collected during an experimental study. We did this frequently in previous chapters when we fit the first order plus dead time (FOPDT) model to data to compute controller tuning values.

Empirical models can describe dynamic process data quite accurately. The disadvantage of empirical modeling is that because it is a data fit and not a process description, extrapolation beyond the bounds of the data used to fit the model is speculative. The mismatch between model predictions and actual process behavior will likely increase as the extrapolation increases.

Theoretical dynamic models are derived from first principles. Because they are based on theory, they often can be extrapolated beyond the bounds of available data with some confidence. This makes them useful for exploring a wide range of operating policies and control strategies. Studies using theoretical models can be performed quickly, safely and inexpensively. They can also form the basis of a training simulator. For example, LOOP-PRO's *Case Studies* simulations are based on theoretical models. The disadvantage of theoretical models is that they can be substantially more challenging and time consuming to develop compared to empirical models.

11.2 Conserved Variables and Conservation Equations

First-principles (theoretical) dynamic models result from conservation equations. Conserved variables include:

- mass
- mass of component *i* (a species balance)
- energy
- momentum

Balances are created by defining a boundary around the process and then computing the following:

$$\text{Accumulation} = \text{In} - \text{Out} + \text{Generation} - \text{Consumption}$$

Note that level, temperature and process variables other than those listed above are *not* conserved.

As developed below, a model describing the dynamic (or time-dependent) behavior of liquid level in a tank begins with a mass balance. The specific assumptions of constant liquid density and constant tank cross-sectional area enable the mass balance to yield a dynamic equation describing liquid level. Similarly, a later example shows that a dynamic temperature model begins with an energy balance.

When deriving a process model, regardless of the final application, good engineering practice dictates that we include the following in the derivation:

- a picture of the process with appropriate labels,
- units used in the model,
- assumptions used in the derivation,
- step-by-step details of the derivation,
- a final dynamic model differential equation including boundary conditions.

These organizational details help prevent mistakes while also documenting the intended use and application of your work. Also, others reviewing your work will benefit from the documentation, and you will also if you revisit your derivation at a later date.

11.3 Mass Balance on a Draining Tank

The process pictured in Fig. 11.1 is a tank that has liquid flowing in the top and freely draining out of a hole in the bottom. As the variable labels indicate, the flow rate in and out can change with time. The cross section of the tank is constant with height, so the volume of the tank at any time t can be related to the height of the tank.

Following good engineering practice, we begin with a picture, units and assumptions used, followed by step-by-step details of the derivation.

Picture:

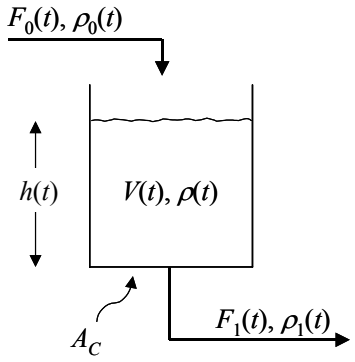


Figure 11.1 - Picture of a draining tank with process variables clearly labeled

Variables with units:

- Liquid flow rate $F(t)$ [=] m^3/s
- Liquid density $\rho(t)$ [=] Kg/m^3
- Liquid level height $h(t)$ [=] m
- Area of tank cross section A_C [=] m^2
- Liquid volume $V(t)$ [=] m^3
- Time t [=] s

Assumptions:

- the process model is restricted to the liquid volumes and flows
- liquid can enter or leave the tank only through the flow streams shown (i.e. no evaporation)
- tank cross sectional area is constant with height, or $V(t) = A_C h(t)$
- liquids are incompressible and thus density is constant, or $\rho_0(t) = \rho_1(t) = \rho(t) = \rho$

Step by step details:

Perform a mass balance on a draining tank experiment that takes time Δt to complete, as follows:

- Mass in by flow: $\overline{\rho}_0(t)\overline{F}_0(t)\Delta t$
- Mass out by flow: $\overline{\rho}_1(t)\overline{F}_1(t)\Delta t$
- Mass generation: 0
- Mass consumption: 0
- Mass accumulation in tank: $\rho(t)V(t)\big|_{t+\Delta t} - \rho(t)V(t)\big|_t$

The bars over the density and flow rate variables, $\overline{\rho}_i(t)$ and $\overline{F}_i(t)$, indicate that these values are averaged over finite time Δt.

Sum the mass balance: $$\rho(t)V(t)\big|_{t+\Delta t} - \rho(t)V(t)\big|_t = \overline{\rho}_0(t)\overline{F}_0(t)\Delta t - \overline{\rho}_1(t)\overline{F}_1(t)\Delta t$$

Divide by Δt: $$\frac{\rho(t)V(t)\big|_{t+\Delta t} - \rho(t)V(t)\big|_t}{\Delta t} = \overline{\rho}_0(t)\overline{F}_0(t) - \overline{\rho}_1(t)\overline{F}_1(t)$$

Take the limit as $\Delta t \to 0$ (as Δt approaches zero) and recognize that the left side becomes the definition of a derivative. We further recognize that the average process variables values $\overline{\rho}_i(t)$ and $\overline{F}_i(t)$ then become point values $\rho_i(t)$ and $F_i(t)$, or:

$$\frac{d\rho(t)V(t)}{dt} = \rho_0(t)F_0(t) - \rho_1(t)F_1(t)$$

We employ the assumption that liquids are incompressible and divide by density across the equation to obtain:

$$\frac{dV(t)}{dt} = F_0(t) - F_1(t)$$

Employing the assumption that the cross-sectional area of the tank is constant with respect to liquid level (liquid height), or $V(t) = A_C\,h(t)$, yields:

$$A_C\,\frac{dh(t)}{dt} = F_0(t) - F_1(t)$$

An ordinary differential equation (ODE) is incomplete without boundary conditions. For dynamic models used in process control, the most useful conditions are initial conditions, or those that define the condition of the process at time $t = 0$. Suppose we know that the initial height of the liquid is steady at h_S, or:

$$h(0) = h_S$$

Then the dynamic process model with initial conditions becomes

$$\boxed{A_C\,\frac{dh(t)}{dt} = F_0(t) - F_1(t) \qquad \text{where } h(0) = h_S} \tag{11.1}$$

Case 1: Assume Drain Flow Proportional to Hydrostatic Head

If we assume the drain flow out of the bottom of the tank is proportional to the height of the liquid in the tank (the hydrostatic head), then we can write

$$F_1(t) = \alpha_1 h(t) \qquad \text{with units } \alpha_1 \text{ [=] } m^2/s$$

Substituting this flow rate into the dynamic process model yields

$$A_C \frac{dh(t)}{dt} + \alpha_1 h(t) = F_0(t)$$

Divide by α_1 to obtain the following:

$$\frac{A_C}{\alpha_1}\frac{dh(t)}{dt} + h(t) = \frac{1}{\alpha_1}F_0(t) \qquad \text{where } h(0) = h_S \tag{11.2}$$

Compare the above to the general first order process model:

$$\tau_P \frac{dy(t)}{dt} + y(t) = K_P u(t)$$

(We assume no dead time for this model and will explore dead time as a separate issue when we introduce Laplace transforms in Chapter 14.)

We see our payoff for this modeling exercise. Specifically, we now know the steady state process gain, K_P, and the process time constant, τ_P, of the draining tank model:

$$K_P = \frac{1}{\alpha_1} \text{ [=] s/m}^2 \qquad \text{and} \qquad \tau_P = \frac{A_C}{\alpha_1} \text{ [=] s}$$

As expected, the process gain has units of $y(t)/u(t)$ and more specifically $h(t)/F_0(t)$ [=] m/(m^3/s), and the time constant has units of time.

Case 2: Assume Drain Flow Proportional to Square Root of Hydrostatic Head

Unfortunately, while Case 1 had a pleasing outcome, drain flow out of a freely draining tank follows much more closely a square root relationship with liquid height rather than the direct proportional relationship presented previously. The square root relationship is expressed as such:

$$F_1(t) = \alpha_1 \sqrt{h(t)}$$

Substituting yields the following process model:

$$A_C \frac{dh(t)}{dt} + \alpha_1 \sqrt{h(t)} = F_0(t) \qquad \text{where } h(0) = h_S \tag{11.3}$$

Comparing the above equation to the general first order process model reveals a problem – there is not a square root relationship, $\sqrt{y(t)}$, in the general first order form. Hence, we cannot directly determine the model process gain and time constant for this case by a simple comparison.

Because the dependent variable, $h(t)$, has a square root relation, this model is a nonlinear ODE. Before we apply linear control theory to this process model, we must linearize it. We discuss the method for linearizing nonlinear equations in Chapter 12.

11.4 Mass Balance on Two Draining Tanks

Consider two freely draining tanks stacked one above the other similar to the Gravity Drained Tanks case study in LOOP-PRO and as shown below in Fig. 11.2. Although the lower tank receives its feed from the upper tank, the dynamics and behavior of the lower tank have no impact on the upper tank. This arrangement is traditionally called *non-interacting* tanks in series.

Picture:

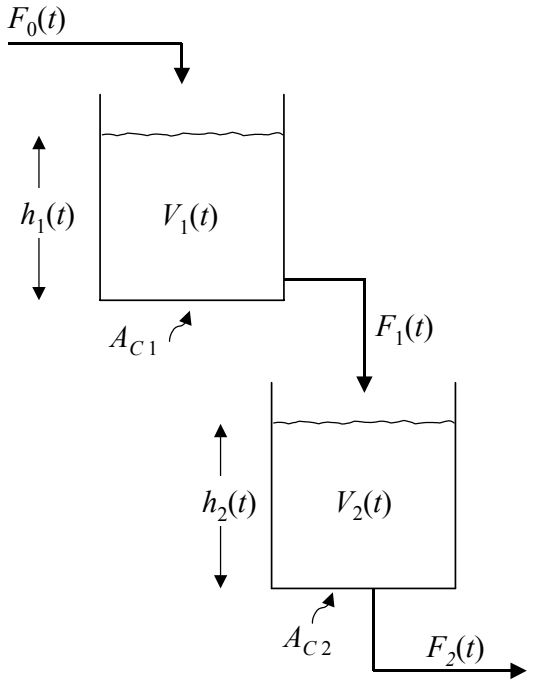


Figure 11.2 - Two non-interacting draining tanks in series with process variables labeled

Variables with units:

- Liquid flow rate $F(t)$ [=] m^3/s
- Liquid density $\rho(t)$ [=] Kg/m^3
- Liquid level height $h(t)$ [=] m
- Area of cross section A_C [=] m^2
- Liquid volume $V(t)$ [=] m^3
- Time t [=] s

Assumptions:

- the process model is restricted to the liquid volumes and flows
- liquid can enter or leave the tank only through the flow streams shown (i.e. no evaporation)
- tank cross sectional area is constant with height, or $V(t) = A_C h(t)$
- liquids are incompressible and thus density is constant, or $\rho_0(t) = \rho_1(t) = \rho(t) = \rho$

Step by step details:

Applying the mass balance derived in Section 11.3 to each tank, we obtain the following:

Tank 1: $$A_{C1}\frac{dh_1(t)}{dt} = F_0(t) - F_1(t) \qquad \text{where } h_1(0) = h_{1,S}$$

Tank 2: $$A_{C2}\frac{dh_2(t)}{dt} = F_1(t) - F_2(t) \qquad \text{where } h_2(0) = h_{2,S}$$

Applying the assumption that tank drain flow rate is proportional to the square root of the hydrostatic head, or

$$F_1(t) = \alpha_1\sqrt{h_1(t)} \qquad \text{and} \qquad F_2(t) = \alpha_2\sqrt{h_2(t)}$$

yields the following model equations:

Tank 1: $$A_{C1}\frac{dh_1(t)}{dt} + \alpha_1\sqrt{h_1(t)} = F_0(t) \qquad \text{where } h_1(0) = h_{1,S}$$

Tank 2: $$A_{C2}\frac{dh_2(t)}{dt} + \alpha_2\sqrt{h_2(t)} = \alpha_1\sqrt{h_1(t)} \qquad \text{where } h_2(0) = h_{2,S}$$

Both of these equations are nonlinear because of the square root relation for the dependent variables, $h_1(t)$ and $h_2(t)$. Also note that $h_2(t)$ does not appear in the first of these *coupled* equations, which supports the earlier statement that the dynamics of the lower tank have no effect on the upper tank.

11.5 Energy Balance on a Stirred Tank with Heater

Here we perform an energy balance on a well-stirred tank of liquid that is being heated with a steam coil. As indicated in the picture, the flow rates, temperatures and other variables can all change with time.

Picture:

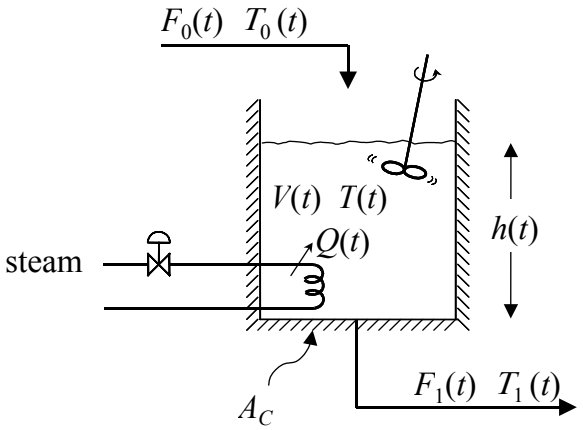


Figure 11.3 - Steam heated mixing tank with process variables labeled

Variable with units:

- Liquid flow rate $F(t)$ [=] cm^3/s
- Liquid density $\rho(t)$ [=] g/cm^3
- Liquid level height $h(t)$ [=] cm
- Area of cross section A_C [=] cm^2
- Liquid volume $V(t)$ [=] cm^3
- Energy in from steam $Q(t)$ [=] cal/s
- Liquid temperature $T(t)$ [=] °C
- Liquid heat capacity C_P [=] cal/g·°C
- Time t [=] s

Assumptions:

- A_C constant with height of liquid, i.e. $V(t) = A_C\, h(t)$
- ρ and C_P constant
- No evaporation
- No other streams enter or leave the vessel than those shown
- Vessel is perfectly insulated
- Liquid is perfectly mixed, i.e. $T_1(t) = T(t)$
- No shaft work (no W_S) from mixer
- Only energy of liquid is considered
- Enthalpy balance = Energy balance (or $\Delta PE = \Delta KE = 0$)
- Reference Temperature is zero, i.e. $T_{ref} = 0$
- No reaction occurs
- No frictional losses; no pressure drop losses

Step by step details:

Perform an energy balance on an experiment that takes time Δt to complete, as follows:

- Energy in by flow: $\rho C_P \overline{F}_0(t)\overline{T}(t)\Delta t$
- Energy out by flow: $\rho C_P \overline{F}_1(t)\overline{T}(t)\Delta t$
- Energy in from steam: $\overline{Q}(t)\Delta t$
- Energy accumulation in tank:

$$E|_{t+\Delta t} - E|_t = H|_{t+\Delta t} - H|_t = \rho A_C C_P [h(t)T(t)|_{t+\Delta t} - h(t)T(t)|_t]$$

Sum the energy balance (conserve the variable):

$$\rho A_C C_P [h(t)T(t)|_{t+\Delta t} - h(t)T(t)|_t] = \rho C_P \overline{F_0}(t)\overline{T_0}(t)\Delta t - \rho C_P \overline{F_1}(t)\overline{T}(t)\Delta t + \overline{Q}(t)\Delta t$$

Divide by Δt and take the limit as $\Delta t \to 0$. Recognize that the average values of the process variables become point values, or $\overline{F}(t) \to F(t), \quad \overline{T}(t) \to T(t), \quad \overline{Q}(t) \to Q(t)$, yielding the following process model:

$$A_C \frac{d\left[h(t)T(t)\right]}{dt} = F_0(t)T_0(t) - F_1(t)T(t) + \frac{Q(t)}{\rho C_P}$$

We can then apply the product rule to the derivative:

$$A_C h(t)\frac{dT(t)}{dt} + A_C T(t)\frac{dh(t)}{dt} = F_0(t)T_0(t) - F_1(t)T(t) + \frac{Q(t)}{\rho C_P}$$

Recall the mass balance for a freely draining tank:

$$A_C \frac{dh(t)}{dt} = F_0(t) - F_1(t)$$

Substituting the mass balance into the energy balance yields

$$A_C h(t)\frac{dT(t)}{dt} + T(t)[F_0(t) - F_1(t)] = F_0(t)T_0(t) - F_1(t)T(t) + \frac{Q(t)}{\rho C_P}$$

We can then eliminate like terms to arrive at the energy balance process model for liquid in the tank:

$$A_C h(t)\frac{dT(t)}{dt} + T(t)F_0(t) = F_0(t)T_0(t) + \frac{Q(t)}{\rho C_P} \qquad \text{where } T(0) = T_S$$

To calculate the tank height used in the energy balance, we must include the mass balance as part of the overall model:

$$A_C \frac{dh(t)}{dt} = F_0(t) - F_1(t) \qquad \text{where} \quad h(0) = h_S$$

Note that the energy balance has ***non-constant*** coefficients, so we cannot draw any conclusions by comparing it directly to the general first order process model with constant coefficients:

$$\tau_P \frac{dy}{dt} + y(t) = K_P u(t)$$

The next chapter shows how to approximate non-constant coefficients with constant coefficients using linearization techniques.

11.6 Species (Component) Balance on a Stirred Tank with Reaction

Here we perform a species (component) balance on a well-stirred tank of liquid where reaction A→B takes place. As indicated in Fig. 11.4, the flow rates, temperatures and other variables can all change with time.

Picture:

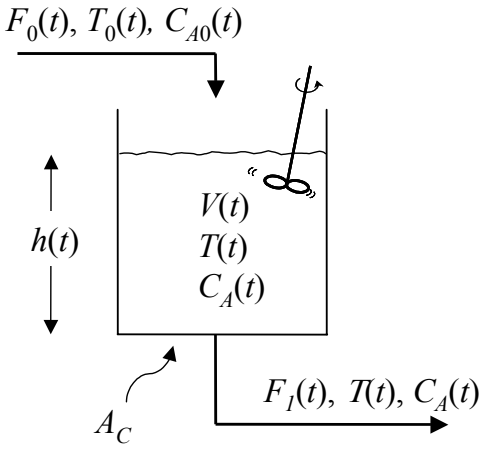


Figure 11.4 - Mixing tank with chemical reaction and process variables labeled

Variables with units (in addition to those in previous example):

- Concentration of A — C_A [=] gmol/cm^3
- Moles of A — n_A [=] gmol
- Activation energy — E [=] cal/gmol
- Gas constant — R [=] cal/gmol·K
- Reaction rate constant — k [=] 1/s
- Liquid flow rate — $F(t)$ [=] cm^3/s
- Liquid density — $\rho(t)$ [=] g/cm^3
- Liquid level height — $h(t)$ [=] cm
- Area of cross section — A_C [=] cm^2
- Liquid volume — $V(t)$ [=] cm^3
- Liquid temperature — $T(t)$ [=] K
- Liquid heat capacity — C_P [=] cal/g·K
- Rate of reaction — r [=] gmol/cm^3·s
- Time — t [=] s

Assumptions (in addition to the previous example):

- There is no heat of reaction (reaction is neither exothermic nor endothermic)
- $A \rightarrow B$ is a first order reaction, i.e. $r(t) = k(t)C_A(t)$
- Arrhenius temperature dependence on reaction rate constant, i.e. $k(t) = k_0 e^{-E/RT(t)}$

Details:

Perform a species (component) balance on an experiment that took time Δt to complete, as follows:

- Moles of A in by flow: $\overline{F}_0(t)\overline{C}_{A0}(t)\Delta t$
- Moles of A out by flow: $\overline{F}_1(t)\overline{C}_A(t)\Delta t$
- A out due to reaction: $\overline{r}(t)V(t)\Delta t$
- Accumulation of A in tank:

$$n_A(t)|_{t+\Delta t} - n_A(t)|_t = V(t)C_A(t)|_{t+\Delta t} - V(t)C_A(t)|_t = A_C[h(t)C_A(t)|_{t+\Delta t} - h(t)C_A(t)|_t]$$

Sum up the species balance and take the limit as $\Delta t \to 0$. Recognize that the average values of the process variables become point values, or $\overline{C}_A(t) \to C_A(t)$, $\overline{F}(t) \to F(t)$, $\overline{T}(t) \to T(t)$, yielding the following process model:

$$A_C \frac{d[h(t)C_A(t)]}{dt} = C_{A0}(t)F_0(t) - C_A(t)F_1(t) - A_C h(t)r(t)$$

Apply the product rule to the derivative and assume an Arrhenius temperature dependence:

$$A_C h(t)\frac{dC_A(t)}{dt} + A_C C_A(t)\frac{dh(t)}{dt} = C_{A0}(t)F_0(t) - C_A(t)F_1(t) - A_C h(t)C_A(t)k_0 e^{-E/RT(t)}$$

Substituting the mass balance into the species balance and eliminating like terms, we arrive at the following process model:

$$A_C h(t)\frac{dC_A(t)}{dt} + C_A(t)[F_0(t) + A_C h(t)k_0 e^{-E/RT(t)}] = C_{A0}(t)F_0(t) \qquad \text{where } C_A(0) = C_{A,S}$$

An energy balance on this process yields

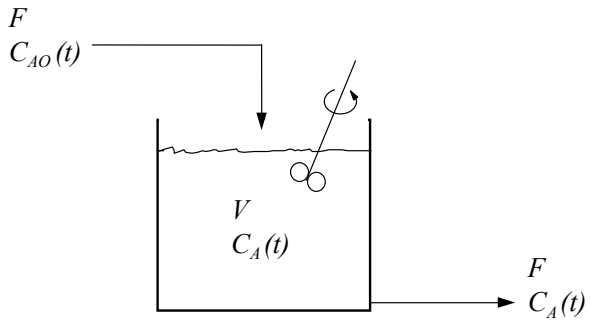
$$A_C h(t)\frac{dT(t)}{dt} + F_0(t)T(t) = F_0(t)T_0(t) \qquad \text{where } T(0)=T_S$$

A mass balance yields the following:

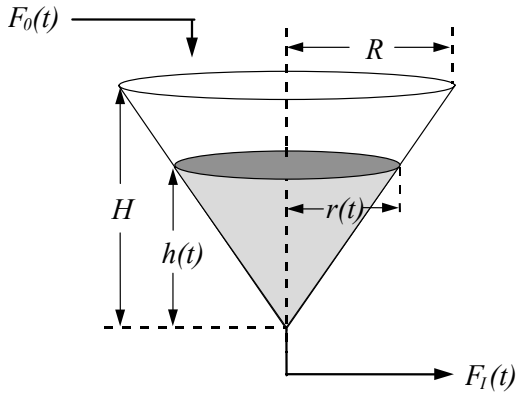
$$A_C \frac{dh(t)}{dt} = F_0(t) - F_1(t) \qquad \text{where } h(0) = h_S$$

11.7 Exercises

Q-11.1 Consider the stirred tank reactor below. Following good engineering practice, derive the steady state process gain and process time constant for this system. You should assume first order reaction kinetics, $r_A(t) = kC_A(t)$.

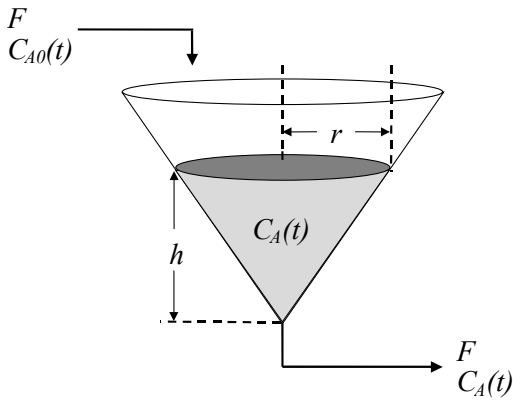


Q-11.2 A constant density fluid flows into a perfect cone tank as pictured below. Showing all steps, derive the dynamic ordinary differential equation (ODE) describing liquid height in the tank. Express your result in the form $\frac{dh(t)}{dt} = ?$



Exit flow, $F_1(t)$, should be assumed to be proportional to the square root of the hydrostatic head, and the geometry of the tank indicates that $\frac{r(t)}{R} = \frac{h(t)}{H}$.

Q-11.3 Consider the stirred reactor below. Following good engineering practice, derive the steady state process gain and process time constant for this system. You should assume first-order reaction kinetics, $r_A(t) = kC_A(t)$. Note that parameters r and h are not time dependent.



12. Linearization of Nonlinear Equations and Deviation Variables

12.1 The Linear Approximation

Popular process control theory requires that all equations used in analysis be linear and have constant coefficients. Recall that all of the process models we derived in the Chapter 11 were nonlinear ordinary differential equations (ODEs) or linear ODEs with nonconstant coefficients.

The distinction between linear and nonlinear ODEs lies in the treatment of the dependent variable, $y(t)$. Specifically, if the dependent variable is raised to a power, inversed, or otherwise manipulated, the equation is nonlinear. Interestingly, the complexity of the forcing function (the right hand side of the ODE that is not a function of the dependent variable) does not influence whether a process displays a linear or nonlinear character. For example:

Linear ODE with constant coefficients:
$$A\frac{dy(t)}{dt} + By(t) = t^2\sin(t) \tag{12.1}$$

Linear ODE with nonconstant coefficients:
$$A(t)\frac{dy(t)}{dt} + B(t)y(t) = u(t) \tag{12.2}$$

Nonlinear ODE with constant coefficients:
$$A\frac{dx(t)}{dt} + \frac{B}{x(t)} = Cu(t) \tag{12.3}$$

Nonlinear ODE with nonconstant coefficients:
$$A(t)\left(\frac{dx(t)}{dt}\right)^2 + B(t)x(t) = C(t)u(t) \tag{12.4}$$

Linearization is a method used to approximate nonlinear equations with nonconstant coefficients as linear equations with constant coefficients.

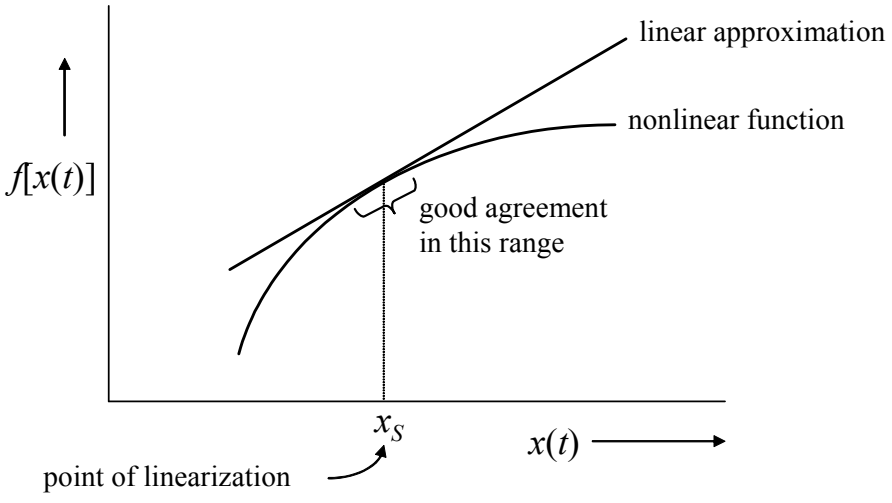


Figure 12.1 - Linear approximation shows good agreement in a narrow operating range

As shown in Fig. 12.1, linearization is a procedure for approximating a nonlinear function with a simple linear function. The linear approximation is exact at one point and has good agreement in a range around that point. The approximation degrades as we move away from the point of linearization.

Fortunately, we get to choose the point around which we will linearize. We choose the point where the process will spend most of its time, the controller design level of operation. This should correspond to the normal or expected set point, which in turn should correspond to the normal or expected value of the process variable.

12.2 Linearization for Functions of One Variable

The method for linearizing a nonlinear equation $f[x(t)]$ is to approximate with a Taylor series expansion around the point x_S:

$$f[x(t)] = f[x_S] + [x(t) - x_S]\frac{d}{dx} f[x(t)]\Big|_{x=x_S} + \underbrace{\frac{[x(t) - x_S]^2}{2!}\frac{d^2}{dx^2} f[x(t)]\Big|_{x=x_S} + \ldots}_{HOT} \qquad (12.5)$$

We consider only the linear terms of the expansion and ignore the higher order terms (*HOT*) of the Taylor series. As shown in the following examples, isolating and linearizing the individual nonlinear terms of a complicated equation and then substituting them back into the original form is often more convenient than linearizing the entire original equation at once.

Example: Tank Liquid Level Model

A mass balance on a gravity drained tank as detailed in Section 11.3 produces the following tank liquid level model:

$$A_C \frac{dh(t)}{dt} = F_0(t) - F_1(t) \qquad \text{where} \quad h(0)=h_S \qquad (12.6)$$

Assume that drain flow is proportional to the square root of hydrostatic head, or $F_1(t) = \alpha h^{1/2}(t)$, and the model becomes the following:

$$A_C \frac{dh(t)}{dt} + \alpha h^{1/2}(t) = F_0(t) \qquad \text{where} \quad h(0)=h_S \qquad (12.7)$$

The nonlinear term in the ODE is $h^{1/2}(t)$. We isolate this nonlinear term and linearize $h^{1/2}(t)$ around the design level h_S (a tank height design value) using the Taylor series approximation:

$$h^{1/2}(t) \cong h_S^{1/2} + [h(t) - h_S]\{\frac{d}{dh}[h^{1/2}(t)]\}\Big|_{h=h_S}$$

$$\cong h_S^{1/2} + [h(t) - h_S]\left(\frac{1}{2h_S^{1/2}}\right)$$

$$\cong h_S^{1/2} - \frac{1}{2} h_S^{1/2} + \frac{1}{2h_S^{1/2}} h(t)$$

The linear approximation then becomes:

$$h^{1/2}(t) \cong \frac{1}{2} h_S^{1/2} + \frac{1}{2h_S^{1/2}} h(t). \tag{12.8}$$

Substitute the linear approximation into the original ODE to obtain our linear process model:

$$\boxed{A_C \frac{dh(t)}{dt} + \frac{\alpha}{2h_S^{1/2}} h(t) = F_0(t) - \frac{\alpha}{2} h_S^{1/2} \qquad \text{where} \quad h(0)=h_S} \tag{12.9}$$

The general form of the dynamic model above is a linear ODE with constant coefficients:

$$A \frac{dy(t)}{dt} + B\, y(t) = C u(t) + E \tag{12.10}$$

We compare this to the general first-order dynamic model (without dead time):

$$\tau_P \frac{dy(t)}{dt} + y(t) = K_P u(t) \qquad \text{where} \quad y(0)= 0$$

Although the tank height model ODE is linear with constant coefficients, we still cannot determine K_P and τ_P by comparison. This is because of the extra constant term, E, on the right-hand side of the equation and the non-zero initial condition, h_S.

12.3 Linearization for Functions of Two Variables

Linearize $f[x(t),y(t)]$ and $g[x(t),y(t)]$ around the point (x_S, y_S) using Taylor series expansion:

$$\frac{dx(t)}{dt} = f\big[x(t), y(t)\big] \qquad \frac{dy(t)}{dt} = g\big[x(t), y(t)\big]$$

As with the single variable expansion, we ignore the higher order Taylor series terms, so the approximations are as follows:

$$f\big[x(t), y(t)\big] \approx f(x_S, y_S) + \big[x(t) - x_S\big]\frac{\partial f}{\partial x}|_{x_S, y_S} + \big[y(t) - y_S\big]\frac{\partial f}{\partial y}|_{x_S, y_S}$$

and

$$g\big[x(t), y(t)\big] \approx g(x_S, y_S) + \big[x(t) - x_S\big]\frac{\partial g}{\partial x}|_{x_S, y_S} + \big[y(t) - y_S\big]\frac{\partial g}{\partial y}|_{x_S, y_S}$$

Again, individually isolating and linearizing the nonlinear terms and then substituting them back into the original equation is often more convenient than linearizing the entire original equation at once.

Example: Consider the Tank Species Balance Model
A species (component) balance on a mixing tank in Section 11.6 produced the following tank concentration model:

$$\frac{dC_A(t)}{dt} = \frac{F}{V}[C_{A0} - C_A(t)] - k_0 e^{-E/RT(t)} C_A(t) \qquad \text{where} \quad C_A(0) = C_{AS}$$

The nonlinear term, $e^{-E/RT(t)}C_A(t)$, is a function of two variables, $T(t)$ and $C_A(t)$. We isolate the nonlinear term and linearize around C_{AS}, T_S (design concentration and temperature values):

Using the Taylor series expansion and recalling that $\frac{d}{dx}e^{u(x)} = e^{u(x)}\frac{du(x)}{dx}$, then

$$e^{-E/RT(t)}C_A(t) \approx e^{-E/RT_S}C_{AS} + \left[T(t) - T_S\right]e^{-E/RT_S}C_{AS}\left(E/RT_S^2\right) + \left[C_A(t) - C_{AS}\right]e^{-E/RT_S}$$

Substituting the linearized term into the original ODE yields the following:

$$\frac{dC_A(t)}{dt} = \frac{F}{V}\left[C_{A0} - C_A(t)\right] - k_0\left\{e^{-E/RT_S}C_{AS} + \left[T(t) - T_S\right]e^{-E/RT_S}C_{AS}\left(E/RT_S^2\right) + \left[C_A(t) - C_{AS}\right]e^{-E/RT_S}\right\}$$

The resulting dynamic model is a linear ODE with constant coefficients of general form:

$$A\frac{dy(t)}{dt} + By(t) = Cu(t) + Dd(t) + E$$

The variable $d(t)$ might be a disturbance variable that impacts the measured process variable $y(t)$. As in the single variable case, we have an extra constant term, E, on the right hand side of the equation.

☆ ☆ ☆

12.4 Defining Deviation Variables

Deviation variables (also called perturbation variables) are used to recast an equation into a more convenient form for control system analysis. Specifically, the time-dependent variables in an equation are recast in terms of their deviation from a fixed value. For controller design, the value we choose is not only the design level of operation, but also the point of linearization, and the point around which we generated dynamic data. In a proper design, all of these have the same value.

For the time-dependent variable $x(t)$, the deviation variable $x^P(t)$ is defined relative to fixed point x_S as such:

$$x^P(t) = x(t) - x_S$$

This is shown graphically in Fig. 12.2:

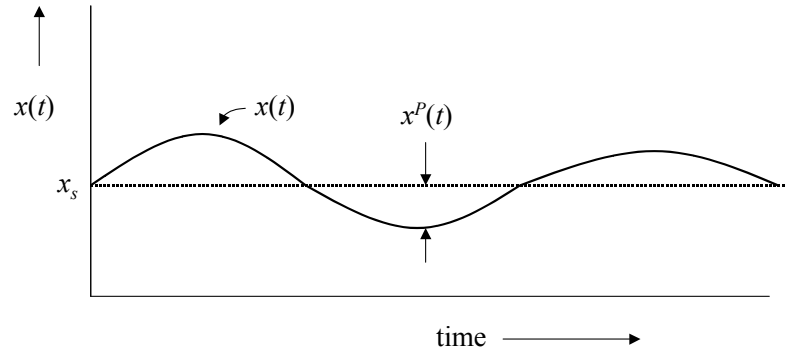


Figure 12.2 - Deviation variable $x^P(t)$ is the deviation of x(t) from constant value x_S

12.5 Deviation Variables Simplify the Equation Form

Deviation variables are useful in preparing equations for control system analysis for two important reasons:

- using them eliminates constant terms, and
- the initial conditions for the ODEs become zero.

To explore this, consider the linear(ized) ODE:

$$\frac{dx(t)}{dt} = Ax(t) + B \qquad \text{where } x(0) = x_S$$

At the point $x(t) = x_S$, the ODE becomes

$$\frac{dx_S}{dt} = Ax_S + B \qquad \text{where } x(0) = x_S$$

Subtracting the second equation from the first, we obtain

$$\frac{d}{dt}[x(t) - x_s] = A[x(t) - x_s] + [B - B] \qquad \text{where } x(0) - x(0) = x_S - x_S = 0$$

We then apply the definition of the deviation variable to obtain the following:

$$\frac{dx^p(t)}{dt} = Ax^p(t) \qquad \text{where} \qquad x^P(0) = 0$$

Note that this form, expressed as a difference from x_S, has no extra constant terms and has an initial condition of zero.

Example: Tank Liquid Level Model
Recall the linearized tank liquid level model:

$$A_C \frac{dh(t)}{dt} + \frac{\alpha}{2h_S^{1/2}} h(t) = F_0(t) - \frac{\alpha}{2} h_S^{1/2} \qquad \text{where } h(0) = h_S$$

We choose the design variables as $h(t) = h_S$ and $F_0(t) = F_{0,S}$. We will use these later to define the deviation variables. Note that these values are not independent from each other because if you maintain the inlet flow rate at $F_{0,S}$ for a sufficient length of time, the liquid level should ultimately steady at h_S.

Substituting these design variables into the original ODE, we obtain

$$A_C \frac{dh_S}{dt} + \frac{\alpha}{2h_S^{1/2}} h_S = F_{0,S} - \frac{\alpha}{2} h_S^{1/2} \qquad \text{where } h(0) = h_S$$

We then subtract the second ODE from the first:

$$A_C \frac{d}{dt}\left[h(t) - h_S\right] + \frac{\alpha}{2h_S^{1/2}}\left[h(t) - h_S\right] = \left[F_0(t) - F_{0,S}\right] - \left[\frac{\alpha}{2} h_S^{1/2} - \frac{\alpha}{2} h_S^{1/2}\right] \quad \text{where } h(0) - h(0) = 0$$

and define the deviation variables:

$$h^P(t) = h(t) - h_S$$
$$F_0^P(t) = F_0(t) - F_{0S}$$

Substituting these, we obtain the linear ODE with constant coefficients and a zero initial condition:

$$A_C \frac{dh^P(t)}{dt} + \frac{\alpha}{2h_S^{1/2}} h^P(t) = F_0^P(t) \qquad \text{where } h^P(0) = 0$$

We can now compare this to the general first order ODE without dead time. We have not previously mentioned it, but the initial condition is zero for this general form:

$$\tau_P \frac{dy(t)}{dt} + y(t) = K_P u(t) \qquad \text{where } y(0) = 0$$

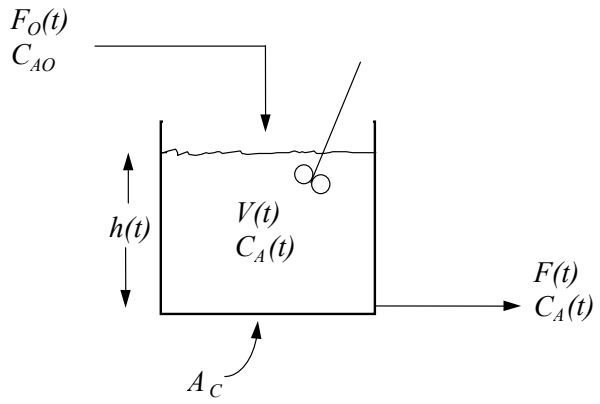
We now have a payoff for this effort. By comparing the model forms, we can determine the steady state process gain and overall time constant for this process:

$$K_P = \frac{2h_S^{1/2}}{\alpha} \qquad \tau_P = \frac{2A_C h_S^{1/2}}{\alpha}$$

These K_P and τ_P values are only exact at point ($h(t)=h_S$, $F_0(t)=F_{0,S}$). They become approximations that decrease in accuracy as $h(t)$ moves away from liquid height h_S or $F_0(t)$ moves away from $F_{0,S}$.

12.6 Exercises

Q-12.1 Consider the process below. A control system is under development where $F_O(t)$ is the manipulated variable and $C_A(t)$ is be the measured variable.



a) Assuming a third order reaction for A → B, or $r(t) = k[C_A(t)]^3$, and a constant cross section, or $V(t) = A_C h(t)$, show all steps to show that the ODE describing this system is

$$A_C \frac{dC_A(t)}{dt} + \frac{F_O(t)}{h(t)} C_A(t) + A_C k\left[C_A(t)\right]^3 = \frac{C_{AO}}{h(t)} F_O(t)$$

with the initial condition $C_A(0) = C_{A,S}$.

b) Linearize this ODE and put it in deviation variable form. Assume that when $F_O(t) = F_{O,S}$, then at steady state $C_A(0) = C_{A,S}$ and $h(0) = h_S$.

c) Near the expected point of operation, what are the (approximate) process gain and time constant values?

Q-12.2 The density of an ideal gas, $\rho(t)$, can be expressed as follows:

$$\rho(t) = \frac{MP(t)}{RT(t)}$$

where M is the molecular weight and R is the ideal gas constant. Showing all steps, linearize this expression around ρ_S to obtain a linear approximation for density as a function of temperature and pressure.

13. Time Domain ODEs and System Behavior

13.1 Linear ODEs

After deriving process models, linearizing them, and simplifying them with deviation variables, we are left with linear ODEs (ordinary differential equations) with constant coefficients:

1st order: $$a_1 \frac{dy(t)}{dt} + a_0 y(t) = Q(t)$$

2nd order: $$a_2 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = Q(t)$$

3rd order: $$a_3 \frac{d^3 y(t)}{dt^3} + a_2 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = Q(t)$$

The coefficients of these equations signal specific dynamic behaviors. By solving these ODEs in the time domain, we learn the relationship between an equation form and the dynamic behavior it implies.

13.2 Solving First Order ODEs

For the general first order ODE of the form

$$\frac{dy(t)}{dt} + P(t)y(t) = Q(t) \qquad \text{where} \quad y(0) = y_S$$

the solution method involves defining an integrating factor, solving a general solution form based on this integrating factor, and then solving for the constant of integration using the initial conditions.

An integrating factor, μ, is defined as follows:

$$\mu = e^{\int P(t)dt}$$

The solution to the ODE is then $$y(t) = \frac{1}{\mu}\left[\int \mu Q(t)dt + C_1\right]$$

where the constant C_1 is computed from the initial condition $y(0) = y_S$.

Example 1: Solve this ODE in the time domain:

$$\frac{dy(t)}{dt} + 2y(t) = 4t \qquad \text{where} \quad y(0) = 1$$

Solution: Using the steps outlined, we first compute the integrating factor, $\mu = e^{\int 2dt} = e^{2t}$

so the solution becomes $$y(t) = \frac{1}{e^{2t}}\left[\int e^{2t}(4t)dt + c_1\right] = e^{-2t}\left[4\int te^{2t}dt + c_1\right]$$

We must then integrate by parts: $\int udv = uv - \int vdu$

$$u = t;\ du = dt; \qquad v = \frac{1}{2}e^{2t};\ \ dv = e^{2t}dt$$

Substituting the result, the solution becomes

$$y(t) = e^{-2t}\left[\frac{4}{2}te^{2t} - \frac{4}{2}\int e^{2t}dt + c_1\right]$$

$$= e^{-2t}\left[2te^{2t} - e^{2t} + c_1\right]$$

$$= 2t - 1 + c_1e^{-2t}$$

Next, we apply initial conditions: @ $t = 0$, $y = 1$, so $1 = -1 + c_1$, and thus $c_1 = 2$,

which yields the total solution: $\boxed{y(t) = 2e^{-2t} + 2t - 1}$

Example 2: Solve this ODE in the time domain:

$$\frac{dx(t)}{dt} = x(t) + \sin t \qquad \text{where } x(0) = 0$$

Solution: We first rewrite the ODE as such:

$$\frac{dx(t)}{dt} - x(t) = \sin t$$

Then, we compute the integrating factor:

$$\mu = e^{\int -1dt} = e^{-t}$$

The solution then becomes $$x(t) = e^{t}\left[\int e^{-t}\sin t dt + c_1\right]$$

From integral tables, we can see that

$$\int e^{-t}\sin t dt = \frac{1}{2}e^{-t}\left[-\sin t - \cos t\right]$$

Substituting this, the solution becomes

$$x(t) = e^t\left[-\frac{1}{2}e^{-t}(\sin t + \cos t) + c_1\right]$$
$$= -\frac{1}{2}(\sin t + \cos t) + c_1 e^t$$

We then apply initial conditions: @ $t = 0$, $x = 0$, so $0 = -\frac{1}{2}[\sin(0) + \cos(0)] + c_1$, thus $c_1 = \frac{1}{2}$,

which yields the total solution: $\boxed{x(t) = \frac{1}{2}\left[e^t - \sin t - \cos t\right]}$

* * *

13.3 Deriving "τ_p = 63.2% of Process Step Response" Rule

Our ability to solve linear first order ODEs now allows us to derive the "τ_p = 63.2% of Process Step Response" rule used in model fitting, as referred to in Section 3.4.

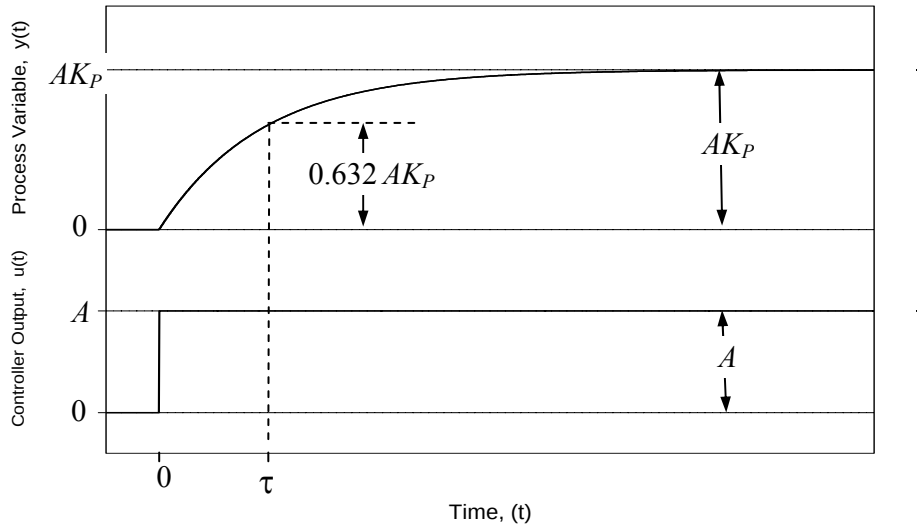


Figure 13.1 – Response of true first order process to a step change in controller output

The figure above shows the open loop step response of a true first order process model:

$$\tau_p \frac{dy(t)}{dt} + y(t) = K_p u(t) \qquad \text{where} \qquad y(0) = 0$$

As shown in Fig 13.1, the measured process variable, $y(t)$, and controller output signal, $u(t)$, are initially at steady state with $y(t) = u(t) = 0$ for $t < 0$. At time $t = 0$, the controller output is stepped to $u(t) = A$, where it remains for the duration of the experiment. Hence, the first order model becomes

$$\tau_p \frac{dy(t)}{dt} + y(t) = AK_p$$

To solve this ODE, rearrange as:

$$\frac{dy(t)}{dt} + \frac{1}{\tau_p} y(t) = \frac{AK_p}{\tau_p}$$

First compute the integrating factor, $\mu = e^{\int (1/\tau_p) dt} = e^{t/\tau_p}$. Solving the ODE using this integrating factor yields

$$y(t) = \frac{1}{e^{t/\tau_p}} \left[\int e^{t/\tau_p} \left(\frac{AK_p}{\tau_p} \right) dt + c_1 \right]$$

$$= e^{-t/\tau_p} \left[\frac{AK_p}{\tau_p} \int e^{t/\tau_p} dt + c_1 \right]$$

Recall that $\int e^{ax} dx = \frac{1}{a} e^{ax}$

and thus $\int e^{t/\tau_P} dt = \tau_P e^{t/\tau_P}$

so $y(t) = e^{-t/\tau_p} \left[AK_p e^{t/\tau_p} + c_1 \right]$

$$= AK_p + c_1 e^{-t/\tau_p}$$

Next, apply the initial condition: @ $t = 0$, $y = 0$, $0 = AK_p + c_1$, and thus $c_1 = -AK_p$.

Substituting and rearranging, we obtain the solution to the ODE:

$$\boxed{y(t) = AK_p \left[1 - e^{-t/\tau_P} \right]}$$

After the passage of one time constant, time $t = \tau_p$, the solution becomes

$$y(\tau_p) = AK_p \left[1 - e^{-\tau_P/\tau_P} \right] = AK_p \left[1 - e^{-1} \right]$$

Therefore, the measured process variable step response at time $t = \tau_p$ is

$$\boxed{y(\tau_p) = 0.632 AK_p}$$

As we set out to show, the measured process variable will have traveled to 63.2% of the total change that it will ultimately experience at time t equals one time constant, τ_P.

13.4 Solving Second Order ODEs

For the general second-order ODE with constant coefficients,

$$a_2 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = Q(t)$$

the total solution, $y(t)$, is the sum of the particular solution, $y_p(t)$, and the complementary solution, $y_c(t)$, or

$$y(t) = y_p(t) + y_c(t)$$

The complementary solution, $y_c(t)$, is the solution of the ODE when $Q(t) = 0$, or

$$a_2 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = 0$$

To solve the complementary solution, form the characteristic equation using operator notation:

$$a_2 s^2 + a_1 s + a_0 = (s - p_1)(s - p_2) = 0$$

The roots of the characteristic equation, p_1 and p_2, are computed using the quadratic equation:

$$p_1, p_2 = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2 a_0}}{2a_2}$$

The coefficients a_2, a_1 and a_0 define the roots of the characteristic equation, and the roots define the form of the complementary solution.

Case 1: If $a_1^2 - 4a_2 a_0 > 0$, then the characteristic equation of the complementary solution will have two distinct real roots, p_1 and p_2, and the solution will have the form

$$y(t) = y_p(t) + c_1 e^{p_1 t} + c_2 e^{p_2 t}$$

Case 2: If $a_1^2 - 4a_2 a_0 = 0$, then the characteristic equation of the complementary solution will have two equal or repeated roots, $p_3 = p_4$, and the solution will have the form

$$y(t) = y_p(t) + (c_1 + c_2 t) e^{p_3 t}$$

Case 3: If $a_1^2 - 4a_2 a_0 < 0$, then the characteristic equation of the complementary solution will have two complex conjugate roots, $p_5 \pm p_6 i$, and the solution will have the form

$$y(t) = y_p(t) + e^{p_5 t}\left[c_1 \cos(p_6 t) - c_2 \sin(p_6 t) \right]$$

Note that the *roots of the characteristic equation* provide important clues about the dynamic behavior described by an ODE. Stability relates to the sign of a root because e^{+t} grows without bound and e^{-t} approaches zero as time increases. And as results in Case 3, a solution with sines and cosines will have a natural tendency to oscillate.

Example 1: Solve this ODE in the time domain:

$$\frac{d^2y(t)}{dt^2} + 4\frac{dy(t)}{dt} + 3y(t) = t \qquad \text{where} \quad y(0) = \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

Complementary Solution: $\dfrac{d^2y(t)}{dt^2} + 4\dfrac{dy(t)}{dt} + 3y(t) = 0$

The characteristic equation is $s^2 + 4s + 3 = 0$ with roots $p_1, p_2 = \dfrac{-4 \pm \sqrt{16-12}}{2} = -1, -3$

so $y_c(t)$ solution has the form $y_c(t) = c_1e^{-t} + c_2e^{-3t}$

Particular Solution: The right hand side of the original ODE indicates that $y_p(t) = c_3 + c_4t$

so $\dfrac{dy_p(t)}{dt} = c_4$ and $\dfrac{d^2y_p(t)}{dt^2} = 0$

Substituting this into the original ODE yields

$$(0) + 4(c_4) + 3(c_3 + c_4t) = t$$

We then equate like terms and solve for constants:

$3c_4t = t$ so $c_4 = \dfrac{1}{3}$

and $4c_4 + 3c_3 = 0$ so $4\left(\dfrac{1}{3}\right) + 3c_3 = 0$ and $c_3 = -\dfrac{4}{9}$

so $y_p(t) = -\dfrac{4}{9} + \dfrac{1}{3}t$

Total Solution: $y(t) = y_p(t) + y_c(t) = -\dfrac{4}{9} + \dfrac{1}{3}t + c_1e^{-t} + c_2e^{-3t}$

We apply initial conditions to determine the constants:

1. @ $t = 0, y = 0$ $\qquad 0 = -\dfrac{4}{9} + 0 + c_1 + c_2$

2. @ $t = 0$, $\dfrac{dy(t)}{dt} = 0$ $\qquad \dfrac{dy(t)}{dt} = 0 = \dfrac{1}{3} - c_1e^0 - 3c_2e^0$

Solving simultaneously yields $\left.\begin{array}{r} c_1 + c_2 = \frac{4}{9} \\ c_1 + 3c_2 = \frac{1}{3} = \frac{3}{9} \end{array}\right\} \; c_1 = \frac{1}{2}, \quad c_2 = -\frac{1}{18}$

and thus the total solution is $\boxed{y(t) = -\frac{4}{9} + \frac{1}{3}t + \frac{1}{2}e^{-t} - \frac{1}{18}e^{-3t}}$

Example 2: Solve this ODE in the time domain:

$$\frac{d^2y(t)}{dt^2} + 2\frac{dy(t)}{dt} + y(t) = t^2 \qquad \text{where} \quad y(0) = 0 \;\text{ and }\; \left.\frac{dy(t)}{dt}\right|_{t=0} = 1$$

Complementary Solution: $\dfrac{d^2y(t)}{dt^2} + 2\dfrac{dy(t)}{dt} + y(t) = 0$

The characteristic equation is $s^2 + 2s + 1 = 0$ with roots $p_1, p_2 = \dfrac{-2 \pm \sqrt{4-4}}{2} = -1, -1$

so $y_c(t)$ solution has the form $y_c(t) = (c_1 + c_2 t)e^{-t}$

Particular Solution: The right hand side of the ODE indicates that $y_p(t) = c_3 + c_4 t + c_5 t^2$

so $\dfrac{dy_p(t)}{dt} = c_4 + 2c_5 t$ and $\dfrac{d^2y_p(t)}{dt^2} = 2c_5$

Substituting this into the original ODE yields

$$(2c_5) + 2(c_4 + 2c_5 t) + (c_3 + c_4 t + c_5 t^2) = t^2$$

equate like terms, $2c_5 + 2c_4 + c_3 = 0$ $\qquad 4c_5 t + c_4 t = 0$ $\qquad c_5 t^2 = t^2$

and solve for constants: $c_5 = 1$ $\qquad c_4 = -4$ $\qquad c_3 = 6$

so $y_p(t) = 6 - 4t + t^2$.

Total Solution: $y(t) = y_p(t) + y_c(t) = 6 - 4t + t^2 + (c_1 + c_2 t)e^{-t}$

Apply initial conditions to determine the constants:

1. $@t = 0, y = 0$ $\qquad 0 = 6 - 0 + 0 + [c_1 + c_2(0)]e^0$

2. $@t = 0, \dfrac{dy}{dt} = 1$ $\qquad 1 = -4 + 2(0) + [-6 + c_2(0)]e^0(-1) + c_2 e^0$

Solving for constants yields: $c_1 = -6$ and $c_2 = -1$

and thus the total solution is $\boxed{y(t) = 6 - 4t + t^2 - (6+t)e^{-t}}$

Example 3: Solve this ODE in the time domain:

$$2\frac{d^2y(t)}{dt^2} + 2\frac{dy(t)}{dt} + y(t) = 2 \quad \text{where} \quad y(0) = \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

Complementary Solution: $2\frac{d^2y(t)}{dt^2} + 2\frac{dy(t)}{dt} + y(t) = 0$

The characteristic equation is $2s^2 + 2s + 1 = 0$ with roots $p_1, p_2 = \frac{-2 \pm \sqrt{4-8}}{4} = -\frac{1}{2} \pm \frac{1}{2}i$

so $y_c(t)$ solution has the form $y_c(t) = c_1 e^{(-\frac{1}{2}+\frac{1}{2}i)t} + c_2 e^{(-\frac{1}{2}-\frac{1}{2}i)t}$

$$= e^{\frac{-t}{2}}\left[c_1 e^{\frac{i}{2}t} + c_2 e^{-\frac{i}{2}t}\right]$$

Recall the identities $e^{iat} = \cos at + i\sin at$, $\cos(-at) = \cos(at)$, and $\sin(-at) = -\sin(at)$

Applying these, the complementary solution becomes

$$y_c(t) = e^{-\frac{t}{2}}\left\{c_1\left[\cos\frac{t}{2} + i\sin\frac{t}{2}\right] + c_2\left[\cos\frac{t}{2} - i\sin\frac{t}{2}\right]\right\}$$

If we let $c_1 = \frac{1}{2}(c_R + ic_I)$ and $c_2 = \frac{1}{2}(c_R - ic_I)$

where c_R and c_I are real constants, then

$$y_c(t) = e^{-\frac{t}{2}}\left\{c_R\cos\frac{t}{2} - c_I\sin\frac{t}{2}\right\}$$

Particular Solution: The right hand side of the original ODE indicates that $y_p(t) = c_3$

so $\frac{dy_p(t)}{dt} = 0$ and $\frac{d^2y_p(t)}{dt^2} = 0$.

Substituting this into the original ODE yields

$2(0) + 2(0) + c_3 = 2$, and thus $c_3 = 2$,

so $y_p(t) = 2$

Total Solution: $y(t) = y_p(t) + y_c(t) = 2 + e^{-\frac{t}{2}}\left\{c_R \cos\frac{t}{2} - c_I \sin\frac{t}{2}\right\}$

We apply initial conditions to determine the constants:

1. @ t=0, y=0 $\quad 0 = 2 + e^0\{c_R \cos 0 - c_I \sin 0\}$ so $c_R = -2$

2. @ t=0, $\frac{dy(t)}{dt} = 0$ $\quad \frac{dy(t)}{dt} = 0 = -\frac{1}{2}e^{-\frac{t}{2}}\left[-2\cos\frac{t}{2} - c_I \sin\frac{t}{2}\right]$

$$+e^{-\frac{t}{2}}\left[-2\left(-\sin\frac{t}{2}\right)\left(\frac{1}{2}\right) - c_I\left(\cos\frac{t}{2}\right)\left(\frac{1}{2}\right)\right]$$

so at $t = 0,\ c_I = 2$.

The total solution becomes $\boxed{y(t) = 2 - 2e^{-\frac{t}{2}}\left\{\cos\frac{t}{2} + \sin\frac{t}{2}\right\}}$

Example 4: Solve this ODE in the time domain:

$$\frac{d^2x(t)}{dt^2} + x(t) = \sin(2t) \text{ where } x(0) = 0 \text{ and } \left.\frac{dx(t)}{dt}\right|_{t=0} = 1$$

Complementary Solution: $\frac{d^2x(t)}{dt^2} + 0\frac{dx(t)}{dt} + x(t) = 0$

The characteristic equation is $s^2 + 0s + 1 = 0$ with roots $p_1, p_2 = \pm i$

so $x_c(t)$ solution has form $x_c(t) = c_1 e^{it} + c_2 e^{-it}$

We recall the identity $e^{iat} = \cos at + i\sin at$

We then let $c_1 = \frac{1}{2}(c_R + ic_I)$ and $c_2 = \frac{1}{2}(c_R - ic_I)$

so $x_c(t) = c_R \cos(t) - c_I \sin(t)$

Particular Solution: The right hand side of the ODE indicates that $x_p(t) = c_3\cos 2t + c_4\sin 2t$

so $$\frac{dx_p(t)}{dt} = -2c_3\sin 2t + 2c_4\cos 2t$$

and $$\frac{d^2x_p(t)}{dt^2} = -4c_3\cos 2t - 4c_4\sin 2t$$

Substituting into the original ODE yields

$$-4c_3\cos 2t - 4c_4\sin 2t + c_3\cos 2t + c_4\sin 2t = \sin 2t$$

We then equate like terms: $-4c_4\sin 2t + c_4\sin 2t = \sin 2t$ $\quad -4c_3\cos 2t + c_3\cos 2t = 0$

Solving for constants yields $c_4 = -\frac{1}{3}$ and $c_3 = 0$

so the particular solution is $x_p(t) = -\frac{1}{3}\sin 2t$

Total Solution: $x(t) = x_p(t) + x_c(t) = -\frac{1}{3}\sin 2t + c_R\cos t - c_I\sin t$

We apply initial conditions to determine the constants:

1. @ $t = 0, x = 0$ $\quad 0 = -\frac{1}{3}\sin 0 + c_R\cos 0 - c_I\sin 0$ $\quad$ so $\quad c_R = 0$

2. @ $t = 0, \left.\frac{dx(t)}{dt}\right|_{t=0} = 1$ $\quad 1 = -\frac{2}{3}\cos 0 - c_I\cos 0$ $\quad$ so $\quad c_I = -\frac{5}{3}$

and thus the total solution is $\boxed{x(t) = -\frac{1}{3}\sin 2t + \frac{5}{3}\sin t}$

13.5 The Second Order Underdamped Form

A general and more useful form of the second order ODE is written

$$\tau_n^2\frac{d^2y(t)}{dt^2} + 2\tau_n\xi\frac{dy(t)}{dt} + y(t) = K_Pu(t) \qquad \text{where} \quad y(0) = \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

and
- $y(t)$ = measured process variable
- $u(t)$ = forcing function (controller output)
- K_P = steady state process gain
- τ_n = natural period of oscillation
- ξ = damping factor

The total solution, $y(t)$, is the sum of the particular solution, $y_P(t)$, and the complementary solution, $y_C(t)$:

$$y(t) = y_p(t) + y_c(t)$$

The complementary solution form is

$$\tau_n^2 \frac{d^2 y(t)}{dt^2} + 2\tau_n \xi \frac{dy(t)}{dt} + y(t) = 0$$

The characteristic equation is thus

$$\tau_n^2 s^2 + 2\tau_n \xi s + 1 = (s - p_1)(s - p_2) = 0$$

with roots

$$p_1, p_2 = \frac{-2\tau_n \xi \pm \sqrt{4\tau_n^2 \xi^2 - 4\tau_n^2}}{2\tau_n^2} = \frac{-\xi \pm \sqrt{\xi^2 - 1}}{\tau_n}$$

As mentioned in Section 13.4, the *roots of the characteristic equation* provide information about the behavior of the system. If the roots yield sines and cosines, there will be a natural tendency for the system to oscillate. If the roots yield even one positive real part, then e^{+t} for that term grows without bound as time increases, leading to an unstable system. If all real parts are negative, then e^{-t} approaches zero for all terms as time increases, indicating that dynamics die out and the system remains stable.

The second order underdamped form written above is useful for a quick evaluation of the nature of the system dynamics based solely on the value of the damping factor, ξ. Since τ_n must be positive (it is a period of time), complex roots and positive or negative real parts arise based solely on the value of the damping factor.

13.6 Roots of the Characteristic Equation Indicate System Behavior

As shown in Fig. 13.2, y_p is a constant for time $t > 0$. This would result from a step change in the forcing function. While not a necessary assumption for the derivations that follow, it does permit the characteristic behaviors of the different cases to be visually apparent.

Case 1: $\xi > 1$ (overdamped) $\qquad (s - p_1)(s - p_2) = 0\,; \quad -p_1, -p_2 = \dfrac{-\xi \pm \sqrt{\xi^2 - 1}}{\tau_n}$

The overdamped case, $\xi > 1$, yields real negative distinct roots, $-p_1$, $-p_2$ (note that the negative signs for p_1, p_2 are shown explicitly).

The total solution is thus $\qquad y(t) = y_p + c_1 e^{-p_1 t} + c_2 e^{-p_2 t}$.

As time passes, the real exponentials $e^{-p_1 t}$ and $e^{-p_2 t}$ approach zero as time grows toward infinity. The lack of imaginary roots leads to a lack of sine and cosine terms and their characteristic oscillatory effect. Thus, $y(t)$ approaches y_p slowly, exponentially and without oscillations (a stable system).

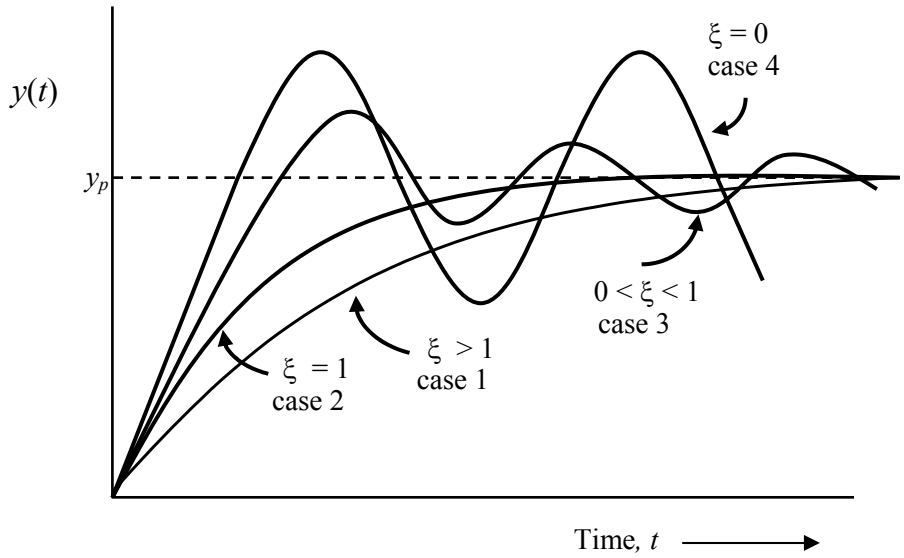


Figure 13.2 - Stable behaviors of underdamped system based on value of ξ

Case 2: ξ = 1 (critically damped) $(s-p_1)(s-p_2)=0\,;\quad -p_1,-p_2=\dfrac{-\xi\pm\sqrt{\xi^2-1}}{\tau_n}=-\dfrac{1}{\tau_n}$

The critically damped case, ξ = 1, yields real negative repeated roots (note that the negative signs for p_1, p_2 are shown explicitly).

The total solution is thus $y(t)=y_p+(c_1+c_2t)e^{-t/\tau_n}$.

As time passes, the real exponential e^{-t/τ_n} approach zero as time grows toward infinity. The lack of imaginary roots leads to a lack of sine and cosine terms and their characteristic oscillatory effect. Thus, $y(t)$ approaches y_p exponentially and without oscillations (a stable system).

Case 3: 0 < ξ < 1 (underdamped) $-p_1,-p_2=\dfrac{-\xi\pm\sqrt{\xi^2-1}}{\tau_n}=\dfrac{-\xi\pm i\sqrt{1-\xi^2}}{\tau_n}$

The underdamped case, 0 < ξ < 1, yields distinct roots with a negative real part and an imaginary part.

The total solution is thus $y(t)=y_p+e^{-\xi t/\tau_n}\left(c_1e^{\frac{i\sqrt{1-\xi^2}}{\tau_n}t}+c_2e^{-\frac{i\sqrt{1-\xi^2}}{\tau_n}t}\right)$.

We then apply the Euler identities, $e^{i\theta}=\cos\theta+i\sin\theta$ and $e^{-i\theta}=\cos\theta-i\sin\theta$

and define $c_1 = 0.5(c_R + ic_I)$ and $c_2 = 0.5(c_R - ic_I)$. The total solution thus becomes

$$y(t) = y_p + e^{-\xi t/\tau_n}\left(c_R \cos(\frac{\sqrt{1-\xi^2}}{\tau_n}t) - c_I \sin(\frac{\sqrt{1-\xi^2}}{\tau_n}t)\right).$$

As time passes and for all values of the damping factor in the range $0 < \xi < 1$, the sines and cosines in the solution indicate that the system has a natural tendency to oscillate. The real exponential $e^{-\xi t/\tau_n}$ approaches zero as time grows toward infinity. Because the term that causes oscillations is multiplied by a term that is decreasing to zero, the result is a damping oscillation. Hence, as time passes, $y(t)$ approaches y_p exponentially and with oscillations (a stable system).

Case 4: $\xi = 0$ (undamped) $\qquad -p_1, -p_2 = \dfrac{-\xi \pm \sqrt{\xi^2 - 1}}{\tau_n} = \pm\dfrac{\sqrt{-1}}{\tau_n} = \pm\dfrac{i}{\tau_n}$

The undamped case, $\xi = 0$, yields repeated roots with an imaginary part and no real part.

The total solution is thus $\qquad y(t) = y_p + \left(c_R \cos(\dfrac{t}{\tau_n}) - c_I \sin(\dfrac{t}{\tau_n})\right)$

As time passes, $y(t)$ oscillates around y_p with constant amplitude. This is due to the lack of a real exponential term to dampen the oscillations as time increases. Oscillations that neither grow nor die out indicate a system at the limit of stability. Any increase in ξ produces a damping oscillation as shown in Case 3. Any decrease in ξ produces an unstable system as shown in Case 5 below.

Case 5: $\xi < 0$ (unstable)

All values of the damping factor, ξ, that are less than zero yield a solution that has a positive real part. As time passes, e^{+t} will grow without bound and the process will display diverging (unstable) behavior. The behaviors of the solutions below are shown in Fig. 13.3.

$-1 < \xi < 0$: $\qquad y(t) = y_p(t) + e^{+\xi t/\tau_n}\left(c_R \cos(\dfrac{\sqrt{1-\xi^2}}{\tau_n}t) - c_I \sin(\dfrac{\sqrt{1-\xi^2}}{\tau_n}t)\right)$

$\xi = -1$: $\qquad y(t) = y_p(t) + (c_1 + c_2 t)e^{+t/\tau_n}$

$\xi < -1$: $\qquad y(t) = y_p(t) + c_1 e^{+p_1 t} + c_2 e^{+p_2 t}$

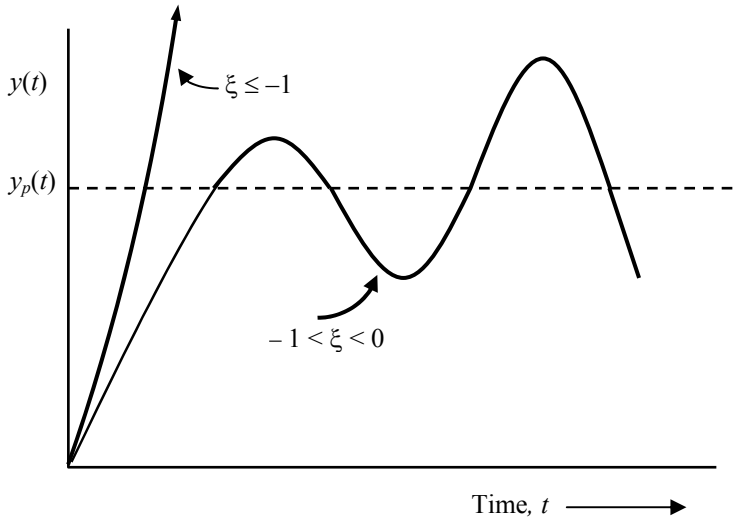


Figure 13.3 - Unstable behaviors of underdamped system based on value of ξ in Case 5

13.7 Exercises

Q-13.1 For each ODE below, determine the natural period of oscillation, τ_n, and damping factor, ξ. Based on the damping factor, describe the inherent nature of the system, e.g.

- is it naturally stable or unstable?
- does it have a natural tendency to oscillate or not?

a) $\dfrac{d^2y(t)}{dt^2} + 4\dfrac{dy(t)}{dt} + 3y(t) = t$ where $y(0) = \left.\dfrac{dy(t)}{dt}\right|_{t=0} = 0$

b) $\dfrac{d^2y(t)}{dt^2} + 2\dfrac{dy(t)}{dt} + y(t) = t^2$ where $y(0) = 0$ and $\left.\dfrac{dy(t)}{dt}\right|_{t=0} = 1$

c) $2\dfrac{d^2y(t)}{dt^2} + 2\dfrac{dy(t)}{dt} + y(t) = 2$ where $y(0) = \left.\dfrac{dy(t)}{dt}\right|_{t=0} = 0$

d) $\dfrac{d^2x(t)}{dt^2} + x(t) = \sin(2t)$ where $x(0) = 0$ and $\left.\dfrac{dx(t)}{dt}\right|_{t=0} = 1$

Q-13.2 Use *Custom Proces*s to explore the second order linear model form:

$$\tau_n^2 \frac{d^2 y(t)}{dt^2} + 2\tau_n \xi \frac{dy(t)}{dt} + y(t) = K_P u(t)$$

Perform this study in open loop to understand how the parameters impact system behavior.

a) Start by clicking on the *Custom Process* button on LOOP-PRO's main screen; then choose Single Loop Process. When the simulation starts, notice that the graphic to the right of the scrolling plots is comprised of a Process button, Disturb(ance) button and Controller button (the C in the white circle).

Click the Process button on the graphic. This opens a "Construct Process and Disturbance Models" form. On the Process Model tab, select Underdamped Linear Model and Self Regulating (Stable) Processes.

Custom Process permits the construction of sophisticated model forms. Here we explore the simple second order linear model form shown above.

For the underdamped self regulating linear process model, enter the following parameters:

Process Gain, K_P	1
Natural Period, τ_{Pn}	10
Damping Factor, ξ	0
Time Constant, τ_P	0
Lead Time, τ_{PL}	0
Dead Time, θ_P	0

We will not be studying the disturbance model, so when you are finished entering the process model parameters, click Done at the bottom of the form to start the simulation.

b) When the damping factor, ξ, equals zero (as is the case above), the process is said to be *undamped.* When forced by a step in the controller output signal, an undamped process will oscillate with a period correlated to the natural period of oscillation, τ_n.

Specifically, the period of oscillation as measured on a strip chart, T, is related to τ_n as

$$\tau_n = \frac{T}{2\pi}$$

Step the controller output from 50 to 51 and let the measured process variable oscillate through a few complete cycles. Does the process display an undamped character?

Pause the process and view a fixed plot. Measure on the plot the amount of time, T, for the process to complete one cycle (from peak to peak or from trough to trough). Use the above relation to confirm that the natural period specified in your *Custom Process* model matches the behavior observed on the plot.

c) Return to your simulation and click the Process button on the graphic. On the "Construct Process and Disturbance Models" menu, change the natural period of oscillation to 15. As you did in step (b) above, step the controller output from 50 to 51. After the measured process variable completes a few complete cycles, pause the process and view a fixed plot.

From the plot, determine T, the time for the process to complete one cycle. Use the above relation of T to τ_n to confirm that the natural period specified in your *Custom Process* model matches the behavior observed on the plot.

d) Explore how the damping factor, ξ, impacts system behavior. Click the Process button on the graphic. On the "Construct Process and Disturbance Models" menu, change the natural period of oscillation, τ_n, to 10 and the damping factor, ξ, to 0.3. Step the controller output from 50 to 51, and when the dynamics die out, back to 50.

Repeat for a damping factor of 0.5, 0.7, 1.0 and 2.0. Describe how the system behavior you observe relates to the chapter discussion about the damping factor.

Now set the damping factor to -0.1 and step the controller output from 50 to 51. Describe how the system behavior you observe relates to the chapter discussion about the damping factor.

14. Laplace Transforms

14.1 Laplace Transform Basics

Transforming functions from the time domain into the Laplace domain provides a convenient means for manipulating and solving linear ODEs with constant coefficients. In particular, Laplace transforms enable us to solve ODEs using algebra instead of calculus. They also provide a straightforward method for handling the mathematical time shift associated with dead time equations. Thus, complicated analysis can be performed in a straightforward manner. As we learned in previous chapters, few processes are accurately described by linear models with constant coefficients, but linearization and deviation variable techniques enable us to recast ODEs into a linear, constant coefficient form.

Laplace transforms map an equation from the time domain (t) into the Laplace domain (s). The definition of the Laplace Transform is

$$\mathcal{L}[f(t)] \equiv \int_0^\infty f(t)e^{-st}dt \equiv F(s)$$

The Laplace independent variable, s, is defined in the complex plane as $s = a + bi$, where a is the real part and b is the imaginary part. To help visualize the complex s plane, plotted below in Fig. 14.1 are the points A = 2 + i and B = – 2 – 2i.

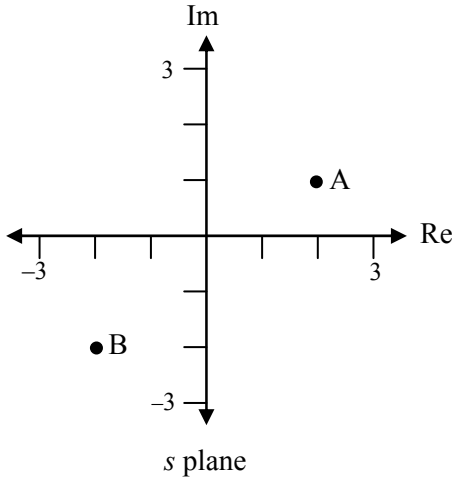


Figure 14.1 – The complex plane (s = a + bi)

While the mapping of a function $f(t) \rightarrow F(s)$ can be derived using the above definition, we are fortunate that tables have been created that list the transforms for common functions (see Appendix B). We detail the derivation of a few entries to show that the table has a basis in the theoretical definition of the Laplace Transform.

Example 1: Show that $f(t) = at \quad \rightarrow \quad F(s) = \dfrac{a}{s^2}$

First apply the definition of the Laplace transform: $\mathcal{L}[at] = \int_0^\infty ate^{-st}dt = a\int_0^\infty te^{-st}dt$

Next, integrate by parts: $\int udv = uv - \int vdu$, so $u = t$; $du = dt$; $v = \dfrac{-e^{-st}}{s}$; $dv = e^{-st}dt$

Substituting, we obtain $\mathcal{L}[at] = a\left[\left.\dfrac{-te^{-st}}{s}\right|_0^\infty - \int_0^\infty \dfrac{-e^{-st}}{s}dt\right] = a\left[(0-0) - \left.\dfrac{-e^{-st}}{s^2}\right|_0^\infty\right]$

The solution is thus $\boxed{\mathcal{L}[at] = \dfrac{a}{s^2}}$

Example 2: Show that $f(t) = e^{-at} \quad \rightarrow \quad F(s) = \dfrac{1}{s+a}$

Applying the definition of a Laplace transform and simplifying, we obtain

$$\mathcal{L}[e^{-at}] = \int_0^\infty e^{-at}e^{-st}dt = \int_0^\infty e^{-(s+a)t}dt = -\frac{1}{s+a}e^{-(s+a)t}\Big|_0^\infty$$

The solution is thus $\boxed{\mathcal{L}[e^{-at}] = \dfrac{1}{s+a}}$

Example 3: Show that $f(t) = \sin\omega t \quad \rightarrow \quad F(s) = \dfrac{\omega}{s^2+\omega^2}$

First apply the definition of a Laplace transform:

$$\mathcal{L}[\sin\omega t] = \int_0^\infty \sin(\omega t)e^{-st}dt$$

Next, recall that $\sin(\omega t) = \dfrac{e^{i\omega t} - e^{-i\omega t}}{2i}$

Simplifying, we obtain $$\mathcal{L}[\sin \omega t] = \int_0^\infty \frac{1}{2i}\left[e^{-(s-i\omega)t} - e^{-(s+i\omega)t}\right] dt$$

$$= \frac{1}{2i}\left[-\frac{e^{-(s-i\omega)t}}{s-i\omega} + \frac{e^{-(s+i\omega)t}}{s+i\omega}\right]_0^\infty = \frac{1}{2i}\left[\frac{1}{s-i\omega} - \frac{1}{s+i\omega}\right]$$

$$= \frac{1}{2i}\left[\frac{s+i\omega-s+i\omega}{s^2+i\omega s-i\omega s+\omega^2}\right]$$

The solution is thus $$\boxed{\mathcal{L}[\sin \omega t] = \frac{\omega}{s^2+\omega^2}}$$

14.2 Laplace Transform Properties

In the previous section, we derived the Laplace transforms for three specific time domain functions. In this section, we explore Laplace transform properties and theorems, including the derivative and integral functions.

Translation Property: Show that $\mathcal{L}[f(t-\theta)] = e^{-\theta s}F(s)$.

We first apply the definition of a Laplace transform:

$$\mathcal{L}[f(t-\theta)] = \int_0^\infty f(t-\theta)e^{-st}dt$$

Note that $e^{-st} = e^{-\theta s}e^{-s(t-\theta)}$, and assume that $f(t) = 0$ for $t < 0$. We can then recast the above as

$$\mathcal{L}[f(t-\theta)] = e^{-\theta s}\int_\theta^\infty f(t-\theta)e^{-s(t-\theta)}d(t-\theta)$$

Next, we define $\tau = t - \theta$ and substitute:

$$\mathcal{L}[f(t-\theta)] = \mathcal{L}[f(\tau)] = e^{-\theta s}\int_0^\infty f(\tau)e^{-s\tau}d\tau$$

We can then see that this is the definition of the Laplace transform, so we conclude that

$$\boxed{\mathcal{L}[f(t-\theta)] = e^{-\theta s}F(s)}$$

The *translation property* is useful for modeling the time shift associated with dead time.

Linearity Property: Show that $\mathcal{L}\left[a_1 f_1(t) + a_2 f_2(t)\right] = a_1 \mathcal{L}\left[f_1(t)\right] + a_2 \mathcal{L}\left[f_2(t)\right]$

Applying the Laplace transform definition, we obtain

$$\mathcal{L}\left[a_1 f_1(t) + a_2 f_2(t)\right] = \int_0^\infty \left[a_1 f_1(t) + a_2 f_2(t)\right] e^{-st} dt$$

$$= a_1 \int_0^\infty f_1(t) e^{-st} dt + a_2 \int_0^\infty f_2(t) e^{-st} dt$$

$$= a_1 \mathcal{L}\left[f_1(t)\right] + a_2 \mathcal{L}\left[f_2(t)\right]$$

We can now conclude that

$$\boxed{\mathcal{L}\left[a_1 f_1(t) + a_2 f_2(t)\right] = a_1 \mathcal{L}\left[f_1(t)\right] + a_2 \mathcal{L}\left[f_2(t)\right]}$$

The *linearity property* states that the Laplace of a sum of functions equals the sum of the Laplace of the individual functions:

First Derivative: Show that $\mathcal{L}\left[\frac{df(t)}{dt}\right] = sF(s) - f(0)$ where $f(0)$ is in the time domain.

First, we apply the Laplace transform definition:

$$\mathcal{L}\left[\frac{df(t)}{dt}\right] = \int_0^\infty \frac{df(t)}{dt} e^{-st} dt$$

Next, we integrate by parts: $\int u\,dv = uv - \int v\,du$, so $u = e^{-st}$; $du = -se^{-st}dt$; $v = f(t)$; $dv = \frac{df(t)}{dt}dt$

Substituting, we obtain

$$\mathcal{L}\left[\frac{df(t)}{dt}\right] = f(t)e^{-st}\Big|_0^\infty + \int_0^\infty sf(t)e^{-st} dt$$

$$= [0 - f(0)] + s\int_0^\infty f(t)e^{-st} dt$$

Recognizing that the last term is the definition of the Laplace Transform, we find that

$$\boxed{\mathcal{L}\left[\frac{df(t)}{dt}\right] = sF(s) - f(0)}$$

Integration: Show that $\mathcal{L}\left[\int_0^t f(t)dt\right] = \frac{1}{s}F(s)$.

Applying the Laplace transform definition, we obtain

$$\mathcal{L}\left[\int_0^t f(t)dt\right] = \int_0^\infty \left[\int_0^t f(t)dt\right] e^{-st}dt$$

We must then integrate by parts: $u = \int_0^t f(t)dt; \quad du = f(t)dt; \quad v = -\frac{1}{s}e^{-st}; \quad dv = e^{-st}dt$

Substituting yields $$\mathcal{L}\left[\int_0^t f(t)dt\right] = -\frac{e^{-st}}{s}\int_0^t f(t)dt\Bigg|_0^\infty + \frac{1}{s}\int_0^\infty f(t)e^{-st}dt$$

Recognizing that the last term is the definition of the Laplace Transform, then we obtain

$$\boxed{\mathcal{L}\left[\int_0^t f(t)dt\right] = \frac{1}{s}F(s)}$$

Final value theorem: $$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s)$$

Initial value theorem: $$\lim_{t\to 0} f(t) = \lim_{s\to\infty} sF(s)$$

Example: What is the final value (limit as t $\to \infty$) of $F(s) = \frac{s+4}{s(s+1)(s+2)(s+3)}$?

Applying the final value theorem, we obtain

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s)$$

$$= \lim_{s\to 0}\left[s\frac{s+4}{s(s+1)(s+2)(s+3)}\right]$$

$$= \frac{4}{6}$$

☆ ☆ ☆

14.3 Moving Time Domain ODEs into the Laplace Domain

For the process control analyses we explore later, the procedure we follow will be to move our time domain equations into the Laplace domain, combine and manipulate them using algebra (which is why we bother changing them into the Laplace domain), and then move the result back into the time domain for implementation. Hence, we must become comfortable with moving time domain equations into and out of the Laplace domain.

We solved the ODEs below in the time domain in chapter 13. While these examples all demonstrate moving from the time domain into the Laplace domain, following the steps in reverse will achieve the opposite. When working through these examples, note that the boundary (initial) conditions are applied early in the transformation process. Recall that applying the boundary conditions is one of the last steps when solving ODEs in the time domain.

Example 1: Move the following time domain equation to the Laplace domain:

$$\frac{dy(t)}{dt} + 2y(t) = 4t \quad \text{where} \quad y(0) = 1$$

Consulting the Laplace transform table in Appendix B, we see that

$$\mathcal{L}\left[\frac{dy(t)}{dt}\right] + 2\mathcal{L}[y(t)] = \mathcal{L}[4t]$$

Applying initial conditions, $$[sY(s) - y(0)] + 2Y(s) = \frac{4}{s^2}$$

and simplifying, we obtain $$Y(s)[s + 2] = \frac{4}{s^2} + 1 = \frac{4 + s^2}{s^2}$$

The equation can now be written in the Laplace domain as

$$\boxed{Y(s) = \frac{4 + s^2}{s^2(s + 2)}}$$

Example 2: Move the following time domain equation into the Laplace domain:

$$\frac{dx(t)}{dt} = x(t) + \sin t \quad \text{where} \quad x(0) = 0$$

Consulting the Laplace transform table in Appendix B, we find that

$$\mathcal{L}\left[\frac{dx(t)}{dt}\right] - \mathcal{L}[x(t)] = \mathcal{L}[\sin t]$$

Applying initial conditions, $[sX(s) - x(0)] - X(s) = \frac{1}{s^2 + 1}$

we can now write the equation in the Laplace domain as

$$\boxed{X(s) = \frac{1}{(s^2 + 1)(s - 1)}}$$

Example 3: Move the following time domain equation into the Laplace domain:

$$\frac{d^2y(t)}{dt^2} + 4\frac{dy(t)}{dt} + 3y(t) = t \quad \text{where} \quad y(0) = \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

Consulting the Laplace transform table, we see that

$$\mathcal{L}\left[\frac{d^2y(t)}{dt^2}\right] + 4\mathcal{L}\left[\frac{dy(t)}{dt}\right] + 3\mathcal{L}[y(t)] = \mathcal{L}[t]$$

Applying initial conditions yields

$$\left[s^2Y(s) - sy(0) - \left.\frac{dy(t)}{dt}\right|_{t=0}\right] + [4sY(s) - 4y(0)] + 3Y(s) = \frac{1}{s^2}$$

Simplifying, we obtain $Y(s)[s^2 + 4s + 3] = \frac{1}{s^2}$

The equation can now be written in the Laplace domain as

$$\boxed{Y(s) = \frac{1}{s^2(s^2 + 4s + 3)} = \frac{1}{s^2(s + 3)(s + 1)}}$$

Example 4: Move the following time domain equation into the Laplace domain:

$$\frac{d^2y(t)}{dt^2} + 2\frac{dy(t)}{dt} + y(t) = t^2 \quad \text{where} \quad y(0) = 0 \quad \text{and} \quad \left.\frac{dy(t)}{dt}\right|_{t=0} = 1$$

Consulting the Laplace transform table in Appendix B, we see that

$$\mathcal{L}\left[\frac{d^2y(t)}{dt^2}\right] + 2\mathcal{L}\left[\frac{dy(t)}{dt}\right] + \mathcal{L}[y(t)] = \mathcal{L}[t^2]$$

Applying initial conditions yields

$$\left[s^2 Y(s) - sy(0) - \left.\frac{dy(t)}{dt}\right|_{t=0} \right] + [2sY(s) - 2y(0)] + Y(s) = \frac{2}{s^3}$$

Simplifying, we obtain $$Y(s)[s^2 + 2s + 1] = \frac{2}{s^3} + 1 = \frac{2 + s^3}{s^3}$$

The equation can now be written in the Laplace domain as

$$\boxed{Y(s) = \frac{2 + s^3}{s^3(s^2 + 2s + 1)} = \frac{2 + s^3}{s^3(s+1)^2}}$$

★ ★ ★

Example 5: Move the following time domain equation into the Laplace domain:

$$2\frac{d^2 y(t)}{dt^2} + 2\frac{dy(t)}{dt} + y(t) = 2 \quad \text{where} \quad y(0) = \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

Consulting the Laplace transform table in Appendix B, we can see that

$$2\mathcal{L}\left[\frac{d^2 y(t)}{dt^2}\right] + 2\mathcal{L}\left[\frac{dy(t)}{dt}\right] + \mathcal{L}[y(t)] = \mathcal{L}[2]$$

Applying initial conditions yields

$$\left[2s^2 Y(s) - 2sy(0) - 2\left.\frac{dy(t)}{dt}\right|_{t=0} \right] + [2sY(s) - 2y(0)] + Y(s) = \frac{2}{s}$$

Simplifying, we obtain $$Y(s) = \frac{2}{s(2s^2 + 2s + 1)} = \frac{1}{s\left(s^2 + s + \frac{1}{2}\right)}$$

The equation can now be written in the Laplace domain as

$$\boxed{Y(s) = \frac{2}{s(2s^2 + 2s + 1)} = \frac{1}{s\left[s - (-\frac{1}{2} + \frac{1}{2}i)\right]\left[s - (-\frac{1}{2} - \frac{1}{2}i)\right]}}$$

★ ★ ★

Example 6: Move the following time domain equation into the Laplace domain:

$$\frac{d^2 x(t)}{dt^2} + x(t) = \sin(2t) \quad \text{where } x(0) = 0 \text{ and } \left.\frac{dx(t)}{dt}\right|_{t=0} = 1$$

Consulting the Laplace transform table in Appendix B, we see that

$$\mathcal{L}\left[\frac{d^2x(t)}{dt^2}\right] + \mathcal{L}[x(t)] \;=\; \mathcal{L}[\sin(2t)]$$

Applying initial conditions yields

$$\left[s^2X(s) - sx(0) - \left.\frac{dx(t)}{dt}\right|_{t=0}\right] + X(s) = \frac{2}{s^2+4}$$

Simplifying, we obtain $$X(s)\left(s^2+1\right) - 1 = \frac{2}{s^2+4}$$

The equation can now be written in the Laplace domain as

$$\boxed{X(s) = \frac{s^2+6}{(s^2+4)(s^2+1)}}$$

14.4 Moving Laplace Domain ODEs into the Time Domain

Now that we have some understanding of Laplace transforms, let's consider process oriented challenges.

Example 1: A process response to a unit step forcing function is $Y(s) = \frac{1}{s(3s+1)}$. What is the original process ODE in the time domain?

Solution: From the Laplace table (Appendix B), we know that

$$\mathcal{L}[\text{unit step}] = \frac{1}{s}$$

Applying this, we can see that our original equation is actually

$$Y(s) = \frac{1}{(3s+1)}\left[\frac{1}{s}\right]$$

Simplifying, we find that $$Y(s)\left[3s+1\right] = \frac{1}{s}$$

so $$3sY(s) + Y(s) = \frac{1}{s}$$

Assuming y(0) = 0, we obtain $$3\left[sY(s) - y(0)\right] + Y(s) = \frac{1}{s}$$

From Appendix B, we can see that the inverse Laplace, $\mathcal{L}^{-1}$, is

$$3\frac{dy(t)}{dt} + y(t) = 1 \qquad \text{where } y(0) = 0$$

Example 2: A process response to a forcing function U(s) is $Y(s) = \frac{K_P}{\tau_P s + 1} U(s)$. What is the original process ODE in the time domain?

Solution: First, we rearrange the equation as such:

$$Y(s)\left[\tau_P s + 1\right] = K_P U(s)$$

$$Y(s)\left[\tau_P s\right] + Y(s) = K_P U(s)$$

Next, we assume that y(0) = 0: $\tau_P\left[sY(s) - y(0)\right] + Y(s) = K_P U(s)$

From the table (Appendix B), we can see that the inverse Laplace transform, $\mathcal{L}^{-1}$, can be written

$$\tau_P \frac{dy(t)}{dt} + y(t) = K_P u(t) \qquad \text{where } y(0) = 0$$

With this result, we note that $\frac{Y(s)}{U(s)} = \frac{K_P}{\tau_P s + 1}$ is called the *transfer function* for the first order without dead time ODE.

Example 3: A process response to a forcing function U(s) is $Y(s) = \frac{K_p}{\tau_n^2 s^2 + 2\tau_n \xi s + 1} U(s)$. What is the original process ODE in the time domain?

Solution: We rearrange the equation as follows:

$$Y(s)\left(\tau_n^2 s^2 + 2\tau_n \xi s + 1\right) = K_p U(s)$$

$$Y(s)\left(\tau_n^2 s^2\right) + Y(s)\left(2\tau_n \xi s\right) + Y(s) = K_p U(s)$$

Next, we assume that y(0) = 0 and $\left.\frac{dy(t)}{dt}\right|_{t=0} = 0$:

$$Y(s)\left(\tau_n^2 s^2 - sy(0) - \left.\frac{dy(t)}{dt}\right|_{t=0}\right) + 2\tau_n\xi\big(sY(s) - y(0)\big) + Y(s) = K_p U(s)$$

From Appendix B, we can see that the inverse Laplace transform, $\mathcal{L}^{-1}$, can be written as

$$\tau_n^2 \frac{d^2y(t)}{dt} + 2\tau_n\xi\frac{dy(t)}{dt} + y(t) = K_p u(t) \qquad \text{where } y(0) = 0 \text{ and } \left.\frac{dy(t)}{dt}\right|_{t=0} = 0$$

Similar to the previous example, we note that $\dfrac{Y(s)}{U(s)} = \dfrac{K_p}{\tau_n^2 s^2 + 2\tau_n\xi s + 1}$ is the *transfer function* for the second order without dead time ODE.

Example 4: Consider the following system: $\dfrac{Y(s)}{U(s)} = \dfrac{3}{s^2 + s + 1}$. Is this system stable? Does it have a natural tendency to oscillate?

Solution: Comparing to the general form of Example 3, we observe that $K_P = 3$; $\tau_n^2 = 1$; $2\tau_n\xi = 1$; thus, $\tau_n = 1$; and $\xi = 0.5$. Since $0 < \xi < 1$, the process is stable and underdamped (which means it does have a natural tendency to oscillate).

14.5 Exercises

Q-14.1 Showing all steps, derive the Laplace transform of cos(ωt).

Q-14.2 Showing all steps, derive the Laplace transform of e^{at}.

15. Transfer Functions

15.1 Process Transfer Functions

A process transfer function, $G_P(s)$, is an equation in the Laplace domain that describes the dynamic response of the measured process variable to changes in the manipulated process variable (controller output signal).

Time Domain: Consider the general time domain ODE describing a second order process:

$$\tau_n^2 \frac{d^2y(t)}{dt^2} + 2\tau_n\xi\frac{dy(t)}{dt} + y(t) = K_P u(t-\theta_P) \qquad \text{where} \quad y(0) = 0,\ \left.\frac{dy(t)}{dt}\right|_{t=0} = 0 \tag{15.1}$$

As we learned in Chapter 13, the characteristic equation is

$$\tau_n^2 s^2 + 2\tau_n\xi s + 1 = (s-p_1)(s-p_2) = 0$$

Also, the total solution is $y(t) = C_1 e^{p_1 t} + C_2 e^{p_2 t} + y_P(t)$

or $y(t) = (C_1 + C_2 t)e^{p_1 t} + y_P(t)$

We can express this as $y(t) = e^{(\text{real part})t}\left[C_R \cos(\text{imag part } t) - C_I \sin(\text{imag part } t)\right] + y_P(t)$

Laplace Domain: As detailed in example 2 below, the transfer function for the ODE of Eq. 15.1 is

$$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P e^{-\theta_P s}}{\tau_n^2 s^2 + 2\tau_n\xi s + 1} = \frac{K_P e^{-\theta_P s}}{(s-p_1)(s-p_2)}$$

Note that the denominator of a transfer function is the characteristic equation of the time domain complementary solution. This is true for all transfer functions, including process transfer functions as shown below.

Recall that the roots of the characteristic equation indicate a system's stability and natural tendency to oscillate. An unstable system results if any root has a positive real part, $e^{+p_1 t}$, because that term will grow without bound as time t grows to infinity. A stable system results if all roots have negative real parts, $e^{-p_1 t}$, as these terms all die out (go to zero) as t grows to infinity. The tendency to oscillate is a consequence of sine and cosine terms in the solution that results from imaginary roots.

Because the denominator of a transfer function is the characteristic equation of the time domain complementary solution, then the roots of the denominator of a transfer function (called poles) similarly indicate system stability and tendency to oscillate. This knowledge will prove useful in control system analysis and design studies in the following chapters.

Example 1: Derive the following FOPDT (first order plus dead time) transfer function:

$$\tau_p \frac{dy(t)}{dt} + y(t) = K_p u(t-\theta_P) \qquad \text{where } y(0) = 0 \tag{15.2}$$

Solution: Consulting the Laplace table (Appendix B), we see that

$$\tau_p[sY(s) - y(0)] + Y(s) = K_p e^{-\theta_P} U(s)$$

Applying the initial condition yields

$$(\tau_p s + 1)Y(s) = K_P e^{-\theta_P} U(s)$$

We can then rearrange to form the FOPDT transfer function:

$$\boxed{G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}}$$

Recall that the complementary equation for Eq. 15.2 is

$$\tau_p \frac{dy(t)}{dt} + y(t) = 0$$

so the characteristic equation is $\tau_p s + 1 = 0$

This confirms the observation above that the denominator of a transfer function is the characteristic equation of the time domain complementary solution.

Example 2: Derive the SOPDT (second order plus dead time) transfer function:

$$\tau_n^2 \frac{d^2 y(t)}{dt^2} + 2\tau_n \xi \frac{dy(t)}{dt} + y(t) = K_P u(t-\theta_P) \qquad \text{where } y(0) = 0,\ \left.\frac{dy(t)}{dt}\right|_{t=0} = 0 \tag{15.3}$$

Answer: Consulting the Laplace table, we can see that

$$\tau_n^2 \left[s^2 Y(s) - sy(0) - \frac{dy(0)}{dt} \right] + 2\tau_n \xi [sY(s) - y(0)] + Y(s) = K_P e^{-\theta_P s} U(s)$$

Applying initial conditions yields $\tau_n^2 s^2 Y(s) + 2\tau_n \xi s Y(s) + Y(s) = K_P e^{-\theta_P s} U(s)$

We can then rearrange to form the SOPDT transfer function:

$$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_P e^{-\theta_P s}}{\tau_n^2 s^2 + 2\tau_n \xi s + 1}$$

Recall that the complementary equation for Eq. 15.3 is

$$\tau_n^2 \frac{d^2 y(t)}{dt^2} + 2\tau_n \xi \frac{dy(t)}{dt} + y(t) = 0$$

so the characteristic equation is $\tau_n^2 s^2 + 2\tau_n \xi s + 1 = 0$

This again confirms the observation that the denominator of a transfer function is the characteristic equation of the complementary solution.

15.2 Controller Transfer Functions

Analogous to a process transfer functions, control algorithm transfer functions, $G_C(t)$, are Laplace domain equations that describe the dynamic behavior of the controller output signal as it responds to changes in the controller error. The general form of a controller transfer function is

$$G_C(s) = \frac{U(s)}{E(s)} \tag{15.4}$$

For all derivations, we observe that at the design level of operation, the control error should be zero. Hence, when $u(t) = u_{bias}$, then $e(t) = 0$. Consistent with this observation, we use the following perturbation variable definitions:

$$u^P(t) = u(t) - u_{bias} \tag{15.5a}$$

$$e^P(t) = e(t) - 0 \tag{15.5b}$$

Example: P-Only Control

The P-Only control algorithm is $u(t) = u_{bias} + K_C e(t)$

Rearranging to group like terms gives us

$$u(t) - u_{bias} = K_C[e(t) - 0]$$

Employing Eq. 15.5 yields $u^P(t) = K_C\, e^P(t)$

Using the Laplace table, we can move into the Laplace domain:

$$U(s) = K_C E(s)$$

This gives us the P-Only transfer function:

$$\boxed{G_C(s) = \frac{U(s)}{E(s)} = K_C} \tag{15.6}$$

Example: PI Control

The PI control algorithm is $u(t) = u_{bias} + K_C e(t) + \frac{K_C}{\tau_I}\int_0^t e(t)dt$

Rearranging to group like terms gives us

$$u(t) - u_{bias} = K_C[e(t) - 0] + \frac{K_C}{\tau_I}\int_0^t [e(t) - 0]dt$$

Employing Eq. 15.5 yields $u^P(t) = K_C e^P(t) + \frac{K_C}{\tau_I}\int_0^t e^P(t)dt$

We can then use the Laplace tables and assume the process starts at steady state, to give us

$$U(s) = K_C E(s) + \frac{K_C}{\tau_I}\left(\frac{1}{s}E(s)\right)$$

This results in the PI transfer function:

$$\boxed{G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s}\right)} \tag{15.7}$$

Example: PID Control

The PID w/ derivative on error control algorithm is

$$u(t) = u_{bias} + K_C e(t) + \frac{K_C}{\tau_I}\int_0^t e(t)dt + K_C\tau_D\frac{de(t)}{dt}$$

Rearranging to group like terms gives us

$$u(t) - u_{bias} = K_C[e(t) - 0] + \frac{K_C}{\tau_I}\int_0^t [e(t) - 0]dt + K_C\tau_D\frac{d[e(t) - 0]}{dt}$$

Employing Eq. 15.5 yields $u^p(t) = K_C e^p(t) + \frac{K_C}{\tau_I}\int_0^t e^p(t)dt + K_C\tau_p\frac{de^P(t)}{dt}$

We then consult the Laplace tables and assume the process starts at steady state to give us

$$U(s) = K_C E(s) + \frac{K_C}{\tau_I}\frac{1}{s}E(s) + K_C\tau_D s E(s)$$

This results in the PID transfer function:

$$\boxed{G_C(s) = \frac{U(s)}{E(s)} = K_C\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)} \qquad (15.8)$$

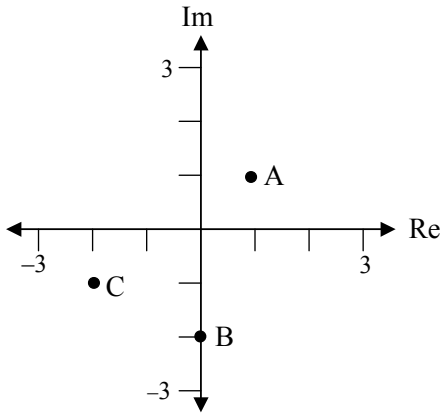
✯ ✯ ✯

15.3 Poles of a Transfer Function and Root Locus

The roots of the denominator of a transfer function are traditionally called the *poles* of the transfer function. Because the Laplace domain is defined in the complex plane (i.e. $s = a$ + bi), a pole has a real part and an imaginary part. A root locus plot shows the location of a system's roots/poles on the complex plane.

Example 1: Locate these points on the *s* plane: A = 1 + *i*; B = –2*i*; C = –2 –*i*

Solution:



s plane

A negative real part of the root means the term $e^{-p_1 t}$ goes to zero as time increases to infinity. Thus, the dynamics will die out and the system will be stable. All of the roots must have negative real parts for this to hold true. If even one root has a positive real part, it will eventually dominate and push the system to instability.

Real parts are negative on the left-hand side of the *s* plane and they are positive on the right-hand side. Consequently, all roots/poles describing a stable system lie in the left-hand side of the complex plane.

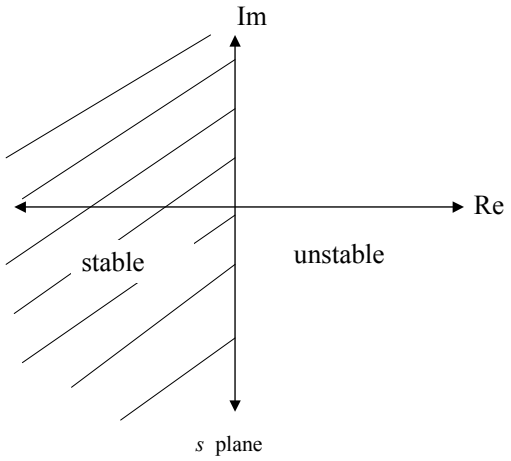


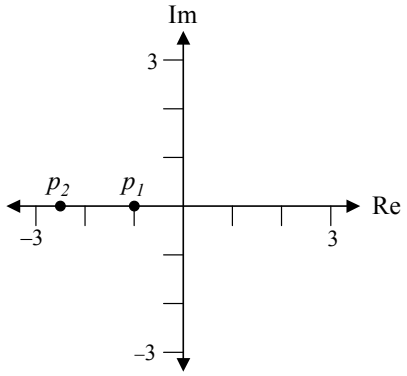
Figure 15.1 – Stable and unstable sides of the complex plane

☆ ☆ ☆

Example 2: What do the poles of the following transfer function indicate about the system behavior?

$$G_{sys}(s) = \frac{1}{(s+1)(s+2.5)}$$

Solution: The roots/poles are $p_1 = -1.0$ and $p_2 = -2.5$ as plotted on the s plane below:



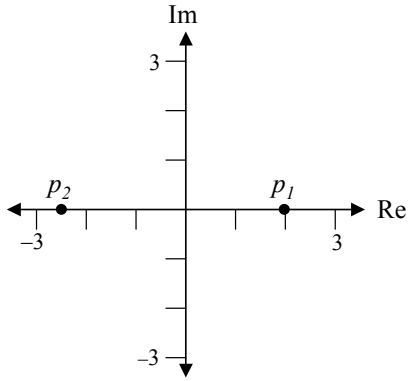
The plot indicates that the system is stable and does not oscillate because the poles are real and negative.

☆ ☆ ☆

Example 3: What do the poles of the following transfer function indicate about the system behavior?

$$G_{sys}(s) = \frac{1}{(s-2)(s+2.5)}$$

Solution: The roots/poles are $p_1 = +2.0$ and $p_2 = -2.5$ as plotted on the s plane below:

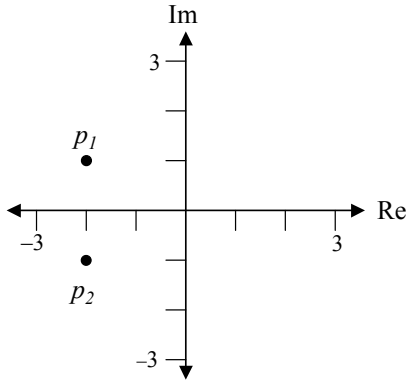


The plot indicates that the system is unstable because one of the poles is positive and hence will grow without bound as time passes. The system will not oscillate as it goes unstable because neither pole has imaginary parts.

Example 4: What do the poles of the following transfer function indicate about the system behavior?

$$G_{sys}(s) = \frac{1}{(s^2 + 4s + 5)}$$

Solution: The roots/poles are $p_1 = -2 + i$ and $p_2 = -2 - i$ as plotted on the s plane below:



Because the real part of both poles is negative, the system is stable. The system will have a natural tendency to oscillate due to the imaginary part of the roots/poles.

15.4 Poles as Complex Conjugates

If the roots/poles have an imaginary component, they will always be present as *complex conjugate* pairs of the form $a \pm bi$. That is, the pair will have the same real part and reflecting imaginary parts.

The property most appealing about the complex conjugate form is that when the pairs are added together or multiplied together, they yield real numbers with no imaginary parts. This is important because the processes we work on are real, and the solutions we devise must also be real (would you want your boss to give you an imaginary raise?).

Example 5: Show that $1 + 2i$ and $1 - 2i$ are complex conjugates.

Answer: Beyond the fact that they have the proper $a \pm bi$ form, when we add and multiply them together, they yield real numbers with no imaginary parts:

$$(1 + 2i) + (1 - 2i) = 2 \quad \text{and} \quad (1 + 2i)(1 - 2i) = (1 + 2i - 2i - 4i^2) = 5$$

15.5 Poles of the Transfer Function Indicate System Behavior

As illustrated in Fig 15.2, as the roots/poles move farther out on the negative real axis, the dynamics become faster. As the roots/poles move farther out on the imaginary axis, the oscillatory nature of the response increases.

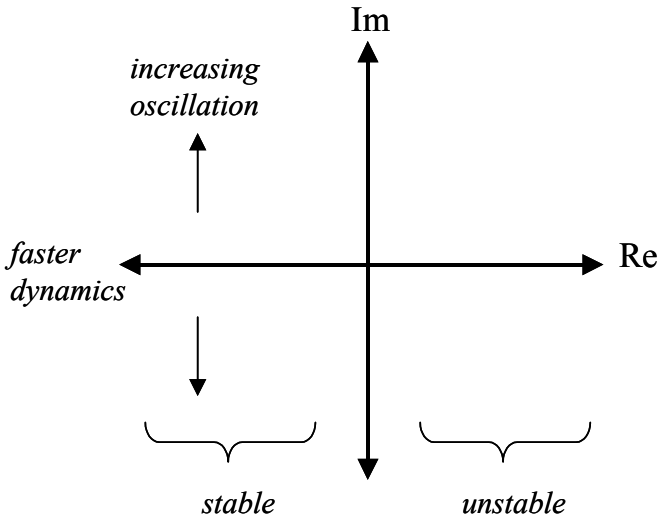
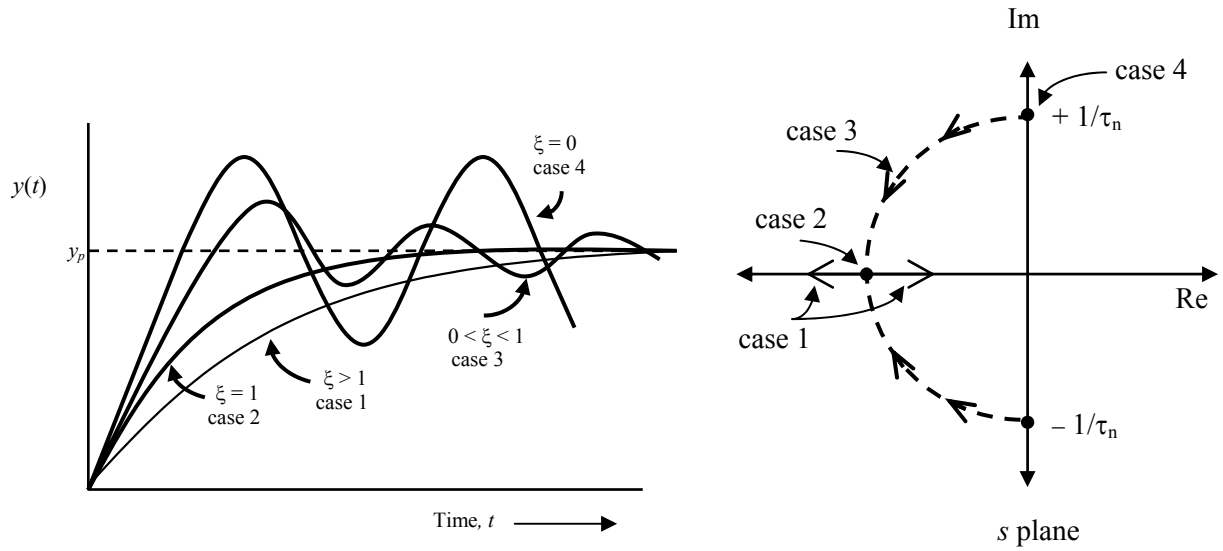


Figure 15.2 –Pole location indicates system behavior

To show that the roots/poles are related to system behavior, we can perform an analysis similar to the one done in the time domain in Chapter 13. Here we consider the same four cases presented earlier in the time domain.



Time Domain *Laplace Domain*

Figure 15.3 - Stable behaviors of underdamped system based on value of ξ

Case 1: ξ > 1 (overdamped) $p_1, p_2 = \dfrac{-\xi \pm \sqrt{\xi^2 - 1}}{\tau_n}$; $y(t) = y_p(t) + C_1 e^{-p_1 t} + C_2 e^{-p_2 t}$

Transfer function: $G(s) = \dfrac{K_P}{(s + p_1)(s + p_2)}$

The overdamped response has distinct roots/poles on the negative real axis. The time domain response becomes slower and slower as ξ gets larger, but it is always stable and never oscillates.

Case 2: ξ = 1 (critically damped) $p_1, p_2 = -\dfrac{1}{\tau_n}$; $y(t) = y_p(t) + (C_1 + C_2 t)e^{-t/\tau_n}$

Transfer function: $G(s) = \dfrac{K_p}{(s + \dfrac{1}{\tau_n})(s + \dfrac{1}{\tau_n})}$

When critically damped, the roots/poles are repeated on the negative real axis. There is no oscillation in the response because there is no imaginary part leading to sine and cosine terms in the roots/poles.

Case 3: 0 < ξ < 1 (underdamped) $p_1, p_2 = \dfrac{-\xi \pm i\sqrt{1-\xi^2}}{\tau_n}$;

$$y(t) = y_p(t) + e^{-\xi t/\tau_n}\left(C_R \cos(\frac{\sqrt{1-\xi^2}}{\tau_n}t) - C_I \sin(\frac{\sqrt{1-\xi^2}}{\tau_n}t)\right)$$

Transfer function:
$$G(s) = \frac{K_P}{\left[s - \left(-\frac{\xi}{\tau_n} + \frac{\sqrt{1-\xi^2}}{\tau_n}i\right)\right]\left[s - \left(-\frac{\xi}{\tau_n} - \frac{\sqrt{1-\xi^2}}{\tau_n}i\right)\right]}$$

As ξ varies in the continuum from zero to one, the roots/poles move from the imaginary axis to the real axis. The roots/poles are always in complex conjugate pairs of the form *a ± bi*. The time domain response shows oscillation that dampens. The closer ξ gets to 1, the greater the damping.

Case 4: ξ = 0 (undamped) $p_1, p_2 = \pm\dfrac{i}{\tau_n}$; $y(t) = y_p(t) + \left(C_R \cos(\frac{t}{\tau_n}) - C_I \sin(\frac{t}{\tau_n})\right)$

Transfer function:
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_p}{\left(s - \frac{i}{\tau_n}\right)\left(s + \frac{i}{\tau_n}\right)}$$

At the limit of stability, the roots/poles are located on the imaginary axis and have no real part. The time domain response shows oscillation that neither grows nor dies out. Thus, the oscillations maintain constant form and continue forever.

Case 5: ξ < 0 (unstable)

All values of the damping factor, ξ, that are less than zero yield a solution that has a positive real part. Hence, as time passes, e^{+t} will grow without bound, an unstable result. All roots/poles will fall on the right-hand side of the *s*-plane.

15.6 Exercises

Q-15.1 What are the steady state gain and time constants for a process described by the following transfer function (pay careful attention to the form of the equation)?

$$G_P(s) = \frac{7}{(s+10)(s+2)}$$

Q-15.2 Starting with the Laplace domain transfer function, show all steps and derive the time domain ODE for the Second Order Plus Dead Time Integrating form:

Laplace: $$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_p e^{-\theta_P s}}{s(\tau_p s + 1)}$$

Time: $$\tau_p \frac{d^2 y(t)}{dt^2} + \frac{dy(t)}{dt} = K_p u(t - \theta_P)$$

Q-15.3 Starting with the time domain ODE, show all steps and derive the Laplace domain transfer function for the Second Order Plus Dead Time Overdamped with Lead Time form:

Time: $$\tau_1 \tau_2 \frac{d^2 y(t)}{dt^2} + (\tau_1 + \tau_2)\frac{dy(t)}{dt} + y(t) = K_p\left[u(t - \theta_p) + \tau_L \frac{du(t - \theta_P)}{dt}\right]$$

Laplace: $$G_P(s) = \frac{Y(s)}{U(s)} = \frac{K_p(\tau_L s + 1)e^{-\theta_P s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

Q-15.4 Starting with the continuous position form of the Integral Only controller, show all steps to derive the I-Only transfer function.

Q-15.5 The dynamics of a process are described by the following ODE, which is in perturbation variable form:

$$6\frac{d^2 y(t)}{dt^2} + 5\frac{dy(t)}{dt} + 4y(t) = 3u(t)$$

a) Showing all steps, determine the transfer function, $G_P(s)$, for this process.

b) Determine the poles of the transfer function.

c) Plot the poles on the complex plane.

d) Based on the pole locations, describe the general dynamic behavior of this process.

Be sure to briefly explain your reasoning.

16. Block Diagrams

16.1 Combining Transfer Functions Using Block Diagrams

Laplace domain transfer functions enable us to manipulate complex equations using simple algebra. For example, consider two non-interacting draining tanks. If we assume that drain flow rate is proportional to hydrostatic head, we can write the equations using perturbation variables:

The tank 1 ODE is $$A_{C1}\frac{dh_1^P(t)}{dt}+\alpha_1 h_1^P(t)=F_0^P(t) \qquad \text{where } h_1^P(0)=0 \tag{16.1}$$

In the Laplace domain, Eq. 16.1 becomes

$$A_{C1}sH_1(s)+\alpha_1 H_1(s)=F_0(s) \tag{16.2}$$

The tank 1 transfer function is thus $$G_{P1}(s)=\frac{H_1(s)}{F_0(s)}=\frac{1/\alpha_1}{(A_{C1}/\alpha_1)s+1} \tag{16.3}$$

The tank 2 ODE is $$A_{C2}\frac{dh_2^P(t)}{dt}+\alpha_2 h_2^P(t)=\alpha_1 h_1^P(t) \quad \text{where } h_2^P(0)=0 \tag{16.4}$$

In the Laplace domain, Eq. 16.4 becomes

$$A_{C2}sH_2(s)+\alpha_2 H_2(s)=H_1(s) \tag{16.5}$$

The tank 2 transfer function is thus $$G_{P2}(s)=\frac{H_2(s)}{H_1(s)}=\frac{\alpha_1/\alpha_2}{(A_{C2}/\alpha_2)s+1} \tag{16.6}$$

With the above as a basis, we write general coupled process ODEs as

Process 1: $$\tau_{P1}\frac{dy_1(t)}{dt}+y_1(t)=K_{P1}u(t);\ y_1(0)=0 \quad \Rightarrow \quad G_{P1}(s)=\frac{Y_1(s)}{U(s)}=\frac{K_{P1}}{\tau_{P1}s+1} \tag{16.7}$$

Process 2: $$\tau_{P2}\frac{dy_2(t)}{dt}+y_2(t)=K_{P2}y_1(t);\ y_2(0)=0 \quad \Rightarrow \quad G_{P2}(s)=\frac{Y_2(s)}{Y_1(s)}=\frac{K_{P2}}{\tau_{P2}s+1} \tag{16.8}$$

Combining the time domain ODE's of Eqs. 16.7 and 16.8 into a single second-order differential equation describing how $y_2(t)$ responds to changes in $u(t)$ requires manipulation of ODEs as follows:

Solve Eq. 16.8 for $y_1(t)$: $$y_1(t)=\frac{\tau_{P2}\frac{dy_2(t)}{dt}+y_2(t)}{K_{P2}} \tag{16.9}$$

Take the derivative of Eq. 16.9:

$$\frac{dy_1(t)}{dt} = \frac{\tau_{P2}\dfrac{d^2y_2(t)}{dt^2} + \dfrac{dy_2(t)}{dt}}{K_{P2}} \tag{16.10}$$

Substitute Eqs. 16.9 and 16.10 into Eq. 16.7:

$$\tau_{P1}\frac{\tau_{P2}\dfrac{d^2y_2(t)}{dt^2} + \dfrac{dy_2(t)}{dt}}{K_{P2}} + \frac{\tau_{P2}\dfrac{dy_2(t)}{dt} + y_2(t)}{K_{P2}} = K_{P1}u(t) \tag{16.11}$$

Multiply both sides by K_{P2} and combine like terms:

$$\boxed{\tau_{P1}\tau_{P2}\frac{d^2y_2(t)}{dt^2} + (\tau_{P1} + \tau_{P2})\frac{dy_2(t)}{dt} + y_2(t) = K_{P1}K_{P2}u(t)} \tag{16.12}$$

In the Laplace domain, we combine the transfer functions using simple algebra, which is the reason for converting from the time domain into the Laplace domain and back:

$$G_{\text{system}}(s) = G_{P1}(s)G_{P2}(s) = \frac{\cancel{Y_1(s)}}{U(s)}\frac{Y_2(s)}{\cancel{Y_1(s)}} = \frac{Y_2(s)}{U(s)} = \left(\frac{K_{P1}}{\tau_{P1}s+1}\right)\left(\frac{K_{P2}}{\tau_{P2}s+1}\right) \tag{16.13}$$

and thus

$$\frac{Y_2(s)}{U(s)} = \frac{K_{P1}K_{P2}}{\tau_{P1}\tau_{P2}s^2 + (\tau_{P1} + \tau_{P2})s + 1} \tag{16.14}$$

As we expect, converting Eq. 16.14 back to the time domain yields Eq. 16.12. This comparison of time domain versus Laplace domain equation manipulation helps demonstrate the benefit of using the Laplace domain in our subsequent analyses.

A block diagram is a convenient way to visualize the combination and manipulation of Laplace domain equations. As shown in Fig. 16.1, we use a summer block (a circle) to add block inputs and a multiplier block (a square) to multiply block inputs:

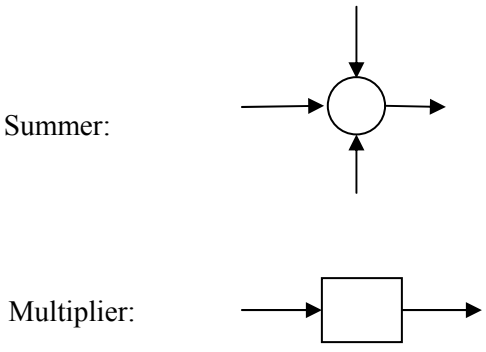
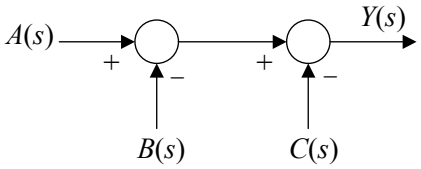


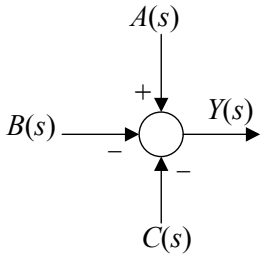
Figure 16.1 – Blocks used to create a Laplace domain block diagram

Example 1: Show this manipulation using block diagrams: $Y(s) = A(s) - B(s) - C(s)$

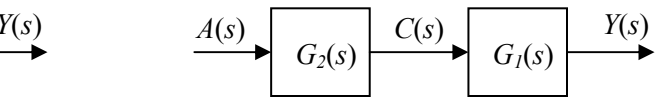
Solution: Below are two of several possibilities:



$[A(s) - B(s)] - C(s) = Y(s)$

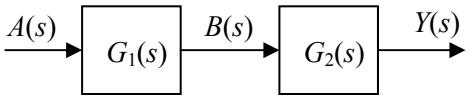


$A(s) - B(s) - C(s) = Y(s)$



Example 2: Show this manipulation using block diagrams: $Y(s) = A(s)G_1(s)G_2(s)$

Solution: Below are two of several possibilities:

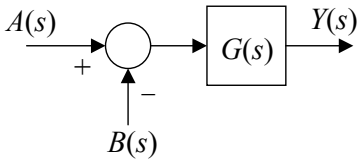


$Y(s) = B(s)G_2(s)$
$B(s) = A(s)G_1(s)$
$Y(s) = A(s)G_1(s)G_2(s)$

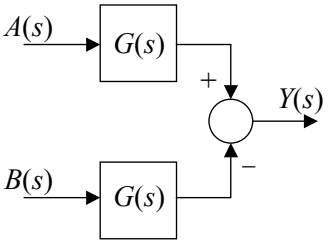
$Y(s) = C(s)G_1(s)$
$C(s) = A(s)G_2(s)$
$Y(s) = A(s)G_1(s)G_2(s)$

Example 3: Show using block diagrams: $Y(s) = [A(s) - B(s)]G(s)$

Solution: Below are two of several possibilities:



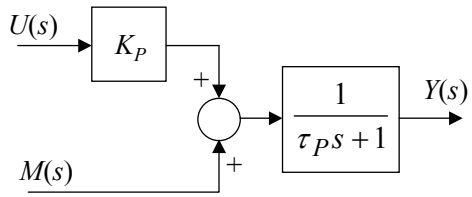
$Y(s) = [A(s) - B(s)]G(s)$



$Y(s) = A(s)G(s) - B(s)G(s) = [A(s) - B(s)]G(s)$

Example 4: Show using block diagrams: $Y(s) = \frac{K_P}{\tau_P s + 1} U(s) + \frac{1}{\tau_P s + 1} M(s)$

Solution: Below is one of several possibilities:



$$Y(s) = [U(s)K_P + M(s)]\left[\frac{1}{\tau_P s + 1}\right] = \frac{K_P}{\tau_P s + 1} U(s) + \frac{1}{\tau_P s + 1} M(s)$$

☆ ☆ ☆

16.2 The Closed Loop Block Diagram

As shown in Fig. 16.2, the closed loop block diagram in the time domain is

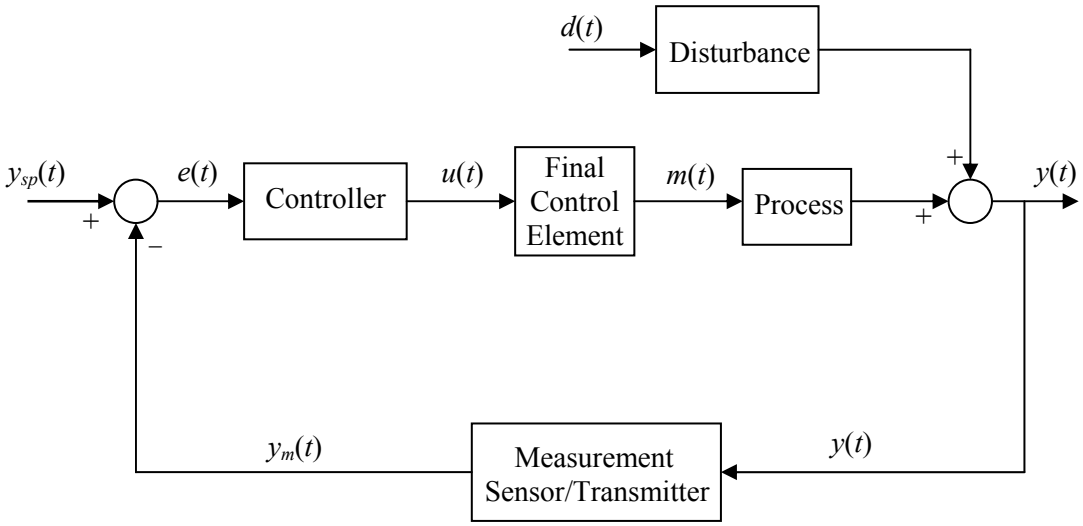


Figure 16.2 – Closed Loop Block Diagram in Time Domain

In the Laplace domain, the closed loop block diagram is as shown in Fig. 16.3:

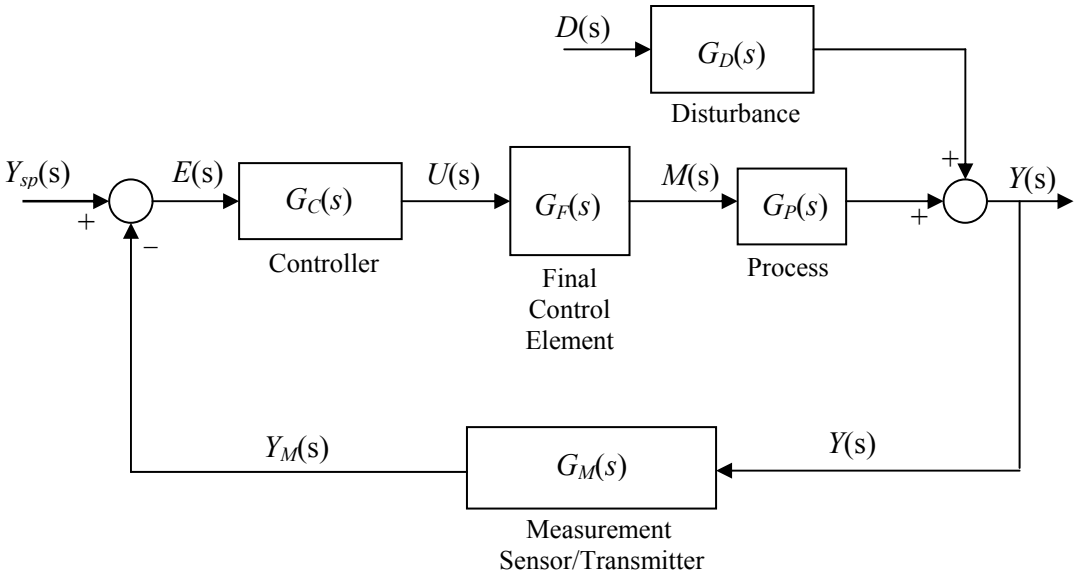


Figure 16.3 – Closed Loop Block Diagram in Laplace Domain

Notice that it is not only the process and controller that have transfer functions describing their dynamic behavior. As shown in the block diagram of Fig. 16.3, final control element (e.g. valve, pump) and measurement sensor also have transfer functions:

$$G_C(s) = \frac{U(s)}{E(s)} \qquad G_F(s) = \frac{M(s)}{U(s)} \qquad G_M(s) = \frac{Y_M(s)}{Y(s)} \tag{16.15}$$

controller final control element measurement sensor

The block diagram shows that:

$$E(s) = Y_{sp}(s) - Y_M(s) \tag{16.16}$$

The block diagram also shows that the transfer function for the measured process variable is somewhat more complicated when a disturbance variable is included. Following the block diagram rules presented above, the transfer function is:

$$Y(s) = M(s)G_P(s) + D(s)G_D(s) \tag{16.17}$$

16.3 Closed Loop Block Diagram Analysis

Building on the principles discussed earlier in this chapter, we can write a series of equations as we step around the closed loop block diagram in an orderly fashion. A convenient place to start in the balance is with process variable $Y(s)$ as it exits the block diagram on the right. The equations thus develop as:

$$Y(s) = M(s)G_P(s) + D(s)G_D(s) \tag{16.18a}$$

$$M(s) = U(s)G_F(s) \tag{16.18b}$$

$$U(s) = E(s)G_C(s) = [Y_{sp}(s) - Y_M(s)]G_C(s) \tag{16.18c}$$

$$Y_M(s) = Y(s)G_M(s) \tag{16.18d}$$

Substituting Eq. 16.18b into Eq. 16.18a, and Eq. 16.18d into Eq. 16.18c yields:

$$Y(s) = U(s)G_F(s)G_P(s) + D(s)G_D(s) \tag{16.19a}$$

$$U(s) = [Y_{sp}(s) - Y(s)G_M(s)]G_C(s) \tag{16.19b}$$

Substituting Eq. 16.19b into Eq. 16.19a yields:

$$Y(s) = [Y_{sp}(s) - Y(s)G_M(s)]G_C(s)G_F(s)G_P(s) + D(s)G_D(s) \tag{16.20}$$

$$= Y_{sp}(s)G_C(s)G_F(s)G_P(s) - Y(s)G_M(s)G_C(s)G_F(s)G_P(s) + D(s)G_D(s) \tag{16.21}$$

Rearrange to obtain

$$Y(s)[1+ G_M(s)G_C(s)G_F(s)G_P(s)] = Y_{sp}(s)G_C(s)G_F(s)G_P(s) + D(s)G_D(s) \tag{16.22}$$

Combining these equations and solving for $Y(s)$ produces the following closed loop Laplace equation:

$$Y(s) = \frac{G_C(s)G_F(s)G_P(s)}{1+G_C(s)G_F(s)G_P(s)G_M(s)}Y_{sp}(s) + \frac{G_D(s)}{1+G_C(s)G_F(s)G_P(s)G_M(s)}D(s) \tag{16.23}$$

Here we realize that a complex transfer function can be constructed from a combination of simpler transfer functions. As this analysis reveals, the closed loop transfer functions are

Process Variable to Set Point (when disturbance is constant):

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{G_C(s)G_F(s)G_P(s)}{1+G_C(s)G_F(s)G_P(s)G_M(s)}$$

Process Variable to Disturbance (when set point is constant):

$$\frac{Y(s)}{D(s)} = \frac{G_D(s)}{1+G_C(s)G_F(s)G_P(s)G_M(s)}$$

With the controller in automatic (closed loop), if the dynamics are disturbance driven or set point driven, the characteristic equation that reveals the inherent dynamic character of the system is the denominator of the transfer function, which in this case is

$$1+G_C(s)G_F(s)G_P(s)G_M(s)=0 \tag{16.24}$$

Recall that the roots of the characteristic equation (the poles of the transfer function) indicate whether or not a system is stable and the degree to which it has tendency to oscillate. The analysis above

reveals that the roots of Eq. 16.24 will provide this same important information for a closed-loop control system.

16.4 Simplified Block Diagram

While the final control element, process and sensor/transmitter have individual dynamics, from a controller's viewpoint it is impossible to separate these different behaviors. A controller sends a signal out on one wire and sees the response in the process variable when the measurement returns on another wire. As a consequence, the individual gains, time constants and dead times all lump together into a single overall dynamic response. A lumped or simplified block diagram can represent this as:

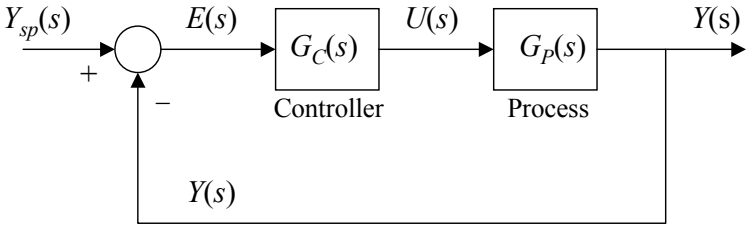


Figure 16.3 – Simplified Closed Loop Block Diagram in Laplace Domain

As before, we write a balance around the closed loop block diagram of Fig. 16.3 that starts and ends with $Y(s)$:

$$Y(s) = U(s)G_P(s)$$

$$U(s) = E(s)G_C(s) = [Y_{sp}(s) - Y(s)]G_C(s)$$

Combining these equations and solving for $Y(s)$ produces the *closed-loop process variable to set point transfer function* that describes the dynamic response of the measured process variable in response to changes in set point:

$$\boxed{\frac{Y(s)}{Y_{sp}(s)} = \frac{G_C(s)G_P(s)}{1 + G_C(s)G_P(s)}} \tag{16.25}$$

The characteristic equation for this closed-loop system is the denominator of the transfer function of Eq. 16.25, or:

$$1 + G_C(s)G_P(s) = 0 \tag{16.26}$$

The roots of Eq. 16.26, which are the poles of the transfer function of Eq. 16.25, indicate whether or not the closed-loop system is stable and the degree to which it has tendency to oscillate.

16.5 The Padé Approximation

Before we continue with our analysis of block diagrams, we recognize the need for a rational expression for dead time in the Laplace domain, $e^{-\theta s}$. This will permit us to employ normal algebraic manipulations during our analysis. The Taylor series expansion for $e^{-\theta s}$ is:

$$e^{-\theta s} = 1 - \theta s + \frac{\theta^2 s^2}{2!} - \frac{\theta^3 s^3}{3!} + \frac{\theta^4 s^4}{4!} + \ldots \tag{16.27}$$

For very small values of dead time we can truncate the series as:

$$e^{-\theta s} \cong 1 - \theta s \tag{16.28}$$

A Padé approximation is a clever expression that more accurately approximates the Taylor series of Eq. 16.27 while providing the rational expression we seek. There are a family of Padé expressions that become increasingly accurate as they increase in complexity. A simple Padé form we use in the next section is exact for the first three terms of the Taylor series expansion and quite close for the fourth term:

$$e^{-\theta s} \cong \frac{2 - \theta s}{2 + \theta s}$$

which we can show using long division yields the series:

$$\frac{2 - \theta s}{2 + \theta s} = 1 - \theta s + \frac{\theta^2 s^2}{2} - \frac{\theta^3 s^3}{4} + \ldots$$

16.6 Closed Loop Analysis Using Root Locus

The poles of interest for our simplified closed loop system are the roots of Eq. 16.26:

$$1 + G_C(s)G_P(s) = 0$$

This analysis assumes that the process behavior, and thus, the process transfer function, remains constant. Adjustable controller tuning provides the ability to move the poles (root location), thereby manipulating closed-loop system behavior.

Example 1: A true first order process without dead time, with a process gain $K_P = 1$ and a time constant $\tau_P = 1$, is under P-Only control. What is the impact of controller gain, K_C, on closed loop system behavior?

Solution: A first order process transfer function is $G_P(s) = \frac{K_P}{\tau_P s + 1}$

From the problem statement, we know that

$$G_P(s) = \frac{1}{s + 1}$$

The P-Only controller transfer function is

$$G_C(s) = K_C$$

Substituting $G_P(s)$ and $G_C(s)$ into the characteristic equation, we obtain

$$1 + G_C(s)G_P(s) = 1 + \frac{K_C}{s+1} = 0$$

Rearranging yields $s + 1 + K_C = 0$

We recall that s is defined in the complex plane as $s = a + bi$, so

$$a + bi + 1 + K_C = 0 + 0i$$

Equating like real term gives us $a + 1 + K_C = 0$

and equating like imaginary terms: $bi = 0i$

We now see that the roots/poles as a function of K_C are

K_C	$a = -K_C - 1$	$b = 0$
0	−1	0
10	−11	0
100	−101	0

We can examine this result on the s plane of Fig. 16.4 and note that the single root always lies on the real axis as long as $K_C \geq -1$. For increasing positive values of K_C, the real root becomes increasingly negative:

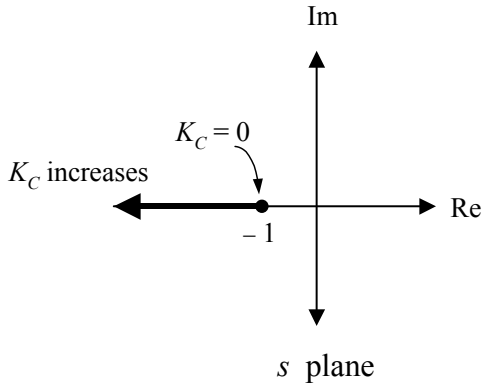


Figure 16.4 – P-Only root locus (root location) in the complex plane

All positive values of controller gain, K_C, yield a solution with no imaginary part. Hence, a true first order system under P-Only control cannot be made to oscillate, no matter how large a K_C value used. It is also unconditionally stable for all positive K_C because the root always remains on the left hand side of the s plane.

It is interesting to note that a true first order system can remain stable even when the controller gain has the wrong sign. For example, if $K_C = -0.5$, then $a = -0.5$ and $b = 0$. This root is located on the left hand side of the s plane, and thus, the system will remain stable (though control would be poor). A value of $K_C = -10$ yields an $a = 9.0$ and $b = 0$, which produces a root located on the right hand side of the s plane, indicating that the system is unstable. As the next example illustrates, even a small value of process dead time dramatically changes the inherent dynamic nature of a closed loop system.

Example 2: A first order plus dead time (FOPDT) process with a process gain, $K_P = 1$, a time constant, $\tau_P = 1$, and a dead time, $\theta_P = 0.1$, is under P-Only control. What is the impact of controller gain, K_C, on closed loop system behavior?

Solution: A FOPDT process transfer function is $G_P(s) = \dfrac{K_P e^{-\theta_P s}}{\tau_P s + 1}$.

From the problem statement, we know that

$$G_P(s) = \frac{e^{-0.1s}}{s+1}$$

The P-Only controller transfer function is

$$G_C(s) = K_C$$

Substituting $G_P(s)$ and $G_C(s)$ into the characteristic equation, we obtain

$$1 + G_C(s)G_P(s) = 1 + \frac{K_C e^{-0.1s}}{s+1} = 0$$

We can then employ the Padé approximation:

$$e^{-0.1s} = \frac{2 - 0.1s}{2 + 0.1s}$$

Substituting the Padé approximation into the characteristic equation gives us

$$1 + \left(\frac{2 - 0.1s}{2 + 0.1s}\right)\frac{K_C}{s+1} = 0$$

Rearranging yields $\quad 0.1s^2 + (2.1 - 0.1K_C)s + 2 + 2K_C = 0$

We can then multiply both sides by 10 and then solve for the roots of the characteristic equation:

$$p_1, p_2 = \frac{-(21 - K_C) \pm \sqrt{(21 - K_C)^2 - 4(20 + 20K_C)}}{2}$$

$$= \frac{-21 + K_C \pm \sqrt{(441 - 42K_C + K_C{}^2 - 80 - 80K_C)}}{2}$$

When the roots are real:

$$= \frac{-21 + K_C \pm \sqrt{K_C{}^2 - 122K_C + 361}}{2}$$

When they have imaginary parts:

$$= \frac{-21 + K_C \pm i\sqrt{-K_C{}^2 + 122K_C - 361}}{2}$$

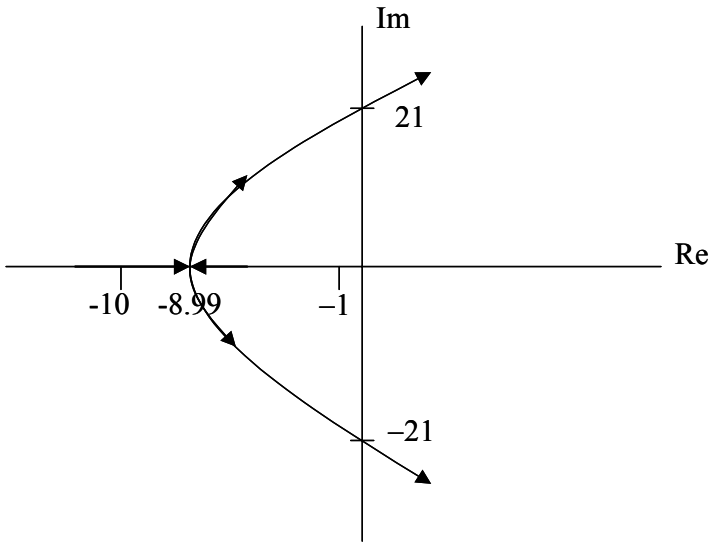
We can now solve for the roots using various values of K_C:

	K_C	P_1	P_2
	0	–1.0	–20
	1	–2.25	–17
	2	–4.0	–15
	3	–8.0	–10
repeated real roots →	3.0345	–8.99	–8.99
	3.04	–8.98 + 0.4*i*	–8.98 – 0.4*i*
	3.2	–8.90 + 2.19*i*	–8.90 – 2.19*i*
	4.0	–8.50 + 5.27*i*	–8.50 – 5.27*i*
	10.0	–5.50 + 13.8*i*	–5.50 – 13.8*i*
	20.0	–0.50 + 20.49*i*	–0.50 – 20.49*i*
limit of stability →	21.0	0 + 20.98*i*	0 – 20.98*i*
	25.0	2 + 22.72*i*	2 – 22.72*i*
	50.0	14.5 + 28.46*i*	14.5 – 28.46*i*

The limit of stability is the point where the roots fall directly on the imaginary axis (the real part of the root is zero). This is considered the limit of stability because as soon as the roots cross over to the positive real part of the *s* plane, the system becomes unstable.

An important observation from this example is that the addition of a small amount of process dead time is enough to transform a process that will not even oscillate (as shown in Example 1) into a process that will oscillate and then go unstable as controller gain, K_C, increases.

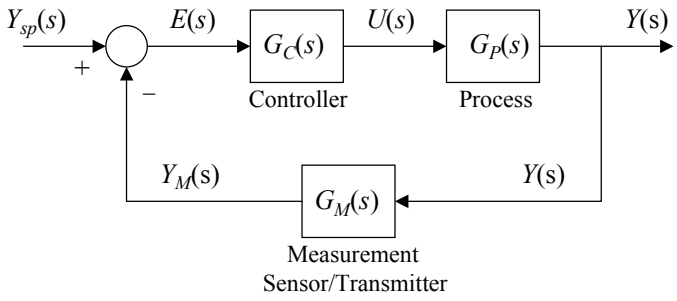
To gauge the accuracy of the Padé approximation, we could construct this problem in *Custom Process*. There we will find that the limit of stability for this system is actually closer to a controller gain $K_C = 16$ rather than the $K_C = 21$ predicted from the above anlaysis. The difference arises because LOOP-PRO does not employ an approximation for dead time in its calculations.



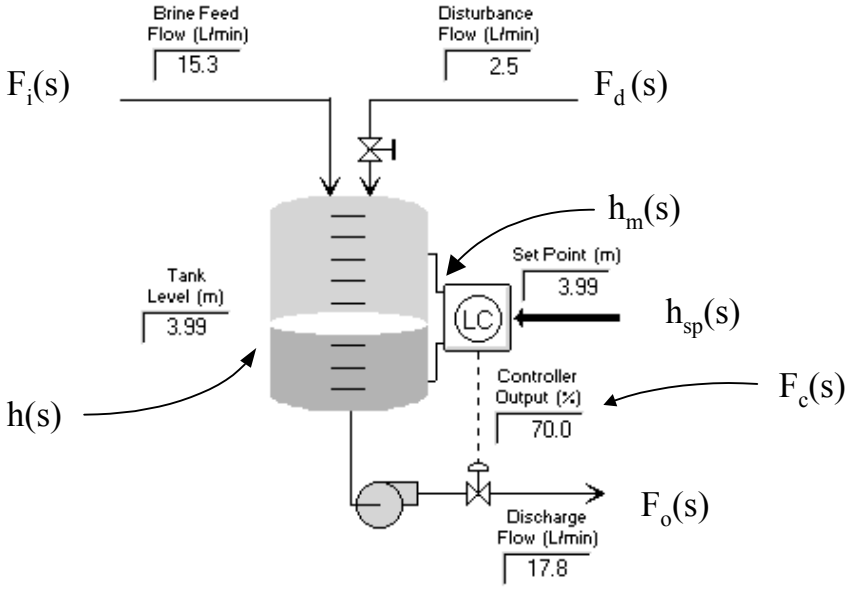
☆ ☆ ☆

16.7 Exercises

Q-16.1 Showing all steps, derive the closed loop "set point to measured process variable" transfer function for this block diagram.



Q-16.2 Draw and label the block diagram for this process. Please use the notation given.



17. Deriving PID Controller Tuning Correlations

17.1 The Direct Synthesis Design Equation

The PID tuning correlations we have used in the book and that are summarized in Appendix C can be derived using the theoretical foundation we have established to this point. Direct synthesis, though challenging to extend to the complete family of PID algorithms, is perhaps the most straightforward method for deriving tuning correlations.

The derivation is based on the simplified block diagram of Fig. 17.1:

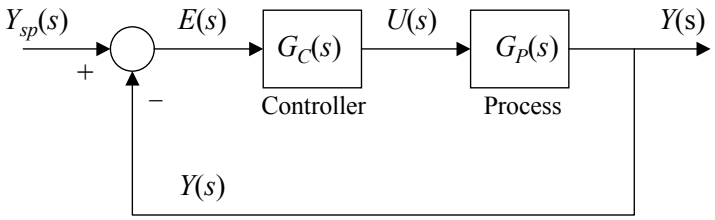


Figure 17.1 – Simplified block diagram

As derived in Chapter 15, the closed loop "process variable to set point" transfer function for this block diagram is:

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{G_C(s)G_P(s)}{1+G_C(s)G_P(s)} \tag{17.1}$$

Solve Eq. 17.1 for the controller transfer function:

$$G_C(s) = \frac{1}{G_P(s)} \frac{Y(s)}{\left[Y_{sp}(s)-Y(s)\right]} \tag{17.2}$$

Divide through by $Y_{SP}(s)$ to arrive at the controller design equation:

$$G_C(s) = \frac{1}{G_P(s)} \frac{\dfrac{Y(s)}{Y_{sp}(s)}}{1-\dfrac{Y(s)}{Y_{sp}(s)}} \tag{17.3}$$

Next, we specify that in closed loop, the measured process variable will rise to meet a step change in set point following a FOPDT shape, or:

$$\boxed{\frac{Y(s)}{Y_{sp}(s)} = \frac{K_{CL}e^{-\theta_C s}}{\tau_C s+1}} \tag{17.4}$$

Figure 17.2 illustrates this desired closed loop response shape:

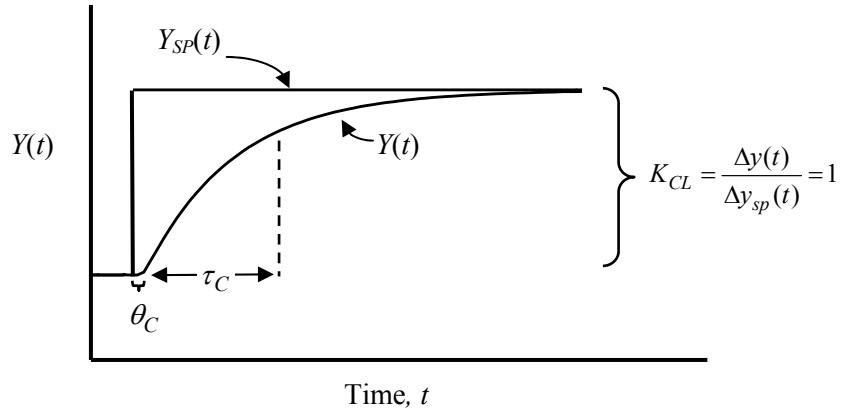


Figure 17.2 – Desired closed loop response of process variable to set point changes

where for Eq. 17.4 and illustrated in Fig. 17.2:

K_{CL} [=] closed loop process variable to set point response gain

θ_C [=] closed loop dead time

τ_C [=] closed loop time constant

• Specify K_{CL}: We want the measured process variable always to equal the set point. Whenever there is a set point change, $\Delta y_{sp}(t)$, then the process variable, Δy(t), should respond quickly and ultimately change in equal magnitude. Hence, recalling that gains are computed from one steady state to the next, we conclude:

$$K_{CL} = \frac{\Delta y(t)}{\Delta y_{sp}(t)} = 1$$

• Specify θ_C: Dead time is always undesirable. Whenever possible, we avoid adding more dead time to a loop. Yet if we are tuning a controller for a process where dead time exists, we cannot ignore it. Consequently, we set the closed loop dead time to the minimum value possible without adding to the process dead time, so:

$$\theta_C(t) = \theta_P(t)$$

• Specify τ_C: The closed loop time constant indicates the speed of the response of a process to set point changes. A popular heuristic for achieving a 10% to 15% overshoot to step changes in set point is:

$$\tau_C \text{ is the larger of } 0.1\tau_P \text{ or } 0.8\theta_P$$

A heuristic for a more conservative "no overshoot" response to set point changes is:

$$\tau_C \text{ is the larger of } 0.5\tau_P \text{ or } 4\theta_P$$

These rules indicate that for small values of dead time, the closed loop process should respond from *two to ten times faster* than the open loop process. To understand how this is possible, consider that when accelerating a car from one velocity to another, you do not move the gas pedal once and wait for the car to speed up and then steady out at the desired velocity (this is how the process time constant is computed in open loop).

Rather, you push the gas pedal past the level where it will ultimately end up and then ease off the pedal as the car approaches the desired speed. Similarly, a controller will send a controller output signal that is beyond its ultimate or final value and then ease off as the measured process variable approaches the new set point. As such, a closed loop process can indeed respond faster than its natural open loop response.

Hence, the desired closed loop response of the measured process variable to changes in set point expressed in Eq. 17.4 becomes:

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{e^{-\theta_P s}}{\tau_C s + 1} \tag{17.5}$$

Substitute Eq. 17.5 into the controller design equation, Eq. 17.3, yields:

$$G_C(s) = \frac{1}{G_P(s)} \frac{\dfrac{e^{-\theta_P s}}{\tau_C s + 1}}{1 - \dfrac{e^{-\theta_P s}}{\tau_C s + 1}} \tag{17.6}$$

which gives our final controller design equation:

$$\boxed{G_C(s) = \frac{1}{G_P(s)} \left(\frac{e^{-\theta_P s}}{\tau_C s + 1 - e^{-\theta_P s}} \right)} \tag{17.7}$$

17.2 Deriving Controller Tuning Correlations Using Direct Synthesis

The method of approach for deriving tuning correlations using design Eq. 17.7 is summarized as:

- transform the process ODE from the time domain into the Laplace domain to determine $G_P(s)$,
- substitute resulting $G_P(s)$ into Eq. 17.7,
- assume a rational expression for $\theta_P(s)$ (e.g. the Padé approximation) and substitute into Eq. 17.7,
- rearrange the updated Eq. 17.7 to match the form of a controller transfer function, $G_C(s)$, from the PID family of algorithms,
- compare forms and deduce the correlations for the tuning parameters.

Example 1: Derive the PI Controller Tuning Correlations

The PI controller tuning correlations are derived assuming our process is reasonably described with a FOPDT model:

$$G_P(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Substitute the process model into Eq. 17.7, the controller design equation:

$$G_C(s) = \left(\frac{\tau_P s + 1}{K_P e^{-\theta_P s}}\right)\left(\frac{e^{-\theta_P s}}{\tau_C s + 1 - e^{-\theta_P s}}\right)$$

$$= \left(\frac{\tau_P s + 1}{K_P}\right)\left(\frac{1}{\tau_C s + 1 - e^{-\theta_P s}}\right)$$

Next assume a very small value of dead time and employ the simplifying approximation:

$$e^{-\theta_P s} \cong 1 - \theta_P s$$

The controller design equation becomes:

$$G_C(s) = \left(\frac{\tau_P s + 1}{K_P}\right)\left(\frac{1}{\tau_C s + 1 - 1 + \theta_P s}\right)$$

$$= \left(\frac{\tau_P s + 1}{K_P}\right)\left(\frac{1}{\tau_C s + \theta_P s}\right)$$

Factoring gives:

$$G_C(s) = \frac{\tau_P}{K_P(\tau_C + \theta_P)}\left(1 + \frac{1}{\tau_P s}\right)$$

Recall the general PI controller transfer function:

$$G_C(s) = K_C\left(1 + \frac{1}{\tau_I s}\right)$$

Comparing these equations reveals that we have derived PI controller tuning correlations if we specify:

$$K_C = \frac{\tau_P}{K_P(\tau_C + \theta_P)} \quad \text{and} \quad \tau_I = \tau_P$$

Example 2: Derive the Interacting PID with Filter Controller Tuning Correlations
The Interacting PID with Filter controller tuning correlations can be derived assuming our process can be described with an FOPDT model:

$$G_P(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Substituting the process model into the controller design equation gives:

$$G_C(s) = \left(\frac{\tau_P s + 1}{K_P e^{-\theta_P s}}\right)\frac{e^{-\theta_P s}}{\tau_C s + 1 - e^{-\theta_P s}}$$

$$= \left(\frac{\tau_P s + 1}{K_P}\right)\frac{1}{\tau_C s + 1 - e^{-\theta_P s}}$$

Assume a small value of dead time and use the Padé 1-1 approximation:

$$e^{-\theta_P s} \cong \frac{2 - \theta_P s}{2 + \theta_P s}$$

so:

$$G_C(s) = \left(\frac{\tau_P s + 1}{K_P}\right)\frac{1}{\tau_C s + 1 - \left(\frac{2 - \theta_P s}{2 + \theta_P s}\right)}$$

$$= \left(\frac{\tau_P s + 1}{K_P}\right)\frac{2 + \theta_P s}{2\tau_C s + \tau_C \theta_P s^2 + 2\theta_P s}$$

Factoring gives:

$$G_C(s) = \left(\frac{\tau_P s}{K_P}\right)\left(1 + \frac{1}{\tau_P s}\right)\frac{2 + \theta_P s}{\left(2\tau_C + \tau_C \theta_P s + 2\theta_P\right)s}$$

$$= \left(\frac{\tau_P}{K_P}\right)\left(1 + \frac{1}{\tau_P s}\right)\frac{2 + \theta_P s}{\left(\tau_C + \theta_P\right)\left(2 + \left(\frac{\tau_C \theta_P}{\tau_C + \theta_P}\right)s\right)}$$

$$= \left(\frac{\tau_P}{K_P\left(\tau_C + \theta_P\right)}\right)\left(1 + \frac{1}{\tau_P s}\right)\frac{2 + \theta_P s}{\left(2 + \left(\frac{\tau_C \theta_P}{\tau_C + \theta_P}\right)s\right)}$$

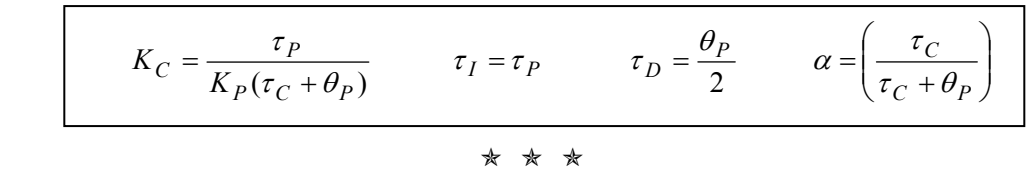
$$G_C(s) = \left(\frac{\tau_P}{K_P\left(\tau_C + \theta_P\right)}\right)\left(1 + \frac{1}{\tau_P s}\right)\left(\frac{1 + \frac{\theta_P}{2}s}{1 + \left(\frac{\tau_C}{\tau_C + \theta_P}\right)\frac{\theta_P}{2}s}\right)$$

Recall the general Interacting PID with Filter controller transfer function:

$$G_C(s) = K_C\left(1+\frac{1}{\tau_I s}\right)\left(\frac{1+\tau_D s}{1+\alpha\tau_D s}\right)$$

Comparing these equations reveals that we have derived Interacting PID with Filter controller tuning correlations if we specify:

$$K_C = \frac{\tau_P}{K_P(\tau_C+\theta_P)} \qquad \tau_I = \tau_P \qquad \tau_D = \frac{\theta_P}{2} \qquad \alpha = \left(\frac{\tau_C}{\tau_C+\theta_P}\right)$$



17.3 Internal Model Control (IMC) Structure

Internal Model Control (IMC), like direct synthesis, can be used to derive PID controller tuning correlations. Figure 17.3 shows a simplified block diagram of the IMC structure. The unique aspect of IMC construction is the placement of a process model, $G_P^*(s)$, in parallel with the actual process it represents.

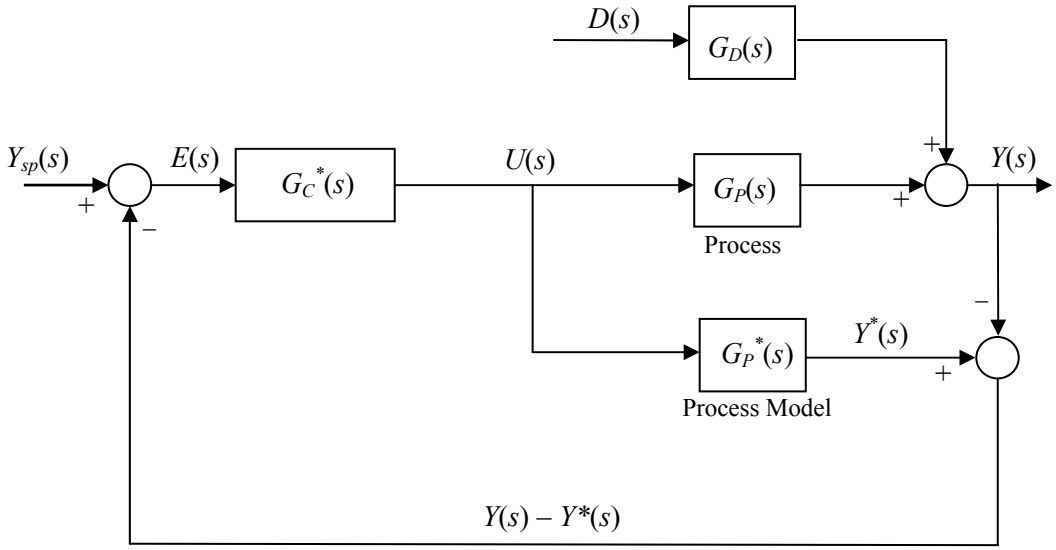


Figure 17.3 – IMC Structure

As shown in the diagram, process model $G_P^*(s)$ receives the actual controller output signal, $U(s)$, and uses it to compute $Y^*(s)$, a prediction of the measured process variable, $Y(s)$. While in theory, the parallel process model must be derived and programmed as part of the controller, we show in the following sections that with certain assumptions, the structure of Fig. 17.3 can be recast into a traditional feedback control architecture. Thus, for the specific cases of interest here, this model is never actually created as a separate entity.

17.4 IMC Closed Loop Transfer Functions

As with direct synthesis, the controller tuning correlations are based on the closed-loop transfer functions. To derive the closed-loop transfer functions, we perform balances on the IMC structure shown in Fig. 17.3 by writing:

$$Y(s) = U(s)G_P(s) + D(s)G_D(s) \tag{17.8}$$

$$Y^*(s) = U(s)G_P^*(s) \tag{17.9}$$

$$U(s) = E(s)G_C^*(s) = \Big[Y_{sp}(s) - Y(s) + Y^*(s) \Big] G_C^*(s) \tag{17.10}$$

Substituting Eqs. 17.8 and 17.9 into Eq. 17.10 yields:

$$U(s) = \Big[Y_{sp}(s) - U(s)G_P(s) - D(s)G_D(s) + U(s)G_P^*(s) \Big] G_C^*(s)$$

$$= Y_{sp}(s)G_C^*(s) - U(s)G_P(s)G_C^*(s) - D(s)G_D(s)G_C^*(s) + U(s)G_P^*(s)G_C^*(s)$$

We solve for U(s):

$$U(s) = \frac{G_C^*(s)}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} Y_{sp}(s) - \frac{G_D(s)G_C^*(s)}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} D(s) \tag{17.11}$$

Substitute Eq. 17.11 into Eq. 17.8 to obtain:

$$Y(s) = \frac{G_C^*(s)G_P(s)}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} Y_{sp}(s) - \frac{G_D(s)G_C^*(s)G_P(s)}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} D(s) + D(s)G_D(s)$$

And rearrange into the closed loop transfer function:

$$\boxed{Y(s) = \frac{G_C^*(s)G_P(s)}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} Y_{sp}(s) + \frac{G_D(s)\Big[1 - G_C^*(s)G_P^*(s) \Big]}{1 + G_C^*(s)\Big[G_P(s) - G_P^*(s) \Big]} D(s)} \tag{17.12}$$

Equation 17.12 yields a set-point tracking (servo response) transfer function assuming a constant disturbance, and disturbance rejection (regulator response) transfer function assuming a constant set point:

Set-Point Tracking:
$$\frac{Y(s)}{Y_{sp}(s)} = \frac{G_C^*(s)G_P(s)}{1+G_C^*(s)\left[G_P(s)-G_P^*(s)\right]} \tag{17.13}$$

Disturbance Rejection:
$$\frac{Y(s)}{D(s)} = \frac{G_D(s)\left[1-G_C^*(s)G_P^*(s)\right]}{1+G_C^*(s)\left[G_P(s)-G_P^*(s)\right]} \tag{17.14}$$

17.5 Deriving Controller Tuning Correlations Using the IMC Method

Three basic steps are used to derive IMC tuning correlations; the first two steps detail the creation of the IMC model. The last step relates the IMC model to a classical feedback controller transfer function to obtain controller tuning correlations.

Step 1: Recall the discussion from Chapter 15 where we established that the poles of a transfer function (the roots of the characteristic equation in the denominator of the transfer function) indicate system stability. If the real part of any root is positive (lies in the right hand side of the complex plane), the system is unstable.

This concept plays a central role in the IMC analysis. The approach we take is to invert the process model to create the controller. One problem with such an approach is that any roots in the numerator (analogous to poles, roots in the numerator of a transfer function are called zeros) of the process model that lie in the right hand of the complex plane, when inverted, become unstable poles. If we permit this to occur, our controller will be unstable.

To avoid creating an unstable controller, factor the process model, $G_P^*(s)$, into an invertible and noninvertible part. The classification is based on the numerator of the transfer function because this becomes the denominator when the model is inverted in Step 2.

The noninvertible part, $G_{P+}^*(s)$, contains all right-hand plane zeros (roots in the numerator of a transfer function that have positive real parts) and the dead time term. The invertible part, $G_{P-}^*(s)$, contains left hand plane zeros (roots in the numerator that have negative real parts) that produce stable behavior when in the denominator of a transfer function. Using this notation, the process model is factored as:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s) \tag{17.15}$$

Step 2: Specify the controller transfer function as:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s) \tag{17.16}$$

where F(s) is a low-pass filter with gain equal to 1. The term "low-pass" is used to indicate that high frequencies (rapid controller output changes) are lost. For deriving tuning correlations, the IMC filter has the form:

$$F(s) = \frac{1}{\left(\tau_C s+1\right)} \tag{17.17}$$

As discussed in section 17.1, the closed loop time constant, τ_C, indicates the speed of the response of a process to set point changes. A popular heuristic for achieving a 10% to 15% overshoot to step changes in set point is:

$$\tau_C \text{ is the larger of } 0.1\tau_P \text{ or } 0.8\theta_P$$

A heuristic for a more conservative "no overshoot" response to set point changes is:

$$\tau_C \text{ is the larger of } 0.5\tau_P \text{ or } 4\theta_P$$

Step 3: Relate the IMC transfer function models to those from classical feedback control. We recall that the closed loop transfer function for a classical feedback control architecture is:

$$Y(s) = \frac{G_P(s)G_C(s)}{1+G_P(s)G_C(s)}Y_{sp}(s) + \frac{G_D(s)}{1+G_P(s)G_C(s)}D(s)$$

We compare set point tracking forms:

IMC: $\dfrac{Y(s)}{Y_{sp}(s)} = \dfrac{G_P(s)G_C^*(s)}{1+G_C^*(s)\left[G_P(s)-G_P^*(s)\right]}$ Classical: $\dfrac{Y(s)}{Y_{sp}(s)} = \dfrac{G_P(s)G_C(s)}{1+G_P(s)G_C(s)}$

And equate the two:

$$G_P(s)G_C^*(s)\left[1+G_P(s)G_C(s)\right] = G_P(s)G_C(s)\left[1+\left(G_P(s)-G_P^*(s)\right)G_C^*(s)\right]$$

$$G_C^*(s)+G_C^*(s)G_P(s)G_C(s) = G_C(s)+G_C^*(s)G_P(s)G_C(s)+G_C^*(s)G_P^*(s)G_C(s)$$

Rearranging, we obtain

$$\boxed{G_C(s) = \frac{G_C^*(s)}{1-G_C^*(s)G_P^*(s)}} \qquad (17.18)$$

We can use Eq. 17.18 to obtain a classical feedback controller from one derived from the IMC structure. This enables us to determine the controller tuning parameters K_P, τ_I, τ_D, and α.

Example: Derive the PI Controller Tuning Correlations using the IMC Method
Assume a FOPDT process model:

$$G_P^*(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s+1}$$

Substitute the first-order expansion for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx 1-\theta_P s$$

so

$$G_P^*(s) = \frac{K_P(1-\theta_P s)}{\tau_P s+1} \qquad (17.19)$$

Factor $G_P^*(s)$ into invertible and noninvertible parts:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so following the discussion above:

$$G_{P+}^*(s) = (1-\theta_P s)$$

$$G_{P-}^*(s) = \frac{K_P}{\tau_P s + 1} \tag{17.20}$$

We can now express the IMC controller model, $G_C^*(s)$, in terms of the invertible process model term and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)} F(s) \tag{17.21}$$

where the IMC filter has the form:

$$F(s) = \frac{1}{\tau_C s + 1} \tag{17.22}$$

Substituting Eqs. 17.20 and 17.22 into Eq. 17.21 yields the controller:

$$G_C^*(s) = \left(\frac{\tau_P s + 1}{K_P}\right)\left(\frac{1}{\tau_C s + 1}\right) = \frac{\tau_P s + 1}{K_P\left(\tau_C s + 1\right)} \tag{17.23}$$

We can relate this IMC controller model, $G_C^*(s)$, to a classical feedback controller model via Eq. 17.18::

$$G_C(s) = \frac{G_C^*(s)}{1 - G_P^*(s)G_C^*(s)}$$

We substitute Eq. 17.19 and 17.23 into Eq. 17.18 and simplify:

$$G_C(s) = \frac{\dfrac{\tau_P s + 1}{K_P\left(\tau_C s + 1\right)}}{1 - \left(\dfrac{K_P\left(1-\theta_P s\right)}{\cancel{\tau_P s + 1}}\right)\left(\dfrac{\cancel{\tau_P s + 1}}{K_P\left(\tau_C s + 1\right)}\right)}$$

$$= \frac{\dfrac{\tau_P s + 1}{K_P\left(\tau_C s + 1\right)}}{1 - \left(\dfrac{\cancel{K_P}\left(1-\theta_P s\right)}{\cancel{K_P}\left(\tau_C s + 1\right)}\right)}$$

$$= \frac{\dfrac{\tau_P s + 1}{K_P \cancel{\left(\tau_C s + 1\right)}}}{\dfrac{\tau_C s + 1 - 1 + \theta_P s}{\cancel{\tau_C s + 1}}}$$

$$= \frac{\tau_P s + 1}{K_P s \left(\tau_C + \theta_P\right)}$$

$$= \frac{\tau_P \cancel{s}}{K_P \cancel{s} \left(\tau_C + \theta_P\right)} + \frac{1}{K_P s \left(\tau_C + \theta_P\right)}$$

$$G_C(s) = \frac{\tau_P}{K_P \left(\tau_C + \theta_P\right)} \left[1 + \frac{1}{\tau_P s} \right] \tag{17.24}$$

Compare Eq. 17.24 to the classical feedback model for a PI controller,

$$G_C(s)_{\text{PI}} = K_C \left[1 + \frac{1}{\tau_I s} \right]$$

we obtain the following controller tuning parameters:

$$\boxed{K_C = \frac{\tau_P}{K_P \left(\tau_C + \theta_P\right)} \quad \text{and} \quad \tau_I = \tau_P}$$

★ ★ ★

Additional tuning correlation derivations for several controllers from the PID family can be found in Appendix A.

17.6 Exercises

Q-17.1 Showing all steps, use the Direct Synthesis method to derive controller tuning correlations for the PID control algorithm

$$G_C(s) = K_C \left(1 + \frac{1}{\tau_I s} + \tau_D s \right)$$

Assume the process is best described by a second order plus dead time model of the form

$$G_P(s) = \frac{K_P e^{-\theta_P s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

and that the process has very little dead time, or $e^{-\theta_P s} \cong 1 - \theta_P s$.

Hint: the resulting correlations will not match those in the Controller Tuning Guide.

Q-17.2 Showing all steps, use the Direct Synthesis method to derive PI controller tuning correlations, assuming the dynamic behavior of a process is described by a first-order without dead time transfer function.

18. Cascade Control

18.1 Architectures for Improved Disturbance Rejection

The most popular architectures for improved regulatory performance are cascade control and the feed forward with feedback trim architecture discussed in the next chapter. Both architectures trade off additional complexity in the form of instrumentation and engineering time in return for a controller better able to reject the impact of disturbances on the measured process variable. Neither architecture benefits nor detracts from set point tracking performance.

As we will soon learn, the feed forward with feedback trim architecture requires an additional sensor and the programming of a dynamic model. The sensor is installed to directly measure changes in the disturbance variable. The model receives the disturbance measurement, computes control actions to counter or negate its impending impact on the measured process variable, and transmits the result to the controller for execution. Because of this architecture, feed forward is useful when one specific disturbance variable is responsible for repeated, costly disruptions to stable operation. It is also useful when a secondary measured process variable cannot be identified to construct a control cascade as discussed in the next section.

18.2 The Cascade Architecture

A feed forward element seems reasonably straightforward in concept; directly measure a nasty disturbance and use a model to instruct the controller how and when to take actions to negate its impact on the measured process variable. Developing and programming a proper dynamic model, however, can be a challenging task that should not be underestimated.

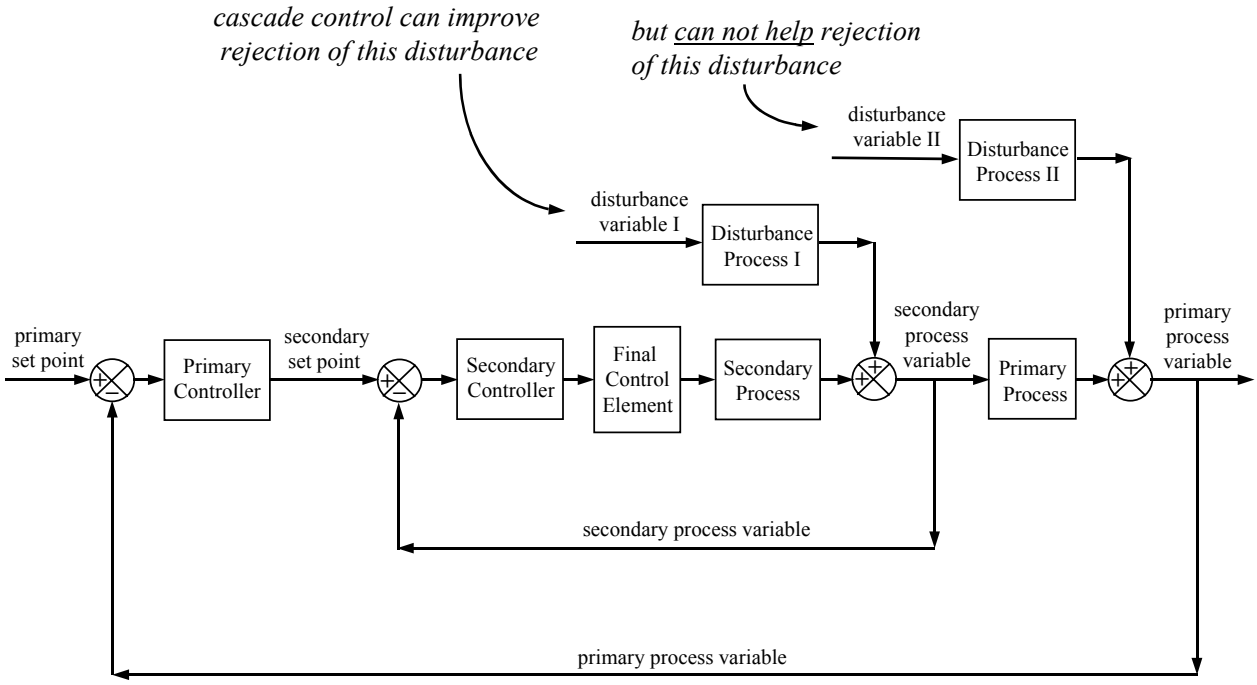


Figure 18.1 – Block diagram of the cascade architecture

Cascade control is more difficult to conceptualize, yet implementation is a familiar task because the architecture is comprised of two ordinary controllers from the PID family. Like feed forward, cascade provides minimal benefits for set point tracking. Before you begin your design, be sure your goal is improved disturbance rejection.

In a traditional feedback loop, a controller adjusts a manipulated variable so the measured process variable remains at set point. The cascade design requires that you identify a *secondary* process variable (we will henceforth call the main process variable associated with original control objective the *primary* variable). This secondary process variable has specific requirements:

- it must be measurable with a sensor,
- the same final control element (e.g. valve) used to manipulate the primary variable must also manipulate the secondary variable,
- the same disturbances that are of concern for the primary variable must also disrupt the secondary variable, and
- the secondary variable must be *inside* the primary process variable, which means it responds well before the primary variable to disturbances and final control element manipulations.

With a secondary process variable identified, a cascade is constructed as shown in Fig. 18.1. The block diagram shows how both Disturbance I and the final control element impact the secondary variable before they affect the primary variable. Notice that the secondary loop has a traditional feedback control structure, except here it is literally nested inside the primary loop.

A cascade requires two sensors and two controllers but only one final control element because the output of the primary controller, rather than going to a valve, becomes the set point of the secondary controller. Because of the nested architecture, *success in a cascade implementation requires that the settling time of the (inner) secondary loop is significantly faster than the settling time of the primary (outer) loop.*

As the above discussion implies, one advantage of a control cascade is that it is not tied to a single disturbance. Rather, the same cascade can address multiple disturbances as long as each impacts the inner secondary variable well before it impacts the outer primary variable. Also, as mentioned before, implementation uses our existing skills because the architecture is comprised of two ordinary controllers from the PID family.

18.3 An Illustrative Example

The Flash Drum Process

To better understand the construction and use of cascade control, consider the flash drum process shown in Fig. 18.2. This process is not a case study in LOOP-PRO, but its behavior is reasonably intuitive, and this makes it useful for a qualitative discussion of the important concepts.

As shown in the figure, the feed stream entering the flash valve is a hot liquid under high pressure. A flash valve produces a large and sudden drop in pressure as the liquid flows toward the flash drum, permitting some of the hot liquid to vaporize. As a result, a vapor phase forms over the liquid inside the drum. Figure 18.2 shows that a vapor stream exits overhead from the drum while a liquid stream exits from the bottom. It is essential that the liquid level never falls so low that vapor is sent down the liquid drain nor rises so high that liquid enters the vapor line.

Given this scenario, our design objective is to control liquid level in the drum. Liquid drain flow rate is selected as the manipulated variable in this example. If liquid level is too high, the controller should increase liquid drain flow rate. If level is too low, the controller should decrease liquid drain flow rate.

The disturbance of concern in this example is the pressure in the vapor phase. As indicated in the figure, the overhead vapor phase pressure changes without warning due to the behavior of some unidentified down stream unit.

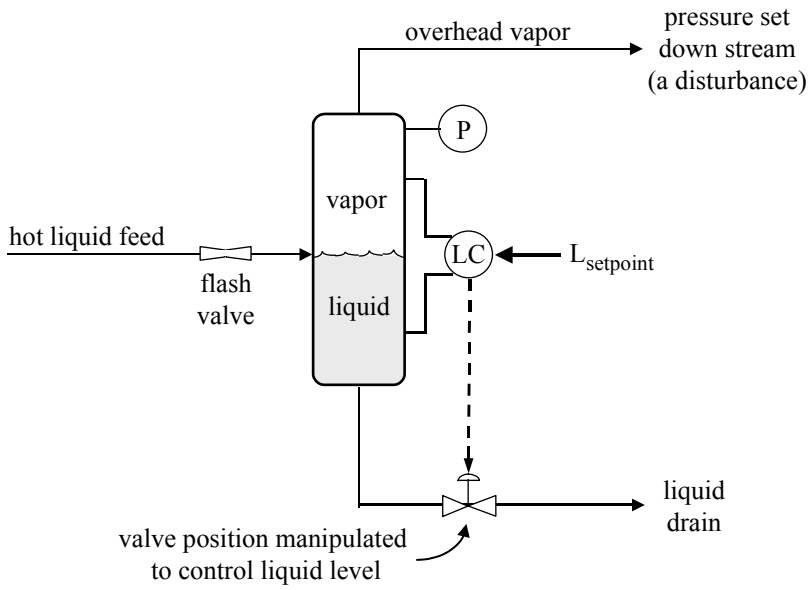


Figure 18.2 – Single-loop level control of a flash drum

Problems with Single Loop Control

The controller of Fig. 18.2 attempts to achieve the design objective by adjusting valve position in the liquid drain stream. If the measured level is higher than set point, the controller signals the valve to open by an appropriate percentage with the expectation that this will increase drain flow rate accordingly. But drain flow rate is a function of several variables, including

- valve position,
- the hydrostatic head (height of the liquid), and
- the pressure of the vapor pushing down on the liquid (a disturbance).

Assume for a moment that the pressure of the vapor phase is constant over time (and the feed flow rate and composition are also constant) then as the drain valve opens and closes, the liquid drain flow rate increases and decreases in a predictable fashion. Hence, the single loop architecture of Fig. 11.2 should provide satisfactory liquid level control performance.

However, as Fig. 18.2 indicates, the pressure of the vapor phase is controlled by a unit down stream of our flash drum. Rather than remaining constant, the pressure of the vapor phase changes over time and this acts as a disturbance to the level control process.

Suppose the vapor phase pressure starts decreasing. This disturbance will cause the pressure pushing down on the liquid interface to decrease. If the valve position remains constant, the liquid drain flow rate will similarly decrease. If the pressure decrease occurs quickly enough, the controller can actually be opening the valve yet the liquid drain flow rate can continue to decrease. Alternatively, if the pressure in the vapor phase starts to increase, the controller can be closing the valve yet the liquid drain flow rate can actually increase.

The pressure change disturbance causes a contradictory outcome that can confound the controller and result in poor control. As the preceding discussion indicates, it is not valve position but liquid drain flow rate which must be adjusted for high performance disturbance rejection.

A Cascade Control Solution

A first step in cascade design is to ensure that our control objective is disturbance rejection. The scenario presented supports that the controller is not intended for set point tracking. In fact, the set point will be constant at mid-drum level during normal operation. Our goal is to maintain liquid level at set point while rejecting the disturbance of pressure changes in the overhead vapor phase.

To implement a cascade we must be able to identify a secondary process variable. Liquid level becomes the primary process variable and controlling it remains the central design objective of our strategy. For the secondary process variable we propose liquid drain flow rate. As required by the cascade design criteria:

- liquid drain flow rate is measurable with a sensor,
- the same valve used to manipulate liquid level (the primary variable) also manipulates the liquid drain flow rate,
- changes in vapor phase pressure that disturb liquid level control also impact the drain flow rate, and
- drain flow rate is *inside* the liquid level in that it responds well before liquid level to changes in valve position and changes in vapor phase pressure.

Figure 18.3 shows a cascade architecture with two controllers (level control and drain flow rate control), two measurement sensors (measuring liquid level and liquid drain flow rate) and one final control element (a valve in the liquid drain stream). Figure 11.4 shows a block diagram of this same level-to-flow cascade.

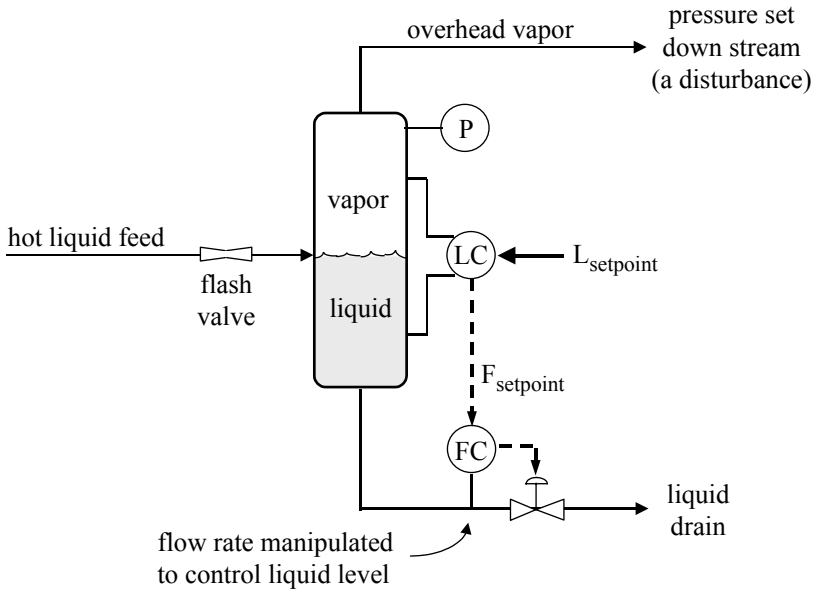


Figure 18.3 – Level-to-flow cascade control of liquid level

Liquid level control, our main objective, is the primary or outer loop. The output of the primary controller is the set point of the secondary controller, which controls liquid drain flow rate by adjusting the valve position. Flow control dynamics are much faster than level control dynamics. Hence, this configuration is consistent with the previously mentioned design criteria that the settling time of the secondary loop must be significantly faster than the settling time of the primary loop.

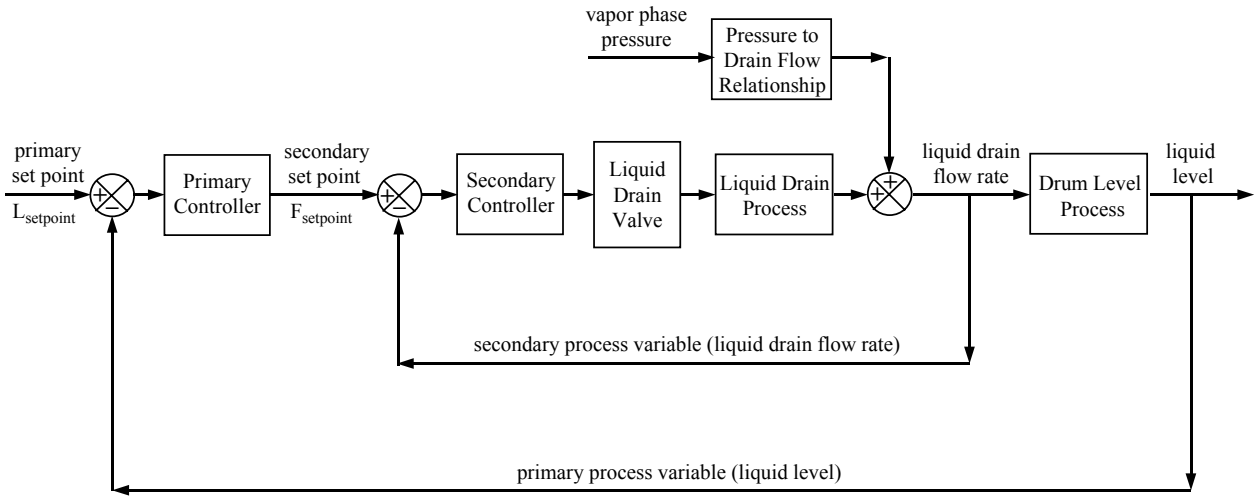


Figure 18.4 – Block diagram of the level-to-flow cascade architecture

With this cascade architecture, if the liquid level is too high, the level controller now specifically calls for an increased liquid drain flow rate rather than simply an increase in valve opening as was the case in the single loop configuration of Fig. 18.2. It is the flow controller that then determines whether this means opening or closing the valve and by how much. Thus, a pressure disturbance in the vapor phase gets addressed quickly by the secondary flow controller and this dramatically improves the disturbance rejection performance of primary control loop.

18.4 Tuning a Cascade Implementation

Cascade loop tuning uses the skills we have developed in previous chapters:

1) Begin with both the primary and secondary controllers in manual mode.
2) Select a P-Only controller for the inner secondary loop (integral action increases settling time and offset is rarely an issue for the secondary process variable).
3) Tune the secondary P-Only controller using a set point tracking criterion (its main job is to respond to set point commands from the primary controller). Test it to ensure a satisfactory set point tracking performance.
4) Leave the secondary controller in automatic; it now literally becomes part of the primary process. Select a controller with integrating action for the primary loop (PI or PID). Use a disturbance rejection design criterion as this is the main job of the primary controller.
5) Tune the primary controller using methods discussed in the previous chapters and test it to ensure acceptable performance.
6) With both controllers in automatic, tuning of the cascade is complete.

18.5 Exploring the Jacketed Reactor Process

As described in Section 2.6 and shown in Fig. 18.5 for the single feedback loop case, the jacketed reactor is a continuously stirred vessel in which an exothermic (heat producing) reaction occurs. Residence time is constant in this well mixed reactor, so the conversion of reactant feed to desired product can be inferred from the temperature of the reactor exit stream.

To control reactor exit stream temperature (the measured process variable), the vessel is enclosed with a jacket through which a cooling liquid flows. The controller manipulates a valve to adjust the cooling liquid flow rate. If the exit stream temperature (and thus conversion) is too high, the controller opens the valve. This increases cooling liquid flow rate, which cools off the reactor and causes the heat producing reaction to slow. Ultimately, the measured temperature of the stream exiting the reactor drops in response to the control action. As shown in Fig. 18.5, the disturbance variable of interest for this process is the temperature of cooling liquid entering the jacket.

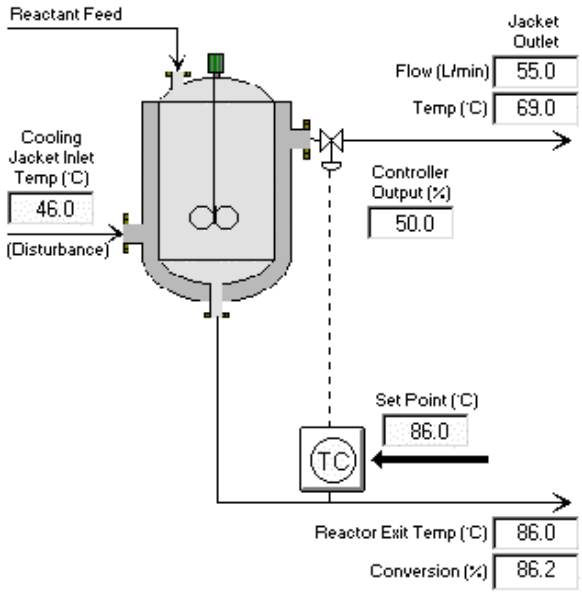


Figure 18.5 – Jacketed reactor process with single loop feedback control architecture

In the study explored here, the control objective is to maintain the reactor exit stream temperature at set point by rejecting disturbances caused by changes in the cooling jacket inlet temperature. The disturbance rejection performance of a single loop PI controller configuration is first presented and then compared to that of a PI to P-Only controller cascade architecture.

For both the single loop and cascade investigation, the design level of operation is a reactor exit stream temperature (measured process variable) of 86°C. The cooling jacket inlet temperature (disturbance) is normally at its design value of 46°C, but our concern is that on occasion, it is known to unexpectedly spike as low as 40°C. An open loop study establishes that a controller output of 50% causes the reactor to steady at the design measured exit stream temperature of 86°C when the cooling jacket inlet temperature is at its expected or design value of 46°C.

18.6 Single Loop Disturbance Rejection in the Jacketed Reactor

Tuning the single loop PI controller shown in Fig. 18.5 follows the same design procedure we have always used in this book. We perturb the controller output, record the data as the process responds, fit a FOPDT (first order plus dead time) model to the data, and use the resulting model parameters in a tuning correlation to determine initial controller tuning values.

As shown in Fig. 18.6, we use a controller output doublet to generate measured process variable data both above and below the design level of operation. Although a variety of dynamic tests would produce an equally useful data set, here the controller output is stepped from the design value of 50% up to 53%, then down to 47%, and finally back to 50%. The measured reactor exit stream temperature exhibits a clear response after each controller output step that dominates any measurement noise.

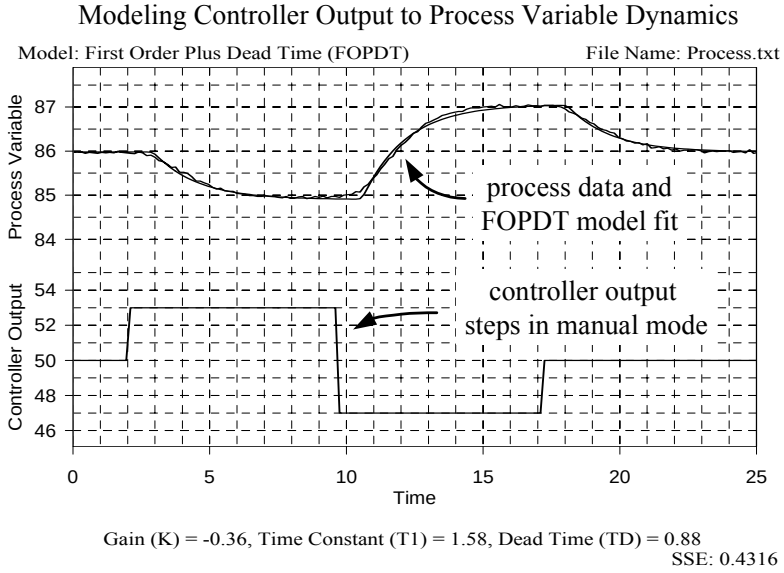


Figure 18.6 – FOPDT model fit of single loop controller output to measured process variable data

A FOPDT fit of the dynamic process data as computed by *Design Tools* is also shown in Fig. 18.6. The model appears to be reasonable and appropriate based on visual inspection, thus providing the following design parameters:

$$\text{Process Gain, } K_P = -0.36\,°\text{C}/\%$$

$$\text{Time Constant, } \tau_P = 1.6 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.88 \text{ min}$$

To use the IMC correlations, we first compute the closed loop time constant. Here we choose aggressive tuning:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = \text{larger of } 0.1(1.6) \text{ or } 0.8(0.88) = 0.70 \text{ min.}$$

Substituting this closed loop time constant and the above FOPDT model parameters into the IMC tuning correlations of Eq. 8.5 yield the following tuning values:

$$\text{Controller Gain, } K_C = -2.8\ \%/°\text{C}$$

$$\text{Reset Time, } \tau_I = 1.6 \text{ min}$$

The disturbance rejection performance of the single loop PI controller using these tuning parameters is shown in Fig. 18.7. The controller label to the upper right of the plot confirms that the PI controller has a direct acting proportional term, an anti-reset windup integral term, and a derivative term that is off.

As shown in Fig. 18.7, the measured reactor exit stream temperature is initially steady at the design set point value of 86°C. To test the controller, the cooling jacket inlet temperature is stepped from its design value of 46°C down to 40°C and back again. As shown, the single loop PI controller is able to maintain reactor exit stream temperature near the constant set point of 86°C, with deviations ranging as high as 2.5°C during the event.

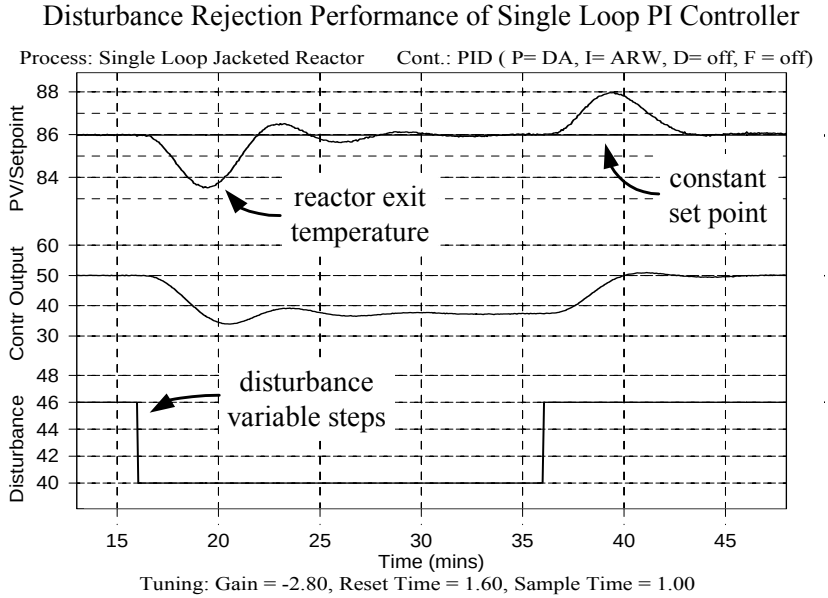


Figure 18.7 – Disturbance rejection in the jacketed reactor under single loop PI control show process variable deviations reach 2.5°C (compare to Fig. 18.13)

18.7 Cascade Disturbance Rejection in the Jacketed Reactor

The open loop dynamic behavior of the cascade jacketed reactor is identical to that of the single loop case. The process graphic for the cascade controller architecture is shown in Fig. 18.8.

The Cascade Architecture

As discussed in the single loop study, our control objective is disturbance rejection, so it is appropriate to consider a cascade architecture. The primary process variable remains the reactor exit stream temperature. To construct a cascade, we need to identify a secondary process variable. As shown in Fig. 18.8, LOOP-PRO uses the cooling jacket outlet temperature. As required for a cascade design:

- cooling jacket outlet temperature is measurable with a sensor,
- the same valve used to manipulate reactor exit stream temperature (the primary variable) also manipulates the cooling jacket outlet temperature,
- changes in cooling jacket inlet temperature that disturb the reactor exit stream temperature also impact the cooling jacket outlet temperature, and
- the cooling jacket outlet temperature is *inside* the reactor exit temperature in that it responds first to changes in valve position and changes in the cooling jacket inlet temperature.

A block diagram of this architecture is shown in Fig. 18.9. Like all cascades, there are two measurements, two controllers and one final control element; the same final control element as in the single loop case.

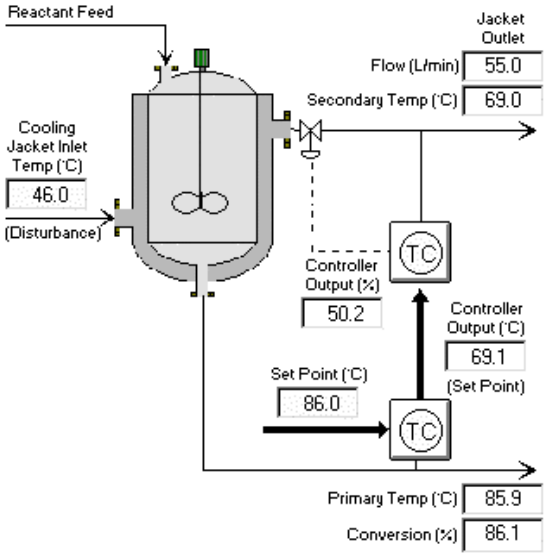


Figure 18.8 – Jacketed reactor process with cascade control architecture

The primary (outer) process is still the reactor and the primary measured process variable is the reactor exit stream temperature. The output of the primary controller is the set point of the secondary controller. The inner (secondary) process is the cooling jacket. The manipulated variable of the secondary loop is the cooling jacket liquid flow rate and the secondary measured process variable is the cooling jacket outlet temperature.

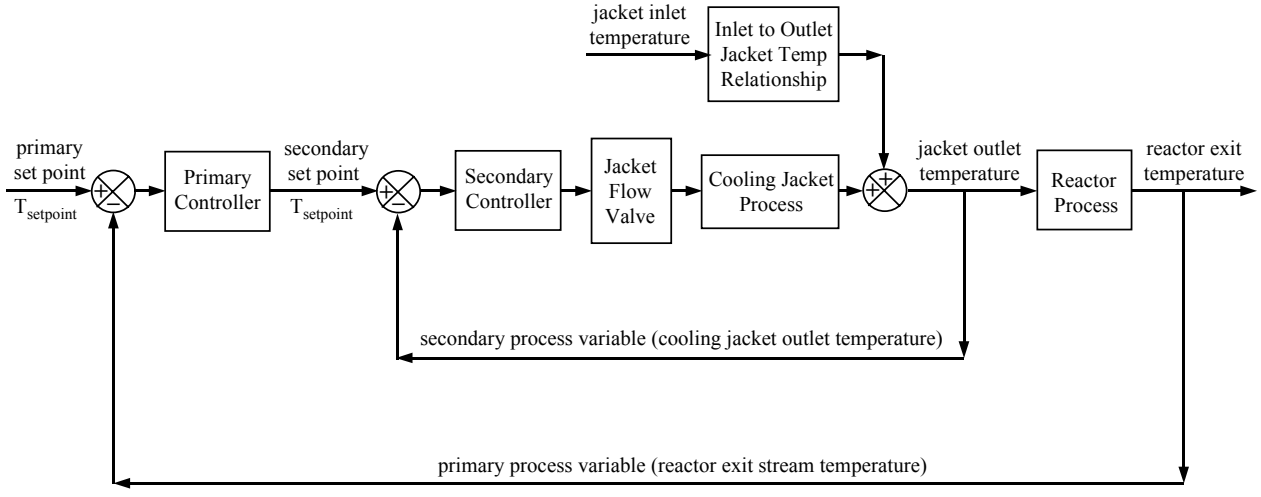


Figure 18.9 – Block diagram of jacketed reactor cascade architecture

Secondary P-Only Controller

To implement a cascade, the secondary controller is tuned while the primary controller is in manual mode. The design operating conditions are the same as those used for the single loop PI controller study. That is, with the cooling jacket inlet temperature at 46°C and the controller output at 50%, the reactor exit stream temperature steadies at the design value of 86°C. We note at these design conditions that the cooling jacket outlet temperature, the secondary measured process variable, steadies at 69°C. Hence, for the secondary controller:

$$y_{\text{setpoint}} = 69°\text{C}$$

The bias is the value of the controller output that, in open loop, causes the measured process variable to steady at its design condition when the disturbances are at their design or expected value. So for the secondary P-Only controller:

$$u_{\text{bias}} = 50\%$$

Starting from steady state at the design level of operation, a doublet is used to generate controller output to secondary process variable dynamic data as shown in Fig. 18.10. The controller output is stepped from the design value of 50% up to 55%, down to 45%, and back to 50%. After each control action, the secondary process variable displays a clear response that dominates measurement noise.

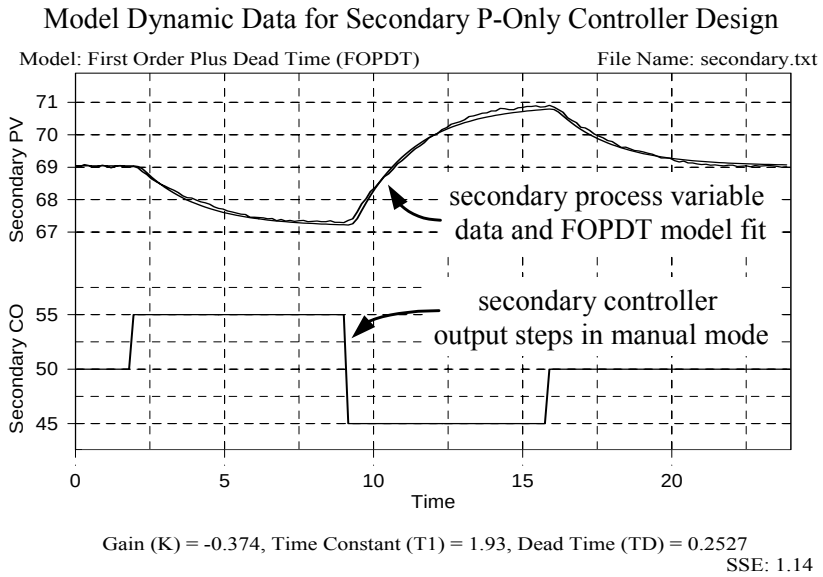


Figure 18.10 – FOPDT fit of controller output to secondary process variable dynamic data

A FOPDT dynamic model fit to the data, also shown in Fig. 11.10, yields the following secondary control loop model parameters.

Process Gain, $K_P = -0.37°\text{C}/\%$

Time Constant, $\tau_P = 1.9$ min

Dead Time, $\theta_P = 0.25$ min

Although disturbance rejection is the overall objective, the goal of the inner secondary loop is to track set point changes computed by the primary controller. Using these FOPDT model parameters in the ITAE for set point tracking correlation (recall that IMC correlations do not exist for P-Only controllers) yields the following P-Only tuning parameter:

$$\text{Controller Gain, } K_C = -6.4 \text{ \%/°C}$$

P-Only set point tracking performance is shown in Fig. 18.11. The primary loop is still in manual mode at this point. As expected for a P-Only controller, offset exists when the set point is not at the design value. The secondary process variable responds quickly and settles rapidly to set point changes so we consider the design of the secondary loop to be complete. The secondary loop is left in automatic and literally becomes part of the primary process. We now tune the primary controller.

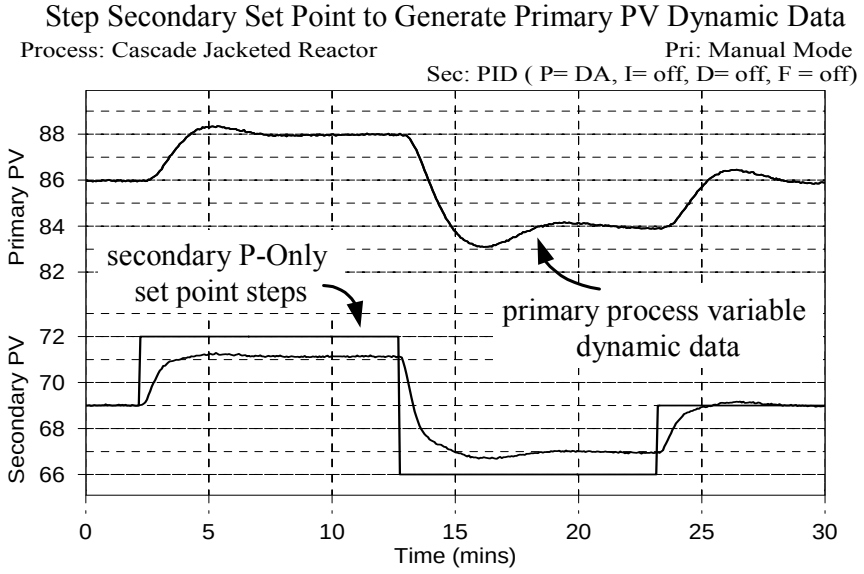


Figure 18.11 – Set point tracking performance of the secondary loop under P-Only control

Primary Loop Control

In previous chapters, we have always generated dynamic process data for controller design by stepping, pulsing or otherwise perturbing the controller output signal. In the cascade architecture, however, the controller output of the primary loop is the set point of the secondary controller. So to design the primary controller, we must step, pulse or otherwise perturb the secondary set point and then fit a model to the corresponding response in the primary measured process variable. Figure 18.11 contains such data and this is used for the design of the primary PI controller.

The set point of the secondary P-Only controller, as shown in Fig. 18.12, is stepped in a doublet from its design value of 69°C up to 72°C, down to 66°C, and back to 69°C. The measured reactor exit stream temperature displays a clear response after each change that dominates the measurement noise. A FOPDT fit of the dynamic data, shown in Fig. 18.12, yields the following primary control loop model parameters.

$$\text{Process Gain, } K_P = 0.70 \ \frac{\text{°C of reactor exit stream}}{\text{°C of cooling jacket outlet stream}}$$

$$\text{Time Constant, } \tau_P = 0.55 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.71 \text{ min}$$

We first compute the closed loop time constant. Here we choose aggressive tuning:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = \text{larger of } 0.1(0.55) \text{ or } 0.8(0.71) = 0.57 \text{ min.}$$

Substituting this closed loop time constant and the above FOPDT model parameters into the IMC correlations for PI control yields the following tuning values:

$$\text{Controller Gain, } K_C = 0.61 \ \frac{\text{°C of cooling jacket outlet stream}}{\text{°C of reactor exit stream}}$$

$$\text{Reset Time, } \tau_I = 0.55 \text{ min}$$

These tuning values are implemented on the primary loop and the design of the cascade is complete.

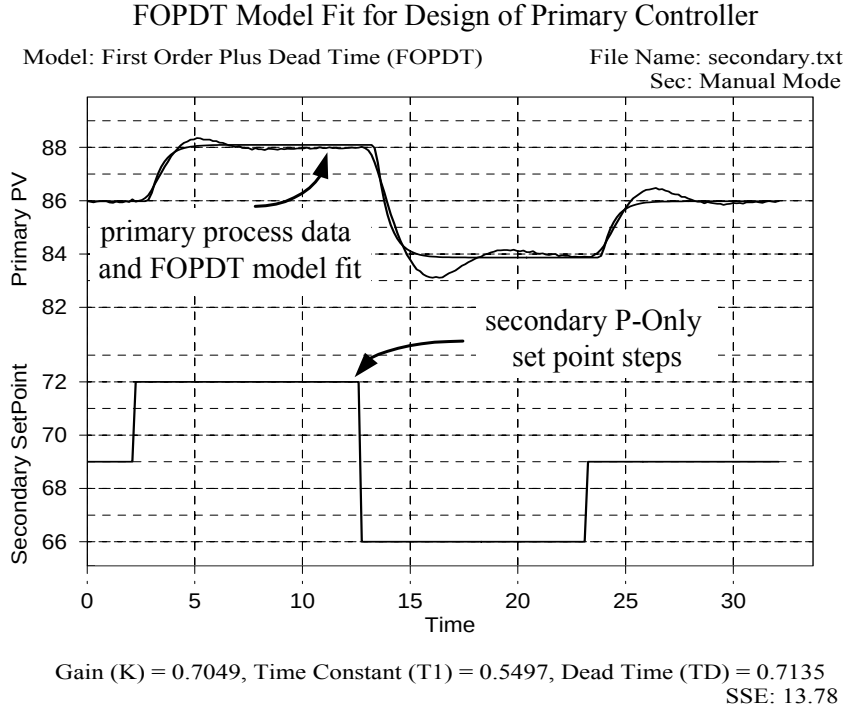


Figure 18.12 – FOPDT model fit of dynamic data from a primary loop doublet test

Figure 18.13 shows the disturbance rejection performance of the cascade using these tuning parameters for the primary PI controller while the secondary loop remains under P-Only control as previously described. Just as in Fig. 18.7, the primary measured process variable (reactor exit stream temperature) is initialized at the design set point value of 86°C. To test the controller, the cooling jacket inlet temperature is again stepped from its design value of 46°C down to 40°C and back again.

One interesting outcome of this cascade implementation is the offset that occurs in the secondary P-Only loop when the process moves away form the design level of operation. Of more importance, however, is that the cascade shows improved performance over the single loop case in maintaining the reactor product temperature near the constant set point of 86°C.

Specifically, while the measured reactor exit temperature deviations for the single loop PI controller range as high 2.5°C during the event, here the cascade limits the maximum deviation to about 1.0°C. This improved performance did not come free, however, as the cascade architecture requires an additional sensor, controller and tuning effort.

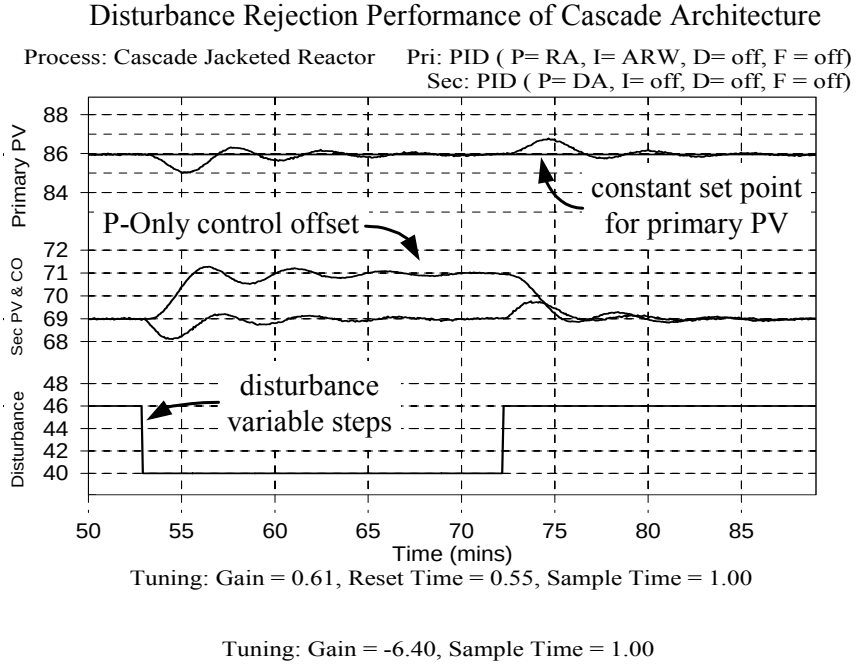


Figure 18.13 – Disturbance rejection in the jacketed reactor with a P-Only to PI cascade Process variable deviations reach 1.0°C (compare to Fig. 11.7)

18.8 Set Point Tracking Comparison of Single Loop and Cascade Control

The cascade architecture does not provide benefit in tracking set point changes and this is illustrated in Fig. 18.14 for the jacketed reactor. The plot on the left shows the set point tracking performance of the single loop PI controller while the plot on the right shows that for the cascade. The lower trace on each plot is the controller output signal sent to the valve located on the jacket cooling outlet stream.

The performance of the single loop controller on the left may reasonably be considered superior to that of the cascade on the right. Do not forget, however, that the single loop PI controller was tuned using the IMC for set point tracking and disturbance rejection correlation. The primary loop of the cascade was tuned using the ITAE for disturbance rejection correlation, which generally provides more aggressive tuning values.

This single example is not sufficient to support a claim that one architecture performs better than the other for set point tracking. We close this chapter, however, by restating once again that before considering a cascade architecture, be sure the controller design objective is the rejection of disturbances.

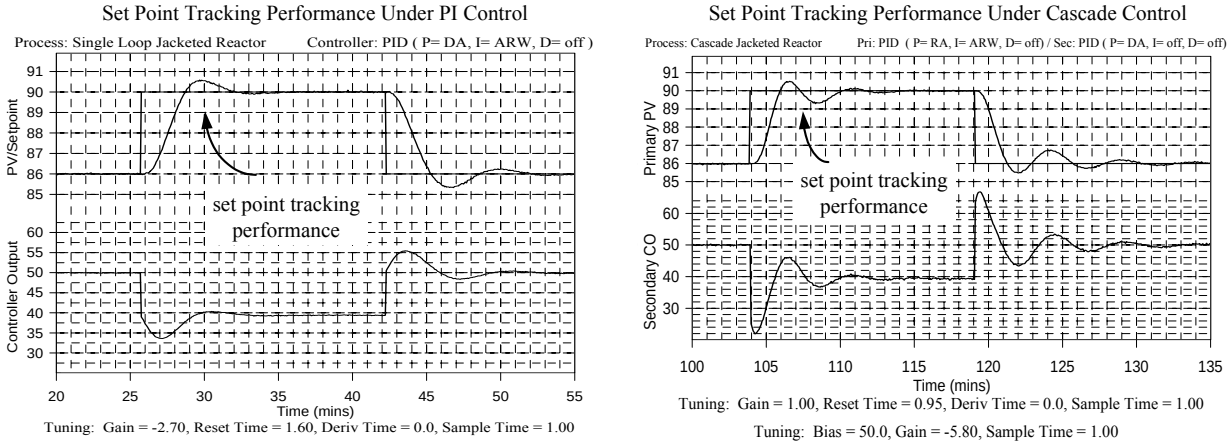


Figure 18.14 – Comparing set point tracking of single feedback loop to that of the cascade

18.9 Exercises

Q-18.1 Repeat the cascade design study presented in this chapter, only for comparison,

a) use a PID algorithm for the outer primary loop
b) use a PI algorithm for the inner secondary loop

Comment on how these changes impact controller performance. Include plots that support your conclusions.

19. Feed Forward Control

19.1 Another Architecture for Improved Disturbance Rejection

As discussed in the previous chapter, the most popular architectures for improved regulatory performance are cascade control and the feed forward with feedback trim architecture discussed here. Like cascade, a feed forward implementation requires additional instrumentation and engineering time and neither benefits nor detracts from set point tracking performance.

Before considering feed forward for your application, be sure your control objective is disturbance rejection. Choose feed forward over cascade if one specific disturbance variable is responsible for repeated, costly disruptions to stable operation, or if an appropriate secondary measured process variable as required for a cascade implementation cannot be identified.

19.2 The Feed Forward Architecture

The process block diagram used in the cascade discussion is shown in Fig. 19.1 with the secondary cascade control loop removed. As we have learned, a cascade architecture can improve the rejection performance of Disturbance I but not Disturbance II because there is no secondary process variable associated with that disturbance. As indicated in this figure, feed forward with feedback trim holds potential for improved disturbance rejection for both classes of disturbance.

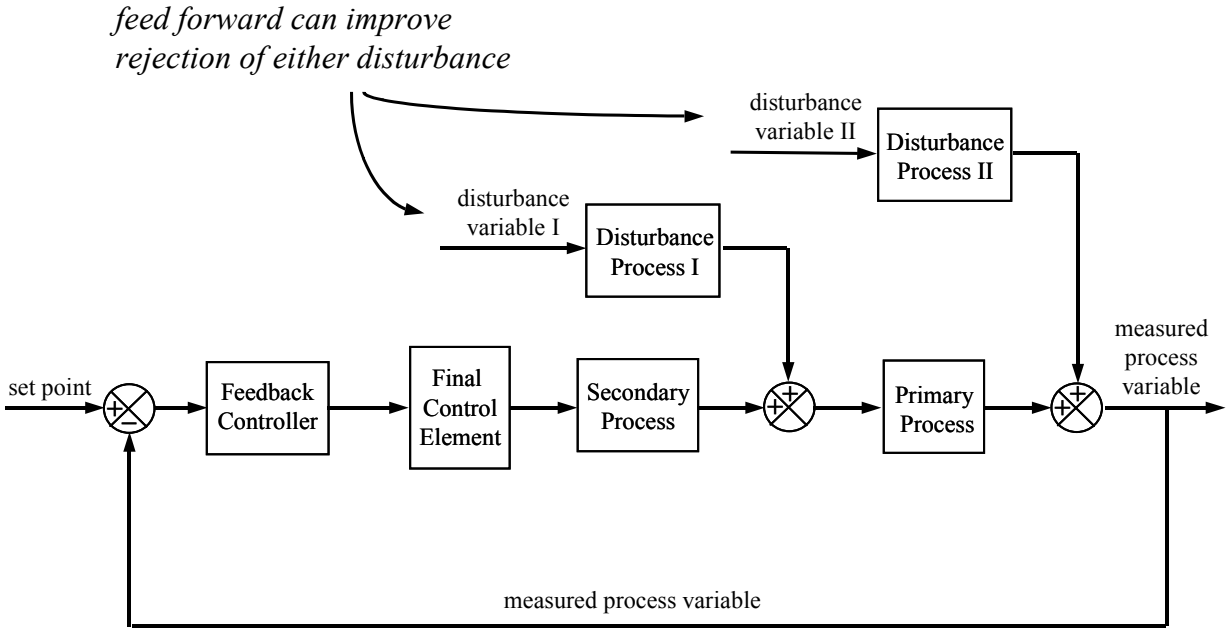


Figure 19.1 - Feed forward controllers do not require a secondary process variable

When a traditional feedback controller works to reject a disturbance, corrective action begins only after the measured process variable has been forced away from set point. Damage to stable operation is in progress before a traditional feedback controller even begins to respond.

Consider that many disturbances originate in some other part of the plant. A measurable series of events occur that cause that "distant" event to ultimately impact your process. From this view, the traditional feedback controller simply starts too late to be effective in reducing or negating the impact of a disturbance.

A feed forward controller gains advantage by using a sensor to directly measure the disturbance *before* it reaches the process. As shown in Fig. 19.2, a feed forward element receives the disturbance measurement signal and uses it to compute and schedule preemptive control actions that will counter the impact of the disturbance just as it reaches the measured process variable.

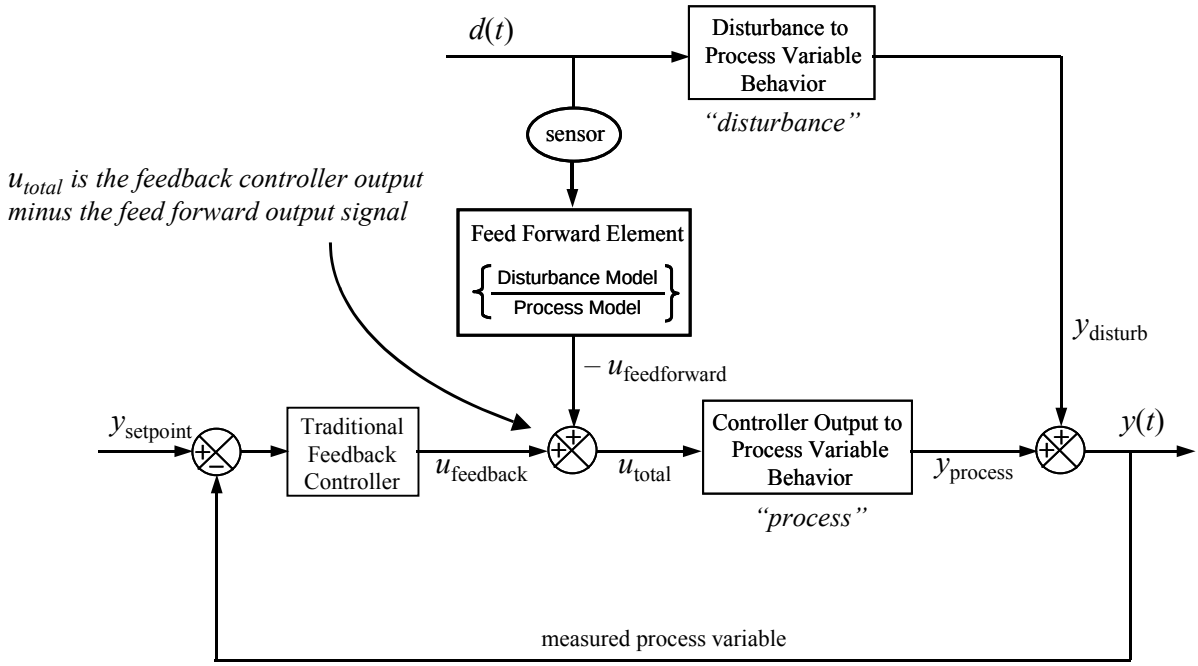


Figure 19.2 - Architecture of a feed forward controller with feedback trim

A feed forward implementation requires the purchase and installation of a sensor and the construction of a feed forward model element. This element is comprised of a disturbance model and a process model. Both models are linear in form. The computation performed by the feed forward element may be thought of as a two step procedure:

- The *disturbance* model receives disturbance measurement, $d(t)$, and predicts an "impact profile," or when and by how much the measured process variable, $y(t)$, will be impacted.

- Given this predicted sequence of disruption to $y(t)$, the *process* model then back calculates a series of control actions, $u_{feedforward}(t)$, that will exactly counteract the disturbance as it arrives so the measured process variable remains constant at set point.

Implementation requires that these linear models be programmed into the control computer. As we have learned, linear models never exactly describe real process behavior. So although a feed forward element can dramatically reduce the impact of a disturbance, it never will succeed in providing perfect disturbance rejection.

To account for model inaccuracies, the feed forward signal is combined with traditional feedback control action, $u_{feedback}(t)$, to create a total controller output, $u_{total}(t)$. The feedback controller provides *trim*. That is, it rejects those portion of the measured disturbance that make it past the feed forward element and reach the measured process variable. The feedback controller also works to reject all other unmeasured disturbances affecting plant operation and provides set point tracking capabilities as needed.

Notice in Fig. 19.2 that the computed feed forward control action, $u_{feedforward}$, is assigned a negative sign. It is subtracted from the feedback output signal to create a total controller output:

$$u_{total}(t) = u_{feedback}(t) - u_{feedforward}(t) \tag{19.1}$$

This makes sense because if the disturbance model predicts that a particular disturbance will cause the measured process variable to, say, move *up* by a certain amount over a period of time, the process model must compute feed forward control actions that cause the measured process variable to move *down* in the same fashion. The negative sign enables "action opposite to prediction" to be taken.

19.3 An Illustrative Example

Flash Drum Process Revisited

To better understand the construction and use of feed forward with feedback trim, we revisit the flash drum process, shown again in Fig. 19.3. We used this example of a disturbance rejection objective in Section 18.3. Since feed forward is an alternative strategy for improved disturbance rejection, we use the same process here so we can compare the two techniques.

As shown in Fig. 19.3, a hot liquid under high pressure flashes as it enters the drum (experiences a large and sudden pressure drop). The result is a vapor and liquid phase. The design objective is to control liquid level in the drum. If level is too high, the controller should increase liquid drain flow rate. If level is too low, the controller should decrease liquid drain flow rate. The disturbance of concern is the pressure in the overhead vapor phase, which can change without warning due to the behavior of some unidentified down stream unit.

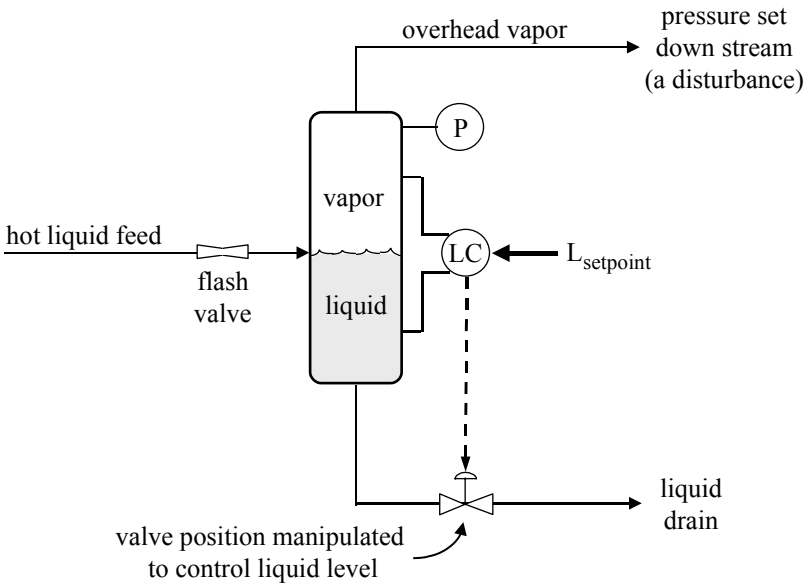


Figure 19.3 – Single-loop level control of a flash drum

Problems with Single Loop Control

The liquid drain flow rate is a function of several variables, including valve position, hydrostatic head (height of the liquid), and pressure of the vapor pushing down on the liquid (a disturbance). Suppose the vapor phase pressure starts decreasing. This disturbance will cause the pressure pushing down on the liquid interface to decrease. If the pressure decrease occurs quickly enough, the controller can actually be opening the valve yet the liquid drain flow rate can continue to decrease. Alternatively, if the pressure in the vapor phase starts increasing, the controller can be closing the valve yet the liquid drain flow rate can actually increase.

A Feed Forward Solution

As shown in Fig. 19.4, a feed forward control strategy includes a sensor that directly measures changes in a specific disturbance variable, and a feed forward model element to compute corrective control actions based on that measurement. In this case, a sensor is installed to detect changes in the pressure of the vapor phase.

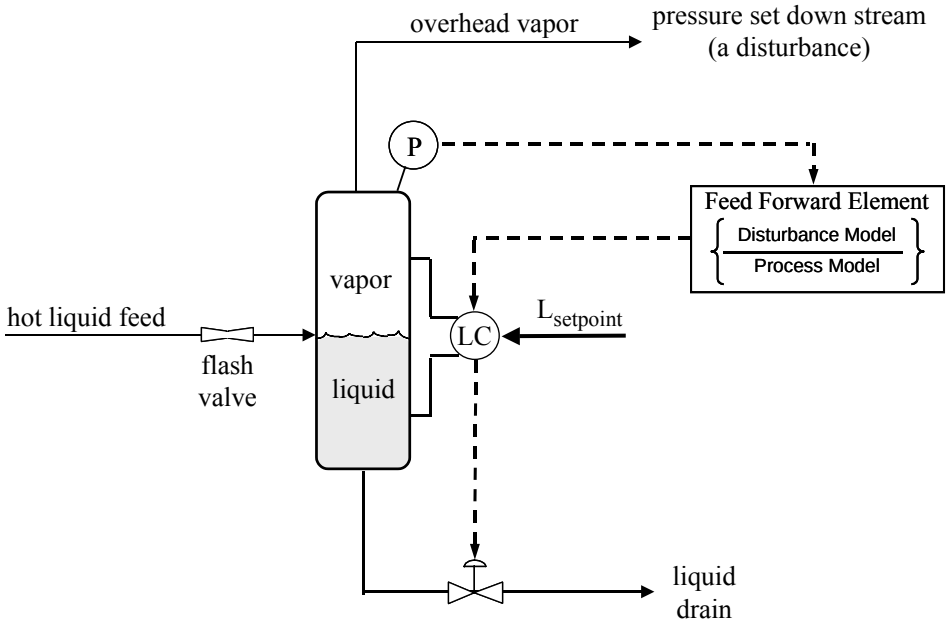


Figure 19.4 – Controlling liquid level with a feed forward with feedback trim architecture

As the pressure signal is received by the feed forward element, the disturbance model continually predicts a liquid level impact profile, or how far and how fast the liquid level will change for a given change in the vapor phase pressure. For example, if the pressure begins to increase, the model will predict how the drum level will fall as liquid is forced out the drain faster.

Based on this impact profile prediction, the process model then computes a precise sequence of control actions (adjustments to the liquid drain valve), as well as the timing of these actions. The goal is to adjust the valve so that drain flow rate does not change as pressure changes, and thus the impending upset to drum level will be canceled.

With reasonably accurate models, the feed forward controller can substantially reduce the impact of pressure changes on liquid level. Perfect elimination of the disturbance is not likely because the linear feed forward dynamic models will not exactly describe the nonlinear and time varying behavior of this (or any real) process.

To make up for plant-model mismatch, the traditional feedback level controller, represented by the LC in the circle attached to the tank, trims the control system. That is, the feedback controller rejects those portions of any pressure disturbance that make it past the feed forward controller and succeed in impacting the liquid level. It also enables rejection of all other disturbances to the liquid level control process (such as changes in feed composition) and provides set point tracking capabilities when required.

19.4 Feed Forward Control Design

Design Criteria

Implementation of a feed forward element requires the installation of a sensor that directly measures the disturbance variable and the programming of a feed forward element comprised of a process and disturbance dynamic model. To benefit from this technology, two design criteria for success are:

1) The process and disturbance dynamic models must reasonably describe the controller output to measured process variable behavior and disturbance to measured process variable behavior respectively, and

2) The process dead time (controller output to measured process variable dead time) must be shorter than the disturbance dead time (disturbance to measured process variable dead time).

The first criterion is rather obvious. If the models don't describe the behavior of the plant, then the feed forward computations will be of questionable value.

The second criterion is more subtle. Suppose a plant has a disturbance dead time that is shorter than the process dead time. Further suppose a disturbance occurs and the feed forward controller instantly responds with appropriate control actions. Because of the dead time difference, the disturbance will reach the measured process variable before the controller manipulations, even though both happen at the same time in this example. The disturbance will already be disrupting the process before the first disturbance rejection control actions even arrive.

At the limit, the control actions need to arrive at the same time as the disturbance (equal dead times) for reasonable disturbance rejection. If conditions are such that control actions can arrive first (if the process dead time is shorter), the feed forward controller can be most effective in rejecting a measured disturbance.

19.5 Feed Forward Control Theory

Obtaining Process and Disturbance Models

Laplace transforms are used in the next sections to make the derivation of the feed forward element mathematically correct. The presentation uses basic math rules, so even if you are not familiar with Laplace transforms, the logic of the presentation will (hopefully) make sense.

Methods for obtaining a process model have been discussed in detail in previous chapters. In short, a process data set is generated by stepping, pulsing or otherwise perturbing the controller output signal, $u(t)$, and recording the measured variable, $y(t)$, as the process responds. The process should initially be at steady state at the design level of operation and the response of the measured process variable should clearly dominate the noise in the measurement signal.

A process model is obtained by fitting the data with a linear dynamic equation ranging from first order without dead time (FO) up through second order with dead time and lead time (SOPDT w/ L). If we call this process model of unspecified form $G_P(s)$, then in the Laplace domain we can say

$$Y(s) = G_P(s)U(s) \tag{19.2}$$

That is, given knowledge of the controller output, Eq. 19.2 permits the expected behavior of the measured process variable to be computed. This equation can rearranged to say that, given a change in the measured process variable, the controller output signal sequence that would cause that change can be back-calculated:

$$U(s) = \left[1/G_P(s)\right]Y(s) \tag{19.3}$$

The disturbance model is created in the identical fashion to the process model except it is the disturbance variable, $d(t)$, that must be perturbed in some fashion. Since disturbance variables are beyond the control of a loop (which is what makes them disturbances), it is not always possible to step or pulse the disturbance at will. Hence, it may be necessary to wait for moments of opportunity and collect data during an actual disturbance event. An alternative is to sift through data logs to find a disturbance event that produced data suitable for model fitting. In any case, a linear model ranging from FO up through SOPDT w/ L is fit to this data. In the Laplace domain, this disturbance model, $G_D(s)$, is expressed:

$$Y(s) = G_D(s)D(s) \tag{19.4}$$

So with knowledge of changes in the disturbance variable (provided by the added sensor), Eq. 19.4 permits the impact profile of each disturbance on the measured process variable to be computed.

Deriving the Feed Forward Element

Once online, the new sensor measures the disturbance variable and the signal is fed through Eq. 19.5, the disturbance model of the feed forward element. This model continually updates Y_{disturb} as labeled in Fig. 19.2, which is a prediction of the impact profile (how and when the measured process variable will be impacted by the disturbance) as:

$$Y_{\text{disturb}}(s) = G_D(s)D(s) \tag{19.5}$$

This predicted sequence of disruption, $Y_{\text{disturb}}(s)$, is then fed to Eq. 19.6, the process model of the feed forward element, to back calculate a series of control actions, $U_{\text{feedforward}}(s)$. The result is a sequence of control actions that will cause the measured process variable to behave just like this predicted disturbance impact profile (we add a negative sign in Eq. 19.8 to make the feed forward actions oppose the disturbance):

$$U_{\text{feedforward}}(s) = \left[1/G_P(s)\right]Y_{\text{disturb}}(s) \tag{19.6}$$

Substituting Eq. 19.5 into Eq. 19.6 completes the "disturbance model divided by process model" feed forward element:

$$U_{\text{feedforward}}(s) = \left[G_D(s)/G_P(s)\right]D(s) \tag{19.7}$$

Eq. 19.7 computes a series of control actions that cause the measured process variable to duplicate the predicted disturbance impact profile. A negative sign is added so the feed forward controller acts in a manner opposite to this, thus canceling the impact of the disturbance. This negative feed forward action is combined with the traditional feedback control output signal to yield the total controller output:

$$U_{\text{total}}(s) = U_{\text{feedback}}(s) - U_{\text{feedforward}}(s) \tag{19.8}$$

19.6 Limits on the Form of the Feed Forward Model

Highest Model Order

The linear model range of FO (first order) up through SOPDT w/ L (second order plus dead time with lead time) are mentioned in the preceding discussion because they are the choices available in LOOP-PRO. Feed forward theory permits third, fourth and higher order linear models to be used in the feed forward element.

When working with a real plant, obtaining data that will yield accurate values for the three parameters of a FOPDT model is surprisingly challenging, and an accurate FOPDT model is usually capable of providing effective feed forward disturbance rejection. Obtaining a data set so rich in dynamic information and absent of disturbance influences that it can yield accurate values for the five parameters of a SOPDT w/ L model is very difficult in real applications and pushes practical implementation near the limit. Only the rarest of applications would benefit from a model more complex than the choices available in LOOP-PRO.

A SOPDT w/ L process model in the time domain has the following form:

$$\tau_{P1}\tau_{P2}\frac{d^2y(t)}{dt^2} + (\tau_{P1} + \tau_{P2})\frac{dy(t)}{dt} + y(t) = K_P\left[u(t-\theta_P) + \tau_{PL}\frac{du(t-\theta_P)}{dt}\right] \qquad (19.9)$$

In the Laplace domain this same process model is expressed as follows:

$$Y(s) = \frac{K_P(\tau_{PL}s + 1)\,e^{-\theta_P s}}{(\tau_{P1}s + 1)(\tau_{P2}s + 1)}\,U(s) \qquad (19.10)$$

The model of Eq. 19.10 is identical to that of Eq. 19.9. Both are linear ordinary differential equations with constant coefficients. If a SOPDT w/ L model is used for both the process and disturbance models in Eq. 19.7, the feed forward element becomes

$$U_{\text{feedforward}}(s) = \left\{\left(\frac{K_D}{K_P}\right)\left[\frac{(\tau_{P1}s+1)(\tau_{P2}s+1)(\tau_{DL}s+1)}{(\tau_{D1}s+1)(\tau_{D2}s+1)(\tau_{PL}s+1)}\right]e^{-(\theta_D-\theta_P)s}\right\}D(s) \qquad (19.11)$$

Notice that both process time constants are in the numerator along with the disturbance lead time and the both disturbance time constants are in the denominator along with the process lead time. This form is referred to as a *dynamic* feed forward element because the time dependent variables, including time constants, lead times and dead times, are included in the feed forward computation.

Dead Time Difference

We discussed previously that the process dead time must be shorter than the disturbance dead time for effective feed forward disturbance rejection. In fact, LOOP-PRO does not even permit entry of a larger process dead time. This is not a limitation of the software but a requirement of the math.

If $\theta_P > \theta_D$, then Eq. 19.11 would be required to compute corrective control actions that are to be implemented *before* the disturbance change was even first detected. Knowledge of the future, an outcome possible in the sterile world of mathematics, is simply not possible in the real world of process control. Formulations that require unknowable information or impossible actions are said to be *unrealizable*.

Suppose for a particular application, you determine that the disturbance dead time is indeed shorter than the process dead time. Best practice is to arbitrarily set the process dead time equal to the disturbance dead time when entering values into the feed forward input form. Disturbance rejection performance will suffer but at least the feed forward calculations will yield control actions that make physical sense.

Model Order Ratio

A second way an impossible or unrealizable feed forward model can result is through mismatching of the time constant and lead time terms for the process and disturbance models. As mentioned previously, both process time constants are in the numerator along with the disturbance lead time and the both disturbance time constants are in the denominator along with the process lead time.

A model that is physically realizable requires that the number of "time" terms in the numerator be less than or equal to the number of such terms in the denominator. Again, this is not a limitation of LOOP-PRO but a consequence that the mathematics must compute actions that are physically possible to implement. There are many ways to create a realizable feed forward element. A few examples of permitted and prohibited forms are illustrated below.

Permitted: FO or FOPDT Process model; SO or SOPDT Disturbance Model

$$U_{\text{feedforward}}(s) = \left\{\left(\frac{K_D}{K_P}\right)\left[\frac{(\tau_{P1}s+1)}{(\tau_{D1}s+1)(\tau_{D2}s+1)}\right]e^{-(\theta_D-\theta_P)s}\right\}D(s) \tag{19.12}$$

Permitted: FO or FOPDT Process model; SO w/ L or SOPDT w/L Disturbance Model

$$U_{\text{feedforward}}(s) = \left\{\left(\frac{K_D}{K_P}\right)\left[\frac{(\tau_{P1}s+1)(\tau_{DL}s+1)}{(\tau_{D1}s+1)(\tau_{D2}s+1)}\right]e^{-(\theta_D-\theta_P)s}\right\}D(s) \tag{19.13}$$

Permitted: SO or SOPDT Process model; SO or SOPDT Disturbance Model

$$U_{\text{feedforward}}(s) = \left\{\left(\frac{K_D}{K_P}\right)\left[\frac{(\tau_{P1}s+1)(\tau_{P2}s+1)}{(\tau_{D1}s+1)(\tau_{D2}s+1)}\right]e^{-(\theta_D-\theta_P)s}\right\}D(s) \tag{19.14}$$

NOT Permitted: SO or SOPDT Process model; SO w/ L or SOPDT w/L Disturbance Model

$$U_{\text{feedforward}}(s) = \left\{\left(\frac{K_D}{K_P}\right)\left[\frac{(\tau_{P1}s+1)(\tau_{P2}s+1)(\tau_{DL}s+1)}{(\tau_{D1}s+1)(\tau_{D2}s+1)}\right]e^{-(\theta_D-\theta_P)s}\right\}D(s) \tag{19.15}$$

This last form is unrealizable because there are more "time" terms in the numerator than denominator, requiring computation of physically impossible actions.

Positive Process Lead Required

The model of Eq. 19.11, called a Laplace domain transfer function, can also be expressed as a rather complicated linear ordinary differential equation with constant coefficients in the time domain. The solution of this differential equation is unstable (diverges) if any of the "time" terms in the denominator are negative.

There is no concern that τ_{D1} or τ_{D2} in the denominator will be negative as time constants by definition are always positive. The process lead time, τ_{PL}, would be negative if the measured process variable displays an inverse (nonminimum phase) response to a step change in the controller output signal. But since τ_{PL} is in the denominator, it cannot be negative or the feed forward computations will go unstable.

If your process displays an inverse response and thus yields a negative τ_{PL} after a model fit, best practice is to use a FOPDT model and approximate the inverse portion of the response as a long dead time. Interestingly, the disturbance can have a negative lead element such as that exhibited by the heat exchanger because the disturbance lead term is in the numerator of the feed forward element.

19.7 Feed Forward Disturbance Rejection in the Jacketed Reactor

The control objective of this investigation is the same as that used in the cascade study of Chapter 18. We seek to maintain the reactor exit stream temperature at set point by rejecting disturbances caused by changes in the cooling jacket inlet temperature. The design level of operation is a measured reactor exit stream temperature of 86°C. The cooling jacket inlet temperature (disturbance), normally at 46°C, is known to spike as low as 40°C. Open loop studies establish that a controller output of 50% causes the reactor to steady at the design measured exit stream temperature of 86°C when the cooling jacket inlet temperature is at its normal (design) value of 46°C.

Single Loop Disturbance Rejection

As detailed in cascade control chapter and shown again in Fig. 19.5, the controller output is stepped from the design value of 50% up to 53%, then down to 47%, and finally back to 50%, forcing the measured reactor exit stream temperature to exhibit a clear response after each controller output step that dominates any measurement noise.

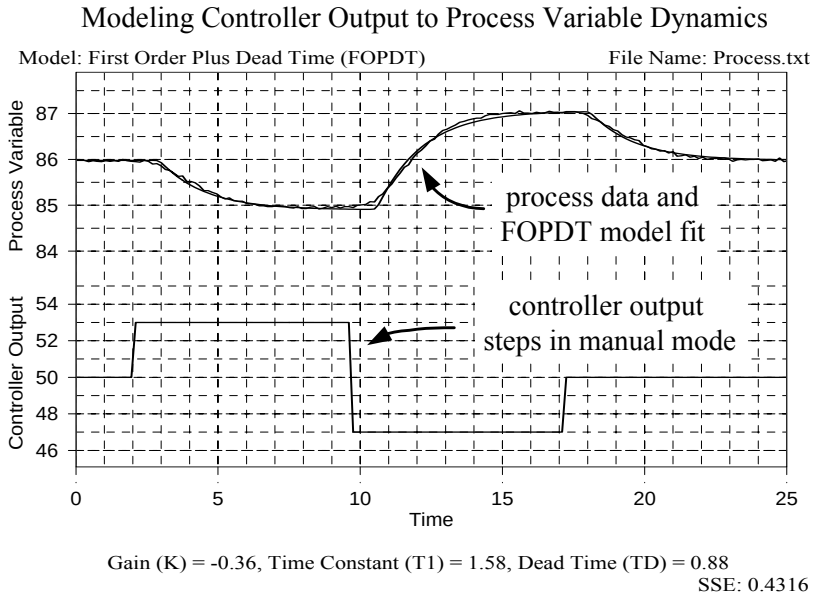


Figure 19.5 – FOPDT model fit of single loop controller output to measured process variable data

A FOPDT fit of the dynamic process data as computed by *Design Tools* yields the model parameters:

$$\text{Process Gain, } K_P = -0.36 \text{ °C/\%} \tag{19.16a}$$

$$\text{Time Constant, } \tau_P = 1.58 \text{ min} \tag{19.16b}$$

$$\text{Dead Time, } \theta_P = 0.88 \text{ min} \tag{19.16c}$$

These parameters were used in the standard IMC tuning correlation to obtain the PI tuning values:

$$\text{Controller Gain, } K_C = -2.7\ \%/°\text{C} \tag{19.17a}$$

$$\text{Reset Time, } \tau_I = 1.6\ \text{min} \tag{19.17b}$$

The disturbance rejection performance of the single loop PI controller using these tuning parameters is shown in Fig. 19.6. The label at the upper right of the plot confirms that the PI controller has a direct acting proportional term, an anti-reset windup integral term, and a derivative term that is off.

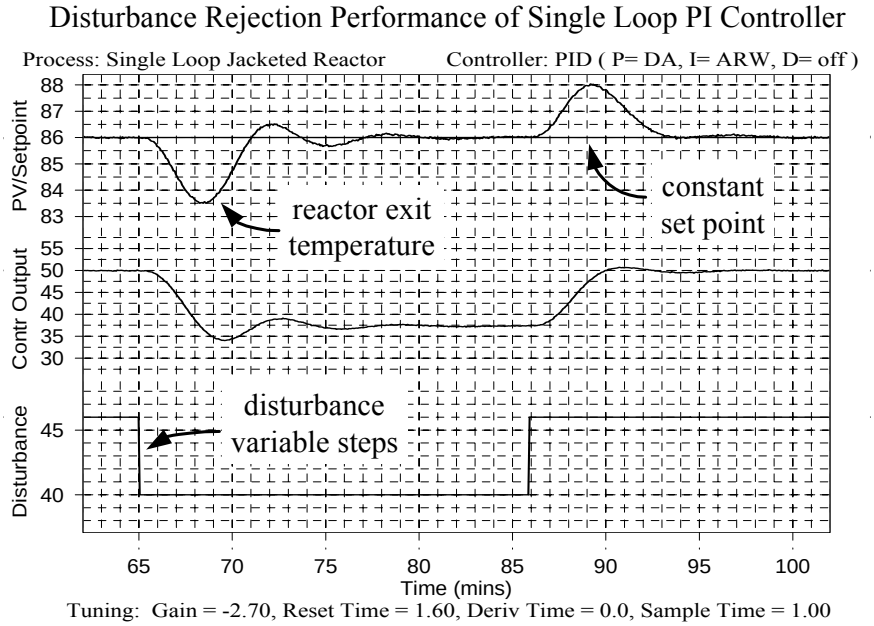


Figure 19.6 – Disturbance rejection in the jacketed reactor under single loop PI control Process variable deviations reach 2.5°C (compare to Fig. 19.9)

The measured reactor exit stream temperature is initially steady at the design set point value of 86°C in Fig. 19.6. To test the controller, the cooling jacket inlet temperature is stepped from its design value of 46°C down to 40°C and back again. As shown, the single loop PI controller is able to maintain reactor exit stream temperature near the constant set point of 86°C, with deviations ranging as high as 2.5°C during the event.

Feed Forward Disturbance Rejection

To construct a feed forward controller, the cooling jacket inlet temperature (the disturbance) is measured as shown in Fig. 19.7. The signal from this disturbance temperature sensor is sent to a feed forward element comprised of a process model and disturbance model.

A FOPDT model reasonably describe the dynamics of the controller output to measured process variable as shown in Fig. 19.5. The parameter values of Eq. 19.16 are thus used to construct the process model of the feed forward element.

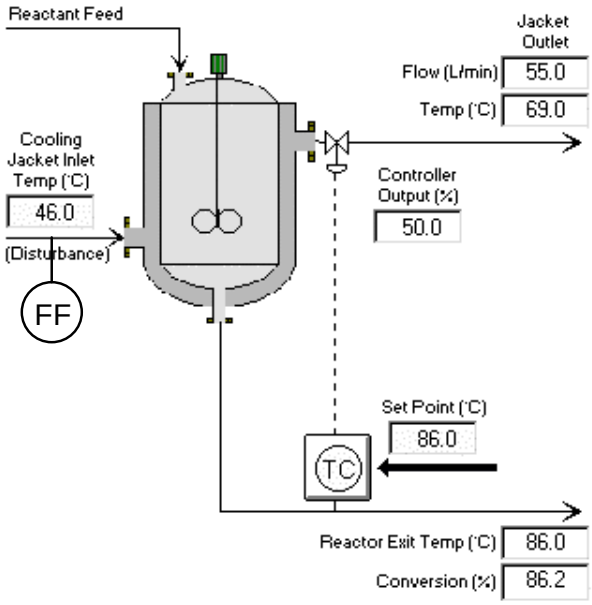


Figure 19.7 – Jacketed reactor with disturbance sensor for feed forward control

Generating disturbance driven data can be problematic on real processes if the disturbance variable cannot be manipulated at will. LOOP-PRO permits such disturbance manipulations. As shown in Fig. 19.8, the cooling jacket inlet temperature is stepped from the design value of 46°C up to 49°C, down to 43°C and back to 46°C. The measured reactor exit stream temperature exhibits a clear response after each step that dominates measurement noise.

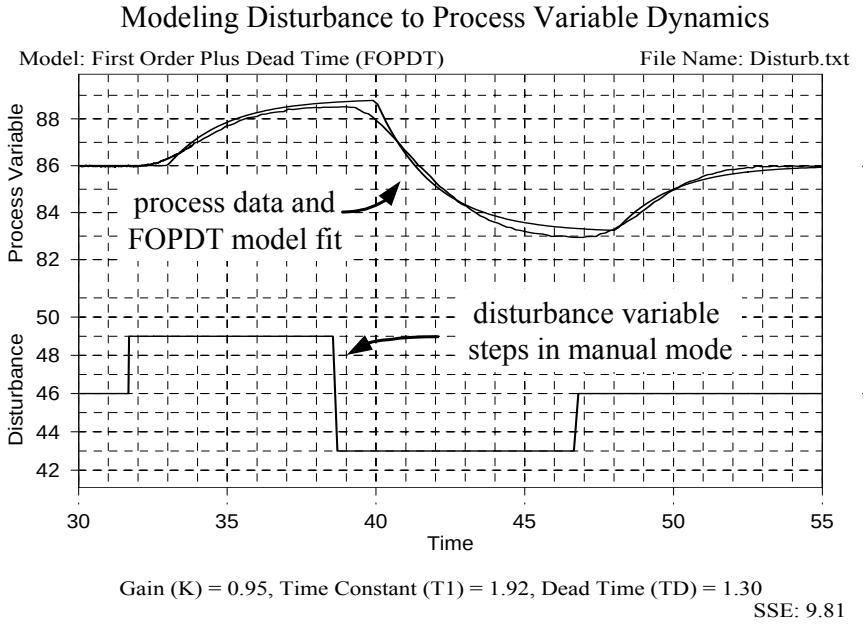


Figure 19.8 – FOPDT model fit of disturbance to measured process variable data

A FOPDT model reasonably describes the disturbance to measured process variable dynamics as shown in Fig. 19.8. The disturbance model parameters used to construct the disturbance model of the feed forward element are thus:

$$\text{Disturbance Gain, } K_D = 0.95 \frac{^{o}\text{C of reactor exit stream temperature}}{^{o}\text{C of cooling jacket inlet temperature}} \tag{19.18a}$$

$$\text{Disturbance Time Constant, } \tau_D = 1.92 \text{ min} \tag{19.18b}$$

$$\text{Disturbance Dead Time, } \theta_D = 1.30 \text{ min} \tag{19.18c}$$

Using the process model of Eq. 19.16 and the disturbance model of Eq. 19.18, the feed forward element is constructed:

$$U_{\text{feedforward}}(s) = \left\{ \left(\frac{0.95}{-0.36} \right) \left[\frac{(1.58s+1)}{(1.92s+1)} \right] e^{-(1.30-0.88)s} \right\} D(s) \tag{19.19}$$

Equation 19.19 is physically realizable because there are as many "time" terms in the denominator as in the numerator and because $\theta_D > \theta_P$.

This feed forward element is combined with the PI feedback controller used earlier in this chapter for the single loop PI controller study (see Fig. 19.5) to yield a feed forward with feedback trim architecture. Figure 19.9 shows the disturbance rejection performance of this controller. Just as in Fig. 19.5, the measured process variable (reactor exit stream temperature) is initialized at the design set point value of 86°C. To test the controller, the cooling jacket inlet temperature is stepped from its design value of 46°C down to 40°C and back again.

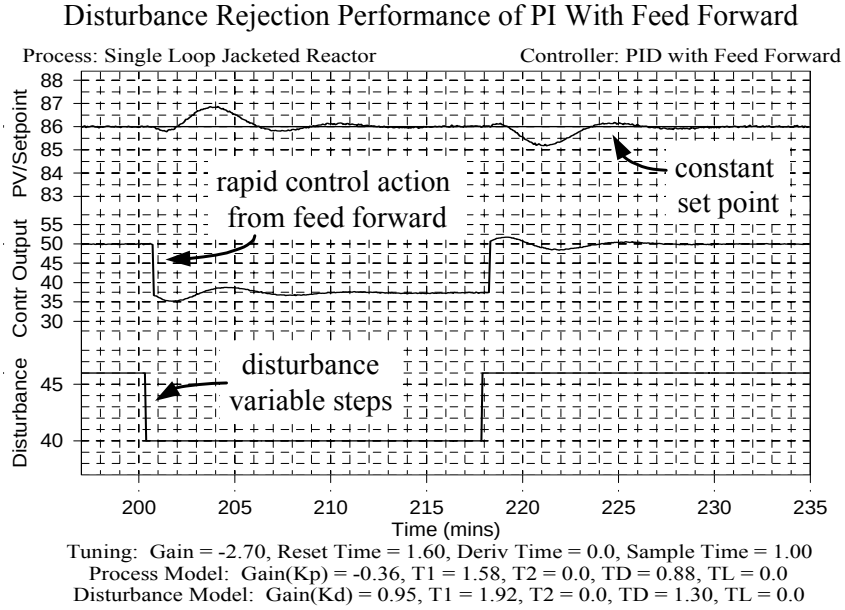


Figure 19.9 – Disturbance rejection in jacketed reactor under feed forward with feedback trim Process variable deviations reach 1.0°C (compare to Fig. 19.5)

The feed controller with feedback trim architecture performs better than the single loop case in maintaining the reactor product temperature near the constant set point of 86°C. Specifically, while

the measured reactor exit temperature deviations for the single loop PI controller range as high as high as 2.5°C during the event, this advanced architecture limits the maximum deviation to less than 1.0°C.

The improved performance did not come free, however, as an additional sensor, controller and tuning effort were required. As shown in Fig. 19.9, the feed forward controller initiated rapid compensating controller just after the disturbance event to minimize its impact on the measured process variable. Perfect disturbance rejection was not achieved because the FOPDT models only approximate the higher order and nonlinear behavior of the jacketed reactor process.

19.8 Static Feed Forward Control

The preceding discussion detailed the implementation of a *dynamic* feed forward controller. A dynamic feed forward controller employs the time dependent model variables in the feed forward computation when calculating a corrective action. These include the time constant and lead time terms that describe the speed of response of the process, and the dead time terms that describe the time delay before the response of the process begins.

A *static* feed forward controller does not consider the time dependent information in the feed forward computation. Only the size of response as described by the ratio of the steady state process gains is used. Hence, the general dynamic form of Eq. 19.11 reduces rather substantially to the static feed forward equation:

$$U_{\text{feedforward}}(s) = \left(\frac{K_D}{K_P}\right) D(s) \tag{19.20}$$

As Eq. 19.20 shows, the feed forward control action is simply a constant multiplied by the current value of the disturbance signal. The constant is computed as the ratio of the disturbance model gain divided by the process model gain.

The benefit of static feed forward is that many commercial controllers are sophisticated enough to permit implementation of this simple algorithm, and hence, there is no need for programming models on the control computer. It should come as no surprise, however, that the loss of time dependent information in computing corrective actions implies some degradation in disturbance rejection performance.

Figure 19.10 shows a side-by-side comparison of disturbance rejection performance in the jacketed reactor for three controller architectures. For all three cases, the jacketed reactor inlet temperature disturbance is stepped from 46°C down to 40°C and back again.

The left most portion of the plot shows disturbance rejection performance using a lone PI controller with no feed forward element. The center of the plot shows rejection of the same disturbance for a PI controller with a static feed forward element. The right most portion shows the performance for a PI controller with a full dynamic feed forward element.

Disturbance rejection performance improves from left to right in Fig. 19.10 because the controller is provided more information in the form of models that describe the behavior of the process and disturbance. And while we are cautious to avoid making broad assumptions from this single example, it is safe to assume that this general trend will hold true in a broad range of applications.

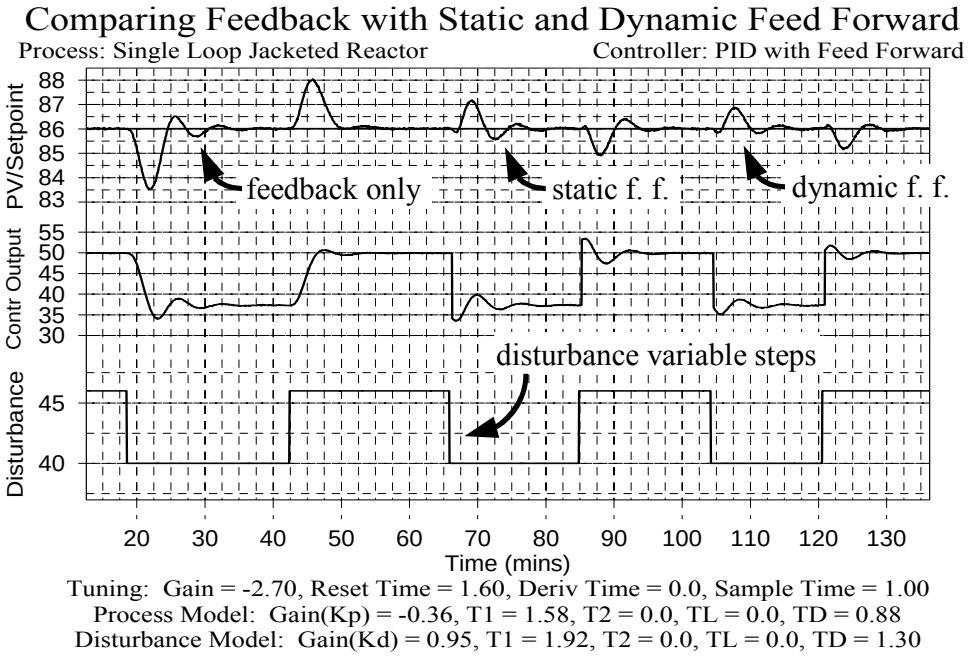


Figure 19.10 – Comparing set point tracking of single feedback loop to that of the cascade

19.9 Set Point Tracking Comparison of Single Loop and Feed Forward Control

The feed forward with feedback trim architecture does not provide benefit in tracking set point changes and this is illustrated in Fig. 19.11 for the jacketed reactor. The trace to the left of the plot shows the set point tracking performance of the single loop PI controller while the trace to the right shows the performance when the feed forward controller is added.

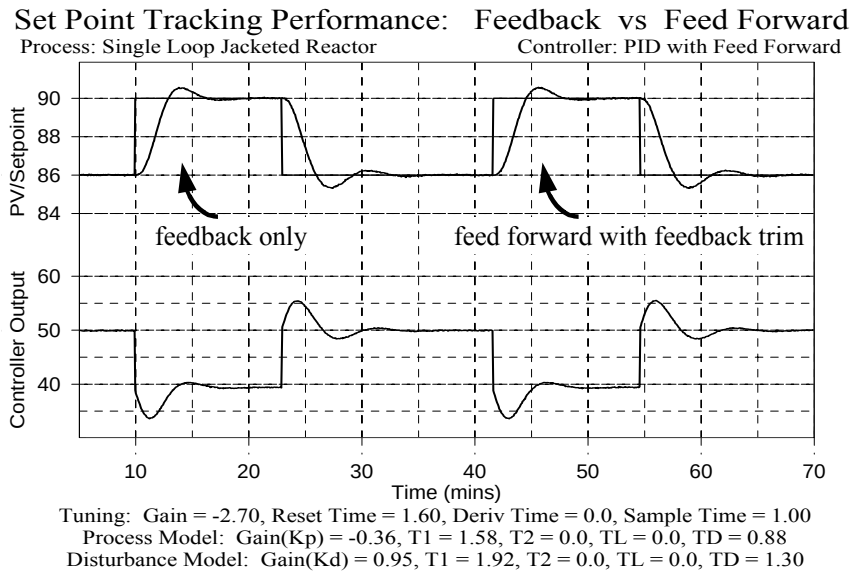


Figure 19.11 – Comparing set point tracking of single feedback loop to that of the cascade

As shown in Fig. 19.11, both architectures provide identical performance in tracking set point changes. This makes sense because for the feed forward with feedback trim architecture shown in Fig. 19.2, as long as the disturbance is constant, the feedback loop works alone to control the process. Since the same feedback loop PI controller tuning was used in both cases, we expect the set point tracking performance to be identical. Based on this observation, be sure your objective is the rejection of one particular disturbance before considering a feed forward architecture.

20. Multivariable Controller Interaction and Loop Decoupling

20.1 Multivariable Process Control

Except for cascade control, the control systems discussed to this point have all been single-loop architectures. That is, the control loops have had a single controller output signal that adjusts a single manipulated variable to impact a single measured process variable. In many real processes, however, there are two or more manipulated variables that when adjusted, each impact more than one measured process variable. These *multivariable* process control systems, with their *loop interactions*, present a new level of complexity for controller design.

LOOP-PRO's distillation column process, shown in Fig. 20.1, offers such a multivariable control challenge. It is a binary distillation column that separates benzene and toluene. The objective is to send a high percentage of benzene (and thus low percentage of toluene) out the top stream, and a low percentage of benzene (and thus high percentage of toluene) out the bottom stream. The column dynamic model employs tray-by-tray mass and energy calculations similar to that proposed by McCune and Gallier [*ISA Transactions*, **12**, 193, (1973)].

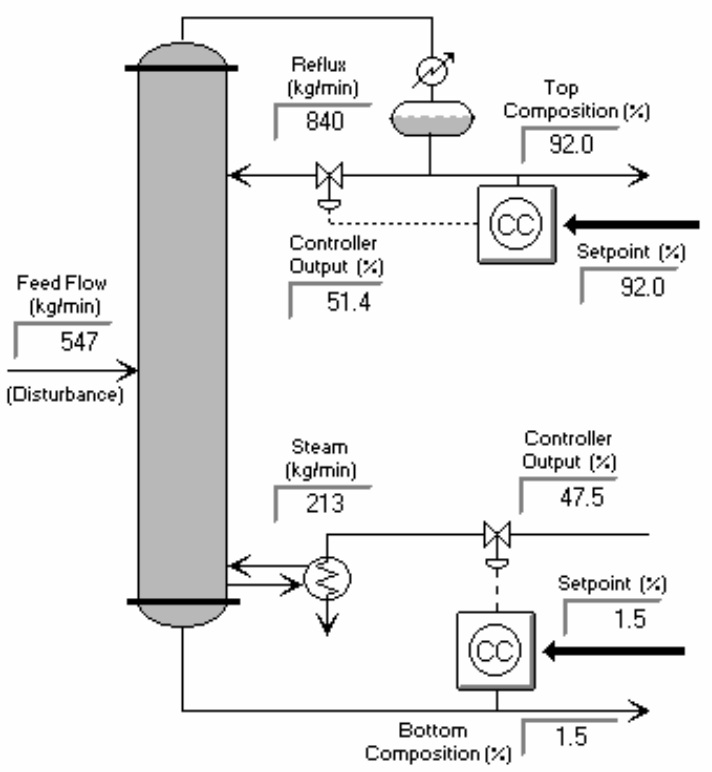


Figure 20.1 - Distillation column has top and bottom control loops that interact

To achieve the desired benzene-toluene separation, the top controller manipulates the reflux rate to control the top (distillate) composition. The bottom controller adjusts the rate of steam to the reboiler to control the bottom stream composition. Any change in feed rate to the column acts as a disturbance to the process. With two manipulated variables and two measured process variables, this is commonly called a two-by-two multi-input multi-output (2x2 MIMO) process.

To illustrate the loop interaction in this MIMO process, suppose the composition (or purity) of benzene in the top stream is below set point. The top controller will respond by increasing the flow rate of cold reflux into the column. This increased reflux flow will indeed increase the purity of benzene in the top stream. However, the additional cold liquid will work its way down the column, begin to cool the bottom, and as a result permit more benzene to flow out the bottom stream.

As the bottom composition moves off set point and produces a controller error, the bottom controller will compensate by increasing the flow of steam into the reboiler to heat up the bottom of the column. While this works to restore the desired purity to the bottom stream, unfortunately, it also results in an increase of hot vapors traveling up the column that eventually will cause the top of the column to begin to heat up.

As the top of the column heats up, the purity of benzene in the top stream again becomes too low. In response, the top controller compensates by further increasing the flow of cold reflux into the top of the column. The controller "fight," or multivariable interaction, begins.

20.2 Control Loop Interaction

Decouplers are essentially feed forward elements designed to reduce controller interaction in MIMO processes. The only difference between a feed forward element and a decoupler is that with a decoupler, the disturbance to be rejected is the action of another control loop on the process.

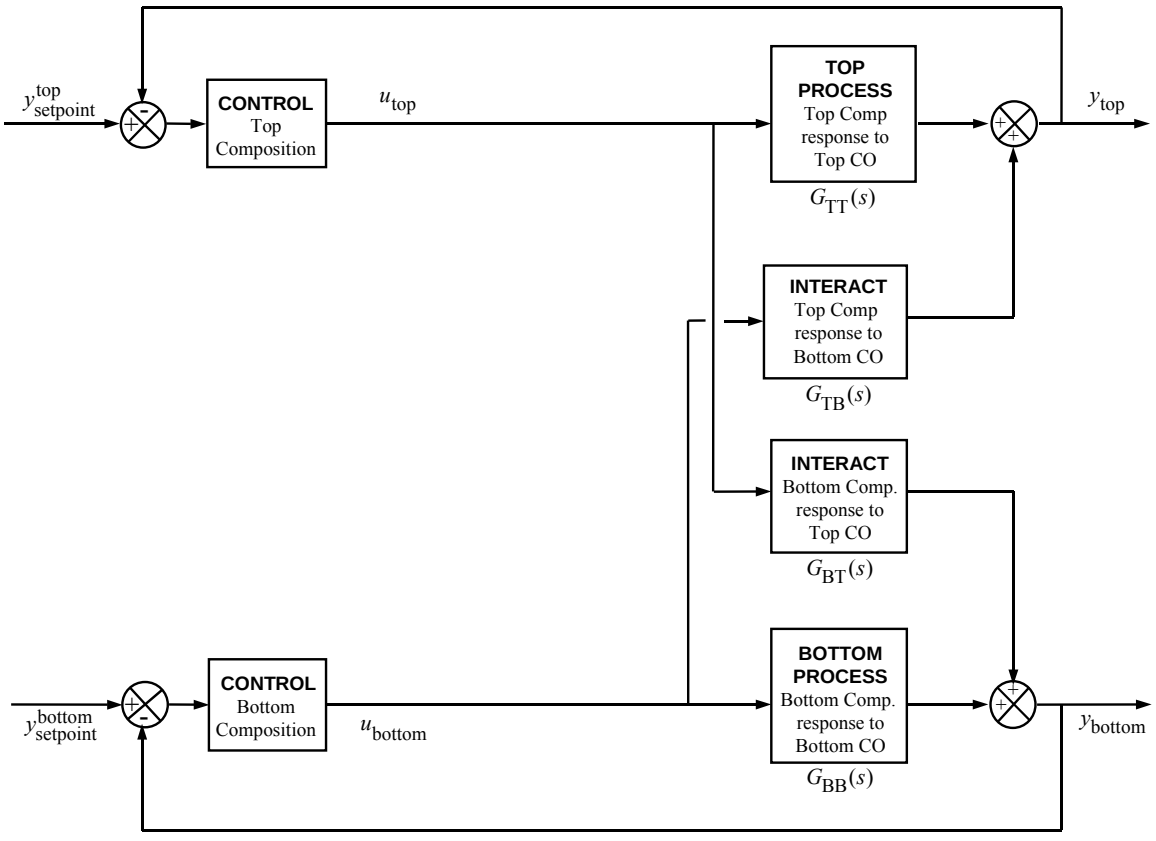


Figure 20.2 - Block diagram of top and bottom distillation control loops with "cross loop" interaction

As shown in Fig 20.2 for the distillation column, the "cross-loop disturbance" of the top stream composition is the bottom controller manipulations of the steam flow rate. The cross-loop disturbance of the bottom composition is the top controller manipulations of the cold reflux rate to the top of the column.

20.3 Decouplers are Feed Forward Controllers

Recall that a feed forward controller gains advantage by using a sensor to directly measure a disturbance while it is still distant from the measured process variable. A feed forward element receives this measurement and uses it to compute preemptive control actions designed to counter the impact of the disturbance just as it reaches the measured process variable.

A decoupler is a feed forward element where the disturbance is the cross-loop controller signal. Thus, a decoupler is comprised of a process model and a cross-loop disturbance model

- The *cross-loop disturbance* model receives the cross-loop controller signal and predicts an "impact profile," or when and by how much the process variable will be impacted.
- Given this predicted sequence of disruption, the *process* model then back calculates a series of control actions that will exactly counteract the cross-loop disturbance as it arrives so the measured process variable remains constant at set point.

Implementation of a decoupler does not require an added sensor to measure the disturbance because the current value of the cross-loop controller signal is readily available for use by the decoupler. However, developing and programming the dynamic process and cross-loop disturbance models is required for decoupler implementation.

To better understand the decoupling computation and its relation to feed forward control, we focus here on the top control loop of the distillation column. Though we concentrate on the logic of the computation in this discussion, Laplace transforms are used to make the derivation of the decoupler mathematically correct.

To develop a process model for the top control loop, a data set is generated by stepping, pulsing or otherwise perturbing the top controller output signal, $u_{top}(t)$, and recording the measured variable, $y_{top}(t)$, as the process responds. *Both loops should be in manual during this data collection exercise.* As always, the process should initially be at steady state at the design level of operation and the response of the top process variable should clearly dominate the noise in the measurement signal.

This process data set is fit with a linear dynamic model ranging from first order without dead time (FO) up through second order with dead time and lead time (SOPDT w/ L). If we call this top composition response to top controller output model $G_{TT}(s)$, then in the Laplace domain we can say:

$$Y_{top}(s) = G_{TT}(s)U_{top}(s) \tag{20.1}$$

That is, given knowledge of the top controller output, Eq. 20.1 permits the expected behavior of the top process variable to be computed. This equation can be rearranged to say that, given a change in the top process variable, the top controller output signal sequence that would cause that change can be back-calculated as:

$$U_{top}(s) = [1/G_{TT}(s)]\, Y_{top}(s) \tag{20.2}$$

The cross-loop disturbance model is created by perturbing the cross-loop controller output, which in this case is the bottom controller, $u_{bottom}(t)$, and recording how $y_{top}(t)$ responds. *Again, both controllers should be in manual during this data collection exercise.* We fit a linear dynamic model to this cross-loop disturbance data and call the resulting top composition response to bottom controller output model $G_{TB}(s)$.

In the Laplace domain we can then say:

$$Y_{top}(s) = G_{TB}(s)U_{bottom}(s) \tag{20.3}$$

Hence, with knowledge of controller signal $u_{bottom}(t)$, Eq. 20.3 lets us compute the impact profile of this cross loop disturbance on the top process variable, $y_{top}(t)$.

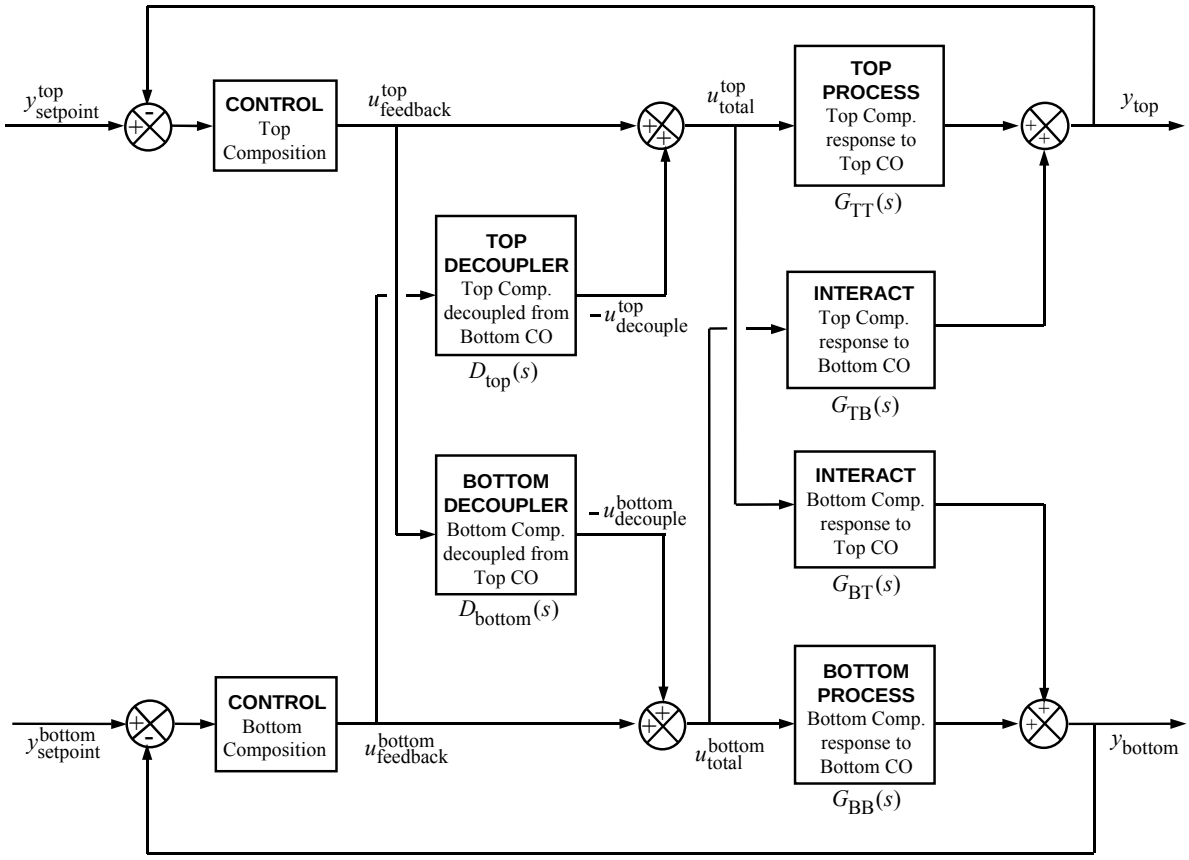


Figure 20.3 - Block diagram of top and bottom distillation control loops with cross loop interaction and decouplers

We now have the models needed to construct the decoupler D_{top}(s) as shown in Fig 20.3. As the bottom controller makes changes in $u^{bottom}_{feedback}(t)$, the signal is sent to Eq. 20.4, which is the cross-loop disturbance model portion of the top decoupler. This model continually updates $y^{*}_{top}(t)$, which is a prediction of the impact profile or how and when the top process variable will be impacted by the actions of the bottom controller. That is:

$$Y^{*}_{top}(s) = G_{TB}(s)U^{bottom}_{feedback}(t) \tag{20.4}$$

This predicted sequence of disruption, $y^*_{\text{top}}(t)$, is then fed to Eq. 20.5, the process model portion of the top decoupler, to back calculate a series of control actions, $u^{\text{top}}_{\text{decouple}}(t)$. This results in a sequence of control actions that will cause the top process variable to behave just like the predicted cross-loop disturbance impact profile (we add a negative sign in Eq. 20.7 to make the decoupling actions oppose the disturbance):

$$U^{\text{top}}_{\text{decouple}}(s) = \left[\frac{1}{G_{\text{TT}}(s)}\right] Y^*_{\text{top}}(s) \tag{20.5}$$

Substituting Eq. 20.4 into Eq. 20.5 completes the top decoupler, $D_{\text{top}}(s)$, that as shown in Eq. 20.6, computes decoupling actions based on knowledge of the bottom controller actions, $u^{\text{bottom}}_{\text{feedback}}(t)$:

$$U^{\text{top}}_{\text{decouple}}(s) = \left[\frac{G_{\text{TB}}(s)}{G_{\text{TT}}(s)}\right] U^{\text{bottom}}_{\text{feedback}}(s) = D_{\text{top}}(s) U^{\text{bottom}}_{\text{feedback}}(s) \tag{20.6}$$

Eq. 20.6 computes a series of control actions that cause the top process variable to duplicate the predicted cross-loop disturbance impact profile. A negative sign is added so the decoupler acts in a manner opposite to this, thus canceling the impact of the cross loop disturbance. This negative decoupling action is combined with the traditional feedback control signal of the top controller to yield the total controller output:

$$U^{\text{top}}_{\text{total}}(s) = U^{\text{top}}_{\text{feedback}}(s) - U^{\text{top}}_{\text{decouple}}(s) \tag{20.7}$$

The bottom decoupler can be derived in an analogous manner as:

$$U^{\text{bottom}}_{\text{decouple}}(s) = \left[\frac{G_{\text{BT}}(s)}{G_{\text{BB}}(s)}\right] U^{\text{top}}_{\text{feedback}}(s) = D_{\text{bottom}}(s) U^{\text{top}}_{\text{feedback}}(s) \tag{20.8}$$

And thus, to implement a 2x2 top and bottom decoupler, we need to develop *four* dynamic process models. For the top decoupler, the FOPDT process and disturbance model parameters entered are:

Process model: $G_{\text{TT}}(s)$ = top composition response to top controller output model
Disturbance model: $G_{\text{TB}}(s)$ = top composition response to bottom controller output model

For the bottom decoupler, the FOPDT process and disturbance model parameters entered are:

Process model: $G_{\text{BB}}(s)$ = bottom composition response to bottom controller output
Disturbance model: $G_{\text{BT}}(s)$ = bottom composition response to top controller output

These models are used to develop two decouplers that must be programmed into the control computer and are implement as shown in Fig 20.3:

$$\text{Top decoupler:} \qquad D_{\text{top}}(s) = \left[\frac{G_{\text{TB}}(s)}{G_{\text{TT}}(s)}\right] \tag{20.9}$$

$$\text{Bottom decoupler:} \quad D_{\text{bottom}}(s) = \left[\frac{G_{\text{BT}}(s)}{G_{\text{BB}}(s)} \right] \tag{20.10}$$

20.4 Distillation Study - Interacting Control Loops

We will study decouplers in the next section. Here we explore the multivariable interactions that occur between the top and bottom controllers on the distillation column when they are designed and implemented as independent or stand-alone loops.

The control objective is the design and tuning of a PI controller that can track set point steps in the top composition from 92% and 94% while the bottom composition remains constant at 1.5%. The design level for this study includes a feed flow rate to the column of 547 Kg/min (note that this is not the default startup value) and the steady state operating conditions:

Top ➔	$u_{\text{top}}(t)$	= 52 %		$y_{\text{top}}(t)$	= 92 %
Bottom ➔	$u_{\text{bottom}}(t)$	= 48 %		$y_{\text{bottom}}(t)$	= 1.5 %

Fig 20.1 shows the distillation column at these design conditions, only in that figure the control loops are in automatic.

Before tuning the loops, we investigate the dynamic behavior of the column using open loop doublet tests. The top controller output is stepped from 52% up to 55%, down to 49% and back to 52%. After the process settles, the bottom controller is stepped from 48% up to 51%, down to 45% and back to 48%. Fig. 20.4 shows the process variable responses to these tests.

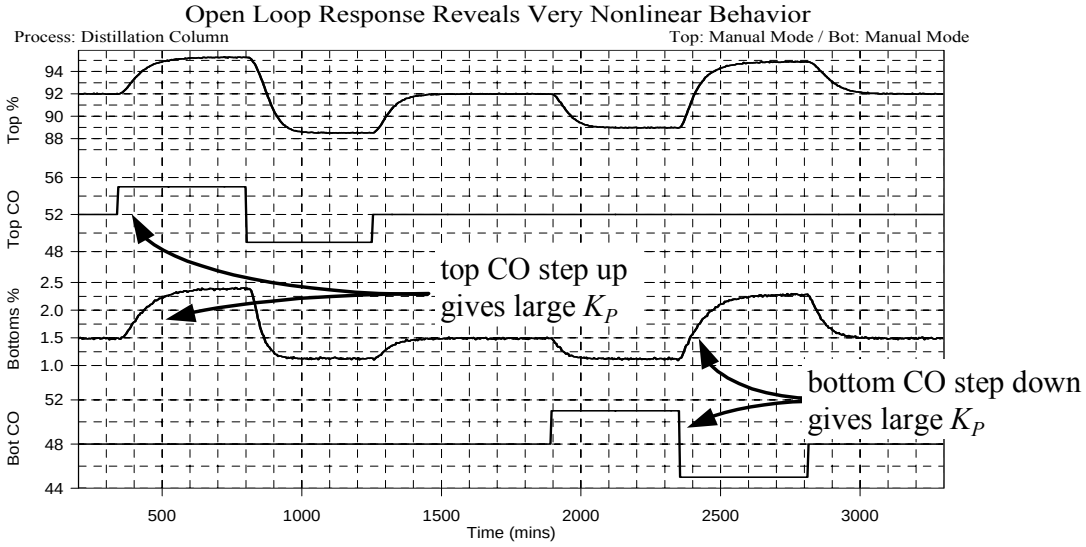


Figure 20.4 - Open loop step tests on the distillation column's top and bottom controller

The upper most trace of Fig. 20.4 shows that the top composition process variable is mildly nonlinear. That is, $y_{\text{top}}(t)$ responds both up and down from the design level of operation in roughly the same magnitudes whether forced by $u_{\text{top}}(t)$ or $u_{\text{bottom}}(t)$.

The bottom composition loop, on the other hand, is extremely nonlinear. In particular, the response of the bottom process variable is *three times larger* when $y_{\text{bottom}}(t)$ is increasing from the design level of operation compared to when it is decreasing. As shown in Fig. 20.4, this is true whether it is $u_{\text{top}}(t)$ or $u_{\text{bottom}}(t)$ that is forcing the response.

This extreme nonlinear behavior provides added control challenges beyond the loop interaction issue. To meet the challenge, we take advantage of the information in the above plot to design the dynamic tests for modeling and tuning. Rather than the standard doublet, we will pulse each variable in one direction only. We choose the direction that yields parameters leading to the most stable controller design.

Recall that tuning correlations compute controller gain, K_C, as proportional to the *inverse* of process gain, K_P. So for a conservative design, we want to use large values of K_P in the correlations to obtain small (and thus less aggressive) values of K_C. Exploiting the information contained in Fig. 13.4, we achieve the largest values of K_P by pulsing $u_{\text{top}}(t)$ up and $u_{\text{bottom}}(t)$ down when generating our dynamic test data.

Top Composition Control

We perform a pulse test to generate dynamic process data for the design of the top composition controller. As shown in Fig. 13.5, the top controller output, $u_{\text{top}}(t)$, is stepped from its design value of 52% up to 54% and back. The bottom controller is in manual mode during this test. *Design Tools* is used to fit a FOPDT model to the data. The process data and model fit are shown in Fig. 20.5.

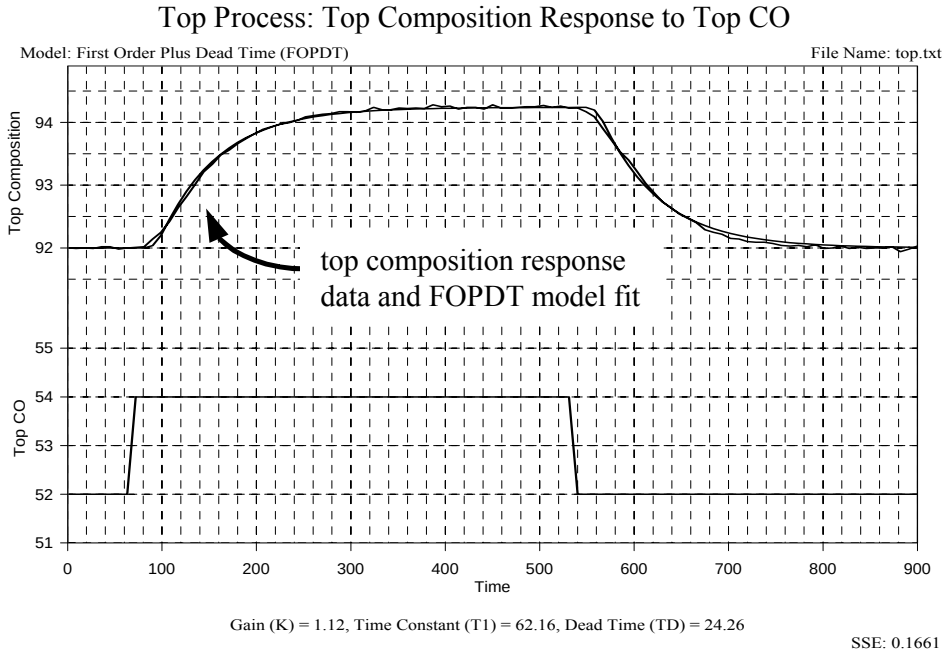


Figure 20.5 - Top composition response to top controller output pulse and FOPDT fit of process data

The FOPDT dynamic model provides a good approximation of the process data. Hence, $G_{\text{DT}}(s)$, the top composition response to top controller output model has the parameters:

Process Gain: $K_{P,\,\text{TT}} = 1.1$ %/%

Overall Time Constant: $\tau_{P,\,\text{TT}} = 62$ minutes

Apparent Dead Time: $\theta_{P,\,\text{TT}} = 24$ minutes

The FOPDT dynamic model parameters are used in the IMC correlation to obtain initial estimates for PI controller tuning. Recalling that the standard IMC correlation uses a closed loop time constant, τ_C, as the larger of $0.1\tau_P$ or $0.8\theta_P$, then *Design Tools* computes the PI tuning parameters:

Controller Gain, $K_{C,\,\text{top}} = 1.3$ %/%

Reset Time, $\tau_{I,\,\text{top}} = 62$ minutes

Returning to the distillation process, a PID controller is selected and these tuning values are entered. We turn derivative action off on the controller design menu, resulting in the desired PI controller form.

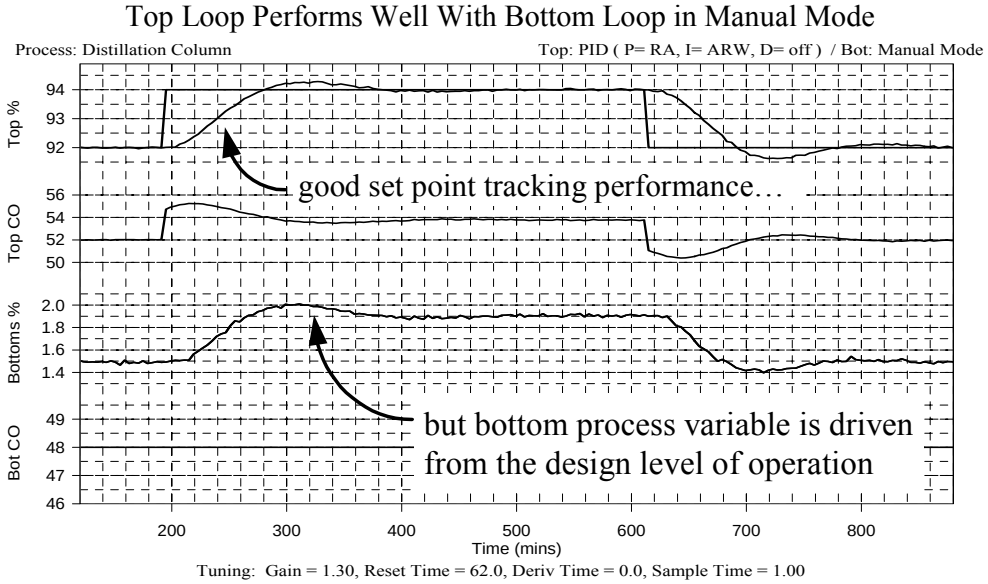


Figure 20.6 - Set point tracking capability of top loop under PI control when bottom loop is in manual mode

Controller performance is tested in tracking set point steps between 92% and 94% as shown in Fig 20.6. The bottom loop remains in manual mode during the evaluation. The top loop shows desirable set point tracking performance, with a rapid response, minimal overshoot and rapid decay. However, the corresponding movement in the bottom composition from the design level of operation shown in Fig. 20.6 highlights the need for a second controller on the bottom composition stream to properly operate the distillation column.

Bottom Control and Loop Interaction

The design of the bottom PI controller is analogous to that of the top controller. A pulse test is performed as shown in Fig. 20.7 by stepping the bottom controller output, $u_{\text{bottom}}(t)$, from its design value of 48% down to 46% and back. The top controller is in manual mode during this test. *Design Tools* is used to fit a FOPDT model to the data and the model fit is also shown in Fig. 20.7.

The FOPDT dynamic model provides a good approximation of the process data. Hence, $G_{BB}(s)$, the bottom composition response to bottom controller output model has the parameters:

Process Gain: $K_{P,\,BB} = -0.22$ %/%

Overall Time Constant: $\tau_{P,\,BB} = 53$ minutes

Apparent Dead Time: $\theta_{P,\,BB} = 14$ minutes

The FOPDT dynamic model parameters are used in the IMC correlation to obtain initial estimates for PI controller tuning. Using the standard IMC correlation, *Design Tools* computes the PI tuning parameters:

Controller Gain, $K_{C,\,bottom} = -9.7$ %/%

Reset Time, $\tau_{I,\,bottom} = 53$ minutes

Returning to the distillation process, a PID controller is selected and the above tuning values are entered. We turn derivative action off on the controller design menu, resulting in the desired PI controller form.

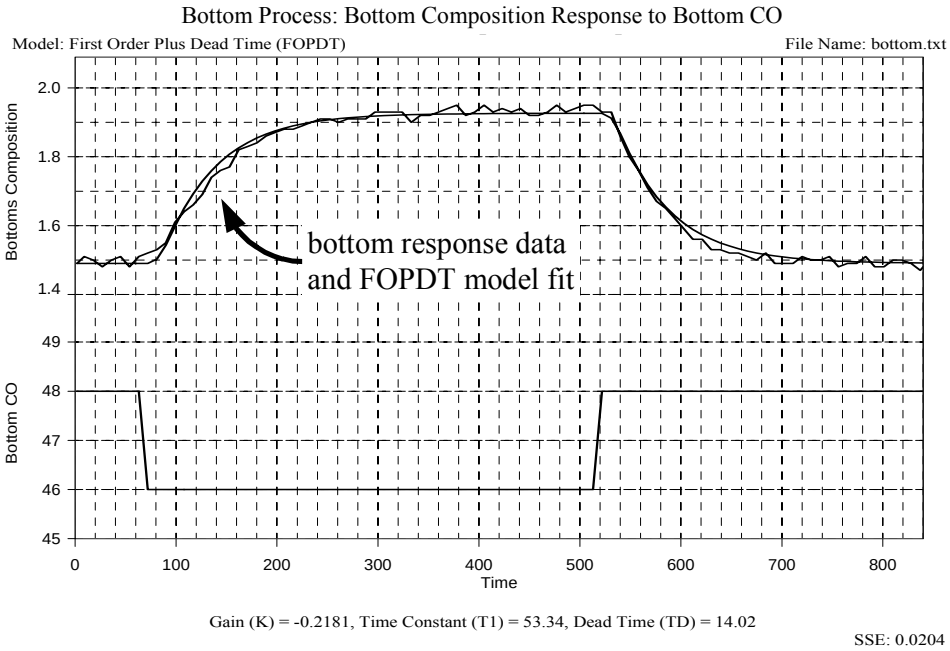


Figure 20.7 - Bottom composition response to bottom controller output pulse and FOPDT fit of process data

The performance of the bottom PI controller in tracking set point steps in bottom composition (not shown) while the top control loop is in manual mode reveals desirable performance. Thus, both the top and bottom control loops perform well when implemented individually while the cross loop controller of each is in manual mode.

Both the top and bottom PI controllers are put in automatic while leaving the controller tuning values unchanged. The same top composition set point step made in Fig 20.6 is repeated. Recall from that figure that when the bottom controller was in manual mode, the top controller was able to complete the set point step response in about 200 minutes. Fig. 20.8 shows that with both controllers in automatic, the set point step response of the top composition drags on for well over 1500 minutes.

The reason for this slow response is evident in Fig. 20.8. Notice how both $u_{top}(t)$ and $u_{bottom}(t)$ continually climb during the transient as each tries to compensate for the actions of the other. That is, as the top controller sends more and more cold reflux down the column to raise the purity of the top composition, the bottom controller responds by sending more and more steam up the column in an attempt to maintain the bottom composition at set point. One is working to cool while the other attempts to heat. This controller fight, or interaction, is a clear detriment to high-performance control.

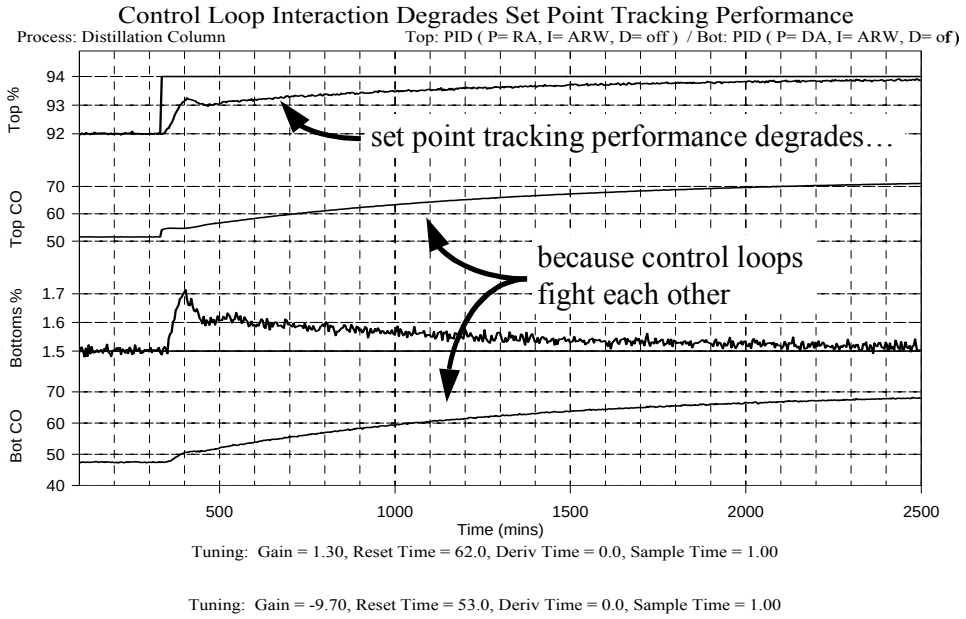


Figure 20.8 - Top and bottom loop fight each other, thus degrading set point tracking performance of top loop

20.5 Distillation Study - Decoupling the Loops

As discussed previously, decouplers are feed forward elements designed to reduce this loop interaction. Each decoupler design requires a process model and cross-loop disturbance model. For the process models, we can use the FOPDT model fits already developed during PI controller tuning and shown in Figs. 20.5 and 20.7. The task of developing cross-loop disturbance models remains.

No additional testing is needed to generate data for the cross-loop disturbance models because during the previous dynamic tests, both the top and bottom loops were in manual mode. Hence, these data sets already include information that describe how the top controller impacts the bottom composition and how the bottom controller impacts the top composition.

The top controller pulse test shown in Fig 20.5 not only forced a response in the top composition, but also impacted the bottom composition. We read that file into *Design Tools*, label the proper columns of data and fit a FOPDT model. The result is shown in Fig. 20.9. The FOPDT dynamic model provides a good approximation of this cross-loop disturbance data. Hence, $G_{BT}(s)$, the bottom composition response to top controller output model has the parameters:

Process Gain: $K_{D,\,BT} = 0.24$ %/%

Overall Time Constant: $\tau_{D,\,BT} = 54$ minutes

Apparent Dead Time: $\theta_{D,\,BT} = 22$ minutes

Similarly, the bottom controller pulse test shown in Fig. 20.7 not only forced a response in the bottom composition, but also impacted the top composition. That data is read into *Design Tools*, the proper columns of data are labeled and a FOPDT model is fit. The result is shown in Fig. 20.10.

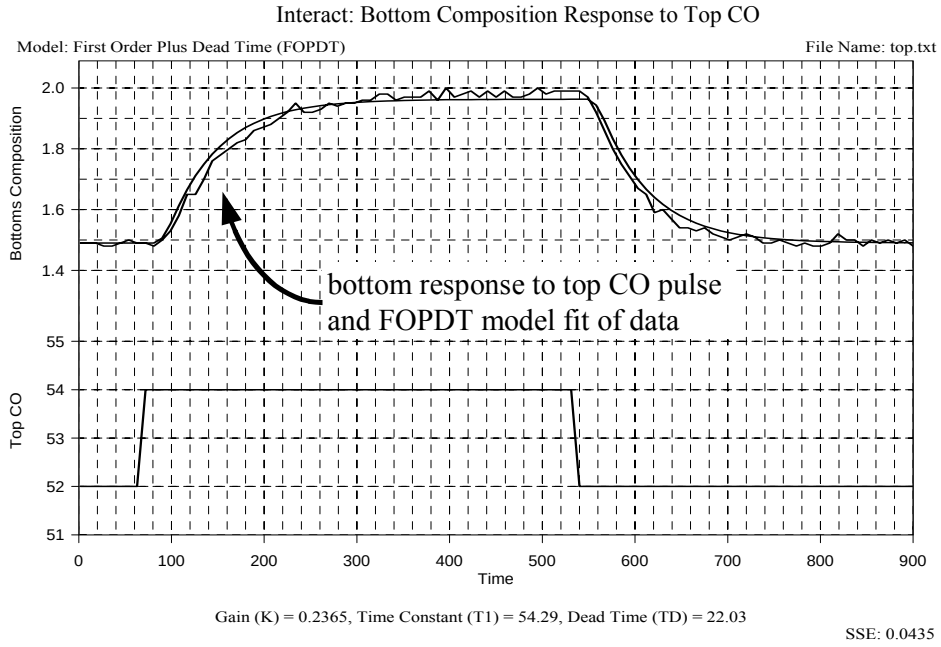


Figure 20.9 - Bottom composition response to top controller output pulse and FOPDT fit of process data

The FOPDT dynamic model provides a good approximation of this cross-loop disturbance data. Hence, $G_{BD}(s)$, the top composition response to bottom controller output model has the parameters:

Process Gain: $K_{D,\,TB} = -1.0$ %/%

Overall Time Constant: $\tau_{D,\,TB} = 63$ minutes

Apparent Dead Time: $\theta_{D,\,TB} = 21$ minutes

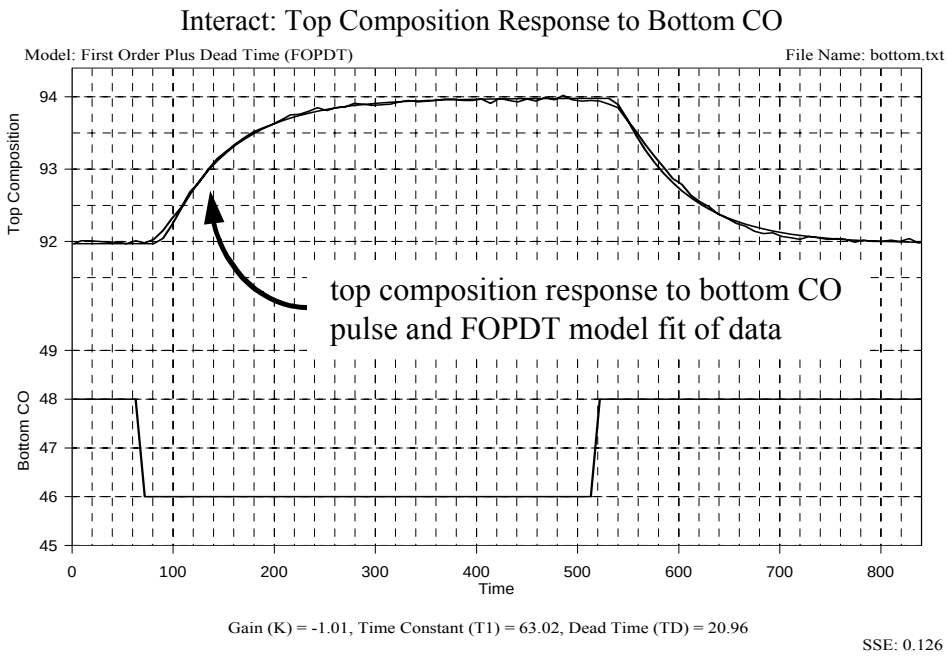


Figure 20.10 - Top composition response to bottom controller output pulse and FOPDT fit of process data

All process models and interaction models are now defined in FOPDT form. Returning to LOOP-PRO, the PID with Decoupler controllers are chosen for both top and bottom loops. The PI tuning parameters for the feed back loops remain as before. The FOPDT decoupler model values required by LOOP-PRO are entered.

For the top decoupler, the FOPDT process model entered is $G_{TT}(s)$ and the cross-loop disturbance model entered is $G_{TB}(s)$. For the bottom decoupler, the FOPDT process model entered is $G_{BB}(s)$ and the cross-loop disturbance model entered is $G_{BT}(s)$.

When the top loop is closed, an error message results telling us that the theory requires that the process dead time must be less than or equal to the disturbance dead time. This is a requirement of the theory and not a limitation of LOOP-PRO (for more information, please review the design criteria discussed in the feed forward chapter). Hence, we set $\theta_{P,\,TT} = \theta_{P,\,TB} = 21$ minutes to result in a mathematically rational design. That dead time change is underlined in Fig. 20.11 below.

Controller performance is again tested by stepping the top set point between 92% and 94% as shown in Fig. 20.11. The bottom loop remains closed during the evaluation. The decouplers succeed in restoring a set point tracking performance that is nearly equal to that of the lone controller as shown in Fig. 20.6. With two controllers and two decouplers, however, the bottom loop is now able to maintain the bottom composition close to set point throughout the event.

While we have achieved our goal of decoupling the control loop interactions, a somewhat disturbing result is the constant rapid movement in the controller output as shown in Fig. 20.11. This control signal behavior will cause the mechanical valves to "chatter" in almost a nervous fashion. It is a situation that likely will not be tolerated because valve wear and the consequent maintenance costs will become an issue in many plants.

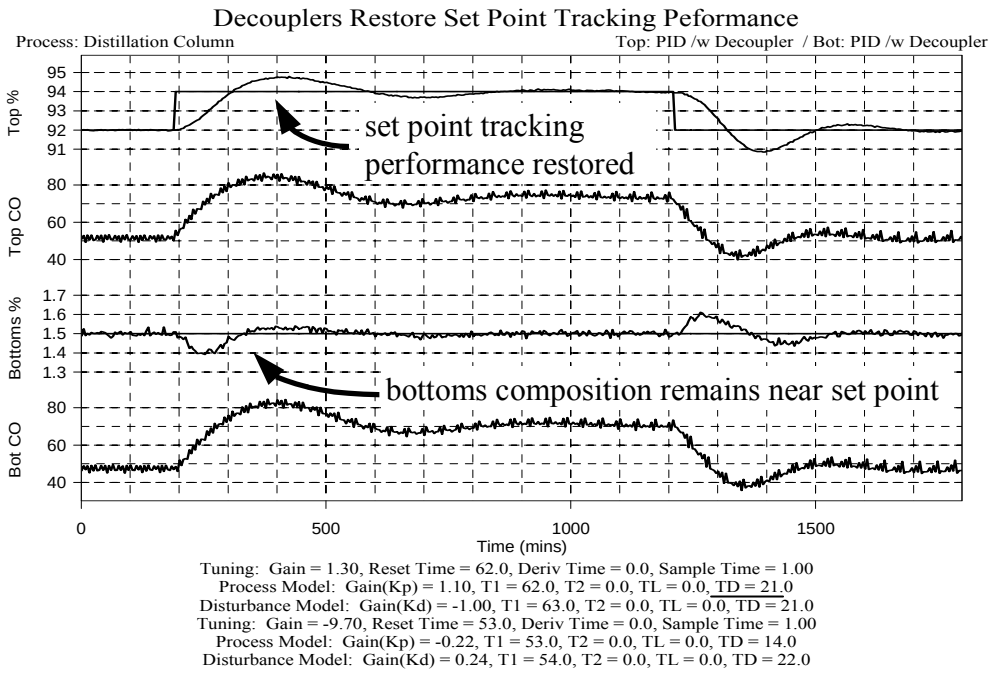


Figure 20.11 - Improved set point tracking capability of top loop and reduced interaction with bottom loop when both are under PI control with decouplers

The reason for this behavior is quite subtle and understanding why it occurs is important. For the models used in the bottom decoupler, the process model gain, $K_{P,\,BB}$, is –0.22 %/% and the cross loop disturbance gain, $K_{D,\,BT}$, is 0.24 %/%. In absolute value, $|K_{D,\,BT}| > |K_{P,\,BB}|$, which says that the top controller (the disturbance) has a larger influence on the bottom composition than does the bottom controller.

The relative size of the gains displayed by the process and used in the decouplers is fundamental to stable control. We will learn more in the next chapter, but it is reasonable to postulate that a decoupler must have at least as much influence on its own process variable as does any disturbance.

Following this assumption, we lower the cross-loop disturbance gain of the bottom loop decoupler so that in absolute value $|K_{D,\,BT}| = |K_{P,\,BB}|$ as underlined in Figure 20.12. Repeating the top set point step test as before, we observe in Fig. 13.12 that decoupling performance is maintained yet the controller output chatter is eliminated.

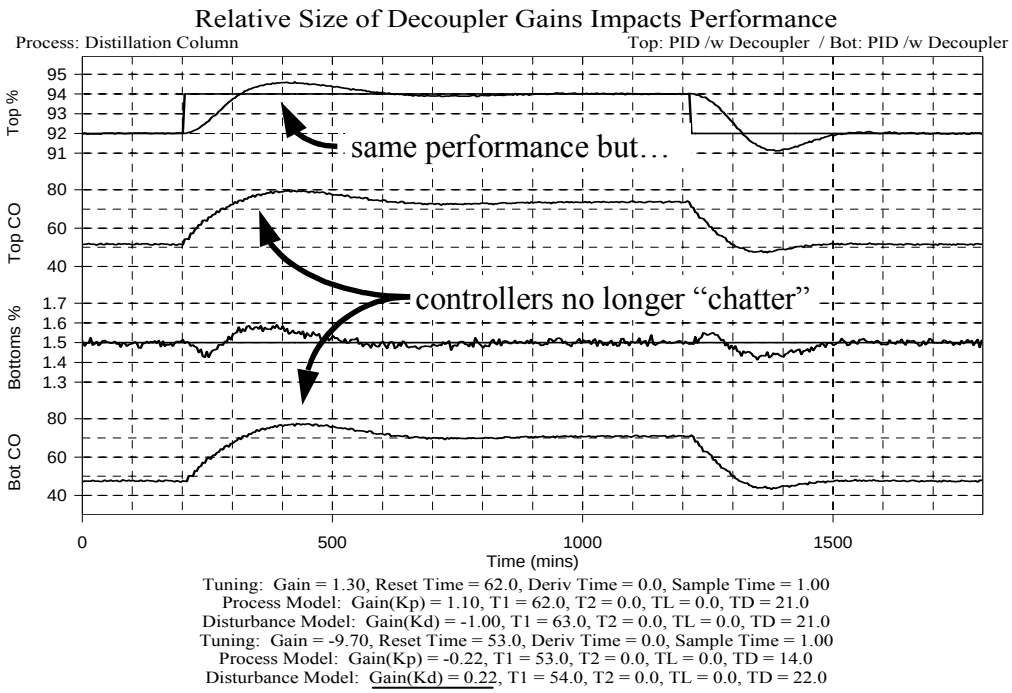


Figure 20.12 - Decoupled loops do not chatter with slight adjustment to one model parameter

21. Modeling, Analysis and Control of Multivariable Processes

21.1 Generalizing 2x2 Multivariable Processes

As we learned when studying the distillation column in the previous chapter, multivariable loop interaction presents a substantial control challenge. To explore in more detail the nature and challenges of multivariable control, we generalize all two-by-two multi-input multi-output (2x2 MIMO) dynamic processes into a standard form.

Following the nomenclature established in the popular texts, G_{ij} represents the dynamic behavior of the i^{th} measured process variable response to the j^{th} controller output signal. Hence, as shown in Fig. 21.1, process G_{11} describes the direct dynamic response of measured process variable PV_1 to changes in controller output CO_1, and interaction G_{21} describes the cross-loop dynamic response of PV_2 to changes in CO_1.

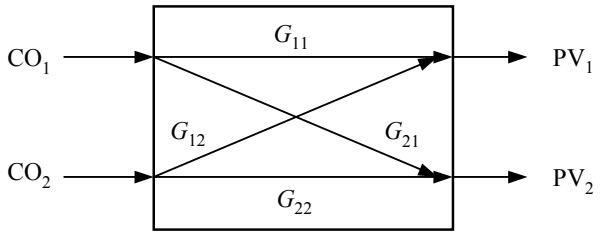


Figure 21.1 - General multivariable architecture

The multi-loop *Custom Process* in LOOP-PRO enables the simulation of a broad range of 2x2 MIMO processes that follow this general form. Figure 21.2 shows the multi-loop *Custom Process* graphic for building such simulations.

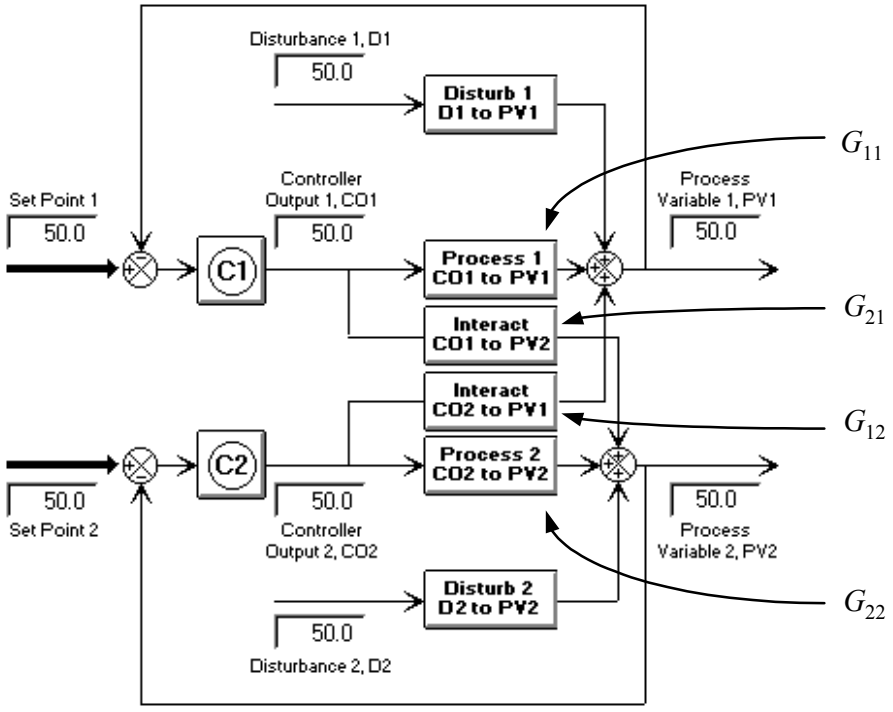


Figure 21.2 - LOOP-PRO*'s Multi-Loop Custom Process*

21.2 Relative Gain as a Measure of Loop Interaction

Before exploring different multivariable process behaviors, we introduce the concept of *relative gain*. Relative gain, λ, is popular because it:

- provides a convenient measure of loop interaction,
- is easy to compute, and
- is dimensionless so it is not affected by the units of the process data.

Relative gain is computed from the steady state gains of the process and cross-loop interaction models that best describe observed process behavior (that result from model fits of process data). Following the nomenclature we have established, relative gain is computed:

$$\lambda = \frac{K_{11}K_{22}}{K_{11}K_{22} - K_{12}K_{21}} \tag{21.1}$$

In the following sections we explore what the size and sign of λ implies for multivariable loop interaction and the ease with which a process can be controlled.

Before starting that study, consider that a 2x2 MIMO process has two controllers (CO_A and CO_B) that adjust two final control elements (F_A and F_B) to regulate two process variables (PV_A and PV_B). The controllers are connected to the process variables and final control elements by wires. Consequently, the control loops can be wired two ways:

1) CO_A manipulates F_A to control PV_A ; CO_B manipulates F_B to control PV_B

2) CO_A manipulates F_A to control PV_B ; CO_B manipulates F_B to control PV_A

Each combination yields a different value of λ for a multivariable process. What we will learn is that:

control loops should always be paired (wired) where possible so the relative gain is positive and as close as possible to one.

21.3 Effect of K_P on Control Loop Interaction

To appreciate the usefulness of relative gain as a measure of cross-loop interaction, consider the variety of 2x2 MIMO cases listed in Table 21.3. These cases are simulated and studied here using LOOP-PRO's *Custom Process* feature shown in Fig 21.2.

All of the direct process and interaction models used in the *Custom Process* simulations are first order plus dead time (FOPDT), and all time constant and dead time parameters for all the simulations in Table 21.3 are:

Process time constant, $\tau_P = 10$
Dead time, $\theta_P = 1$

Also, all of the investigations use two PI controllers with no decoupling and with:

Controller gain, $K_C = 5$
Reset time, $\tau_I = 10$

For all examples, when one PI controller is put in automatic while the other is in manual mode, that controller tracks set point changes quickly and with little oscillation. The issue we study here is process behavior when *both* PI controllers are put in automatic at the same time.

For each simulation case study, the direct process and cross-loop interaction gains are:

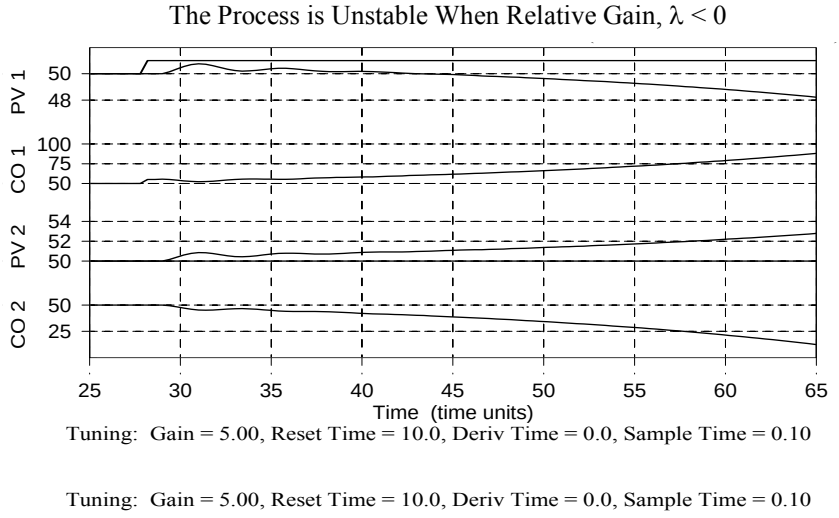
Case	direct $CO_1 \rightarrow PV_1$ K_{11}	cross-loop $CO_1 \rightarrow PV_2$ K_{21}	cross-loop $CO_2 \rightarrow PV_1$ K_{12}	direct $CO_2 \rightarrow PV_2$ K_{22}	λ
1	1	1.1	1.1	1	- 4.8
2a	0	1.1	0.5	1	0.0
2b	-1	3.0	1.1	1	0.2
2c	1	-3.0	0.5	1	0.4
3	1	-1.1	0.5	1	0.6
4	1	0	0.5	1	1.0
5	1	1.1	0.5	1	2.2
6	1	1.1	.85	1	15.4

Table 21.3 - Exploring relative gain, λ, as a measure of loop interaction

Case 1: $\lambda < 0$

When the cross-loop interaction gains are larger then the direct process gain, as is true for Case 1 in Table 21.3, then each controller has more influence on its cross-loop measured process variable than it does on its own direct measured process variable. As listed in the table, the relative gain, λ, computed by Eq. 21.1 for this case is negative.

Figure 21.4 shows the set point tracking performance of the Case 1 process when both loops are under PI control with no decoupling (remember that for all simulations, $\tau_P = 10$ and $\theta_P = 1$, with $K_C = 5$ and $\tau_I = 10$). As each controller works to keep its direct measured process variable on set point, every control action causes an even larger disruption in the cross-loop process variable. And the harder each controller works, the worse the situation. As shown in Fig. 21.4, the result is an unstable, diverging system.



Tuning: Gain = 5.00, Reset Time = 10.0, Deriv Time = 0.0, Sample Time = 0.10

Figure 21.4 - $\lambda < 0$ indicates incorrect loop pairing and an unstable process under PI control

A negative relative gain implies that the *loop pairing* is incorrect. That is, each controller is wired to the wrong measured process variable. The best course of action is to switch the controller wiring. This switches the cross-loop gains in Table 4.3 to direct process gains and vice versa.

Switching the loop pairing recasts Case 1 into a process with a relative gain of $\lambda = 5.8$. This loop interaction behavior is somewhere between Case 5 and Case 6. As we will learn, a process with this relative gain is challenging to control, but it is closed-loop stable and the loops can be decoupled following standard methodologies.

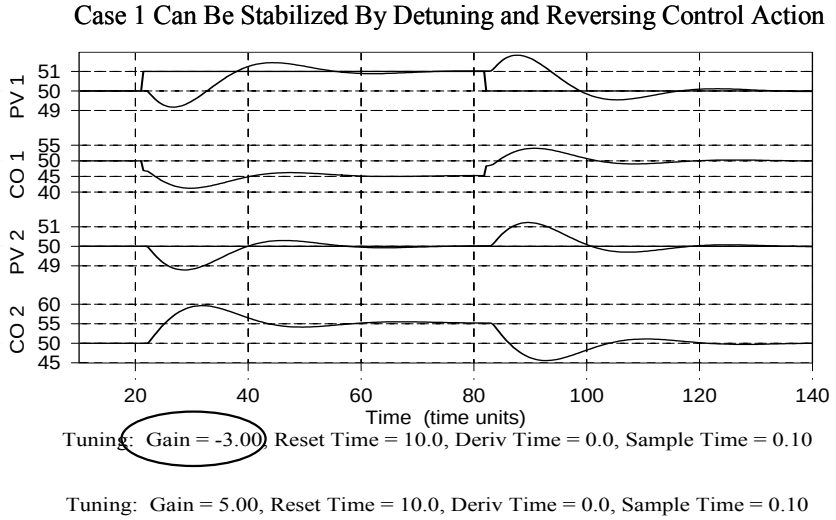


Figure 21.5 - Case 1 can be stabilized by detuning one of the loops and reversing its action

As an alternative to switching the pairing, one of the controllers can be detuned (the K_C can be lowered) and the control action on that controller reversed (e.g. if it is direct acting make it reverse acting). This tricks the control system into making reasonable corrective actions. Figure 21.5 shows that the Case 1 process can be stabilized by detuning and reversing the action of loop 1 as indicated by the K_C circled in the figure. The resulting set point tracking performance is rather unsatisfactory, however, and we do not recommend this approach as a general solution.

Case 2: $0 < \lambda \leq 0.5$

For the relative gain to be *exactly zero* ($\lambda = 0$), one of the direct process gains must be zero. A direct process gain of zero means that a controller has no impact on the measured process variable it is wired to. Clearly there can be no regulation if a controller has no influence.

Case 2a in Table 21.4 has $K_{11} = 0$, implying that CO_1 has no influence on PV_1. Yet because the cross-loop gain K_{12} is not zero, changes in CO_2 will disrupt PV_1. If a measured process variable can be disrupted but there is no means to control it, the result is an unstable process under PI control (no figure shown). Because both cross-loop gains are not zero in Case 2a, the loop pairing should be switched in this case to give each controller direct influence over a measured process variable. This would recast Case 2a into a process with a $\lambda = 1.0$, which is the interaction measure most desired. We study such a process in Case 4 below.

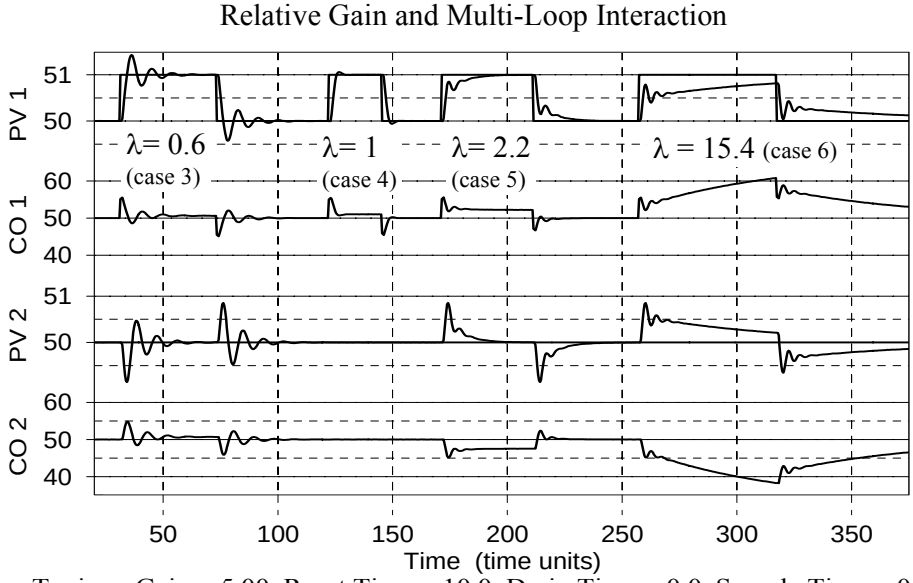
When the relative gain is *near zero* ($0 < \lambda \leq 0.5$), then at least one of the cross-loop gains is large on an absolute basis (e.g. Case 2b and 2c). Under PI control with no decoupling and using the base tuning values of $K_C = 5$ and $\tau_I = 10$, both of these processes are unstable and show considerable loop interaction (no figure shown). Detuning both controllers to $K_C = 2$ and $\tau_I = 10$ restores stability but control-loop interaction is still significant.

Again, the best course of action is to switch loop pairing. With the wiring switched, Case 2b yields $\lambda = 0.8$ and Case 2c yields $\lambda = 0.6$, putting both relative gains closer to the desired value of one. While both processes still display loop interaction, the processes become stable under PI control with no decoupling, even with fairly aggressive PI controller tuning.

Case 3: $0.5 \leq \lambda < 1$

When the relative gain is between 0.5 and one, the cross-loop interactions cause each control action to be reflected and amplified in both process variables. As shown in the left most set point steps of Fig. 21.6 for a case where $\lambda = 0.6$, this interaction leads to a measured process variable response that includes significant overshoot and slowly damping oscillations.

This amplifying interaction exists when stepping the set point of either loop. It grows more extreme and ultimately leads to an unstable process as λ approaches zero (see Case 2). The interaction becomes less pronounced as λ approaches one (see Case 4).



Tuning: Gain = 5.00, Reset Time = 10.0, Deriv Time = 0.0, Sample Time = 0.10

Figure 21.6 - Impact of λ on PI control loop interaction with no decoupling

Case 4: $\lambda = 1$

A relative gain of one occurs when either or both of the cross-loop gains are zero. In Case 4, K_{21} is zero so controller output CO_1 has no impact on the cross-loop measured process variable PV_2. However, since K_{12} is not zero as listed in Table 21.4, changes in CO_2 will impact PV_1.

The second set point steps in Fig. 21.6 show the control performance of the Case 4 process when the set point of PV_1 is changed. As expected, the set point tracking actions of CO_1 have no impact on PV_2. While not shown, a set point step in PV_2 would cause a some cross-loop disruption in PV_1 because of loop interaction.

When both cross-loop gains are zero, the loops do not interact. Such a system is naturally and completely decoupled and the controllers should be designed and tuned as single loop processes.

Case 5: $\lambda > 1$

Opposite to the observations of Case 3, when the relative gain is greater than one, the control loops fight each other. Specifically, the cross-loop interactions act to restrain movement in the measured process variables, prolonging the set point response. The third set point steps in Fig. 21.6 illustrate this behavior for a case where $\lambda = 2.2$. We also observed this prolonged interaction in the distillation case study in the previous chapter.

As stated earlier, a process with a relative gain that is positive and close to one displays the smallest loop interactions (is better behaved). For Case 5, switching the loop pairing would yield a very undesirable negative λ. This means that the loops are correctly paired and the significant loop interaction is unavoidable.

Case 6: $\lambda >> 1$

As the cross-loop gain product, $K_{12}K_{21}$, approaches the direct process gain product, $K_{11}K_{22}$, the relative gain grows and the restraining effect on movement in the measured process variables discussed in Case 5 becomes greater. This is illustrated in the right most set point steps in Fig. 21.6 for a case where $\lambda = 15.4$. Again, switching the loop pairing would yield a negative λ so the loops are correctly paired and the significant loop interaction is unavoidable. Interestingly, as the cross-loop gains grow to the point that their product is larger then the direct process gain product (when $K_{12}K_{21} > K_{11}K_{22}$), then λ becomes negative and we circle back to Case 1.

21.4 Effect of τ_P on Control Loop Interaction

The relative gain provides a convenient means for predicting how the size and sign of the direct process and cross-loop gains will impact loop interaction on a 2x2 process. Here we learn that while τ_P is fundamental to loop interaction dynamics, it rarely plays as important a role as the gains. There is no simple equation analogous to relative gain we can use for this study, so we learn by example.

To explore the role of cross-loop time constants on loop interaction, we define a set of 2x2 processes where for all cases:

Steady State Gains: $K_{11} = 1.0$, $K_{21} = 0.5$, $K_{12} = 0.5$, $K_{22} = 1.0$
Dead Times: $\theta_{11} = 1.0$, $\theta_{21} = 1.0$, $\theta_{12} = 1.0$, $\theta_{22} = 1.0$

All of the investigations use two PI controllers with no decoupling and with:

Controller gain, $K_C = 5$
Reset time, $\tau_I = 10$

Based on Eq. 21.1, all three cases have a relative gain $\lambda = 1.3$, which describes a process with some loop interactions, but as explained in Case 5, is generally well-behaved.

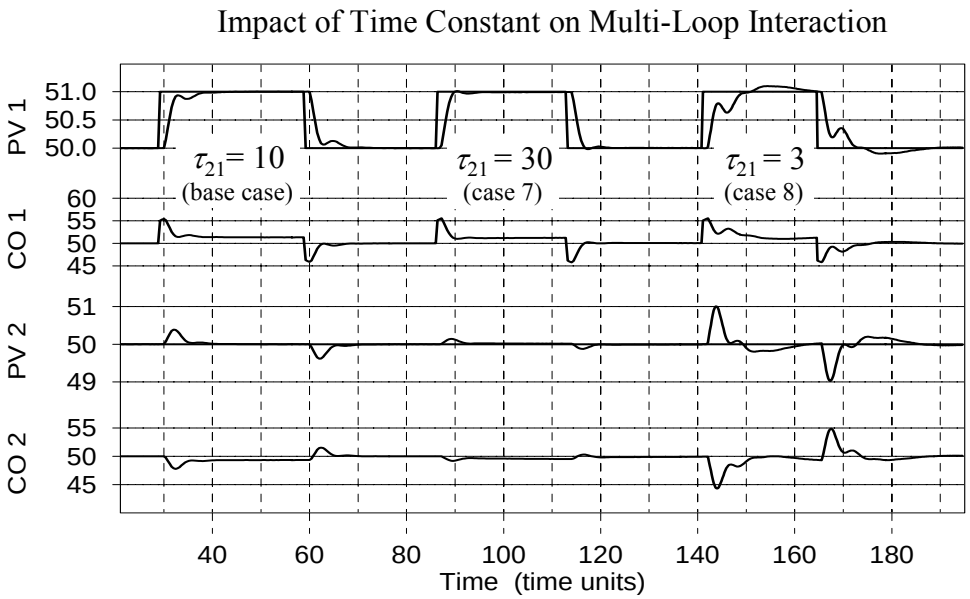
Case	direct $CO_1 \rightarrow PV_1$ τ_{11}	cross-loop $CO_1 \rightarrow PV_2$ τ_{21}	cross-loop $CO_2 \rightarrow PV_1$ τ_{12}	direct $CO_2 \rightarrow PV_2$ τ_{22}
base	10	10	10	10
7	10	30	10	10
8	10	3	10	10

Table 21.7 - Exploring the impact of cross-loop time constants on loop interaction

With the gains and dead times as listed above, we investigate a base case and two variations that focus on time constants. In particular, the direct process and cross-loop interaction time constants explored here are listed in Table 21.7.

Base Case

The base case in Table 21.7 establishes the dynamic behavior when the direct process and cross-loop time constants have equal influence. As shown in the left most set point steps of Fig. 21.8, the loop interactions for the base case are modest and display some restraint in the movement of the measured process variables. This prolongs the set point response as discussed in Case 5 and shown to the right in Fig. 21.6 above.



Tuning: Gain = 5.00, Reset Time = 10.0, Deriv Time = 0.0, Sample Time = 0.10

Figure 21.8 - Impact of τ_P on PI control loop interaction with no decoupling

Case 7: Cross-Loop τ_P Large

This process, with $\tau_{21} = 30$ while $\tau_{11} = \tau_{22} = \tau_{12} = 10$, considers a case where one of the cross-loop time constants is large relative to the direct process time constants. A short time constant implies a fast reaction time. Hence, CO_1 and CO_2 can correct for errors in their direct process variables quickly, while the disruptive influence of CO_1 on cross-loop process variable PV_2 proceeds slowly in comparison.

Because each controller can quickly impact its own direct process variable, each is capable of correcting for the disruptive effects of the slow moving cross-loop interaction as it arrives. And consequently, as shown in the second set point steps of Fig. 21.8, performance is improved relative to the base case when using two PI controllers with no decoupling and with $K_C = 5$ and $\tau_I = 10$. Specifically, the set point tracking performance of PV_1 in case 7 shows less oscillation as it moves to the new set point, and the size and speed of disruption of CO_1 on PV_2 is lessened in comparison to the base case.

Case 8: Cross-Loop τ_P Small

Opposite to Case 7, with $\tau_{21} = 5$ while $\tau_{11} = \tau_{22} = \tau_{12} = 10$, Case 8 in Table 21.7 considers a process where one of the cross-loop time constants is small relative to the direct process time constants. Thus, CO_1 is able to disrupt cross-loop process variable PV_2 faster than CO_2 is able to correct the problem.

Because the cross-loop disruption is fast relative to the controller's ability to regulate its own direct process variable, the third set point steps of Fig. 21.8 show that control performance degrades relative to the base case. As shown in the figure, the set point tracking performance of PV_1 in case 8 shows greater oscillations and a larger overshoot as it moves to the new set point. Also shown is that the disruption of CO_1 on PV_2 is significantly greater here than the base case.

21.5 Effect of θ_P on Control Loop Interaction

Similar to the time constant, dead time plays a secondary role in loop interaction dynamics compared to the relative gains. We continue to learn by investigation with a set of 2x2 processes where for all cases:

Steady State Gains: $K_{11} = 1.0$, $K_{21} = 0.5$, $K_{12} = 0.5$, $K_{22} = 1.0$
Time Constants: $\tau_{11} = 10$, $\tau_{21} = 10$, $\tau_{12} = 10$, $\tau_{22} = 10$

controlled by two PI controllers with no decoupling and with:

Controller gain, $K_C = 5$
Reset time, $\tau_I = 10$

Based on Eq. 21.1, all three cases have a relative gain $\lambda = 1.3$, which describes a process with some loop interactions, but as explained in Case 5, is generally well-behaved. We again explore a base case and two variations where:

Base Case

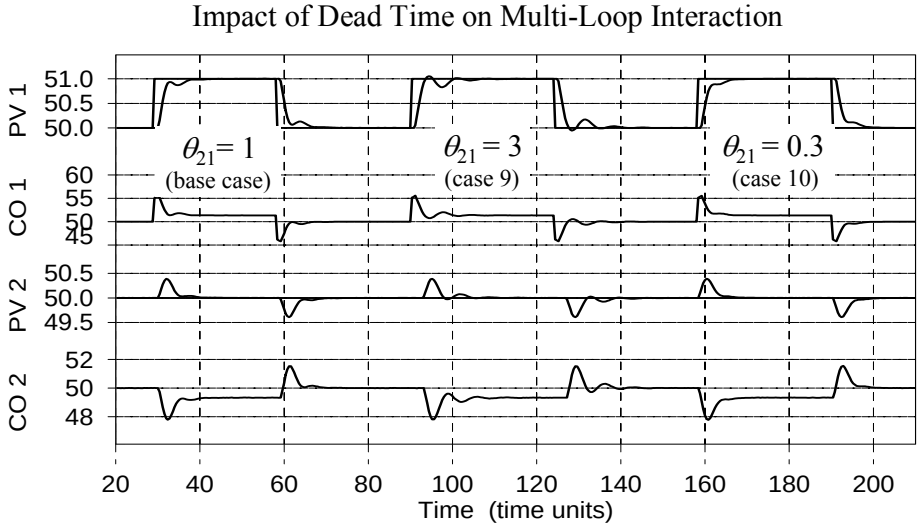
The base case in Table 21.9 establishes the dynamic behavior when the direct process and cross-loop dead times are equal. As shown in the left most set point steps of Fig. 21.10, the loop interactions for the base case are identical to those shown in the base case of Fig. 21.8 above.

Case	direct $CO_1 \rightarrow PV_1$ θ_{11}	cross-loop $CO_1 \rightarrow PV_2$ θ_{21}	cross-loop $CO_2 \rightarrow PV_1$ θ_{12}	direct $CO_2 \rightarrow PV_2$ θ_{22}
base	1	1	1	1
9	1	3	1	1
10	1	0.3	1	1

Table 21.9 - Exploring the impact of cross-loop dead time on loop interaction

Case 9: Cross-Loop θ_P Large

This case, with $\theta_{21} = 3$ while $\theta_{11} = \theta_{22} = \theta_{12} = 1$, considers a process where one of the cross-loop dead times is large relative to the direct process dead times. Hence, there is a short delay before a change in CO_1 impacts PV_1 and a long delay before it begins disrupting cross-loop variable PV_2.



Tuning: Gain = 5.00, Reset Time = 10.0, Deriv Time = 0.0, Sample Time = 0.10

Figure 21.10 - Impact of θ_P on PI control loop interaction with no decoupling

When the second set point steps occur in the middle of Fig. 21.10, CO_1 begins moving PV_1 to track the change. As a consequence, CO_1 also imparts a disruptive event to PV_2. Because of the cross-loop dead time, there is a relatively long delay before the disruption finally impacts PV_2.

When the disruption to PV_2 does begin, CO_2 takes action to reject the disturbance. And as CO_2 changes, the cross-loop interaction causes the control actions to be reflected back to PV_1. The extended delay in one of the steps of this back and forth interaction produces an "echo effect" in the set point response of PV_1. This can be seen in Fig. 21.10 as a more pronounced second dip in the response compared to that evident in the base case.

Case 10: Cross-Loop θ_P Small

Opposite to case 9, with $\theta_{21} = 0.3$ while $\theta_{11} = \theta_{22} = \theta_{12} = 1$, we consider a case where one of the cross-loop dead times is small relative to the direct process dead times. Hence, there is a relatively short delay before a change in CO_1 imparts a cross-loop disruption to PV_2.

When the disruption to PV_2 begins, CO_2 takes action to reject the disturbance and these actions are reflected back to PV_1. As shown in the right most right most set point steps of Fig. 21.10, the shorter delay has little impact on loop interaction. The plot reveals that controller performance is very similar to that of the base case.

21.6 Decoupling Cross-Loop K_P Effects

Recall that a decoupler is a feed forward element where the measured disturbance is the actions of a cross-loop controller. Analogous to a feed forward controller, a decoupler is comprised of a process model and a cross-loop disturbance model. The cross-loop disturbance model receives the cross-loop controller signal and predicts an "impact profile," or when and by how much the process variable will be impacted. Given this predicted sequence of disruption, the process model then back calculates a series of control actions that will exactly counteract the cross-loop disturbance as it arrives so the measured process variable, in theory, remains constant at set point.

Here we explore how *perfect decouplers* can reduce cross-loop interaction. A perfect decoupler employs the identical models in the decoupler as is used for the process simulation. Be aware that in real-world applications, no decoupler model exactly represents the true process behavior. Hence, the decoupling capabilities shown here must be considered as the best possible performance. We revisit the "no decoupler" cases explored earlier in this chapter as summarized in Table 21.3.

Case 1: $\lambda < 0$

A negative relative gain implies that the loop pairing is incorrect. Decoupling is not explored for Case 1 in Table 21.3 because the best course of action is to switch the controller wiring to produce a 2x2 process with a relative gain of $\lambda = 5.8$. This loop interaction behavior is somewhere between Case 5 and Case 6 discussed below.

Case 2: $0 < \lambda \leq 0.5$

A relative gain of *exactly zero* ($\lambda = 0$) implies that at least one controller has no impact on the measured process variable it is wired to. There can be no regulation if a controller has no influence. Hence, decoupling becomes meaningless for this case.

When the relative gain is *near zero* ($0 < \lambda \leq 0.5$), PI controllers with no decoupling must be detuned to stabilize the 2x2 system. Even when the PI controllers are detuned, perfect decouplers (the identical models are used in the decouplers as are used for the process simulation) result in an unstable system (no figure shown). Detuning the decouplers (lowering the disturbance model gain) will restore stability but interaction remains significant and general performance is poor. Again, the best course of action is to switch loop pairing.

Case 3: $0.5 \leq \lambda < 1$

When the relative gain is between 0.5 and one, the cross-loop interactions cause each control action to be reflected and amplified in both process variables. As shown in the left most set point steps of Fig. 21.11 for the case of $\lambda = 0.6$, PI controllers with perfect decouplers virtually eliminate cross loop interactions. This is not surprising since the relative gain is positive and close to one.

Case 4: $\lambda = 1$

A relative gain of one occurs when either or both of the cross-loop gains are zero. In Case 4 of Table 21.3, K_{21} is zero so controller output CO_1 has no impact on the cross-loop measured process variable PV_2. Consequently, a perfect decoupler will provide no benefit for this loop. And as shown in Fig. 21.11 for the middle set point steps, while a perfect decoupler causes no harm, a decoupler implemented on a real process will likely have imperfect models and would then actually create loop interaction.

Table 21.3 shows that K_{12} is not zero, however, so changes in CO_2 will impact PV_1. A perfect decoupler will virtually eliminate cross loop interaction for information flow in this direction (no figure shown). Thus, the Case 4 MIMO process can address the 2x2 loop interaction with a single decoupler on the CO_2 to PV_1 loop.

Case 5: $\lambda > 1$

When the relative gain is greater than one, the cross-loop interactions act to restrain movement in the measured process variables. The third set point steps in Fig. 21.11 for the case where $\lambda = 2.2$ illustrate that perfect decouplers substantially eliminate both this restraining effect and the level of loop interaction. Again, this is not surprising since the relative gain is positive and reasonably close to one.

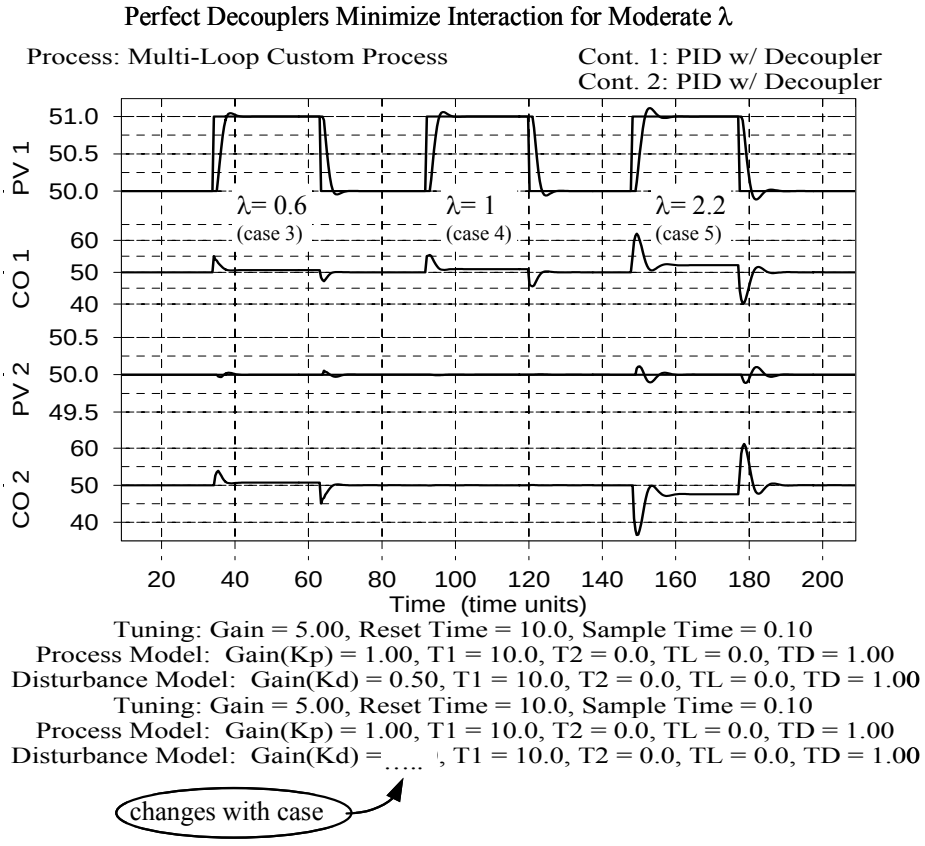


Figure 21.11 - Perfect decouplers can virtually eliminate cross-loop interaction when λ is near 1.

Case 6: $\lambda >> 1$

As relative gain grows larger, the restraining effect on movement in the measured process variables due to loop interaction becomes greater. Case 6 in Table 21.3 is interesting because K_{21} is greater than both K_{11} and K_{22}. That is, controller output CO_1 has a greater influence on cross-loop process variable PV_2 then it does on its own direct process variable PV_1. Further, the influence of this disturbance on PV_2 is large compared to the actions CO_2 must take to reject it. Switching loop pairing offers no benefit as this makes the relative gain negative.

With perfect decouplers as shown in the *right* set point steps of Fig. 21.12 (the decoupler employs the identical models as are used for the process simulation), the system is unstable. This cannot be addressed by detuning the PI controller. Even with lower values for controller gain, K_C, the system is unstable. As we postulate at the end of the distillation case study in the previous chapter, a decoupler must have as much influence on its own process variable as does any disturbance.

Hence, we detune the decoupler by lowering the cross-loop disturbance gain of the bottom loop so that in absolute value, $K_{21} \le K_{22}$ and $K_{21} \le K_{11}$. Repeating the set point steps in the *left* set point steps of Fig. 21.12 reveal a stable and reasonably decoupled system.

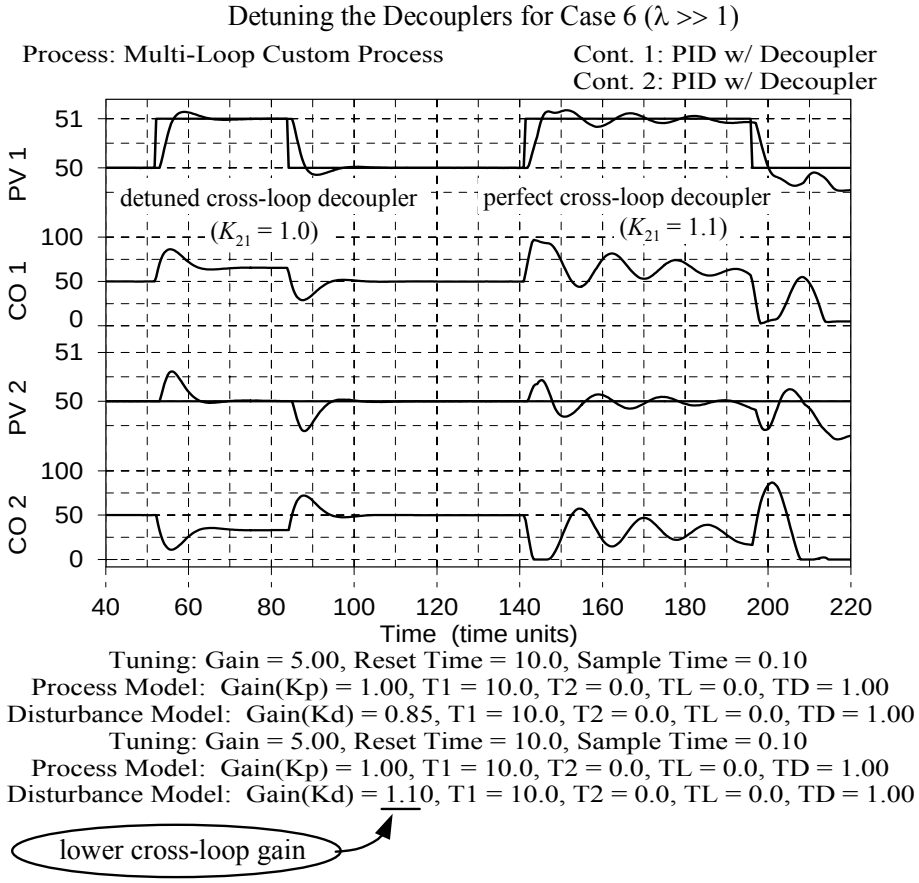


Figure 21.12 - Perfect decouplers must be detuned to produce a stable system for large λ

21.7 Decoupling Cross-Loop τ_P Effects

We continue exploring how perfect decouplers can reduce cross-loop interaction and now focus on perfect decoupling of time constant effects. We revisit the "no decoupler" cases explored earlier in this chapter as summarized in Table 21.7. Recall that all three cases have a relative gain $\lambda = 1.3$, which describes a process with some loop interactions but is generally well-behaved.

Base Case

The base case in Table 21.7 is a system where the direct process and cross-loop time constants have equal influence. As shown in the left most set point steps of Fig. 21.13, perfect decouplers virtually eliminate cross-loop interaction for this process.

Case 7: Cross-Loop τ_P Large

Case 7 in Table 21.7, with $\tau_{21} = 30$ while $\tau_{11} = \tau_{22} = \tau_{12} = 10$, considers a case where one of the cross-loop time constants is large relative to the direct process time constants. Here, the disruptive influence of CO_1 on cross-loop process variable PV_2 proceeds relatively slowly. This enables perfect decouplers, as shown in the middle set point steps of Fig. 21.13, to effectively eliminate cross-loop interaction for this process.

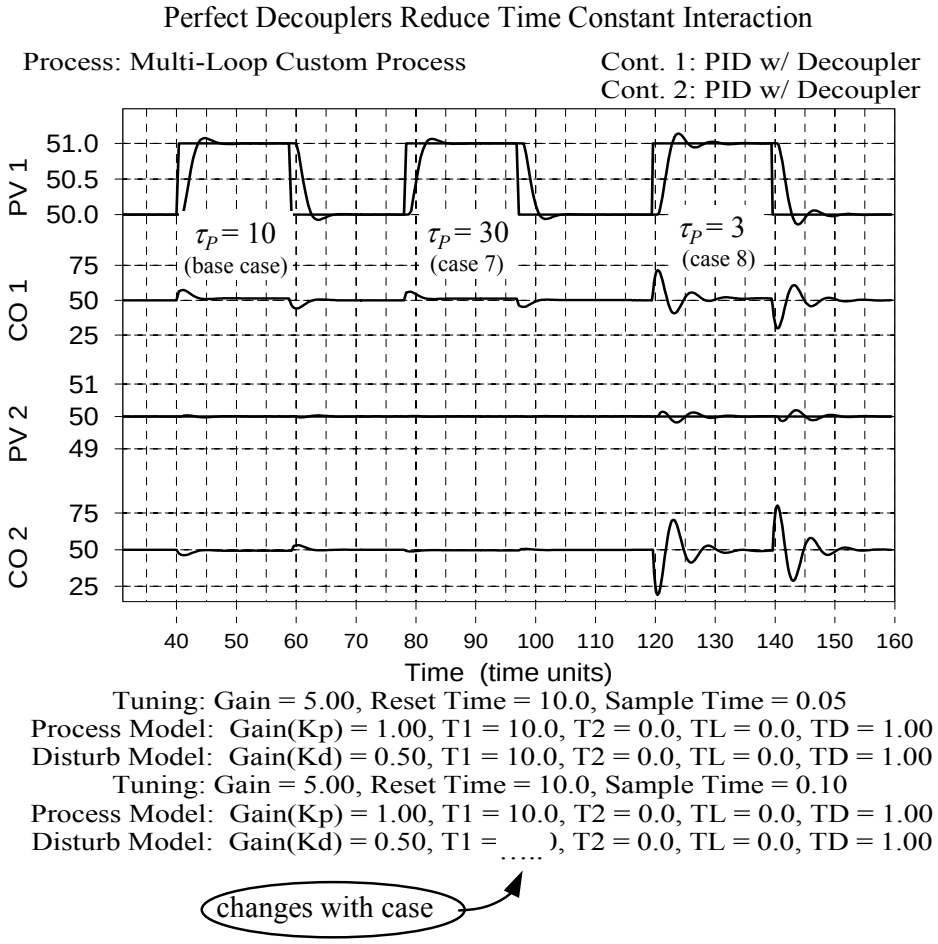


Figure 21.13 - Decoupling loop interaction due to time constant effects

Case 8: Cross-Loop τ_P Small

With $\tau_{21} = 5$ while $\tau_{11} = \tau_{22} = \tau_{12} = 10$, Case 8 in Table 21.7 considers a process where CO_1 is able to disrupt cross-loop process variable PV_2 faster than CO_2 is able to correct the problem. In spite of this challenge, perfect decouplers are able to dramatically reduce the loop interaction as shown in the right most set point steps of Fig. 21.13. We learn from this study that small cross-loop time constants present the greatest decoupling challenge.

21.8 Decoupling Cross-Loop θ_P Effects

We finish by exploring how perfect decouplers can reduce cross-loop interaction and decouple dead time effects. We revisit the "no decoupler" cases explored earlier in this chapter as summarized in Table 21.9. Recall that all three cases have a relative gain $\lambda = 1.3$, which describes a process with some loop interactions but is generally well-behaved.

Base Case

The base case in Table 21.9 is identical to the base case of Table 21.7. That is, the dead times all have equal influence. As shown in the left most set point steps of Fig. 21.14, perfect decouplers virtually eliminate cross-loop interaction for this process.

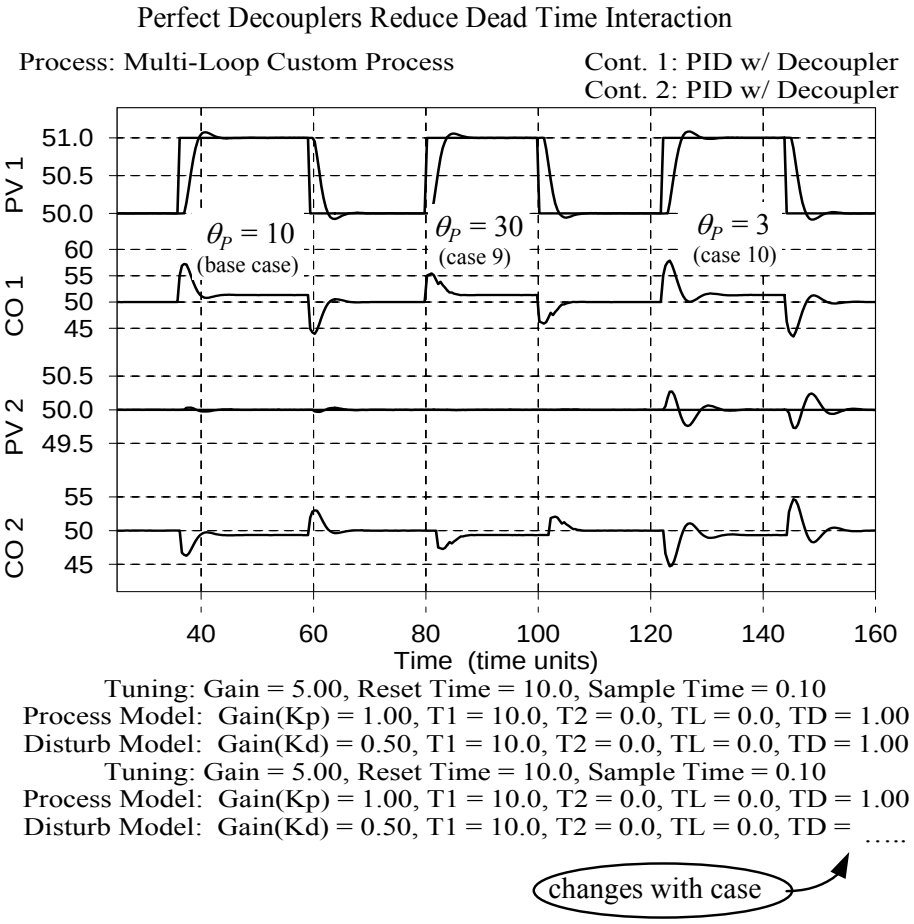


Figure 21.14 - Decoupling loop interaction due to dead time effects

Case 9: Cross-Loop θ_P Large

Case 9 in Table 21.9, with $\theta_{21} = 3$ while $\theta_{11} = \theta_{22} = \theta_{12} = 1$, considers a process where there is a short delay before a change in CO_1 impacts PV_1 and a long delay before it begins disrupting cross-loop variable PV_2. This additional time enables perfect decouplers, as shown in the middle set point steps of Fig. 21.14, to eliminate cross-loop interaction for this process.

Case 10: Cross-Loop θ_P Small

With $\theta_{21} = 0.3$ while $\theta_{11} = \theta_{22} = \theta_{12} = 1$, Case 10 considers a process where one of the cross-loop dead times is small relative to the direct process dead times. We cannot implement perfect decouplers here because the theory requires that we enter a process dead time in the decoupler model that is less than or equal to the disturbance dead time. This is not a limitation of LOOP-PRO. To do otherwise would be seeking to compute a corrective control action *before* the disturbance change first occurs.

Suppose for a particular application, you determine that the disturbance dead time is indeed shorter than the process dead time. Best practice is to arbitrarily set the process dead time equal to the disturbance dead time when entering values into the decoupler input form. An approach that yields identical performance is to set the disturbance dead time equal to the process dead time (the approach we take here). Decoupler performance will suffer but at least the calculations will yield control actions that make physical sense.

Thus, even though the process simulation has the cross loop dead time of $\theta_{21} = 0.3$, we enter $\theta_{21} = \theta_{22} = 1.0$ in the decoupler input form. The right most set point steps of Fig. 21.14 show that while loop interaction is reduced, the comprise we were forced to make with the decoupler model parameters prevents complete decoupling. We learn from this study that small cross-loop dead times present the greatest decoupling challenge.

22. Model Based Smith Predictor For Processes with Large Dead Time

22.1 A Large Dead Time Impacts Controller Performance

Dead time is considered large only in comparison to the time constant of a process. As dead time grows large relative to the time constant ($\theta_P \geq \tau_P$), it becomes increasingly difficult to achieve a tight set point tracking performance with traditional PID control. One example of this performance loss is that as dead time increases, the rise time and settling time must lengthen to maintain a desired peak overshoot ratio.

Suppose a process has a dead time equal to the time constant ($\theta_P = \tau_P$) and the controller sample time is ten times per time constant ($T = 0.1\,\tau_P$). For such a process, 10 full samples (one dead time) must pass after a control action before the sensor detects any impact. And every subsequent control action encounters this tremendous delay. The controller tuning must be sluggish enough to reflect this dead time. Otherwise, too much corrective action can accumulate in the dead time "pipeline," leading to large oscillations and even a dangerous unstable operation.

To visualize this idea, consider the tank temperature control process shown in Fig. 22.1. A hot and cold liquid stream combine at the entrance to a pipe, travel its length, and spill into a tank. The control objective is to maintain temperature in the tank by adjusting the flow rate of hot liquid entering the pipe (this case study is not available in LOOP-PRO but is illustrative for this discussion).

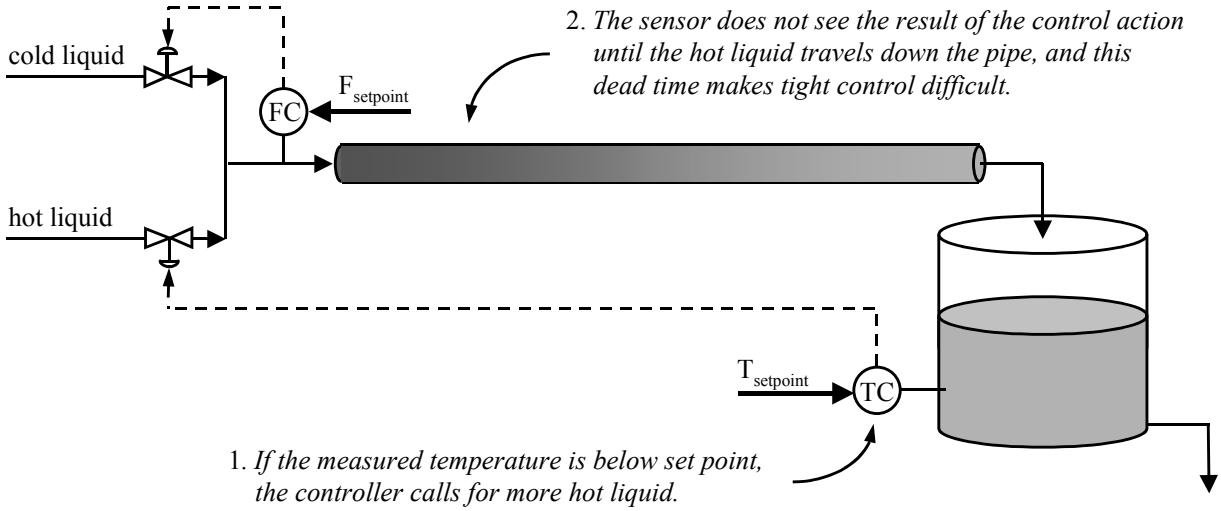


Figure 22.1 – The large dead time from the pipe makes tight temperature control difficult

If tank temperature is below set point (too cold), the controller opens the hot liquid valve to compensate and the temperature of the liquid entering the pipe increases. The temperature sensor in the tank does not immediately see the result of this action, however, because the additional hot liquid is still a pipe length away.

If the controller is tuned for a rapid response (a large K_C and small τ_I), it will open the hot liquid valve more and more in an attempt to raise tank temperature. The pipe fills with ever hotter liquid until the first hot liquid makes its way down the pipe and finally spills into the tank.

When tank temperature eventually rises to set point, the controller steadies the hot liquid flow rate entering the pipe to stop any additional temperature rise. But the pipe is now filled with hot liquid and it continues spilling into the tank. As tank temperature rises yet further, the controller begins closing the valve to cool the tank. Again, because of the delayed response, the controller will fill the pipe with too-cold liquid before the actions are finally revealed in the tank.

The dead time from the pipe combined with an overly aggressive controller causes the tank temperature to cycle between too-hot and too-cold. One solution is to *detune* the controller. That is, decrease the controller gain and/or increase the reset time to make the action more sluggish.

A sluggish controller makes small control actions at a slow pace. This enables the controller to literally wait out the delay between action and response. Detuning addresses the problem of over-correction just described, but sluggish tuning is associated with a poor control performance.

22.2 Predictive Models as Part of the Controller Architecture

An alternative solution is model predictive control (MPC), which has the potential to markedly improve closed loop performance in the presence of large dead time. Model predictive controllers incorporate a dynamic process model as part of the architecture of the control algorithm. For success in implementation, the model must reasonably describe the controller output to measured process variable dynamic behavior. We already have experience developing dynamic process models because the FOPDT (first order plus dead time) form used for controller tuning is suitable for MPC implementation, though it is the simplest of the models available.

The function of the dynamic model is to predict the future value of the measured process variable based on the current state of the process and knowledge of recent control actions. A control action is "recent" if the dynamic response it caused in the process is still in progress.

If the predicted measured process variable does not match a desired set point, this future error enables corrective control actions to be taken immediately, and before the predicted problem actually occurs. Thus, the model predictive controller exploits process knowledge contained in the dynamic model to compute current control actions based on a predicted future.

In theory, a perfect model can eliminate the negative influence of dead time on controller performance. In the practical world, as we will see later in this chapter, MPC can certainly provide a performance benefit in the presence of large dead time. This benefit comes at a price, however. The designer must identify an appropriate dynamic model form, fit the model parameters to process data, and program the result into the control computer. The model predictions must then be properly sequenced with the computations of the feedback loop to create an integrated MPC architecture.

22.3 The Smith Predictor Control Algorithm

The Smith predictor is the simplest implementation of general MPC theory (a variant of the Smith predictor can be created from MPC theory by choosing both the near and far prediction horizon as $\theta_P/T + 1$ samples, a control horizon of one, and a constant set point profile). The Smith predictor architecture, illustrated in Fig. 22.2, is comprised of an ideal (or no dead time) process model block and a separate dead time model block. The computation proceeds as follows:

1) The ideal process model receives the current value of the controller output, $u(t)$, and computes $y_{ideal}(t)$, which is a model prediction of what the measured process variable, $y(t)$, would be if there were no dead time in the process (or alternatively, a prediction of what $y(t)$ will be one dead time into the future).

2) $y_{ideal}(t)$ is then fed into a dead time model where it is stored until one dead time, θ_P, passes. At the instant that the $y_{ideal}(t)$ is stored, a previously stored $y_{process}(t)$ is released. This $y_{process}(t)$ is the value of $y_{ideal}(t)$ that was computed and stored one dead time ago. Hence, $y_{process}(t)$ is a model prediction of the current value of $y(t)$.

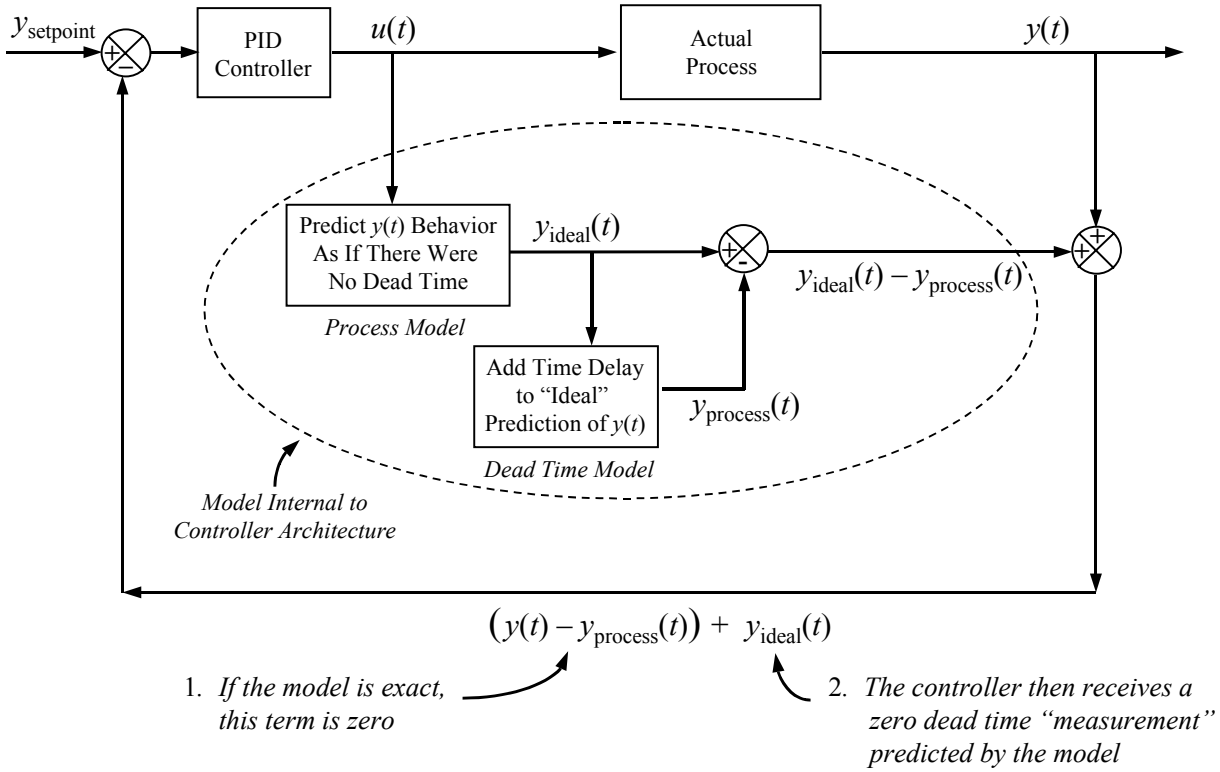


Figure 22.2 - The Smith predictor model based controller architecture

The ideal and dead time model predictions are combined with the actual process measurement prior to every control action to form the Smith controller error, $e^*(t)$, as:

$$e^*(t) = y_{setpoint}(t) - \left[y(t) - y_{process}(t) + y_{ideal}(t)\right] \tag{22.1}$$

If the model *exactly* describes the dynamic behavior of the actual process, then:

$$y(t) - y_{process}(t) = 0 \tag{22.2}$$

or

$$\left[y(t) - y_{process}(t)\right] + y_{ideal}(t) = y_{ideal}(t) \tag{22.3}$$

and so for a perfect model, the Smith predictor error, $e^*(t)$, going to the controller is:

$$e^*(t) = y_{setpoint}(t) - y_{ideal}(t) \tag{22.4}$$

Hence, if the model exactly describes process behavior, then the Smith predicted error going to the controller, rather than being the traditional $e(t) = y_{setpoint}(t) - y(t)$, becomes the difference between the set point and a prediction of what the measured process variable would be if there were no dead time in the process.

Of course, the dynamic model will never exactly describe the true behavior of a process and the logic presented above does not reflect this. It is safe to assume, however, that the ability of the Smith predictor to reduce the influence of dead time on controller performance is directly related to how well the model indeed describes the actual process dynamics. It should also be noted that a very poor match between the model predictions and the actual process dynamics invites disaster, including creating an unstable closed loop system for an otherwise well behaved process.

22.4 Exploring the Smith Predictor Controller

To isolate and explore the impact of dead time on controller performance and to establish the benefits of the Smith predictor control algorithm, we use LOOP-PRO's *Custom Process* module. This study proceeds as follows:

- a second order without dead time process is created in *Custom Process* and a PI controller is designed and validated. The control objective is a 10% peak overshoot ratio and complete settling of dynamics in one cycle of the measured process variable when the set point is stepped from 50% up to 55%.
- significant dead time is added to the original second order without dead time process and the resulting degradation in controller performance is explored.
- a Smith predictor is designed and implemented to compensate for the added dead time and to restore the original set point tracking performance.

PI Control of a Second Order Without Dead Time Process

The second order without dead time process is implemented in *Custom Process* with the parameters:

Process gain, $K_P = 1.2$
First time constant, $\tau_{P1} = 10$ min
Second time constant, $\tau_{P2} = 7$ min
Process dead time, $\theta_P = 0$ min

Thus, the process has a steady state gain of 1.2 (no units specified) and two time constants (which is why it is second order) of 10 minutes and 7 minutes.

As shown in Fig 22.3, controller tuning is based on a pulse test where the controller output is stepped from 50% up to 55% and back again. A *Design Tools* fit of a FOPDT model to the pulse test data is also shown in the figure (a doublet test is normally recommended for controller design so data is generated both above and below the design level of operation to average nonlinear effects. Since *Custom Process* simulations are linear, a pulse test provides equally valid data.)

The FOPDT model parameters computed by *Design Tools* and shown at the bottom of the Fig. 22.3 are:

Process gain, $K_P = 1.2$ (unitless)
Overall time constant, $\tau_P = 13.7$ min
Apparent dead time, $\theta_P = 3.7$ min

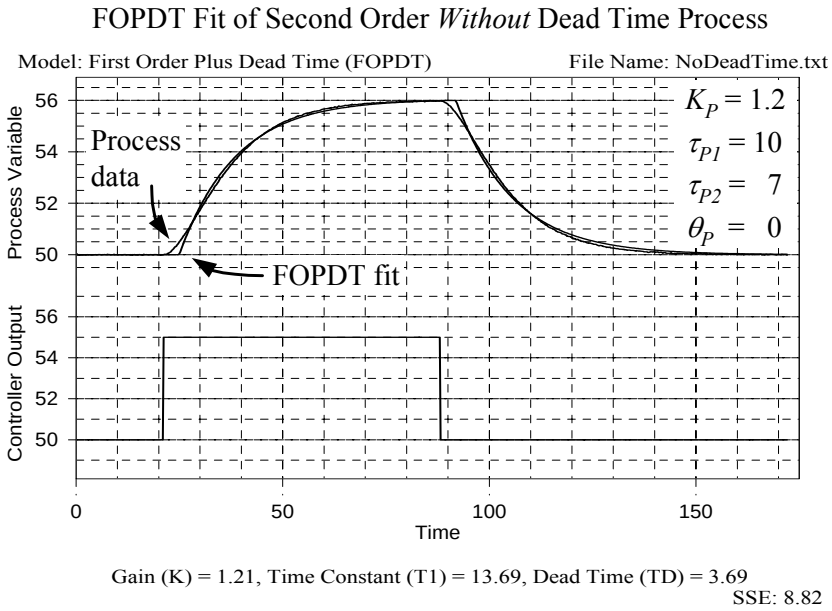


Figure 22.3 - FOPDT model of second order without dead time pulse test data

It is interesting to note that a FOPDT model fit of data from this second order without dead time process yields a rather significant apparent dead time (where "significant" dead time exists only in comparison to the time constant of the process). This apparent dead time is the natural result of approximating the dynamics of a higher than first order processes with the FOPDT form.

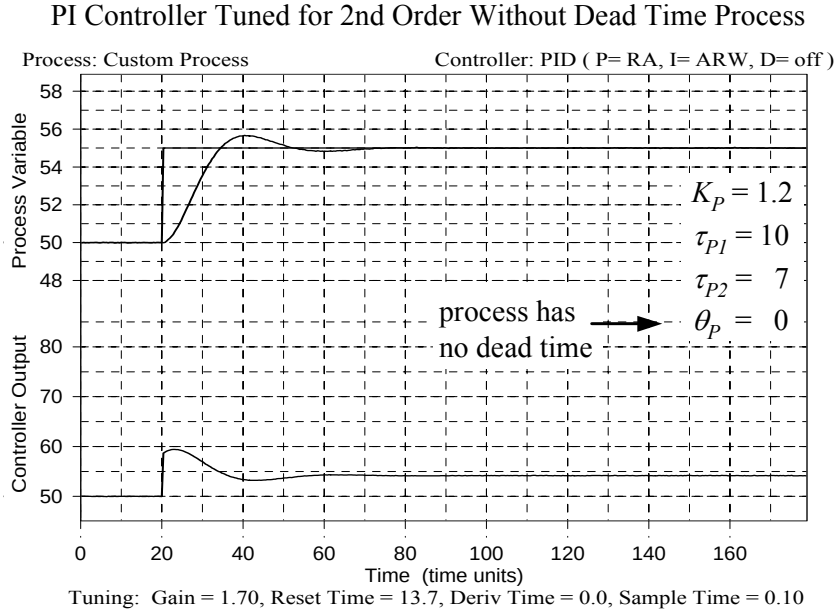


Figure 22.4 - IMC tuned PI controller produces the design 10% overshoot, a rise time of 15 minutes and complete settling in 50 minute

Performance Degrades When Dead Time is Added

The *Custom Process* simulation is now modified by adding 5 minutes of dead time to the original model. To appreciate the impact of dead time on controller performance, the PI tuning parameters used to generate Fig. 22.4 are retained and the same set point step test is performed.

As shown in Fig. 22.5, the addition of dead time results in seriously degraded performance. The peak overshoot ratio has climbed from 10% up to about 70%, and complete settling of dynamics has increased from 50 minutes up to well over 150 minutes with multiple cycles of the measured process variable.

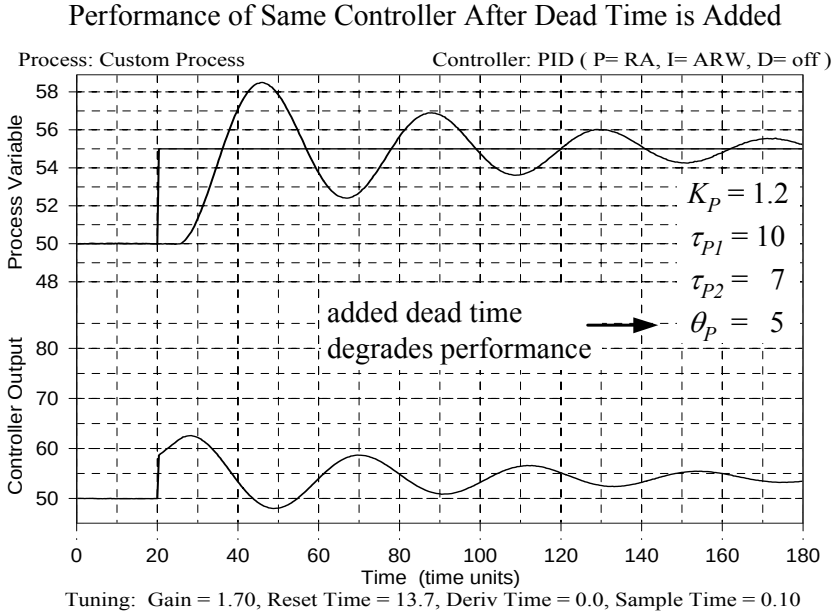


Figure 22.5 – Using the PI tuning of Fig. 22.4, the added dead time causes a 70% overshoot and complete settling in excess of 150 minutes

One method of achieving the design performance criteria of a 10% peak overshoot ratio and complete settling within one cycle of the measured process variable is to detune the controller. Since the process has been altered to include significant dead time, the most efficient way to determine the more sluggish tuning values is to follow the identical design procedure just detailed.

Hence, as shown in Fig. 22.6, a FOPDT model is fit to pulse test data collected form the second order plus dead time (SOPDT) process. At the bottom of the Fig. 22.6 are the FOPDT model parameters computed by *Design Tools*:

Process gain, $K_P = 1.2$ (unitless)
Overall time constant, $\tau_P = 13.7$ min
Apparent dead time, $\theta_P = 8.7$ min

Using IMC tuning in this study, the closed loop time constant, τ_C, is computed as the larger of $0.1\tau_P$ or $0.8\theta_P$, and thus τ_C = 2.95 minutes. Substituting this τ_C and the above FOPDT model parameters into the IMC correlations yields the PI tuning values:

Controller Gain, $K_C = 1.7$ (unitless)
Reset Time, $\tau_I = 13.7$ min

The capability of the PI controller in tracking step changes in set point for the second order without dead time process is shown in Fig. 22.4. For a set point step from 50% up to 55%, the controller achieves the desired peak overshoot ratio of about 10% with a rise time of about 15 minutes and complete settling in one cycle of the measured variable in about 50 minutes. This control performance becomes the base case for the subsequent investigations.

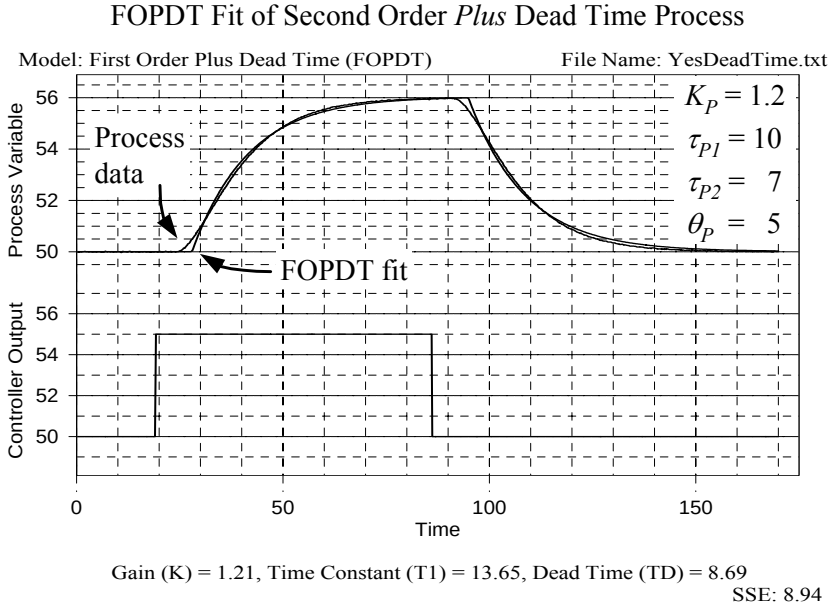


Figure 22.6 - FOPDT model of second order with dead time pulse test data

The new PI tuning values computed from the IMC correlations for the SOPDT process are:

Controller Gain, $K_C = 0.7$ (unitless)
Reset Time, $\tau_I = 13.7$ min

As shown in Fig. 22.7, this tuning produces the design performance criteria of a 10% overshoot ratio and complete settling within one cycle of the measure variable. However, rise time has doubled from the base case 15 minutes up to about 30 minutes. Complete settling now occurs in 80 minutes, a dramatic increase over the 50 minutes of the no dead time base case of Fig. 22.4.

The Smith Predictor Restores Performance
LOOP-PRO permits implementation of a PI with Smith predictor controller by selecting that option from the list of controllers and entering the PI tuning values and the Smith predictor model values where indicated in the controller design form. The model fit shown in Fig. 22.6 from our previous controller design provides a FOPDT model that describes the dynamics of the SOPDT process. This will serve as our predictive process model.

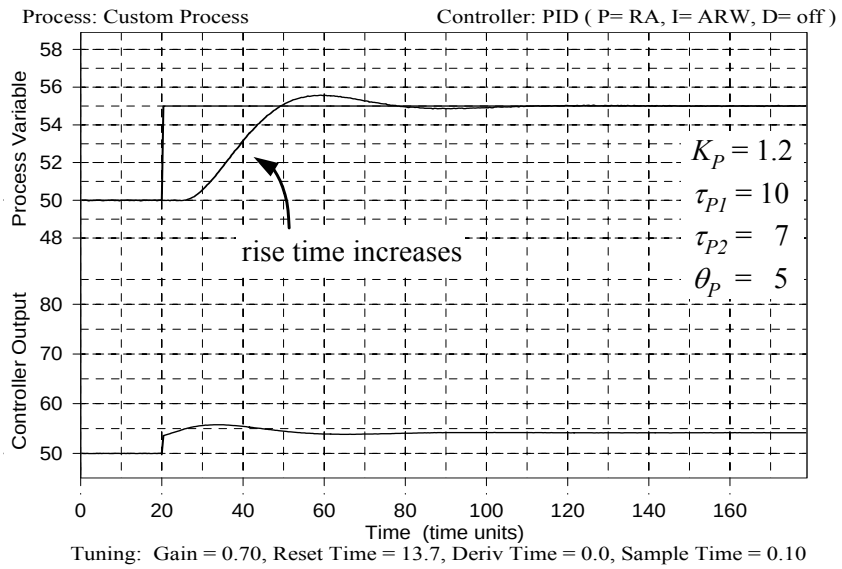


Figure 22.7 - Detuning the controller produces the design 10% overshoot, but rise time has doubled to 30 minutes and complete settling now takes 80 minutes

Tuning presents a challenge, however, because there are no correlations readily available for PI controllers used in a Smith predictor architecture. Here we get creative by recognizing that the Smith predictor theoretically eliminates the impact of dead time on control performance. Thus, we will use the standard IMC tuning correlations, but in the calculation we will employ an artificial or theoretical minimum value for dead time.

Recall that sample time, T, should ideally be less than or equal to $0.1\,\tau_P$. And for commercial control equipment, one sample time always passes between a control action and when the controller receives the next measurement. Hence, the minimum that dead time can be on a commercial control system, in theory anyway, is one sample time.

So in the IMC tuning correlation, we will use the process gain and time constant from the actual FOPDT fit of the process data as shown in Fig. 22.6, but for dead time in the correlation we will use $\theta_P = 0.1\,\tau_P = 1.4$ min, or:

Process gain, $K_P = 1.2$ (unitless)
Overall time constant, $\tau_P = 13.7$ min
Apparent dead time, $\theta_P = 1.4$ min

Using these parameters in the IMC correlation yields the tuning values for the PI with Smith predictor architecture:

Controller Gain, $K_C = 4.1$ (unitless)
Reset Time, $\tau_I = 13.7$ min

As shown in Fig. 22.8, the PI with FOPDT Smith predictor produces very close to the design performance criteria of a 10% overshoot ratio and complete settling within one cycle of the measure variable. Rise time approaches 20 minutes, which equals the base case 15 minutes plus the 5 minutes of dead time added to the base case process. Complete settling occurs in 55 minutes and this equals our base case settling of 50 minutes plus the 5 minutes of dead time.

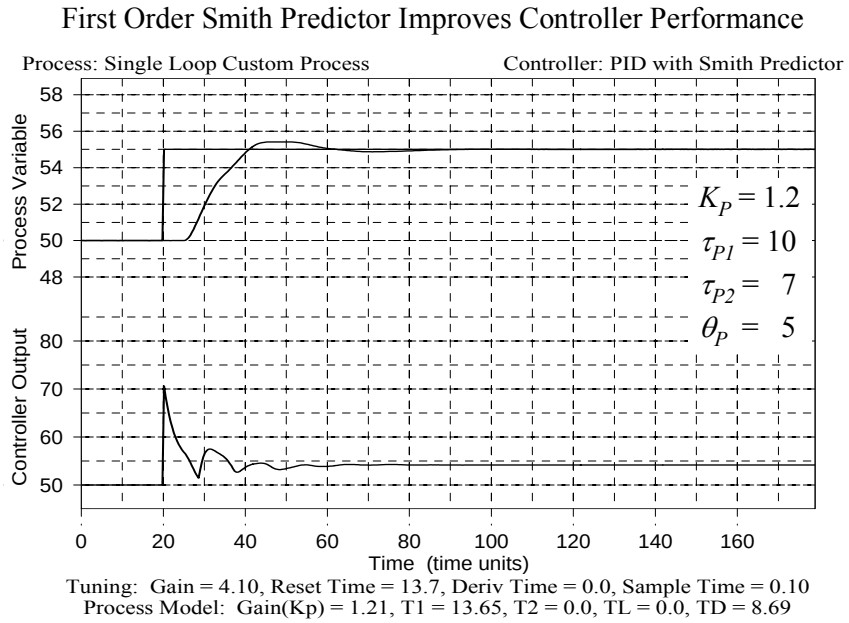


Figure 22.8 - PI controller with FOPDT Smith predictor produces the design 10% overshoot, a rise time of 20 minutes and complete settling in 55 minutes

Though the design performance criteria are met, one observation is that the controller works uncomfortably hard to achieve this success as evidenced by the large swings in the controller output signal trace. As we learn in Fig. 22.9, this behavior is the result of using the FOPDT model to predict the dynamics of a second order plus dead time process.

In Fig. 22.9 we use the original PI controller tuning values from Fig. 22.4, but now the identical SOPDT model used to simulate the process in *Custom Process* is used as the predictive process model in the Smith predictor architecture. The potential of the MPC architecture is demonstrated in that the exact controller output signal and measured process variable traces evolve here as they did in the base case of Fig. 22.4. The only difference is that there is a 5 minute shift in the measured process variable in Fig. 22.9 because of the process dead time added to the simulation.

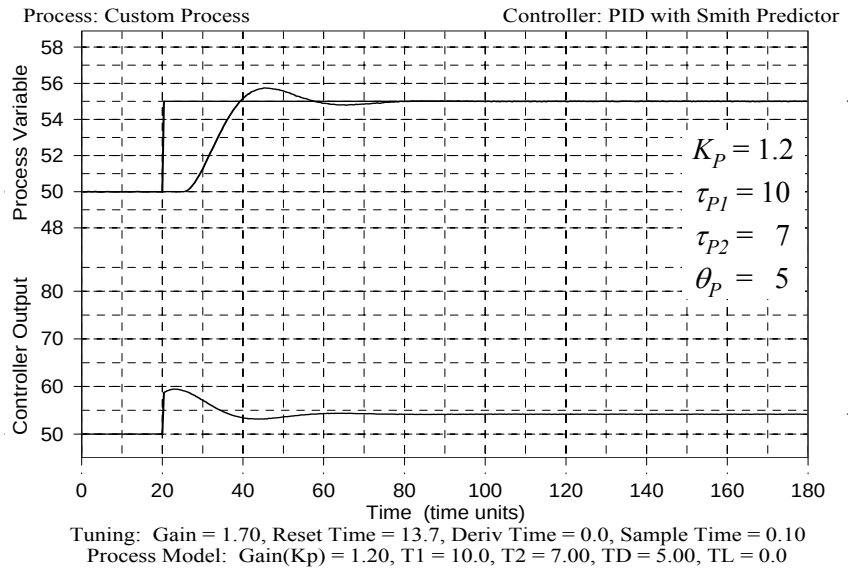


Figure 22.9 - PI controller with SOPDT Smith predictor produces the design 10% overshoot, a rise time of 20 minutes and complete settling in 55 minutes

23. DMC - Single Loop Dynamic Matrix Control

23.1 Model Predictive Control

Model predictive control (MPC) refers to a class of control algorithms that have a dynamic model of the process programmed into the control architecture. The function of the model is to predict the future behavior of the process based upon past controller moves and the current state of the process. At each sample time, the next controller move is computed from a comparison of this predicted future behavior with the desired set point trajectory.

Expanding this explanation, the MPC strategy begins with a performance objective function, or mathematical equation that defines "good" control. This objective function typically combines controller error and controller effort into a single formulation. If the measured process variable is maintained at its set point over the predicted future (controller error is zero), the controller performance is certainly good. If the control effort (the size of each control move) is small, then the mechanical components of the final control element won't wear excessively and the process will not experience unsettling sudden changes. Thus, by finding the mathematical minimum of the objective function, control actions of modest size will be computed to drive the future predicted controller error to zero.

Future set point tracking performance in the objective function is computed using the dynamic process model. The model predicts the future of the measured process variable using past and future (yet to be computed) controller output moves. The objective function is minimized by computing a series of future controller output moves over the control horizon (the distance into the future being considered) that balance set point tracking performance with controller effort. Only the first of these controller output moves is implemented before repeating the entire procedure at the next sample time.

Model predictive control is growing in popularity for its reasonably intuitive approach, its ability to control processes with complex dynamics such as large dead time, inverse response and unstable open loop dynamics, and its convenient handling of multivariable control challenges. MPC offers additional benefits because process constraints and variable set point trajectories can be directly addressed in the control calculations. Unfortunately, the design of a single loop MPC strategy requires specifying at least five adjustable tuning parameters prior to implementation. For multivariable implementations, the tuning burden grows substantially.

23.2 Dynamic Matrix Control

Dynamic matrix control (DMC) is one MPC strategy experiencing rapid growth in the chemical process industries. DMC does not compete with classical PID controllers. Rather, it is typically implemented in a hierarchy above a set of traditional PID loops. Such an approach is well suited for multivariable control, where the actions of x controller outputs each affect y process variables. It also permits the flexible handling of constraints. These can be specified explicitly as hard constraints (e.g., process limits) or implicitly as soft constraints (e.g., economic objectives).

Here we focus on the design and tuning of a single-loop DMC controller to illustrate design concepts and provide a handle on the tuning strategy, both of which extend directly to implementation of the multivariable case. The tuning strategy for a DMC controller can be reduced to a ‘recipe’ as detailed in Table 23.1. After summarizing the theory, the use of this table is demonstrated in an example.

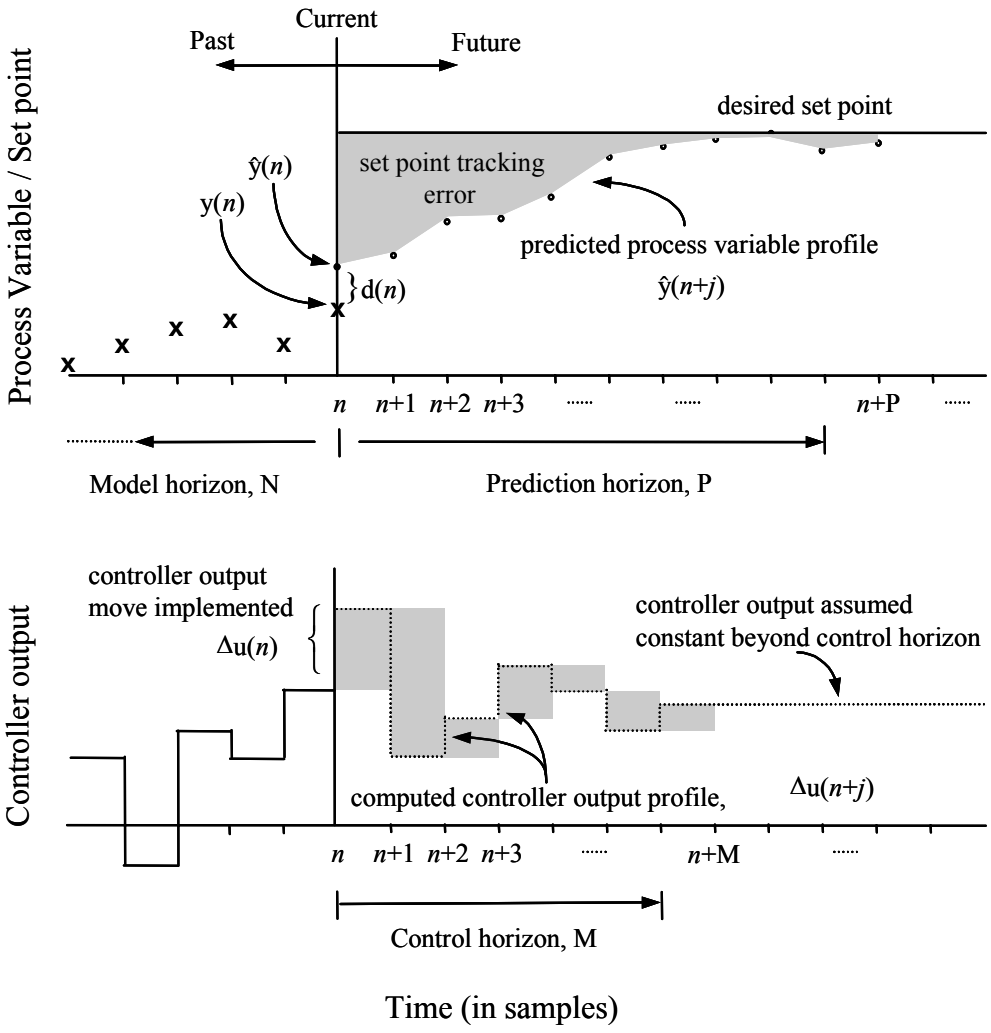


Figure 23.1 - The moving horizon architecture

DMC is a 'moving horizon' MPC strategy. As shown in Fig. 23.1, moving horizon controllers employ a model internal to the controller architecture to predict the future measured process variable profile, $\hat{y}(n+j)$ (j = 1, 2, …, P). P is the prediction horizon and represents the number of sample times into the future over which DMC predicts the measured process variable.

The predicted process variable profile, $\hat{y}(n+j)$, is computed using the recent history of controller output moves, $\Delta u(n-j)$ (j = 1, 2, …, N) and a finite step response process model:

$$\underbrace{\hat{y}(n+j)}_{\substack{\text{Predicted process}\\ \text{variable profile}}} = y_{ss} + \underbrace{\sum_{i=1}^{j} a_i\, \Delta u(n+j-i)}_{\substack{\text{Effect of current and}\\ \text{future moves}}} + \underbrace{\sum_{i=j+1}^{N-1} a_i\, \Delta u(n+j-i)}_{\text{Effect of past moves}} + \underbrace{d(n)}_{\substack{\text{Prediction}\\ \text{error}}} \qquad (23.1)$$

In Eq. 23.1, y_{ss} is the initial steady state of the measured process variable and $\Delta u_i = u_i - u_{i-1}$ is the change in the controller output at the i^{th} sample time. Also, a_i (i = 1, 2, …, N) are the unit step response coefficients as discussed in the next section. N is the model horizon and represents the number of past controller output moves used by DMC to predict the future process variable profile.

The prediction error in Eq. 23.1, $d(n)$, is the difference between the prediction of the current value of the process variable and its actual measured value, or $d(n) = y(n) - \hat{y}(n)$. This prediction error is added to the predicted process variable profile, $\hat{y}(n+j)$, to correct for any unmeasured disturbances or inaccuracies due to plant-model mismatch.

The predicted process variable profile computed above is subtracted from the set point profile, $y_{sp}(n+j)$, over the next P sample times, squared, and then summed to yield the sum of the squares of the set point tracking error. The DMC objective function, to be minimized at every sample time, is then represented as

$$Min\ \mathrm{J} = \underbrace{\sum_{j=1}^{\mathrm{P}} \left\{ \mathrm{y_{sp}}(n+j) - \hat{\mathrm{y}}(n+j) \right\}^2}_{\substack{Set\ point\ tracking \\ error}} + \lambda \underbrace{\sum_{i=1}^{\mathrm{M}} \left\{ \Delta \mathrm{u}(n+i-1) \right\}^2}_{\substack{Penalty\ on\ controller \\ output\ move\ sizes}} \tag{23.2}$$

Here, λ is a positive constant that weighs the controller output move sizes relative to the sum of the squares of the set point tracking error. The penalty on the controller output move sizes is introduced into the DMC objective function to suppress otherwise aggressive control actions when the control horizon, M, is greater than one.

Before the optimization problem posed in Eq. 23.2 can be solved, the set point tracking error must be expressed in terms of the controller output moves that have already occurred and those to be determined, as follows:

$$\begin{aligned} & y_{sp}(n+j) - \hat{y}(n+j) \\ & \quad = \underbrace{y_{sp}(n+j) - y_{ss} - \sum_{i=j+1}^{N-1} a_i \Delta u(n+j-i) - d(n)}_{\substack{\text{Predicted error based on} \\ \text{past moves, } e(n+\mathrm{j})}} - \underbrace{\sum_{i=1}^{j} a_i \Delta u(n+j-i)}_{\substack{\text{Effect of current and future} \\ \text{moves to be determined}}} \end{aligned} \tag{23.3}$$

where j = 1, 2, …, P. Eq. 3 is a linear system of P equations that can be represented in a matrix form, assuming that the controller output is held constant beyond the control horizon, M:

$$\boldsymbol{y_{sp}} - \hat{\boldsymbol{y}} =$$

$$\underbrace{\begin{bmatrix} e(n+1) \\ e(n+2) \\ e(n+3) \\ \vdots \\ e(n+M) \\ \vdots \\ e(n+P) \end{bmatrix}_{P\times 1}}_{\substack{\text{Predicted error based} \\ \text{on past moves, } \bar{e}}} - \underbrace{\begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & & 0 \\ a_3 & a_2 & a_1 & \ddots & 0 \\ \vdots & \vdots & \vdots & & 0 \\ a_M & a_{M-1} & a_{M-2} & & a_1 \\ \vdots & \vdots & \vdots & & \vdots \\ a_P & a_{P-1} & a_{P-2} & \cdots & a_{P-M+1} \end{bmatrix}_{P\times M}}_{\text{DMC Dynamic Matrix, } A} \underbrace{\begin{bmatrix} \Delta u(n) \\ \Delta u(n+1) \\ \Delta u(n+2) \\ \\ \Delta u(n+M-1) \end{bmatrix}_{M\times 1}}_{\substack{\text{Current and future controller} \\ \text{output moves to be determined, } \Delta\bar{u}}} \quad (23.4)$$

or in a compact matrix notation as:

$$\bar{\boldsymbol{y}}_{\boldsymbol{sp}} - \hat{\boldsymbol{y}} = \bar{\boldsymbol{e}} - \boldsymbol{A}\,\Delta\bar{\boldsymbol{u}} \tag{23.5}$$

Here, $\bar{\boldsymbol{y}}_{\boldsymbol{sp}}$ is the vector of future set points, $\hat{\boldsymbol{y}}$ is the vector comprising the predicted process variable profile, $\bar{\boldsymbol{e}}$ is the vector of predicted errors over the next P sampling intervals based on past controller output moves, $\boldsymbol{A}$ is the DMC dynamic matrix, and $\Delta\bar{\boldsymbol{u}}$ is the vector of M controller output moves to be determined.

With the transformation of the set point tracking error as in Eq. 23.5, the DMC objective function can be alternatively written as:

$$\underset{\Delta\bar{\boldsymbol{u}}}{Min}\ J = \left[\bar{\boldsymbol{e}} - \boldsymbol{A}\Delta\bar{\boldsymbol{u}}\right]^{\mathrm{T}}\left[\bar{\boldsymbol{e}} - \boldsymbol{A}\Delta\bar{\boldsymbol{u}}\right] + \left[\Delta\bar{\boldsymbol{u}}\right]^{\mathrm{T}}\lambda\left[\Delta\bar{\boldsymbol{u}}\right] \tag{23.6}$$

In the unconstrained case, Eq. 23.6 represents a least squares optimization problem that has a closed form solution representing the DMC control law (e.g., García and Morshedi, 1986):

$$\Delta\bar{\boldsymbol{u}} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda\mathbf{I})^{-1}\boldsymbol{A}^{\mathrm{T}}\bar{\boldsymbol{e}} \tag{23.7}$$

Adding constraints to this classical formulation (Eq. 23.6) produces the Quadratic Dynamic Matrix Control (QDMC) (e.g., García and Morshedi, 1986) algorithm. The constraints considered in this work include:

$$\hat{y}_{min} \le \hat{y} \le \hat{y}_{max} \tag{23.8a}$$

$$\Delta\bar{\boldsymbol{u}}_{min} \le \Delta\bar{\boldsymbol{u}} \le \Delta\bar{\boldsymbol{u}}_{max} \tag{23.8b}$$

$$\bar{\boldsymbol{u}}_{min} \le \bar{\boldsymbol{u}} \le \bar{\boldsymbol{u}}_{max} \tag{23.8c}$$

23.3 The DMC Process Model

DMC requires that the dynamic process model be expressed both as the finite step response model (Eq. 23.1) and the DMC dynamic matrix (Eq. 23.4). Both forms rely on the unit step response coefficients, a_i (i = 1, 2, …, N), generated from the actual process.

Step response data is generated by introducing a positive step in the controller output with the process at steady state and the controller in manual mode. From the instant the step change is made,

the process variable response is recorded as it evolves and settles at a new steady state. For a step of arbitrary size, the response data is normalized by dividing through by the size of the controller output step to yield the unit step response. For processes with a modest steady-state gain, a unit step in the controller output can be used directly to obtain the unit step response. In either case, it is necessary to make the controller output step large enough such that noise in the process variable measurement does not mask the true process behavior.

Discrete points at every sample time along the unit step response are collected (Fig. 23.2) to yield the unit step response coefficients:

$$\text{Unit step response coefficients} = [a_1,\ a_2,\ a_3,\ \cdots,\ a_N] \tag{23.9}$$

The unit step response coefficients in Eq. 23.9 are used in Eq. 23.1 to compute the predicted process variable profile at every sample time. Also, the first P of the N coefficients are cast in a matrix form as shown in Eq. 23.4 to obtain the DMC dynamic matrix. This matrix can then be used directly in the DMC control law (Eq. 23.7). Notice, however, that both the sample time, T, and the prediction horizon, P, have to be known before the finite step response model or the DMC dynamic matrix can be designed.

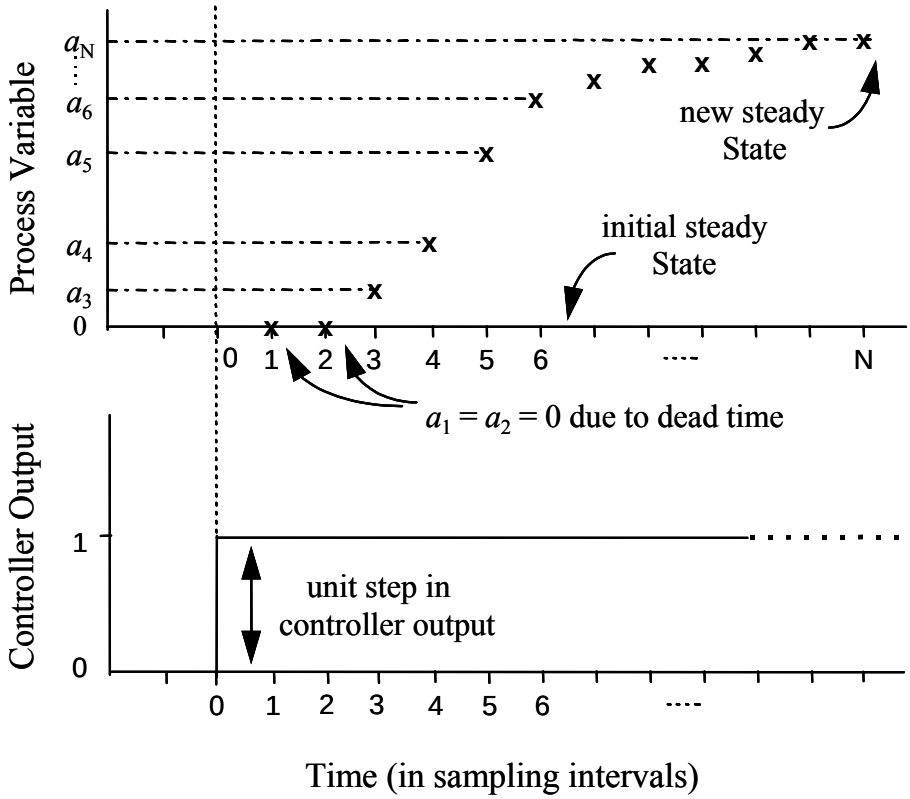


Figure 23.2 - Generating the unit step response coefficients

23.4 Tuning DMC

Tuning DMC, even for a single-input single-output (SISO) process, is challenging because of the number of adjustable parameters that affect closed loop performance. As listed in Appendix C, these

include: a finite prediction horizon, P; a model horizon, N; a control horizon, M; a move suppression coefficient, λ; and a sample time, T.

Step 1 involves fitting a first order plus dead time (FOPDT) model to actual controller output to measured process variable data. Reasonable estimates of the FOPDT model parameters, i.e., the steady state gain, K_p, overall time constant, τ_p, and effective dead time, θ_p, are essential to the success of this tuning strategy. It is important to stress that this model is used in the tuning strategy only and that Eq. 23.1 through 23.7 used in the implementation of DMC are formulated from the actual process data obtained as described above (Fig. 23.2 and Eq. 23.9).

Step 2 involves the selection of an appropriate sample time, T. The estimated FOPDT model parameters provide a convenient way to select T. If the designer does not have complete freedom to select sample time equal to the value computed, then it should be picked as close as possible to this recommended value.

Step 3 computes a model horizon, N, and a prediction horizon, P, from τ_p, θ_p and T. Note that both N and P cannot be selected independent of the sample time, T. Also, it is imperative that N be equal to the open loop process settling time in samples to avoid truncation error in the predicted process variable profile.

Step 4 requires the specification of a control horizon, M. Recommended values of M are such that $M \times T$ is larger than the time required for the slowest open loop response to reach 60% of the steady state. A convenient way to select M is to compute an integer value using τ_p and T as shown in Appendix C. Selecting $M > 1$ can be very useful to the practitioner since this provides advance knowledge of the impending controller output moves.

Step 5 involves computation of a move suppression coefficient, λ. With $M = 1$, the need for a move suppression is eliminated and λ is set equal to zero. However, if $M > 1$, a positive move suppression coefficient of appropriate magnitude is essential to suppress otherwise aggressive control action.

As with all controllers, it may be necessary to perform final on-line tuning. The best single tuning parameter for performance adjustment is the move suppression coefficient, λ. Increasing λ produces smaller controller output move sizes and slower process variable response.

23.5 Example Implementation

The design and tuning of DMC are demonstrated on a *perfect DMC* controller. A *perfect DMC* controller employs the identical models in the DMC step response model as is used for the process simulation. Be aware that in real-world applications, most DMC step response models will differ from the true process behavior. Hence, the DMC controller capabilities shown here should be considered as the best possible performance.

The ideal process simulation is a second order plus dead time model (SOPDT) overdamped linear model of the form:

Process Gain, K_P	=	2.0
First Time Constant, τ_{P1}	=	20.0 time units
Second Time Constant, τ_{P2}	=	10.0 time units
Dead Time, θ_P	=	10.0 time units

The first step in tuning is to fit a FOPDT model to process data. For this example, *Design Tools* yields the model parameters for the ideal process as $K_p = 2.0$ PV units/CO unit, $\tau_p = 24.1$ time units, $\theta_p = 13.8$ time units. Based on the criteria presented in Step 2, a sample time of 7.0 time units is selected. For the FOPDT parameters estimated and the value of T selected above, the prediction horizon, P, and the model horizon, N, are computed in Step 3 to be 19, the open loop settling time of

the ideal process in samples. Next, a control horizon, *M*, of 5 is computed in Step 4. Finally, in Step 5, an appropriate move suppression coefficient of 23.9 is computed for the ideal process.

With the tuning parameters selected, the unit step response coefficients are determined for the ideal process for a sample time of 7.0 time units. In the initialization of the DMC controller, the step response coefficients are generated for you based on the values of *T*, *P* and *N*. The model used to generate the step response coefficients is entered as a transfer function on the design menu. The user wants to use the transfer function that best represents the process. In this case, the transfer function will be a SOPDT overdamped model. The model parameters listed above are entered into the design menu. With the design complete, DMC is implemented and its performance tested for the ideal process.

The closed loop performance achieved for the ideal process, with DMC tuned using the tuning strategy in the appendix, is shown in Fig. 23.3. The top half of this plot shows the process variable response for a step change in set point. The lower half of the plot shows the controller output moves made by DMC to achieve the desired set point tracking.

Desirable closed loop performance is defined as the process variable response to a step change in set point exhibiting a modest peak overshoot (say, less than 10%). Also, the corresponding controller output move sizes should not exceed 2 to 3 times the final change in controller output. As shown in Fig. 23.3, such a performance is achieved.

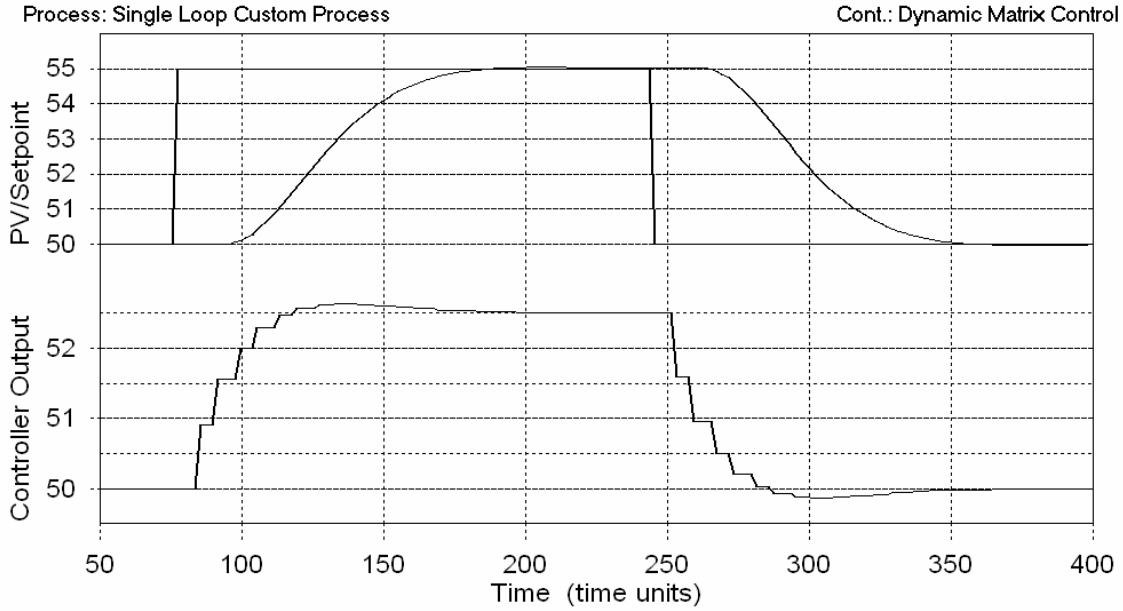


Figure 23. 3 - DMC performance for the ideal process

Effect of Tuning Parameters on DMC Performance

The significance of the tuning strategy can be illustrated in a sensitivity study where the effect of each tuning parameter on closed loop performance is individually explored. For comparison, the DMC tuning and design demonstrated in Fig. 23.3 is labeled as the 'base case' in Figures 23.4 through 23.10.

From left to right, Fig. 23.4 shows the impact of reducing the model horizon, *N*, from 38 to 19 (base case) and then to 10 and finally to 5. With the reduced value of *N*, the now truncated step response model in DMC is able to predict the effect of controller output moves for only 5 sample times (plus the dead time) after a set point change. Beyond 5 sample times, even though the past

controller output moves continue to impact the process variable, their effect is not realized in what amounts to an incorrect computation from the truncated model profile. As in this case, such truncation generally results in unpredictable and undesirable controller performance.

From left to right, Fig. 23.5 shows the impact of reducing the prediction horizon, P, from 19 to 8 and finally to 3. For small values of P the process variable response shows a longer rise time. Note that this change occurs abruptly as the value of P becomes very small rather than occurring gradually as the value of P decreases. This performance change results because: i) the controller output moves computed by DMC seek to eliminate the predicted error over a fewer number of future sampling intervals, ii) the inability of DMC with a small P to predict the process behavior far enough ahead to realize that the past moves will bring the process variable to the new set point and beyond, and iii) the presence of a rather large move suppression coefficient, appropriate for the base case (P = 19), but resulting in very small moves for the case with the reduced prediction horizon (P = 3).

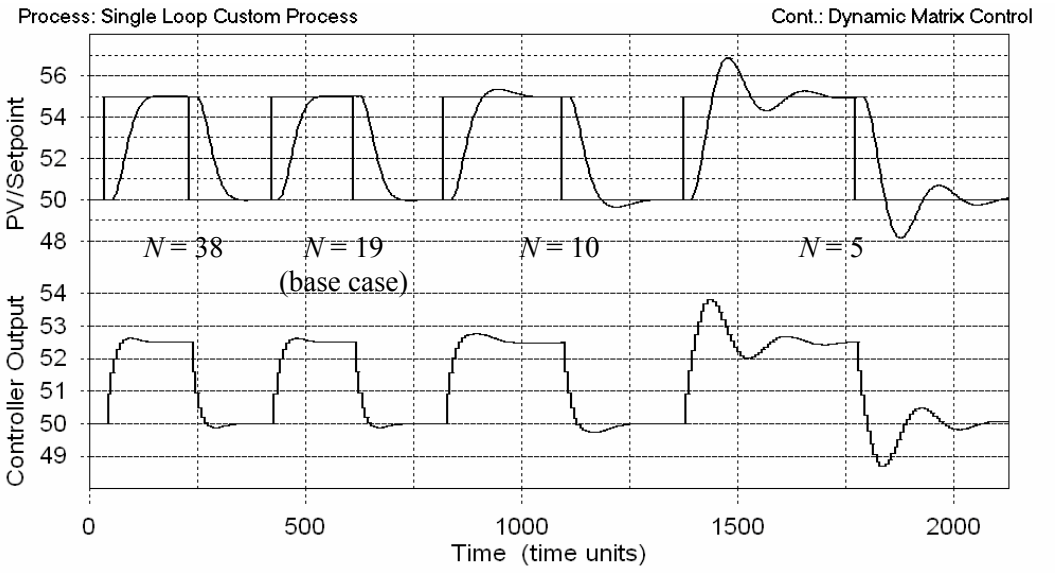


Figure 23.4 – Effect of Varying Only the Model Horizon, N, from the Base Case

As stated previously, the appropriate value for the move suppression coefficient is related to the prediction horizon, P. From left to right, Fig. 23.6 shows the impact of reducing the move suppression coefficient from 192 to 23.9 (base case) and then to 3 and finally to 0.375 for constant values of sample time (T = 7.0 time units), model horizon (N = 19), prediction horizon (P = 19), and control horizon (M = 5). As shown below, increasing the move suppression coefficient for a constant prediction horizon can dramatically increase the rise time, resulting in a slower response and poor control. This illustrates the impact of the move suppression coefficient upon the nature of the response.

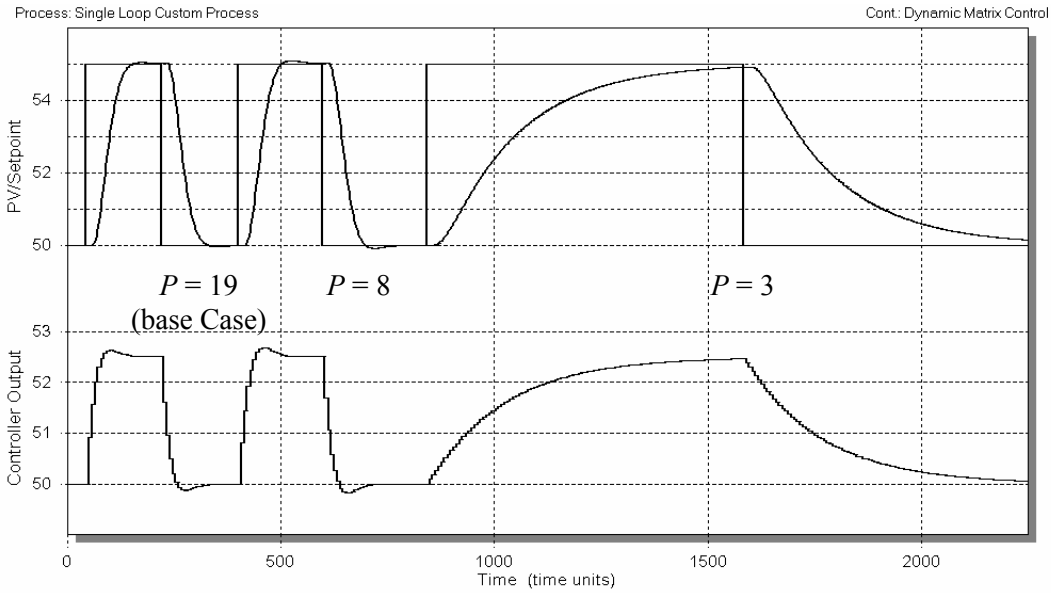


Figure 23.5 – Effect of Varying Only the Prediction Horizon, P, from the Base Case

As the move suppression coefficient decreases, the controller output moves become increasingly aggressive resulting in a process variable response that oscillates with some overshoot. Decreasing the move suppression coefficient reduces the penalty on the move sizes made by DMC resulting in more aggressive moves. Mathematically, this can also be explained as the ineffectiveness of a small move suppression coefficient in conditioning the inherently ill-conditioned system matrix, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, in the DMC control law (Eq. 23.7).

The model and prediction horizons, N and P, respectively, should have the same value. Figures 23.4 through 23.5 demonstrate the individual effects of changing N and P away from one another, but the effect of changing both parameters together is also very interesting. From left to right, Fig. 23.7 illustrates the impact of reducing both N and P from 38 to 19 (base case) to 10 to 5 and finally to 3 for constant values of sample time (T = 7.0 time units), control horizon (M = 5), and move suppression coefficient (λ = 23.9). Notice how the response becomes more oscillatory as N and P decrease until becoming overdamped and sluggish once P equals 3. Therefore, the coupled impact of a low prediction horizon (P = 3) and a large move suppression coefficient (λ = 23.9) overtakes the gradual effect of decreasing both the model and prediction horizons.

From left to right, Fig. 23.8 illustrates the effect of reducing the control horizon, M, from 10 to 5 (base case) to 3 and finally to 1 with the sample time (T = 7.0 time units), model and prediction horizons ($N = P$ = 19), and move suppression coefficient (λ = 23.9) held constant. The difference in performance between the base case and the other cases is rather insignificant, especially in the presence of a significant move suppression coefficient.

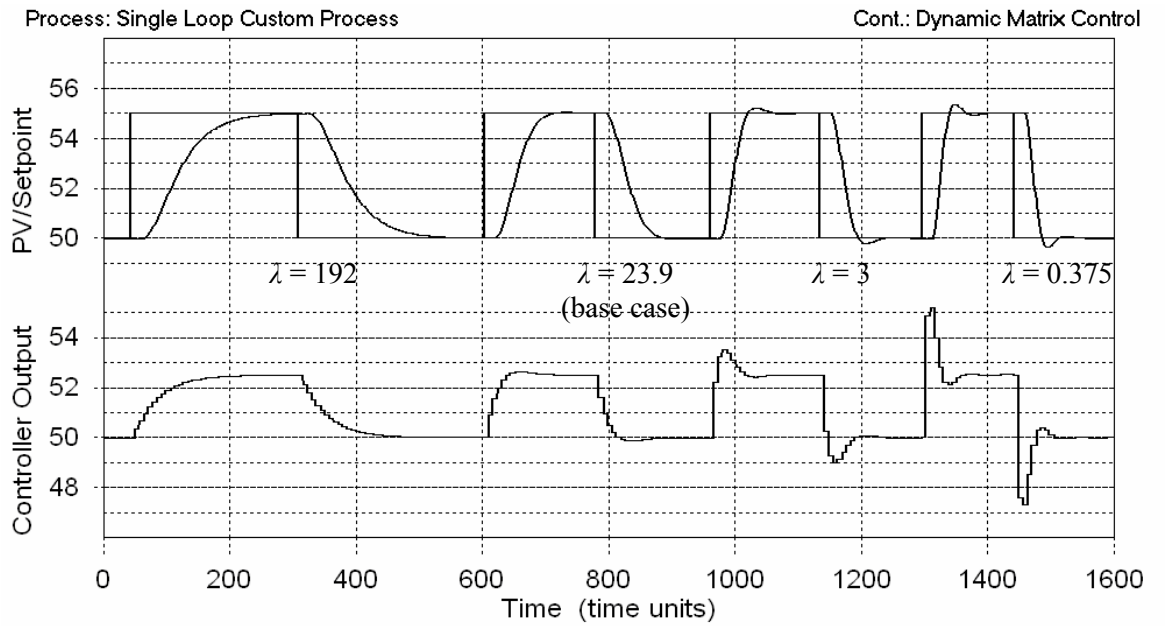


Figure 23.6 – Effect of Varying Only the Move Suppression Coefficient, λ, from the Base Case

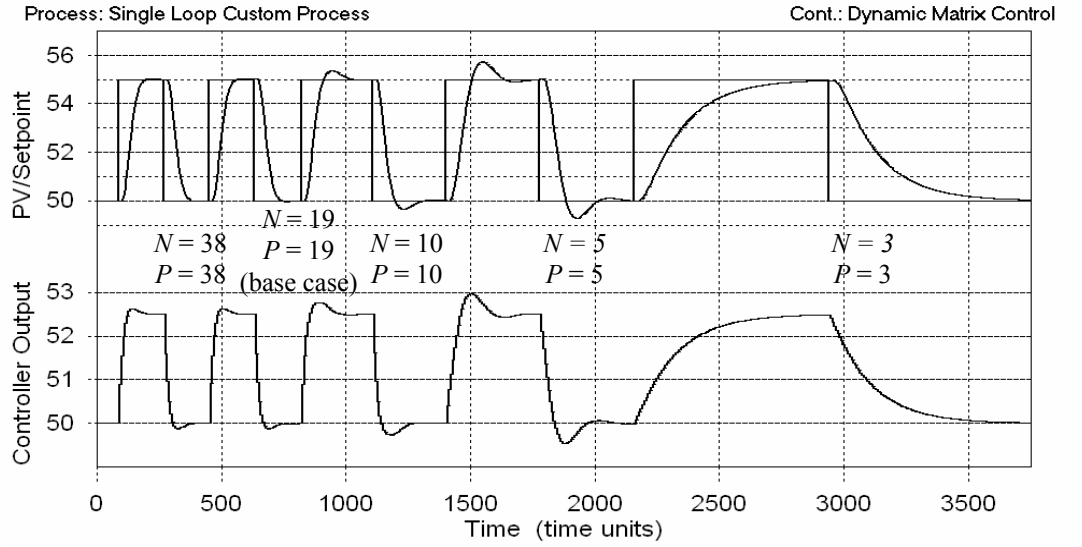


Figure 23.7 – Effect of Varying the Model and Prediction Horizons Together from the Base Case

For the base case with its large control horizon ($M = 5$), the faster process variable response, slight increase in overshoot and increased activity of controller output moves is due to the added degrees of freedom from the extra moves computed at every sample time. With a larger number of moves computed at every sampling interval, DMC can afford to make more aggressive first moves that are compensated for by the extra moves available. Since only the first move is actually implemented while the others are discarded for a new set of M moves computed at the subsequent sample time, the result is a more aggressive response.

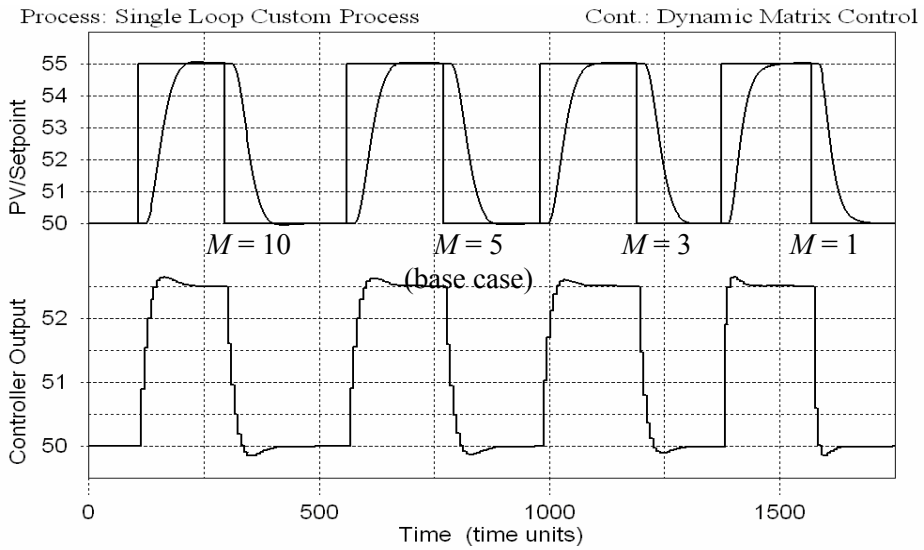


Figure 23.8 – Effect of Varying Only the Control Horizon, M, from the Base Case

From left to right, Fig. 23.9 demonstrates the impact of changing the sample time, T, from 14 to 7 (base case) to 3.5 to 1.75 time units. Note that these changes occurred without changing any of the other DMC parameters from their base case values (as calculated based for $T = 7$ time units). Notice how the response grows more oscillatory as T decreases.

Similarly, Fig. 23.10 also demonstrates the impact of changing T from 14 to 7 (base case) to 3.5 to 1.75 time units. However, note that the other DMC tuning parameters were changed along with the sample time, using the applicable tuning rules found in the appendix. In Table 23.1 below, you will find the DMC tuning parameters used for each step, from left to right, in Fig. 23.10.

Example	*T* (time units)	*N*	*P*	*M*	λ
(a)	14	9	9	2	5.66
(b)	7	19	19	5	23.9
(c)	3.5	39	39	11	97.34
(d)	1.75	78	78	22	366.7

Table 23.1 – Changes in DMC Tuning Parameters with Sample Time, T

It is important to mention here that when the sample time is altered, the unit step response model internal to the DMC architecture should also be appropriately updated to eliminate plant-model mismatch. LOOP-PRO calculates the step response model from a linear model of the process and T, both input by the user. The linear model used does not change with T, but the step response model computed from it by LOOP-PRO does. However, the step response model should change whenever T changes if the step response model is input directly into the controller without such a calculation. Specifically, the updated model should be formed from unit step response coefficients, a_i, collected at the new sampling interval, T.

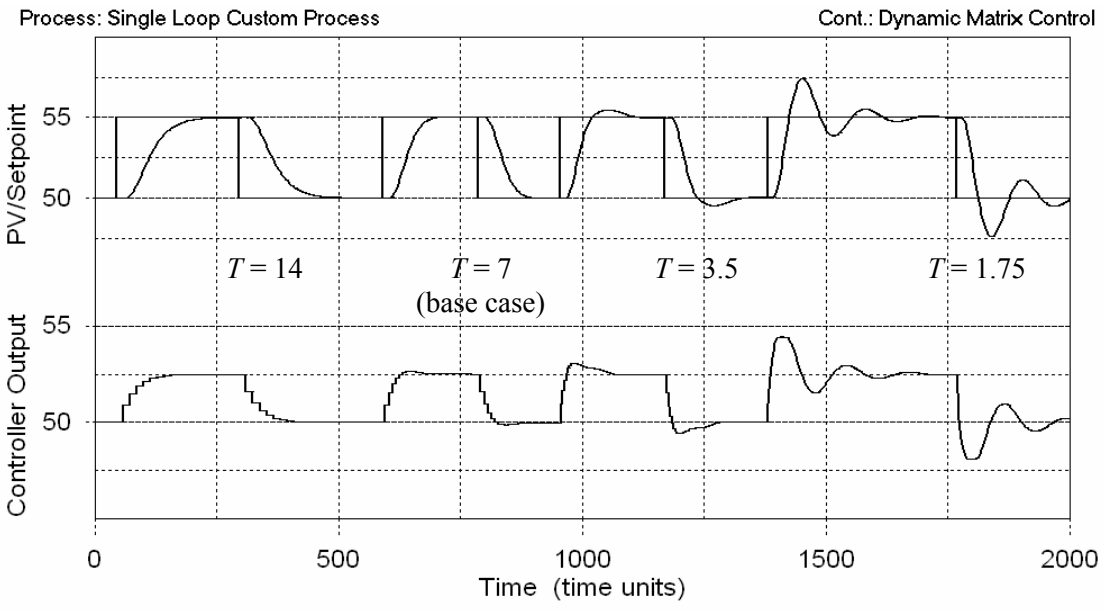


Figure 23.9 – Effect of Varying Only the Sample Time, T, from the Base Case

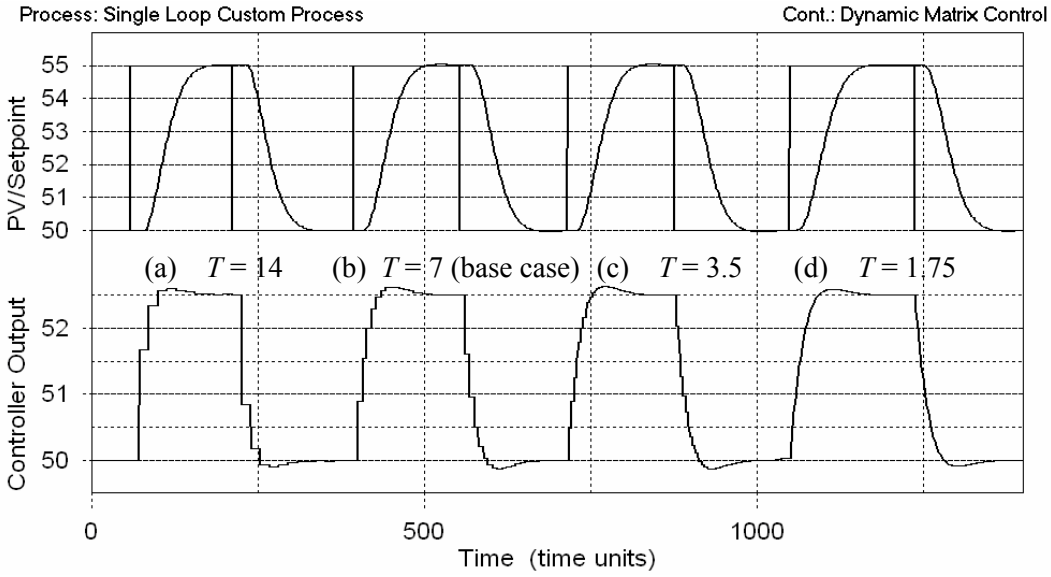


Figure 23.10 – Effect of Varying the DMC Tuning Parameters with T from the Base Case

Notice how the response obtained in Fig. 23.9 shows a highly erratic set of controller output moves accompanied with an irregular, underdamped process variable response while the responses in Fig. 23.10 are quite similar. The cause for this difference in behavior can be noticed from the tuning strategy found in Appendix C, where the computation of all of the tuning parameters depends on the choice of the sample time. Hence, changing the sample time without updating all of the other DMC tuning parameters adversely affects closed loop performance. Figures 23.9 and 23.10 above demonstrate the importance of changing N, P, M, and λ along with any changes in the sample time, T.

Effect of Constraints on DMC Performance

The significance of constraints can be illustrated using the following example. For both step tests shown below, the tuning parameters are given as $T = 7.0$ time units, $P = N = 19$, $M = 5$, and $\lambda = 23.9$, and the constraints are:

$$45 \le \hat{y} \le 60$$
$$-1.0 \le \Delta\overline{\boldsymbol{u}} \le 1.0$$
$$50.0 \le \overline{\boldsymbol{u}} \le 52.5$$

The first plot in Fig. 23.11 shows the performance achieved by DMC tuned for the case when $T = 7$, $P = N = 19$, $M = 5$, and $\lambda = 23.9$ without the above constraints. The second plot represents the performance when constraints listed above are included in the controller design. Notice that when constraints are included the move sizes computed by DMC for the controller output are not greater than 0.5, and the controller output is constrained at values of 50% and 52.5%. When constraints are included, the process variable response is significantly slower than it was without constraints.

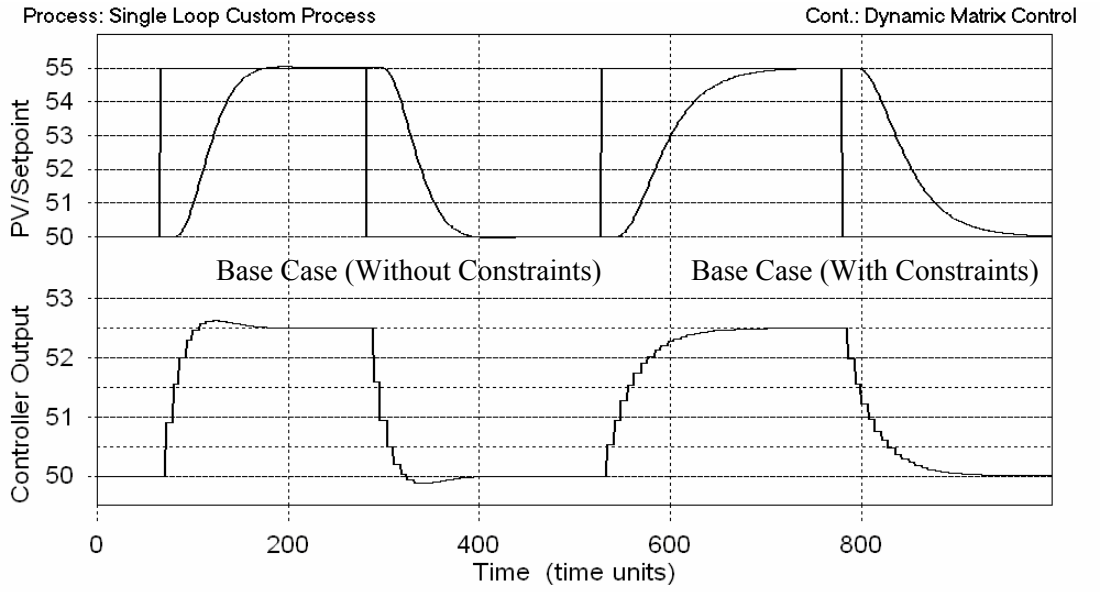


Figure 23.11 – Effect of Constraints on the Base Case

23.6 Chapter Nomenclature

a_i	=	i^{th} unit step response coefficient
$\boldsymbol{A}$	=	dynamic matrix
d	=	prediction error
e	=	predicted error
$\overline{\boldsymbol{e}}$	=	vector of predicted errors
I	=	index
$\mathbf{I}$	=	identity matrix
j	=	index of sampling intervals
k	=	discrete dead time
K_p	=	process gain
M	=	control horizon (number of controller output moves)
n	=	current sampling interval

N	=	model horizon (process settling time in samples)
P	=	prediction horizon
T	=	sampling interval
u	=	controller output
y	=	process variable
y_{ss}	=	initial steady state of process variable
$\hat{y}$	=	predicted process variable
y_{sp}	=	process variable set point

Greek Symbols

Δu_i	=	change in controller output at the ith sampling interval
θ_p	=	effective dead time of process
λ	=	move suppression coefficient (controller output weight)
τ_p	=	overall process time constant

23.7 Tuning Strategy for Single Loop DMC

1. Approximate the manipulated-to-measured-process-variable dynamics with a first order plus dead time (FOPDT) model:

$$\tau_p \frac{dy(t)}{dt} + y(t) = K_p u(t - \theta_p) \quad \text{or} \quad \frac{y(s)}{u(s)} = \frac{K_p e^{-\theta_p s}}{\tau_p s + 1}$$

2. Select the sampling interval as close as possible to:

$$T = 0.1\,\tau_p \quad \text{or} \quad T = 0.5\,\theta_p \quad \text{whichever is larger}$$

3. Compute the prediction horizon, P, and the model horizon, N, as the process settling time in samples (rounded to the next integer):

$$P = N = \frac{5\tau_p}{T} + k \qquad \text{where:} \quad k = \frac{\tau_p}{T} + 1$$

4. Select the control horizon, M, as the time in samples required for the open loop response to reach 60% of the steady state:

$$M = \frac{\tau_p}{T} + k$$

5. Compute the move suppression coefficient:

$$\lambda = \begin{cases} 0 & M = 1 \\ \dfrac{M}{10}\left(\dfrac{3.5\,\tau_p}{T} + 2 - \dfrac{(M-1)}{2}\right) K_p^2 & M > 1 \end{cases}$$

6. Implement DMC using the traditional step response matrix of the actual process and the above parameters.

25. Non-Self Regulating (Integrating) Processes

25.1 Open Loop Behavior of Integrating Processes

Up to this point we have explored only self regulating processes. Self regulating processes naturally seek a steady state operating level if the manipulated and disturbance variables are held constant for a sufficient period of time. The left plot of Fig. 25.1 shows the ideal open loop behavior of a self regulating process. Here the controller output (CO) and process variable (PV) are initially at steady state. The CO is stepped up and back from this steady state while the disturbances (not shown) remain quiet. The PV responds to the step, but ultimately returns to its original operating level. This is a key characteristic of a self regulating process.

Some processes are non-self regulating and these are commonly referred to as *integrating processes*. Non-self regulating (integrating) processes are very challenging to control. After studying this chapter, you may come to realize that some of the level, temperature, pressure, pH and other loops you work with are integrating in nature.

The plot to the right in Fig. 25.1 shows the ideal open loop behavior of an integrating process. As shown, the PV steadies at a different operating level when the CO returns to its original value after the step. If the CO had not been stepped back to its original value (not shown), the PV would not settle at a new steady state, but rather, would continue to drift upward to extreme and possibly dangerous levels. Changes in any of the disturbance variables can also cause the PV to drift. Hence, *integrating processes are rarely operated in open loop* for very long.

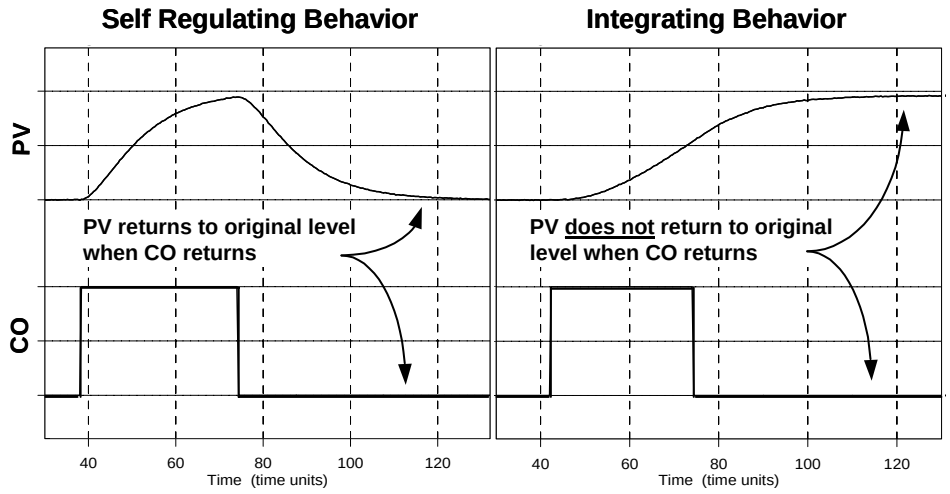


Figure 25.1 – Open loop behavior of self regulating (left) and integrating (right) processes

While the methodology for tuning a PID controller is the same for both self regulating and integrating processes, the specifics are quite different. As we have learned in previous chapters, the general controller tuning method for all processes is comprised of these steps:

- Collect dynamic process test data near the design level of operation
- Fit a linear dynamic model to the process test data
- Use the parameters from the model fit in correlations to obtain controller tuning values
- Test the tuning online and fine tune if necessary

Control Station's Pumped Tank *Case Study* is an integrating process. As shown in Fig. 25.2, the measured process variable is liquid level. To maintain level, the controller output manipulates flow rate out of the bottom of the tank by adjusting a throttling valve at the discharge of a constant pressure pump. The disturbance variable is the flow rate of a secondary feed to the tank.

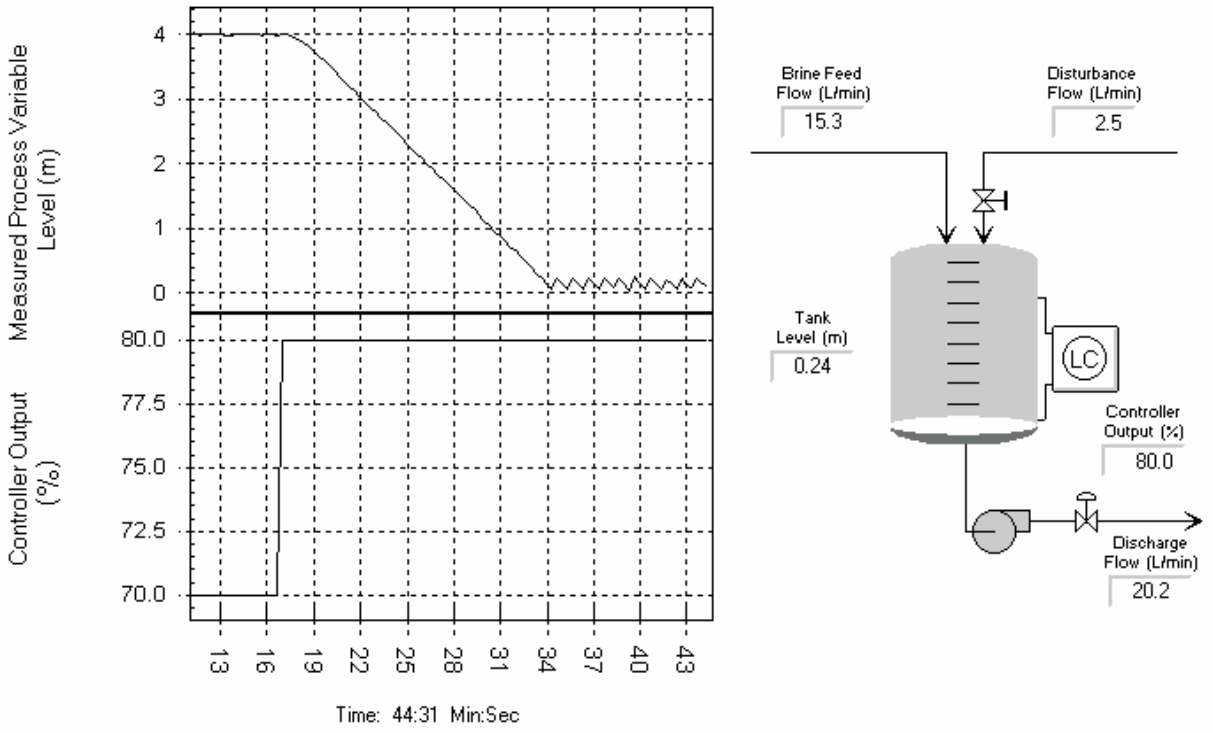


Figure 25.2 – Pumped Tank process

When the Pumped Tank controller output signal is increased from 70% to 80% as shown in Fig 25.2, the discharge flow rate out of the bottom of the tank increases. The total feed flow rate to the tank remains unchanged, however, and since flow out becomes greater than flow in, the tank level begins to fall. As the situation persists, liquid level continues to fall until the tank is drained.

Conversely, if the controller output were to be decreased enough to cause flow rate out to be less than flow rate in, the tank level would rise until full. If this were a real process, the tank would overflow and spill, creating safety and profitability issues. As we will learn in the next sections, this non-self regulating behavior makes the control of integrating processes quite challenging.

25.2 The FOPDT Integrating Model

The familiar first order plus dead time dynamic model does a good job of describing self regulating process behavior but it is not able to reasonably model the behavior of integrating processes. For controller tuning, integrating process behavior is best described with the FOPDT Integrating model form:

$$\frac{dy(t)}{dt} = K_P^* u\left(t - \theta_P\right) \tag{25.1}$$

Individual values for the familiar process gain, K_P, and process time constant, τ_P, cannot be identified for the FOPDT Integrating model. Instead, an integrator gain, K_P^*, is defined that has units of the ratio of the process gain to the process time constant, or:

$$K_P^* = \frac{K_P}{\tau_P} \qquad \text{where} \quad K_P^* \ [=] \ y(t)/[u(t) * \text{time}] \tag{25.2}$$

Analogous to our FOPDT investigations, the FOPDT Integrating model parameters K_P^* and θ_P in Eq. 25.1 can be determined both graphically and using Control Station's *Design Tools.*

25.3 Modeling Data From Integrating Processes

The graphical method of fitting a FOPDT Integrating model to process data requires a data set that includes at least two constant values of controller output, u_1 and u_2. Both u_1 and u_2 must be held constant long enough such that the slope of the measured process variable response trend can be visually identified in the data.

The FOPDT Integrating model describes the process behavior at each value of constant controller output u_1 and u_2 as:

$$\left.\frac{dy(t)}{dt}\right|_1 = K_P^* \, u_1 \left(t - \theta_P\right) \tag{25.3}$$

and

$$\left.\frac{dy(t)}{dt}\right|_2 = K_P^* \, u_2 \left(t - \theta_P\right) \tag{25.4}$$

Subtracting Eq. 25.3 from Eq. 25.4 and solving for K_P^* yields:

$$K_P^* = \frac{\left.\frac{dy(t)}{dt}\right|_2 - \left.\frac{dy(t)}{dt}\right|_1}{u_2 - u_1} = \frac{\text{slope}_2 - \text{slope}_1}{u_2 - u_1} \tag{25.5}$$

The dead time, θ_P, is then estimated from the plot using the same method described in Chapter 3.

Example 1: Graphical Estimation of Model Parameters for Pumped Tank Process

Figure 25.3 shows open loop data collected from the Pumped Tank case study of Fig. 25.2. The controller output is stepped from 68% up to 80%, causing the liquid level to fall. The controller output is then stepped from 80% down to 65%, causing an upward slope in the liquid level.

The slope of each segment is calculated as the change in liquid level divided by the change in time. From the data in Fig 25.3 we compute:

$$\text{slope}_1 = \left.\frac{dy(t)}{dt}\right|_1 = \frac{\Delta PV_1}{\Delta t_1} = \frac{3.1 - 4.1}{20 - 16} = -0.25 \ \frac{\text{m}}{\text{min}} \tag{25.6}$$

and

$$\text{slope}_2 = \left.\frac{dy(t)}{dt}\right|_2 = \frac{\Delta PV_2}{\Delta t_2} = \frac{3.4 - 3.0}{30 - 26} = 0.10 \ \frac{\text{m}}{\text{min}} \tag{25.7}$$

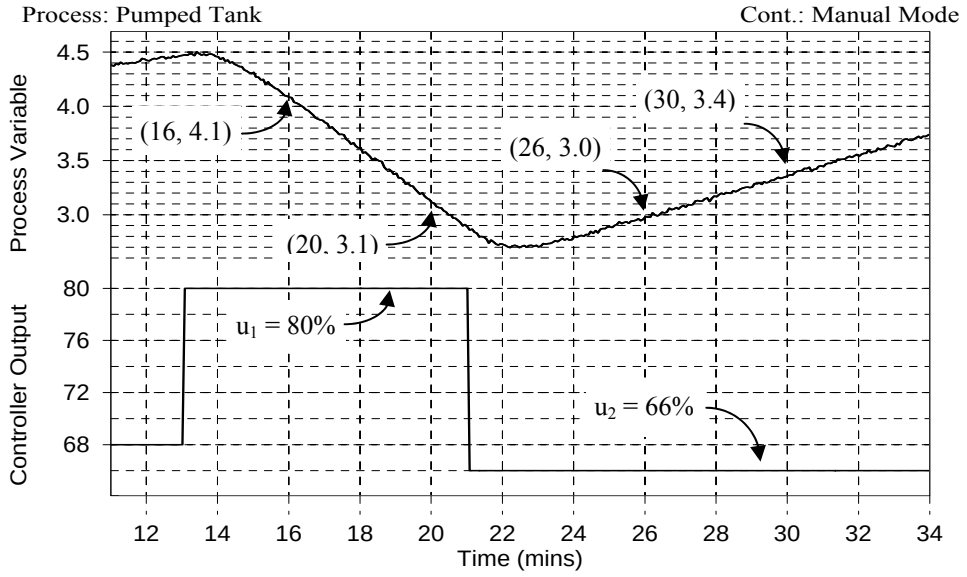


Figure 25.3 – Open loop data for Pumped Tank processes

Using the two slopes computed in Eq. 25.6 and 25.7 along with their respective controller output values in Eq. 25.3 yields the integrator gain, K_P^*, for the pumped tank process:

$$K_P^* = \frac{\text{slope}_2 - \text{slope}_1}{u_2 - u_1} = \frac{0.10 - (-0.25)}{66 - 80} = -0.025 \ \frac{\text{m}}{\% \cdot \text{min}}$$

The process dead time is estimated using the same method outlined in Chapter 3. That is, θ_P is computed as the difference in time from when the controller output signal was stepped (t_{Ustep}) and the time when the measured process variable starts showing a clear response to that change (t_{Ystart}). From the plot we estimate:

$$\theta_P = t_{Ystart} - t_{Ustep} = 22 - 21 = 1.0 \text{ min}$$

Substituting these model parameters into Eq. 25.1 yields the FOPDT Integrating dynamic model describing the pumped tank process behavior at this particular level of operation:

$$\boxed{\frac{dy(t)}{dt} = -0.025u\left(t - 1.0\right)}$$

☆ ☆ ☆

Design Tools offers two tools that automate the FOPDT Integrating model fit. The choice of which tool to use depends on the nature of the process data you have collected. These should be either:

- process data that starts at an initial steady state, or
- process data that has two periods of constant controller output

A process is considered to be at steady state if both the controller output signal and measured process variable are substantially constant for a period of time before the collection of dynamic data

begins. For integrating processes, this situation might occur if the process is already under feedback control and the disturbances are quiet. With data that starts at steady state, *Design Tools* can conduct an automated model parameter search like those we have conducted numerous times throughout the book using the FOPDT model.

If the process data does not start at steady state, as might be expected with open loop data, then the data must include at least two periods where the controller output signal is held constant long enough such that the slope of the measured process variable response is clearly evident in the data. For such data, *Design Tools* offers assistance in performing the graphical analysis like that conducted in example 1.

Example 2: Using *Design Tools'* Graphical Analysis to Determine Model Parameters
The use of the graphical analysis tool to fit an FOPDT Integrating model for the same open loop data used in Example 1 is shown in Fig. 25.4. After selecting the FOPDT Integrating model in *Design Tools* and then specifying that the process data has two periods of constant controller output, a plot is presented along with adjustable nodes, slopes and tie lines.

Two controller output (CO) nodes are displayed that should be clicked and dragged to match the two values of constant controller output expected in the data. Both CO nodes have tie lines to identify their associated measured process variable (PV) slope bar. Each slope bar has two end point nodes. As shown, these should be clicked and dragged so that each bar approximates the corresponding sloped segment on the graph.

With the six nodes (two CO and four PV) properly positioned, the FOPDT Integrating model parameters K_P^* and θ_P result. The PV slope bars may need to be tweaked up or down until the computed model overlays the data to your satisfaction.

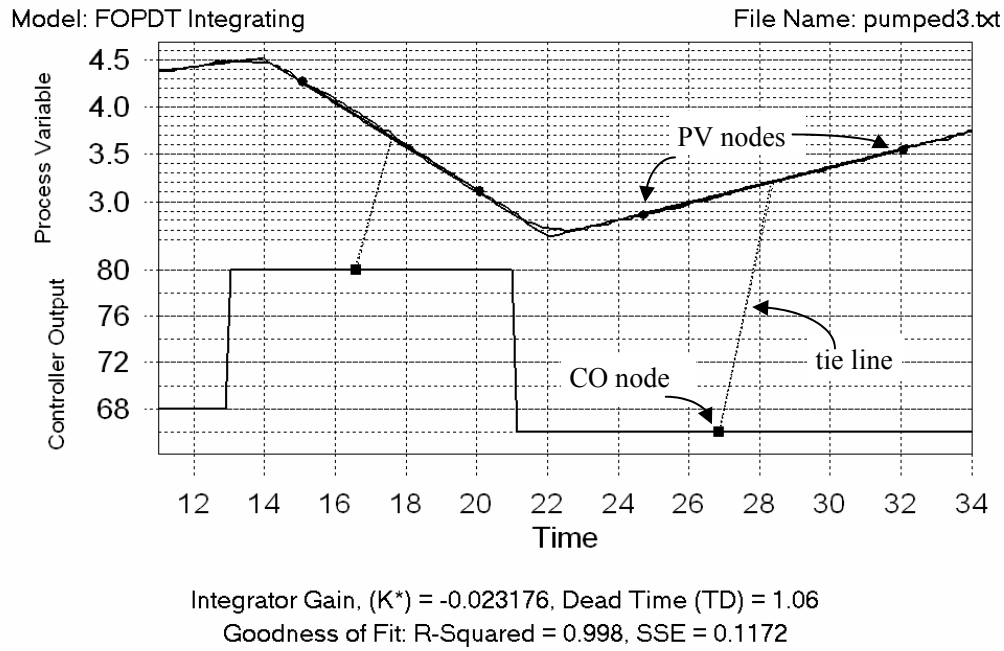


Figure 25.4 – Using the graphical analysis tool to fit a FOPDT Integrating dynamic model

As shown in Fig 25.4, with the nodes properly positioned over the process data, the graphical analysis tool yields FOPDT Integrating model parameters that are satisfactorily close to those computed in example 1:

$$K_P^* = -0.023 \ \frac{\text{m}}{\% \cdot \text{min}} \qquad \theta_P = 1.2 \text{ min}$$

☆ ☆ ☆

Example 3: Using *Design Tools'* Automated FOPDT Integrating Model Fit

When process data starts at steady state, such as that shown in Fig 25.5 for the pumped tank process, then the most convenient modeling tool is the automated model fit like we have used in previous chapters. The data in Fig 25.5 was collected with the disturbance flow rate quiet (unchanging) and the process under P-Only control. A set point step initiated the dynamic event and the data below was collected during the closed-loop response.

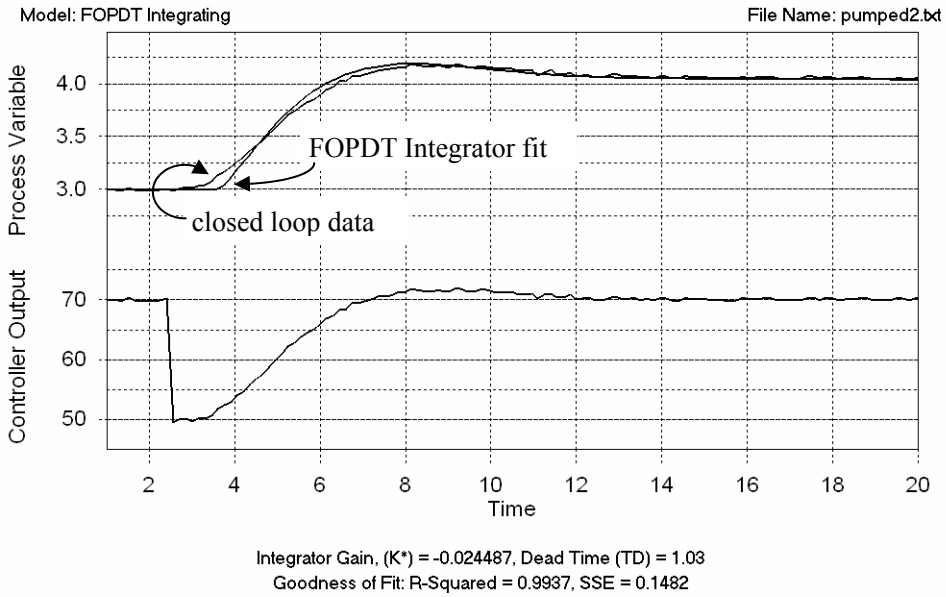


Figure 25.5 – Using the Graphical Integrator tool to determine model parameters

The *Design Tools* automated fit yields the FOPDT Integrating model parameters:

$$K_P^* = -0.024 \ \frac{\text{m}}{\% \cdot \text{min}} \qquad \theta_P = 1.0 \text{ min}$$

These are again consistent with the parameter values computed in the previous two examples.

☆ ☆ ☆

25.4 Designing and Implementing Controllers for Integrating Processes

Using the controller tuning guide for integrating processes found in Appendix D.2, we can determine the parameters for any type of controller we wish to design.

Example 5: Use IMC Tuning Correlations to Design a PI Controller for an Unstable Process
We start with the model parameters obtained by *Design Tools* in Example 2:

$$K_P^* = -0.023 \ \frac{\text{m}}{\% \cdot \text{min}} \qquad\qquad \theta_P = 1.2 \text{ min}$$

We first find the value of τ_C from the dead time, using Standard Tuning (see Appendix D.2):

$$\tau_C = \theta_P \sqrt{10} = 1.2\sqrt{10} = 3.8 \text{ min}$$

Next, we can substitute K_P^*, θ_P, and τ_C into Eq. 25.6 to obtain the controller tuning values:

$$K_C = \frac{1}{K_p^*} \frac{2\tau_C + \theta_P}{(\theta_P + \tau_C)^2} = \frac{1}{-0.023}\left[\frac{2(3.83)+1.2}{(1.2+3.8)^2}\right] = -15.3 \frac{\%}{\text{m}}$$

$$\tau_I = 2\tau_C + \theta_P = 2(3.8) + 1.2 = 8.9 \text{ min}$$

Because *Design Tools* also uses the IMC-derived correlations to determine the tuning parameters, these values should match the ones given by the program.

☆ ☆ ☆

Interaction of Tuning Parameters
The two PI tuning parameters (K_C and τ_I) interact with each other, meaning that if one is changed, the response of the other may also be different. Figure 25.5 shows the base case performance, where the actual controller gain and reset time computed by the tuning correlations are used. The effect on controller performance when K_C is changed is then presented.

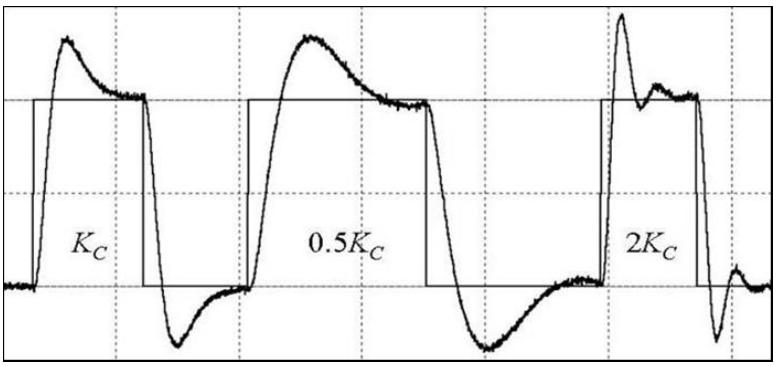


Figure 25.5 – step response performance while varying K_C for integrating process

While the first step response shows performance you may like to improve, the subsequent steps show that performance degrades further by adjusting controller gain. Integrating process are indeed a control challenge.

Appendix A: Derivation of IMC Tuning Correlations

A.1 Self Regulating Processes

For a general discussion of the derivation methodology and the details of the PI tuning correlation derivation, please see Chap. 17, "Deriving PID Controller Tuning Correlations."

A.1.a Ideal PID Control

Start with the general FOPDT process model:

$$G_P^*(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_P^*(s) = \frac{K_P\left(1 - 0.5\theta_P s\right)}{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s) G_{P-}^*(s)$$

so

$$G_{P+}^*(s) = \left(1 - 0.5\theta_P s\right)$$

and

$$G_{P-}^*(s) = \frac{K_P}{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}$$

Now express the IMC controller, $G_C^*(s)$, in terms of $G_{P-}^*(s)$ and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)} F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C:

$$F(s) = \frac{1}{\tau_C s + 1}$$

Substitute $F(s)$ and $G_{P-}^*(s)$ to express the IMC controller as:

$$G_C^*(s) = \left(\frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P} \right) \left(\frac{1}{\tau_C s + 1} \right) = \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C s + 1)}$$

Relate this IMC controller, $G_C^*(s)$, to a classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G_C^*(s)}{1 - G_P^*(s)G_C^*(s)}$$

Substitute the above equations for $G_C^*(s)$ and $G_P^*(s)$ and simplify:

$$G_C(s) = \frac{\dfrac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C s + 1)}}{1 - \left(\dfrac{\cancel{(\tau_P s + 1)(1 + 0.5\theta_P s)}}{\cancel{K_P}(\tau_C s + 1)} \right) \left(\dfrac{\cancel{K_P}(1 - 0.5\theta_P s)}{\cancel{(\tau_P s + 1)(1 + 0.5\theta_P s)}} \right)}$$

$$= \frac{\dfrac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C s + 1)}}{1 - \dfrac{(1 - 0.5\theta_P s)}{(\tau_C s + 1)}}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C s + 1 - 1 + 0.5\theta_P s)}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C s + 0.5\theta_P s)}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P(\tau_C + 0.5\theta_P)s}$$

Factor this into a form we can compare with the ideal PID controller form:

$$G_C(s) = \frac{\tau_P s + 1 + 0.5\theta_P \tau_P s^2 + 0.5\theta_P s}{K_P(\tau_C + 0.5\theta_P)s}$$

$$= \frac{(\tau_P + 0.5\theta_P)s + 1 + 0.5\theta_P \tau_P s^2}{K_P(\tau_C + 0.5\theta_P)s}$$

$$G_C(s) = \frac{\left(\tau_P + 0.5\theta_P\right)}{K_P\left(\tau_C + 0.5\theta_P\right)}\left(1 + \frac{1}{\left(\tau_P + 0.5\theta_P\right)s} + \frac{0.5\theta_P\tau_P}{\left(\tau_P + 0.5\theta_P\right)}s\right)$$

Compare this to the classical feedback form for an ideal PID controller:

$$G_C(s)_{\text{PID Ideal}} = K_C\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{\left(\tau_P + 0.5\theta_P\right)}{K_P\left(\tau_C + 0.5\theta_P\right)}, \quad \tau_I = \left(\tau_P + 0.5\theta_P\right), \text{ and } \quad \tau_D = \frac{\tau_P\theta_P}{2\left(\tau_P + 0.5\theta_P\right)}$$

A.1.b Interacting PID Control

Start with the general FOPDT process model:

$$G_P^*(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_P^*(s) = \frac{K_P\left(1 - 0.5\theta_P s\right)}{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so

$$G_{P+}^*(s) = \left(1 - 0.5\theta_P s\right)$$

and

$$G_{P-}^*(s) = \frac{K_P}{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}$$

Now express the IMC controller, $G_C^*(s)$, in terms of $G_{P-}^*(s)$ and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C:

$$F(s) = \frac{1}{\tau_C s + 1}$$

Substitute $F(s)$ and $G^*_{P\_}(s)$ to express the IMC controller as:

$$G^*_C(s) = \left(\frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P} \right) \left(\frac{1}{\tau_C s + 1} \right) = \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C s + 1)}$$

Relate this IMC controller, $G^*_C(s)$, to the classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G^*_C(s)}{1 - G^*_P(s) G^*_C(s)}$$

Substitute the above equations for $G^*_C(s)$ and $G^*_P(s)$ and simplify:

$$G_C(s) = \frac{\dfrac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C s + 1)}}{1 - \left(\dfrac{\cancel{(\tau_P s + 1)(1 + 0.5\theta_P s)}}{\cancel{K_P}(\tau_C s + 1)} \right) \left(\dfrac{\cancel{K_P}(1 - 0.5\theta_P s)}{\cancel{(\tau_P s + 1)(1 + 0.5\theta_P s)}} \right)}$$

$$= \frac{\dfrac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C s + 1)}}{1 - \dfrac{(1 - 0.5\theta_P s)}{(\tau_C s + 1)}}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C s + 1 - 1 + 0.5\theta_P s)}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C s + 0.5\theta_P s)}$$

$$= \frac{(\tau_P s + 1)(1 + 0.5\theta_P s)}{K_P (\tau_C + 0.5\theta_P) s}$$

Factor this into a form we can compare with the interacting PID form:

$$G_C = \frac{\tau_P}{K_P\left(\tau_C + 0.5\theta_P\right)}\left(1+\frac{1}{\tau_P s}\right)\left(0.5\theta_P s + 1\right)$$

Compare this to the classical feedback form for an interacting PID controller,

$$G_C(s)_{\text{PID Interact}} = K_C\left(1+\frac{1}{\tau_I s}\right)\left(\tau_D s + 1\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{\tau_P}{K_P\left(\tau_C + 0.5\theta_P\right)}, \quad \tau_I = \tau_P \text{ , and } \quad \tau_D = 0.5\theta_P$$

A.1.c Ideal PID with Filter Control

Start with the general FOPDT process model:

$$G_P^*(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so $$G_{P+}^*(s) = e^{-\theta_P s}$$

and $$G_{P-}^*(s) = \frac{K_P}{\tau_P s + 1}$$

Express the IMC controller model, $G_C^*(s)$, in terms of $G_{P-}^*(s)$ and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C:

$$F(s) = \frac{1}{\tau_C s + 1}$$

Substitute $F(s)$ and $G_{P-}^*(s)$ to express the IMC controller as:

$$G_C^*(s) = \left(\frac{\tau_P s + 1}{K_P}\right)\left(\frac{1}{\tau_C s + 1}\right) = \frac{\tau_P s + 1}{K_P\left(\tau_C s + 1\right)}$$

Relate this IMC controller, $G_C^*(s)$, to a classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G_C^*(s)}{1 - G_P^*(s)G_C^*(s)}$$

Substitute the above equations for $G_C^*(s)$ and $G_P^*(s)$ and simplify:

$$G_C(s) = \frac{\dfrac{\left(\tau_P s + 1\right)}{K_P\left(\tau_C s + 1\right)}}{1 - \left(\dfrac{K_P e^{-\theta_P s}}{\left(\tau_P s + 1\right)}\right)\left(\dfrac{\left(\tau_P s + 1\right)}{K_P\left(\tau_C s + 1\right)}\right)}$$

$$= \frac{\dfrac{\left(\tau_P s + 1\right)}{K_P\left(\tau_C s + 1\right)}}{1 - \dfrac{e^{-\theta_P s}}{\left(\tau_C s + 1\right)}}$$

$$= \frac{\tau_P s + 1}{K_P\left(\tau_C s + 1 - e^{-\theta_P s}\right)}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_C(s) = \frac{\tau_P s + 1}{K_P\left(\tau_C s + 1 - \dfrac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}\right)}$$

$$= \frac{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C s + 0.5\theta_P \tau_C s^2 + 1 + 0.5\theta_P s - 1 + 0.5\theta_P s\right)}$$

$$= \frac{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C s + 0.5\theta_P \tau_C s^2 + \left(\tau_C + \theta_P\right)s\right)}$$

Factor into a form we can compare to the ideal PID with filter form:

$$G_C(s) = \frac{\left(1 + 0.5\theta_P s + \tau_P s + 0.5\theta_P \tau_P s^2\right)}{K_P\left(\tau_C + \theta_P\right)s\left(\frac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1\right)}$$

$$= \frac{\left[\left(\tau_P + 0.5\theta_P\right)s + 1 + 0.5\theta_P \tau_P s^2\right]}{K_P\left(\tau_C + \theta_P\right)s\left(\frac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1\right)}$$

$$G_C(s) = \frac{\left(\tau_P + 0.5\theta_P\right)}{K_P\left(\tau_C + \theta_P\right)}\left(1 + \frac{1}{\left(\tau_P + 0.5\theta_P\right)s} + \frac{\tau_P \theta_P}{2\left(\tau_P + 0.5\theta_P\right)}s\right)\frac{1}{\left(\frac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1\right)}$$

Compare this to the classical feedback form for an PID with filter controller:

$$G_C(s)_{\text{PID Ideal, Filter}} = K_C\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)\left(\frac{1}{\alpha\tau_D s + 1}\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{\left(\tau_P + 0.5\theta_P\right)}{K_P\left(\tau_C + \theta_P\right)} \;,\; \tau_I = \tau_P + 0.5\theta_P \;,\; \tau_D = \frac{\tau_P \theta_P}{2\left(\tau_P + 0.5\theta_P\right)} \text{, and } \alpha = \frac{\tau_C\left(\tau_P + 0.5\theta_P\right)}{\tau_P\left(\tau_C + \theta_P\right)}$$

A.1.d Interacting PID with Filter Control

Start with the general FOPDT process model:

$$G_P^*(s) = \frac{K_P e^{-\theta_P s}}{\tau_P s + 1}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so $$G_{P+}^*(s) = e^{-\theta_P s}$$

and $$G_{P-}^*(s) = \frac{K_P}{\tau_P s + 1}$$

Now express the IMC controller, $G_C^*(s)$, in terms of $G_{P-}^*(s)$ and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)} F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C:

$$F(s) = \frac{1}{\tau_C s + 1}$$

Substitute $F(s)$ and $G_{P-}^*(s)$ to express the controller model as:

$$G_C^*(s) = \left(\frac{\tau_P s + 1}{K_P} \right) \left(\frac{1}{\tau_C s + 1} \right) = \frac{\tau_P s + 1}{K_P \left(\tau_C s + 1 \right)}$$

Relate this IMC controller, $G_C^*(s)$, to the classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G_C^*(s)}{1 - G_P^*(s) G_C^*(s)}$$

Substitute equations for $G_C^*(s)$ and $G_P^*(s)$ and simplify:

$$G_C(s) = \frac{\dfrac{\left(\tau_P s + 1 \right)}{K_P \left(\tau_C s + 1 \right)}}{1 - \left(\dfrac{K_P e^{-\theta_P s}}{\left(\tau_P s + 1 \right)} \right) \left(\dfrac{\left(\tau_P s + 1 \right)}{K_P \left(\tau_C s + 1 \right)} \right)}$$

$$= \frac{\dfrac{\left(\tau_P s + 1 \right)}{K_P \left(\tau_C s + 1 \right)}}{1 - \dfrac{e^{-\theta_P s}}{\left(\tau_C s + 1 \right)}}$$

$$= \frac{\tau_P s + 1}{K_P \left(\tau_C s + 1 - e^{-\theta_P s} \right)}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_C(s) = \frac{\tau_P s + 1}{K_P\left(\tau_C s + 1 - \dfrac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}\right)}$$

$$= \frac{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C s + 0.5\theta_P \tau_C s^2 + 1 + 0.5\theta_P s - 1 + 0.5\theta_P s\right)}$$

$$= \frac{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C s + 0.5\theta_P \tau_C s^2 + \left(\tau_C + \theta_P\right)s\right)}$$

Now factor this in such a way that we can compare it to the interacting PID with filter form:

$$G_C(s) = \frac{\left(\tau_P s + 1\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C + \theta_P\right)s\left(\dfrac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1\right)}$$

$$= \frac{\tau_P\left(1 + \dfrac{1}{\tau_P s}\right)\left(1 + 0.5\theta_P s\right)}{K_P\left(\tau_C + \theta_P\right)\left(\dfrac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1\right)}$$

$$= \frac{\tau_P}{K_P\left(\tau_C + \theta_P\right)}\left(1 + \frac{1}{\tau_P s}\right)\left(\frac{0.5\theta_P s + 1}{\dfrac{\tau_C \theta_P}{2\left(\tau_C + \theta_P\right)}s + 1}\right)$$

Comparing this with the classical feedback model for interacting PID with filter control,

$$G_C(s)_{\text{PID Interact, Filter}} = K_C\left(1 + \frac{1}{\tau_I s}\right)\left(\frac{\tau_D s + 1}{\alpha\tau_D s + 1}\right)$$

we obtain the following controller tuning parameters:

$$K_C = \frac{\tau_P}{K_P\left(\tau_C + \theta_P\right)}\ ,\ \ \tau_I = \tau_P\ ,\ \ \tau_D = 0.5\theta_P\ \text{, and}\ \ \alpha = \frac{\tau_C}{\left(\tau_C + \theta_P\right)}$$

Note: The derivation of PI tuning correlations for self-regulating processes can be found in Chap 17.

A.2 Non-Self Regulating Processes

A.2.a Ideal PID Control

Start with the general FOPDT process model for integrating processes:

$$G_P^*(s) = \frac{K_P^* e^{-\theta_P s}}{s}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_P^*(s) = \frac{K_P^*\left(1 - 0.5\theta_P s\right)}{s\left(1 + 0.5\theta_P s\right)}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s) G_{P-}^*(s)$$

so

$$G_{P+}^*(s) = 1 - 0.5\theta_P s$$

and

$$G_{P-}^*(s) = \frac{K_P^*}{s\left(1 + 0.5\theta_P\right)}$$

Now express the IMC controller, $G_C^*(s)$, in terms of the invertible terms, $G_{P-}^*(s)$, and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)} F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C, and the noninvertible process model, $G_{P+}^*(s)$, as:

$$F(s) = \frac{\left(2\tau_C - G_{P+}^{*\,'}(0)\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

and since

$$G_{P+}^{*\,'}(0) = \frac{d\left(G_{P+}^*\right)}{ds}(0) = -0.5\theta_P$$

then
$$F(s) = \frac{\left(2\tau_C + 0.5\theta_P\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

Substitute $F(s)$ and $G^*_{P-}(s)$ to express the IMC controller as:

$$G^*_C(s) = \frac{s\left[1 + 0.5\theta_P s\right]\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K^*_P\left(\tau_C s + 1\right)^2}$$

Relate this IMC controller, $G^*_C(s)$, to a classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G^*_C(s)}{1 - G^*_P(s)G^*_C(s)}$$

Substitute the above equations for $G^*_C(s)$ and $G^*_P(s)$:

$$G_C(s) = \frac{s\left[1 + 0.5\theta_P s\right]\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K^*_P\left[\left(\tau_C s + 1\right)^2 - \left(1 - 0.5\theta_P s\right)\left(1 + s\left(0.5\theta_P + 2\tau_C\right)\right)\right]}$$

Factor this into a form we can compare with the Ideal PID controller form:

$$G_C(s) = \frac{s\left[1 + 0.5\theta_P s\right]\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K^*_P\left[1 + 2\tau_C s + {\tau_C}^2 s^2 - \left(1 + 0.5\theta_P s + 2\tau_C s - 0.5\theta_P s - 0.25\theta_P^2 s^2 - \theta_P \tau_C s^2\right)\right]}$$

$$= \frac{s\left(0.5\theta_P s + 1 + 0.25\theta_P^2 s^2 + 0.5\theta_P s + \tau_C \theta_P s^2 + 2\tau_C s\right)}{K^*_P\left[s^2\left(\tau_C + 0.5\theta_P\right)^2\right]}$$

$$= \frac{s\left[1 + \left(2\tau_C + \theta_P\right)s + \left(\tau_C\theta_P + 0.25\theta_P^2\right)s^2\right]}{K^*_P\left[s^2\left(\tau_C + 0.5\theta_P\right)^2\right]}$$

$$= \frac{1}{K^*_P}\left[\frac{1}{\left(\tau_C + 0.5\theta_P\right)^2}\right]\left[\frac{1}{s} + \frac{\left(2\tau_C + \theta_P\right)s}{s} + \frac{\left(\tau_C\theta_P + 0.25\theta_P^2\right)s^2}{s}\right]$$

$$= \frac{1}{K^*_P}\left[\frac{2\tau_C + \theta_P}{\left(\tau_C + 0.5\theta_P\right)^2}\right]\left[1 + \frac{1}{\left(2\tau_C + \theta_P\right)s} + \frac{\left(\tau_C\theta_P + 0.25\theta_P^2\right)}{\left(2\tau_C + \theta_P\right)}s\right]$$

Compare this to the classical feedback form for an ideal PID controller,

$$G_C(s)_{\text{PID Ideal}} = K_C\left(1+\frac{1}{\tau_I s}+\tau_D s\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{1}{K_P^*}\frac{2\tau_C+\theta_P}{\left(\tau_C+0.5\theta_P\right)^2} \;,\quad \tau_I = 2\tau_C+\theta_P \;,\text{ and }\quad \tau_D = \frac{\left(\tau_C\theta_P+0.25\theta_P^2\right)}{\left(2\tau_C+\theta_P\right)}$$

A.2.b Interacting PID Control

Start with the general FOPDT process model for integrating processes:

$$G_P^*(s) = \frac{K_P^* e^{-\theta_P s}}{s}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1-0.5\theta_P s}{1+0.5\theta_P s}$$

so

$$G_P^*(s) = \frac{K_P^*\left(1-0.5\theta_P s\right)}{s\left(1+0.5\theta_P s\right)}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so

$$G_{P+}^*(s) = 1-0.5\theta_P s$$

and

$$G_{P-}^*(s) = \frac{K_P^*}{s\left(1+0.5\theta_P\right)}$$

Now express the IMC controller, $G_C^*(s)$, in terms of the invertible terms, $G_{P-}^*(s)$, and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C, and the noninvertible process model, $G_{P+}^*(s)$, as:

$$F(s) = \frac{\left(2\tau_C - G_{P+}^{*}{}'(0)\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

and since $$G_{P+}^{*}{}'(0) = \frac{d\left(G_{P+}^{*}\right)}{ds}(0) = -0.5\theta_P$$

then $$F(s) = \frac{\left(2\tau_C + 0.5\theta_P\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

Substitute $F(s)$ and $G_{P-}^{*}(s)$ to express the IMC controller as:

$$G_C^{*}(s) = \frac{s\left[1 + 0.5\theta_P s\right]\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}\left(\tau_C s + 1\right)^2}$$

Relate this IMC controller, $G_C^{*}(s)$, to a classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G_C^{*}(s)}{1 - G_P^{*}(s)G_C^{*}(s)}$$

Substitute the above equations for $G_C^{*}(s)$ and $G_P^{*}(s)$:

$$G_C(s) = \frac{s\left[1 + 0.5\theta_P s\right]\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}\left[\left(\tau_C s + 1\right)^2 + \left(1 - 0.5\theta_P s\right)\left(1 + s\left(0.5\theta_P + 2\tau_C\right)\right)\right]}$$

Factor this into a form we can compare with the Interacting PID controller form:

$$G_C(s) = \frac{s\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}\left[1 + 2\tau_C s + s^2{\tau_C}^2 - \left(1 - 0.5\theta_P s\right)\left(1 + 0.5\theta_P s + 2\tau_C s\right)\right]}\left[1 + 0.5\theta_P s\right]$$

$$= \frac{s\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}\left[1 + 2\tau_C s + s^2{\tau_C}^2 - \left(1 + 2\tau_C s + 0.25\theta_P^2 s^2 + \theta_P\tau_C s^2\right)\right]}\left[1 + 0.5\theta_P s\right]$$

$$= \frac{s\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}\left[s^2{\tau_C}^2 + 0.25\theta_P^2 s^2 + \theta_P\tau_C s^2\right]}\left[1 + 0.5\theta_P s\right]$$

$$= \frac{s\left[1 + s\left(0.5\theta_P + 2\tau_C\right)\right]}{K_P^{*}s^2\left[{\tau_C}^2 + 0.25\theta_P^2 + \theta_P\tau_C\right]}\left[1 + 0.5\theta_P s\right]$$

$$= \frac{1 + s\left(0.5\theta_P + 2\tau_C\right)}{K_P^* s\left[\tau_C{}^2 + 0.25\theta_P^2 + \theta_P\tau_C\right]}\left[1 + 0.5\theta_P s\right]$$

$$= \frac{1}{K_P^*}\left[\frac{\left(0.5\theta_P + 2\tau_C\right)}{\left(\tau_C + 0.5\theta_P\right)^2} + \frac{1}{\left(\tau_C + 0.5\theta_P\right)^2 s}\right]\left[1 + 0.5\theta_P s\right]$$

$$= \frac{1}{K_P^*}\frac{\left(0.5\theta_P + 2\tau_C\right)}{\left(\tau_C + 0.5\theta_P\right)^2}\left[1 + \frac{1}{\left(2\tau_C + 0.5\theta_P\right)s}\right]\left[1 + 0.5\theta_P s\right]$$

Compare this to the classical feedback form for an interacting PID controller,

$$G_C(s)_{\text{PID Interact}} = K_C\left(1 + \frac{1}{\tau_I s}\right)\left(\tau_D s + 1\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{1}{K_P^*}\frac{\left(2\tau_C + 0.5\theta_P\right)}{\left(\tau_C + 0.5\theta_P\right)^2}\ ,\quad \tau_I = 2\tau_C + 0.5\theta_P\ ,\ \text{and}\ \ \tau_D = 0.5\theta_P$$

A.2.c Ideal PID with Filter Control

Start with the general FOPDT process model for integrating processes:

$$G_P^*(s) = \frac{K_P^* e^{-\theta_P s}}{s}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so $$G_{P+}^*(s) = e^{-\theta_P s}$$

and $$G_{P-}^*(s) = \frac{K_P^*}{s}$$

Now express the IMC controller, $G_C^*(s)$, in terms of the invertible terms, $G_{P-}^*(s)$, and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C, and the noninvertible process model, $G^*_{P+}(s)$, as:

$$F(s) = \frac{\left(2\tau_C - G^*_{P+}{}'(0)\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

and since

$$G^*_{P+}{}'(0) = \frac{d\left(G^*_{P+}\right)}{ds}(0) = -\theta_P$$

then

$$F(s) = \frac{\left(2\tau_C + \theta_P\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

Substitute $F(s)$ and $G^*_{P-}(s)$ to express the IMC controller as:

$$G^*_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K^*_P\left(\tau_C s + 1\right)^2}$$

Relate this IMC controller, $G^*_C(s)$, to a classical feedback controller model, $G_C(s)$:

$$G_C(s) = \frac{G^*_C(s)}{1 - G^*_P(s)G^*_C(s)}$$

Substitute the above equations for $G^*_C(s)$ and $G^*_P(s)$:

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K^*_P\left(\tau_C s + 1\right)^2\left[1 - \left(\frac{\cancel{s}\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{\cancel{K^*_P}\left(\tau_C s + 1\right)^2}\right)\left(\frac{\cancel{K^*_P}e^{-\theta_P s}}{\cancel{s}}\right)\right]}$$

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K^*_P\left[\left(\tau_C s + 1\right)^2 - e^{-\theta_P s}\left(1 + s\left(2\tau_C + \theta_P\right)\right)\right]}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K^*_P\left[\left(\tau_C s + 1\right)^2 - \left(\frac{1 + 0.5\theta_P s}{1 - 0.5\theta_P s}\right)\left(1 + s\left(2\tau_C + \theta_P\right)\right)\right]}$$

Factor this into a form we can compare with the ideal PID controller with filter controller form:

$$G_C(s) = \frac{s\left[1+s\left(2\tau_C+\theta_P\right)\right]\left[1+0.5\theta_P s\right]}{K_P^*\left[\left(\tau_C s+1\right)^2\left(1+0.5\theta_P s\right)-\left(1+s\left(2\tau_C+\theta_P\right)\right)\left(1-0.5\theta_P s\right)\right]}$$

$$= \frac{2\tau_C s^2+\theta_P s^2+s+\tau_C\theta_P s^3+0.5\theta_P^2 s^3+0.5\theta_P s^2}{K_P^*\left[\tau_C^2 s^2+0.5\theta_P\tau_C^2 s^3+2\tau_C\theta_P s^2+0.5\theta_P^2 s^2\right]}$$

$$= \frac{\cancel{s^2}\left[2\tau_C+\theta_P+\frac{1}{s}+\tau_C\theta_P s+0.5\theta_P^2 s+0.5\theta_P\right]}{K_P^*\cancel{s^2}\left[\tau_C^2+0.5\theta_P\tau_C^2 s+2\tau_C\theta_P+0.5\theta_P^2\right]}$$

$$= \frac{\left[2\tau_C+1.5\theta_P\right]\left[1+\frac{1}{\left(2\tau_C+1.5\theta_P\right)s}+\frac{0.5\theta_P^2+\tau_C\theta_P}{\left(2\tau_C+1.5\theta_P\right)}s\right]}{K_P^*\left[\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2\right]\left[\frac{0.5\theta_P^2\tau_C}{\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2}s+1\right]}$$

$$G_C(s) = \frac{\left(2\tau_C+1.5\theta_P\right)}{K_P^*\left(\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2\right)}\left[1+\frac{1}{\left(2\tau_C+1.5\theta_P\right)s}+\frac{0.5\theta_P^2+\tau_C\theta_P}{\left(2\tau_C+1.5\theta_P\right)}s\right]\frac{1}{\left[\frac{0.5\theta_P^2\tau_C}{\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2}s+1\right]}$$

Compare this to the classical feedback model for an ideal PID controller with filter,

$$G_C(s)_{\text{PID Ideal, Filter}} = K_C\left(1+\frac{1}{\tau_I s}+\tau_D s\right)\left(\frac{1}{\alpha\tau_D s+1}\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{\left(2\tau_C+1.5\theta_P\right)}{K_P^*\left(\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2\right)},\quad \tau_I = 2\tau_C+1.5\theta_P,\quad \tau_D = \frac{0.5\theta_P^2+\tau_C\theta_P}{2\tau_C+1.5\theta_P},$$

$$\alpha = \frac{0.5\tau_C^2\left(2\tau_C+1.5\theta_P\right)}{\left(\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2\right)\left(0.5\theta_P+\tau_C\right)}$$

A.2.d Interacting PID with Filter Control

Start with the general FOPDT process model for integrating processes:

$$G_P^*(s) = \frac{K_P^* e^{-\theta_P s}}{s}$$

Factor $G_P^*(s)$ into invertible terms, $G_{P-}^*(s)$, and noninvertible terms, $G_{P+}^*(s)$. Recall that an invertible term will not yield positive poles (positive roots of the denominator of the transfer function indicating unstable behavior) when taken in the reciprocal:

$$G_P^*(s) = G_{P+}^*(s)G_{P-}^*(s)$$

so $$G_{P+}^*(s) = e^{-\theta_P s}$$

and $$G_{P-}^*(s) = \frac{K_P^*}{s}$$

Now express the IMC controller, $G_C^*(s)$, in terms of the invertible terms, $G_{P-}^*(s)$, and a first-order filter term, $F(s)$:

$$G_C^*(s) = \frac{1}{G_{P-}^*(s)}F(s)$$

The filter term can be expressed in terms of a closed-loop time constant, τ_C, and the noninvertible process model, $G_{P+}^*(s)$, as:

$$F(s) = \frac{\left(2\tau_C - G_{P+}^{*\,'}(0)\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

and since $$G_{P+}^{*\,'}(0) = \frac{d\left(G_{P+}^*\right)}{ds}(0) = -\theta_P$$

then $$F(s) = \frac{\left(2\tau_C + \theta_P\right)s + 1}{\left(\tau_C s + 1\right)^2}$$

Substitute $F(s)$ and $G_{P-}^*(s)$ to express the IMC controller as:

$$G_C^*(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K_P^*\left(\tau_C s + 1\right)^2}$$

Relate this IMC controller, $G_C^*(s)$, to a classical feedback controller, $G_C(s)$:

$$G_C(s) = \frac{G_C^*(s)}{1 - G_P^*(s)G_C^*(s)}$$

Substitute the above equations for $G_C^*(s)$ and $G_P^*(s)$:

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K_P^*\left[\left(\tau_C s + 1\right)^2 - e^{-\theta_P s}\left(1 + s\left(2\tau_C + \theta_P\right)\right)\right]}$$

Substitute the 1/1 Padé approximation for $e^{-\theta_P s}$:

$$e^{-\theta_P s} \approx \frac{1 - 0.5\theta_P s}{1 + 0.5\theta_P s}$$

so

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]}{K_P^*\left[\left(\tau_C s + 1\right)^2 - \left(\frac{1 + 0.5\theta_P s}{1 - 0.5\theta_P s}\right)\left(1 + s\left(2\tau_C + \theta_P\right)\right)\right]}$$

Factor this into a form we can compare with the interacting PID controller with filter controller form:

$$G_C(s) = \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]\left[1 + 0.5\theta_P s\right]}{K_P^*\left[\left(\tau_C s + 1\right)^2\left(1 + 0.5\theta_P s\right) - \left(1 + s\left(2\tau_C + \theta_P\right)\right)\left(1 - 0.5\theta_P s\right)\right]}$$

$$= \frac{s\left[1 + s\left(2\tau_C + \theta_P\right)\right]\left[1 + 0.5\theta_P s\right]}{K_P^*\left[s^2\tau_C^2 + 2\theta_P\tau_C s^2 + 0.5\theta_P s^3\tau_C^2 + 0.5\theta_P^2 s^2\right]}$$

$$= \frac{\left[1 + s\left(2\tau_C + \theta_P\right)\right]\left[1 + 0.5\theta_P s\right]}{K_P^*\left[s\tau_C^2 + 2\theta_P\tau_C s + 0.5\theta_P s^2\tau_C^2 + 0.5\theta_P^2 s\right]}$$

$$= \frac{\left[1 + s\left(2\tau_C + \theta_P\right)\right]\left[1 + 0.5\theta_P s\right]}{K_P^*\left[0.5\theta_P s^2\tau_C^2 + s\left(\tau_C^2 + 2\theta_P\tau_C + 0.5\theta_P^2\right)\right]}$$

$$= \frac{1}{K_P^*}\left[\frac{\left(1 + s\left(2\tau_C + \theta_P\right)\right)}{s\left(\tau_C^2 + 2\theta_P\tau_C + 0.5\theta_P^2\right)}\right]\left[\frac{\left(1 + 0.5\theta_P s\right)}{\left(\frac{0.5\theta_P\tau_C^2}{\left(\tau_C^2 + 2\theta_P\tau_C + 0.5\theta_P^2\right)}s\right) + 1}\right]$$

$$G_C(s) = \left[\frac{\left(2\tau_C + \theta_P\right)}{K_P^*\left(\tau_C^2 + 2\theta_P\tau_C + 0.5\theta_P^2\right)}\right]\left[1 + \frac{1}{\left(2\tau_C + \theta_P\right)s}\right]\left[\frac{0.5\theta_P s + 1}{\left(\frac{0.5\theta_P\tau_C^2}{\tau_C^2 + 2\theta_P\tau_C + 0.5\theta_P^2}\right)s + 1}\right]$$

Compare this to the classical feedback process model for an interacting PID controller with filter,

$$G_C(s)_{\text{PID Interact, Filter}} = K_C\left(1+\frac{1}{\tau_I s}\right)\left(\frac{\tau_D s+1}{\alpha\tau_D s+1}\right)$$

we obtain the following controller tuning correlations:

$$K_C = \frac{2\tau_C+\theta_P}{K_P^*\left(\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2\right)}\ ,\ \ \tau_I = 2\tau_C+\theta_P\ ,\ \tau_D = 0.5\theta_P\ ,\ \text{and}\ \ \alpha = \frac{\tau_C^2}{\tau_C^2+2\tau_C\theta_P+0.5\theta_P^2}$$

Appendix B: Table of Laplace Transforms

Time Domain	Laplace Domain
$f(t)$	$F(s)$
$af(t)$	$aF(s)$
$\delta(t)$ (impulse)	1
a (step)	$\frac{a}{s}$
at (ramp)	$\frac{a}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
$\sin(\omega t)$	$\frac{\omega}{s^2+\omega^2}$
$\cos(\omega t)$	$\frac{s}{s^2+\omega^2}$
e^{-at}	$\frac{1}{s+a}$
$\frac{t^{n-1}e^{-at}}{(n-1)!}$	$\frac{1}{(s+a)^n}$
$\sinh(\omega t)$	$\frac{\omega}{s^2-\omega^2}$
$\cosh(\omega t)$	$\frac{s}{s^2-\omega^2}$
$\frac{df(t)}{dt}$	$sF(s)-f(0)$
$\frac{d^2f(t)}{dt^2}$	$s^2F(s)-sf(0)-\left.\frac{df(t)}{dt}\right\|_{t=0}$
$\int_0^t f(t)dt$	$\frac{1}{s}F(s)$
$f(t-\theta)$	$e^{-\theta s}F(s)$

Appendix C: DMC Controller Tuning Guides

C.1 DMC Tuning Guide for *Self Regulating* (Stable) Processes

Fit a first order plus dead time (FOPDT) dynamic model to process data. "Process" is defined to include all dynamic information from the output signal of the controller through the measured response signal of the process variable.

Generate process data by forcing the measured process variable with a change in the controller output signal. For accurate results:

- the process must be at steady state before forcing a dynamic response; the first data point in the file must equal that steady state value
- the data collection sample rate should be ten times per time constant or faster ($T \le 0.1\,\tau_P$)
- the controller output should force the measured process variable to move at least ten times the noise band

Use *Design Tools* to fit a FOPDT dynamic model to the process data set. A FOPDT model has the form:

Time Domain: $\tau_P \dfrac{dy(t)}{dt} + y(t) = K_P\, u(t - \theta_P)$

Laplace Domain: $\dfrac{Y(s)}{U(s)} = \dfrac{K_P\, e^{-\theta_P s}}{\tau_P s + 1}$

where:

$y(t)$	=	measured process variable signal
$u(t)$	=	controller output signal
K_P	=	process gain; units of $y(t)/u(t)$
τ_P	=	process time constant; units of time
θ_P	=	process dead time; units of time

also:

T	=	DMC sample time; units of time
N	=	DMC model horizon; samples
P	=	DMC prediction horizon; samples
M	=	DMC control horizon; samples
λ	=	DMC move suppression coefficient; unitless

These correlations provide a starting point for tuning. Final tuning may require online trial and error. "Best" tuning is defined by you and your knowledge of the capabilities of the process, desires of management, goals of production, and impact on other processes.

DMC Tuning

1. Approximate the manipulated-to-measured-process-variable dynamics with a first order plus dead time (FOPDT) model shown above.
2. Select the sample time as close as possible to:
$$T = 0.1\,\tau_p \quad \text{or} \quad T = 0.5\,\theta_p \quad \text{whichever is larger}$$
3. Compute the prediction horizon, P, and the model horizon, N, as the process settling time in samples (rounded to the next integer):
$$P = N = \text{Int}\left(\frac{5\tau_p}{T}\right) + k \quad \text{where } k = \text{Int}\left(\frac{\theta_p}{T}\right) + 1$$
4. Select the control horizon, M, as the time in samples required for the open loop response to reach 60% of the steady state:
$$M = \text{Int}\left(\frac{\tau_p}{T}\right) + \text{k}$$
5. Compute the move suppression coefficient:
$$\lambda = \begin{cases} 0 & M = 1 \\ \dfrac{M}{10}\left(\dfrac{3.5\,\tau_p}{T} + 2 - \dfrac{(M-1)}{2}\right) K_p^2 & M > 1 \end{cases}$$
6. Implement DMC using the traditional step response matrix of the actual process and the above parameters.

C.2 DMC Tuning Guide for *Integrating* (Non-Self Regulating) Processes

Fitt a FOPDT Integrating dynamic model to process data. "Process" is defined to include all dynamic information from the output signal of the controller through the measured response signal of the process variable.

Integrating processes are unstable so the process should already be in closed loop. If not, stabilize the process with a simple P-Only controller and generate process data with a set point step. For accurate results:
- the process must be at steady state before forcing a dynamic response; the first data point recorded must equal that steady state value
- the controller output move from the set point step should force the process variable to move at least ten times the noise band

Use *Design Tools* to fit a FOPDT Integrating dynamic model to the process data set. A FOPDT Integrating model has the form:

Time Domain: $\frac{dy(t)}{dt} = K_P^* \, u(t - \theta_P)$ **Laplace Domain**: $\frac{Y(s)}{U(s)} = \frac{K_P^* \, e^{-\theta_P s}}{s}$

where:
- $y(t)$ = measured process variable signal
- $u(t)$ = controller output signal
- K_P^* = integral gain; units of $y(t)/(u(t) \cdot time)$
- θ_P = process dead time; units of time

also:
- T = DMC sample time; units of time
- N = DMC model horizon; samples
- P = DMC prediction horizon; samples
- M = DMC control horizon; samples
- λ = DMC move suppression coefficient; unitless

These correlations provide a starting point for tuning. Final tuning may require online trial and error. "Best" tuning is defined by you and your knowledge of the capabilities of the process, desires of management, goals of production, and impact on other processes.

DMC Tuning

1. Approximate the manipulated-to-measured-process-variable dynamics with a first order plus dead time integrating (FOPDT Integrating) model shown above.

2. Select the sample time as close as possible to:
$$T \leq \; 0.5\theta_p$$

3. The closed-loop time constant for a FOPDT integrating model can be approximated as:
$$\tau_{CL} = \theta_p \sqrt{10}$$

4. Compute the prediction horizon, P, and the model horizon, N, as the process settling time in samples (rounded to the next integer):
$$P = N = \text{Int}\left(\frac{5\tau_{CL}}{T}\right) + k \;\text{ where }\; k = \text{Int}\left(\frac{\theta_p}{T}\right) + 1$$

5. Select the control horizon, M, as the time in samples required for the open loop response to reach 60% of the steady state:
$$M = \text{Int}\left(\frac{\tau_{CL}}{T}\right) + k$$

6. Compute the move suppression coefficient:
$$\lambda = \begin{cases} 0 & \text{M=1} \\ \frac{1}{10M}\left(\frac{M^2(P-k+1)^3}{3} - 0.08M^3(P-k+1)^2\right) \left[K_p^* T\right]^2 & \text{M>1} \end{cases}$$

7. Implement DMC using the traditional step response matrix of the actual process and the above parameters.

Appendix D: PID Controller Tuning Guides

D.1 PID Tuning Guide for *Self Regulating* (Stable) Processes

Begin by fitting a first order plus dead time (FOPDT) dynamic model to process data. "Process" is defined to include all dynamic information from the output signal of the controller through the measured response signal of the process variable.

Generate process data by forcing the measured process variable with a change in the controller output signal. For accurate results:

- the process must be at steady state before forcing a dynamic response; the first data point recorded must equal that steady state value
- the data collection sample rate should be ten times per time constant or faster ($T \le 0.1\,\tau_P$)
- the controller output should force the measured process variable to move at least ten times the noise band

Use *Design Tools* to fit a FOPDT dynamic model to the process data set. A FOPDT model has the form:

Time Domain: $\tau_P \dfrac{dy(t)}{dt} + y(t) = K_P\, u(t - \theta_P)$

Laplace Domain: $\dfrac{Y(s)}{U(s)} = \dfrac{K_P\, e^{-\theta_P s}}{\tau_P s + 1}$

where:
- $y(t)$ = measured process variable signal
- $u(t)$ = controller output signal
- K_P = process gain; units of $y(t)/u(t)$
- τ_P = process time constant; units of time
- θ_P = process dead time; units of time

also:
- K_C = controller gain; units of $u(t)/y(t)$
- τ_I = controller reset time; units of time
- τ_D = controller derivative time; units of time
- α = derivative filter constant; unitless

Values of K_P, τ_P and θ_P that describe the dynamic behavior of your process are important because:

- they are used in correlations (listed below) to compute initial PID controller tuning values K_C, τ_I, τ_D and α
- the sign of K_P indicates the action of the controller ($+K_P \rightarrow$ reverse acting; $-K_P \rightarrow$ direct acting)
- the size of τ_P indicates the maximum desirable loop sample time (be sure sample time $T \le 0.1\,\tau_P$)
- the ratio θ_P / τ_P indicates whether a Smith predictor would show benefit (useful when $\theta_P \ge \tau_P$)
- the model itself is used in feed forward, Smith predictor, decoupling and other model-based controllers

These correlations provide an excellent start for tuning. Final tuning may require online trial and error. "Best" tuning is defined by you and your knowledge of the capabilities of the process, desires of management, goals of production, and impact on other processes.

IMC (lambda) Tuning

- Aggressive Tuning: τ_C is the larger of $0.1\,\tau_P$ or $0.8\,\theta_P$
- Moderate Tuning: τ_C is the larger of $1.0\,\tau_P$ or $8.0\,\theta_P$
- Conservative Tuning: τ_C is the larger of $10\,\tau_P$ or $80\,\theta_P$

* This is an ITAE correlation as no P-Only IMC exists

	K_C	τ_I	τ_D	α
P-Only*	$K_C = \dfrac{0.2}{K_P}(\tau_p/\theta_p)^{1.22}$			
PI	$\dfrac{1}{K_P}\dfrac{\tau_P}{(\theta_P + \tau_C)}$	τ_P		
PID Ideal	$\dfrac{1}{K_P}\left(\dfrac{\tau_P + 0.5\theta_P}{\tau_C + 0.5\theta_P}\right)$	$\tau_P + 0.5\theta_P$	$\dfrac{\tau_P\theta_P}{2\tau_P + \theta_P}$	
PID Interacting	$\dfrac{1}{K_P}\left(\dfrac{\tau_P}{\tau_C + 0.5\theta_P}\right)$	τ_P	$0.5\theta_P$	
PID Ideal w/filter	$\dfrac{1}{K_P}\left(\dfrac{\tau_P + 0.5\theta_P}{\tau_C + \theta_P}\right)$	$\tau_P + 0.5\theta_P$	$\dfrac{\tau_P\theta_P}{2\tau_P + \theta_P}$	$\dfrac{\tau_C(\tau_P + 0.5\theta_P)}{\tau_P(\tau_C + \theta_P)}$
PID Interacting w/filter	$\dfrac{1}{K_P}\left(\dfrac{\tau_P}{\tau_C + \theta_P}\right)$	τ_P	$0.5\theta_P$	$\dfrac{\tau_C}{\tau_C + \theta_P}$

D.2 PID Tuning Guide for *Integrating* (Non-Self Regulating) Processes

Begin by fitting a first order plus dead time integrating (FOPDT Integrating) dynamic model to process data. "Process" is defined to include all dynamic information from the output signal of the controller through the measured response signal of the process variable.

Integrating processes are unstable so the process should already be in closed loop. If not, stabilize the process with a simple P-Only controller and generate process data with a set point step. For accurate results:

- the process must be at steady state before forcing a dynamic response; the first data point recorded must equal that steady state value
- the controller output movement from the set point step should force the process variable to move at least ten times the noise band

Use *Design Tools* to fit a FOPDT Integrating dynamic model to the process data set. A FOPDT Integrating model has the form:

Time Domain: $\frac{dy(t)}{dt} = K_P^* \, u(t - \theta_P)$

Laplace Domain: $\frac{Y(s)}{U(s)} = \frac{K_P^* \, e^{-\theta_P s}}{s}$

where:

- $y(t)$ = measured process variable signal
- $u(t)$ = controller output signal
- K_P^* = integrator gain; units of $y(t)/(u(t)\cdot time)$
- θ_P = process dead time; units of time

also:

- K_C = controller gain; units of $(u(t)\cdot time)/y(t)$
- τ_I = controller reset time; units of time
- τ_D = controller derivative time; units of time
- α = derivative filter constant; unitless

Values of K_P^* and θ_P that describe the dynamic behavior of your process are important because:

- they are used in correlations (listed below) to compute initial PID controller tuning values K_C, τ_I, τ_D and α
- the sign of K_P^* indicates the action of the controller ($+K_P^* \rightarrow$ reverse acting; $-K_P^* \rightarrow$ direct acting)

These correlations provide an excellent start for tuning. Final tuning may require online trial and error. "Best" tuning is defined by you and your knowledge of the capabilities of the process, desires of management, goals of production, and impact on other processes.

IMC (lambda) Tuning

Standard Tuning: $\tau_C = \theta_P \sqrt{10}$

Conservative Tuning: $\tau_C = 5\theta_P \sqrt{10}$

* This is a Ziegler-Nichols process reaction curve (PRC) correlation as no P-Only IMC exists

	K_C	τ_I	τ_D	α
P-Only*	$\frac{1}{K_p^* \theta_P}$			
PI	$\frac{1}{K_p^*} \frac{2\tau_C + \theta_P}{(\theta_P + \tau_C)^2}$	$2\tau_C + \theta_P$		
PID Ideal	$\frac{1}{K_p^*}\left(\frac{2\tau_C + \theta_P}{(\tau_C + 0.5\theta_P)^2}\right)$	$2\tau_C + \theta_P$	$\frac{0.25\theta_P^2 + \tau_C \theta_P}{2\tau_C + \theta_P}$	
PID Interact	$\frac{1}{K_p^*}\left(\frac{2\tau_C + 0.5\theta_P}{(\tau_C + 0.5\theta_P)^2}\right)$	$2\tau_C + 0.5\theta_P$	$0.5\theta_P$	
PID Ideal w/filter	$\frac{1}{K_P^*}\left(\frac{2\tau_C + 1.5\theta_p}{\tau_C^2 + 2\tau_C\theta_p + 0.5\theta_P^2}\right)$	$2\tau_C + 1.5\theta_P$	$\frac{0.5\theta_P^2 + \tau_C \theta_P}{2\tau_C + 1.5\theta_P}$	$\frac{(0.5\tau_C^2)(2\tau_C + 1.5\theta_P)}{(\tau_C^2 + 2\tau_C\theta_P + 0.5\theta_P^2)(0.5\theta_P + \tau_C)}$
PID Interact w/filter	$\frac{1}{K_P^*}\left(\frac{2\tau_C + \theta_P}{\tau_C^2 + 2\tau_C\theta_P + 0.5\theta_P^2}\right)$	$2\tau_C + \theta_P$	$0.5\theta_P$	$\frac{\tau_C^2}{\tau_C^2 + 2\tau_C \theta_P + 0.5\theta_P^2}$

Index

G

H

I

J

L

M

N

O

P

R

S

T

U