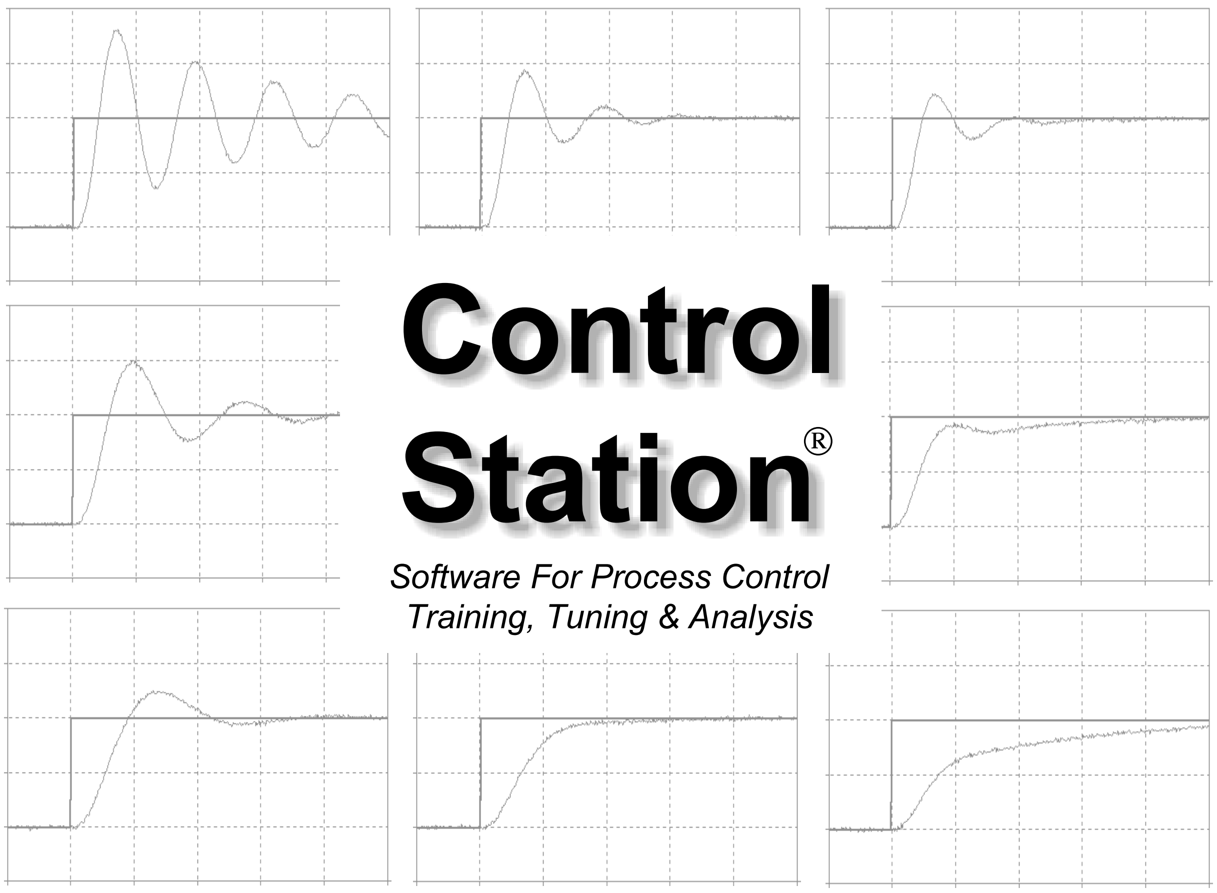


Control Station®

Software For Process Control Training, Tuning & Analysis

Hands-on Workshop Series
using Control Station version 2.5

Doug Cooper

Table Of Contents

Workshop 1: Exploring Dynamics of the Gravity Drained Tanks

Objective: *To generate open loop step test data and learn how to describe the observed dynamic process behavior with a first order plus dead time (FOPDT) model. Also, to learn about the nonlinear nature of processes.*

Reference: "Practical Process Control" Chapters 1-3

1) We always begin a controller design by analyzing the dynamic behavior of the process, or how the measured process variable responds to changes in the controller output signal. Here we analyze the dynamic behavior of the gravity drained tanks process.

   Click the *Case Studies* button on the main Control Station screen. From the pop-up list of processes, click "Gravity Drained Tanks" to start the simulation. Study the graphic and observe how a change in the controller output signal will cause the valve position to change, which changes the flow rate of liquid into the top tank, which ultimately changes the level in the lower tank. Liquid level in the lower tank is our measured process variable.

2) The formal approach for analyzing the behavior of a process is to fit a dynamic model to process data. Good data starts with the process at steady state. The controller output signal is then moved far enough and fast enough so that it forces a clear response in the measured process variable.

   For a step test, the controller output signal is stepped from one value to another and the response of the measured process variable signal is recorded from initial steady state to final steady state.

   At the upper right of the draining tanks graphic on your screen, locate the white number box below the Controller Output label. The number displayed should be 70, corresponding to a 70% output signal from the controller.

   The most convenient way to step the controller output signal is to click once on this white number box. Alternatively, you may double click (click twice in rapid succession) the box.

   If you clicked once, the box will turn blue. For the first step test, type 80 into the box and press Enter. This will cause the controller output to step from its current value of 70% up to the new value of 80%, increasing the flow of liquid into the top tank and causing liquid level to rise.

   If you double clicked, a pop-up menu gives you the option of performing a step, oscillate, ramp or PRBS experiment. With the Step option active, type 80 as the Controller Output Value and then click Done at the bottom of the pop-up menu.

3) Watch as the process responds. When the measured process variable (liquid level) reaches its new steady state, click the "Pause" icon on the tool bar above the strip charts to stop the simulation.

   To view a full screen plot, click the printer icon (with floating label "View and Print Plot") on the tool bar. Alternatively, click File on the menu list and choose View and Print Plot. With the plot displayed, click the Plot Options icon on the plot tool bar and use the various options to create a plot best suited for graphical analysis.

   (Aside: When reading graphs, your results will be more precise if the plot axes are adjusted so the data of interest fills the screen. Also, data points can be read most accurately if tick marks are positioned frequently across the grid.)

Workshop 1: Dynamics of the Gravity Drained Tanks (cont.)

4) Using the methodology described in the Practical Process Control book, fit a first order plus dead time (FOPDT) dynamic model to the process response plot. 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. The work sheet on page 4 will help you in these calculations. Record your results:

   Process gain, $K_P =$ Time constant, $\tau_P =$ Dead time, $\theta_P =$

5) Enter a controller output of 50% and let the measured level reach steady state. Repeat the above procedure for a second step in the controller output from 50 up to 60% and record your results:

   Process gain, $K_P =$ Time constant, $\tau_P =$ Dead time, $\theta_P =$

6) The two steps in the controller output (70→80% and 50→60%) were both of the same size. Are the model parameters the same for these two steps? How/why are they different?

7) Set the controller output to 20% and let the measured level in the lower tank steady out (you should notice a low alarm condition). Step the controller output from 20% up to 40%. When the measured level reaches a steady value, make a next step up to 60%. Continue making steps, next up to 80% and finally up to 100%. Let the process complete its response between each step.

   View a plot of this response over the whole range of operation. The change in the controller output is constant at each step. If the process is linear, then the measured liquid level response will be the same for each step. If the process is nonlinear, then the K_P, τ_P and θ_P will change as operating level changes. Is the gravity drained tanks a linear or nonlinear process?

(optional)

8) Return to the main Control Station screen, click the *Case Studies* button and start the Heat Exchanger process. Take a moment to study the process graphic. The controller output adjusts the flow rate of cooling liquid on the shell side. The measured process variable is stream temperature exiting the exchanger on the tube side.

   Step the controller output signal from 39% up to 50%. This causes an increase in the cooling flow rate and hence a decrease in the measured exit stream temperature. When the process response is complete, click the "Pause" icon on the tool bar above the strip charts to pause the simulation.

   Using the skills you have learned, fit a FOPDT model to the data and record the model parameters:

   Process gain, $K_P =$ Time constant, $\tau_P =$ Dead time, $\theta_P =$

   What is a notable difference in the process gain for the heat exchanger compared to that of the gravity drained tanks?

9) Perform the experiments necessary to determine if the heat exchanger is a linear or nonlinear process. What experiments did you perform and what do you conclude?

Workshop 1: Dynamics of the Gravity Drained Tanks (cont.)

This work sheet follows the terminology and procedures detailed in the Practical Process Control book. In this exercise, $u(t)$ = controller output and $y(t)$ = measured process variable (level in the lower tank)

Controller output step from 70% → 80%	Controller output step from 50% → 60%
$u_1 =$ 70% $\Rightarrow$ $y_1 =$	$u_1 =$ 50% $\Rightarrow$ $y_1 =$
$u_2 =$ 80% $\Rightarrow$ $y_2 =$	$u_2 =$ 60% $\Rightarrow$ $y_2 =$
$\Delta u =$ $\Delta y =$	$\Delta u =$ $\Delta y =$
$K_P = (\Delta y / \Delta u) = \boxed{}$	$K_P = (\Delta y / \Delta u) = \boxed{}$
$t_{Y\text{start}} =$	$t_{Y\text{start}} =$
$y_{63.2} = y_1 + 0.632(\Delta y) =$	$y_{63.2} = y_1 + 0.632(\Delta y) =$
$t_{63.2} =$	$t_{63.2} =$
$\tau_P = t_{63.2} - t_{Y\text{start}} = \boxed{}$	$\tau_P = t_{63.2} - t_{Y\text{start}} = \boxed{}$
$t_{U\text{step}} =$	$t_{U\text{step}} =$
$\theta_P = t_{Y\text{start}} - t_{U\text{step}} = \boxed{}$	$\theta_P = t_{Y\text{start}} - t_{U\text{step}} = \boxed{}$

Workshop 2: P-Only Control of Tank Level

Objective: To design and test a P-Only controller for tracking changes in set point and rejecting disturbances. Also, to explore how controller gain impacts P-Only performance.

Reference: "Practical Process Control" Chapters 4-5

1) All controllers should be designed for a specific level of operation, which includes specifying the expected or design value for the set point, and typical or baseline values for the important process disturbances. Here we design a controller for the gravity drained tanks process. Click the *Case Studies* button on the main Control Station screen and from the pop-up list of processes, click "Gravity Drained Tanks" to start the tanks simulation.

   For this P-Only level control design, consider the case where the design level of operation is a measured liquid level in the lower tank of 3.3 m. The flow rate of the disturbance stream out of the bottom of the lower tank is expected to be about 2.0 L/min during typical operation. When your controller is put in automatic, the set point should be at the design level of the measured process variable:

   $y_{\text{setpoint}} =$

   A P-Only controller design requires determination of the controller bias, or the value of the controller output that causes the measured liquid level in the lower tank to steady at the design value of 3.3 m when the disturbance flow rate is at its design value of 2.0 L/min. Change the controller output, search for this value and record your result:

   $u_{\text{bias}} =$

2) The next step, similar to workshop 1, is to describe the dynamic behavior of this process with a first order plus dead time (FOPDT) dynamic model. That is, we need to compute the steady state process gain, K_P, overall time constant, τ_P, and apparent dead time, θ_P, for the process at the design level of operation.

   In step 1 above it was (hopefully) found that a controller output of 64% causes the measured level in the lower tank to steady at the design value of 3.3 m. Because this process is nonlinear, best practice is to perform an experiment that includes data both a little below and a little above this design level.

   Set the controller output to 60% and wait until the process steadies. With this as a starting point, step the controller output up to 68%. The result is a data set that on average describes the design level of 3.3 m. When the process response is complete, click the "Pause" icon on the tool bar above the strip charts to pause the simulation.

   View a full screen plot of the response by clicking the printer icon (with floating label "View and Print Plot") on the tool bar. Alternatively, click File on the menu list and choose View and Print Plot. With the plot displayed, click the Plot Options icon on the plot tool bar and use the various options to create a plot best suited for graphical analysis.

   Compute and record the FOPDT process model parameters:

   Process gain, $K_P =$ Time constant, $\tau_P =$ Dead time, $\theta_P =$

Workshop 2: P-Only Control of Tank Level (cont.)

3) One important reason for fitting a FOPDT model to dynamic process data is because the resulting model parameters can be used in correlations to estimate controller tuning values. One popular controller design correlation is the ITAE (Integral Time Weighted Absolute Error) for Set Point Tracking, which in the form below yields an initial estimate of the P-Only controller gain, K_C, when set point tracking is the primary objective.

   Compute and record P-Only controller gain. Remember it has a sign (plus or minus) and units.

   ITAE for Set Point Tracking tuning relation: $$K_C = \frac{0.2021}{K_P}\left(\theta_P/\tau_P\right)^{-1.2188}$$

   $K_C =$

4) Now validate your completed controller design. Click the level controller icon (the LC in the white circle on the gravity drained tanks graphic). When the controller design menu opens, note that "Manual Mode" is displayed at the top of the menu indicating that no controller is in service (the control loop is open). Click "Manual Mode" and from the selection of controllers, click "PID" to enable a proportional-integral-derivative controller.

   The default configuration is a PI controller. Thus, the proportional and integral modes are “On” while the derivative mode is “Off.” Implement a P-Only controller by turning the integral mode off. To do this, click “ON: Integral with Anti-Reset Windup” (located half way down the controller design menu) and choose “off: Integral Mode.” Tuning values entered for the integral and derivative terms have no impact on controller computations when these modes are off.

   With this P-Only configuration, click "Set Point," then "Bias" and finally "Controller Gain" and enter the values you computed for these selections. When finished, click "Done" at the bottom of the controller design menu to put your controller in automatic. If you have performed the steps correctly, the strip chart should show the white measured level signal tracing (with some noise or random error) over the straight yellow set point line. There should be no sustained offset.

5) Explore the performance of this controller in tracking set point steps. The set point is most conveniently changed by clicking on the "Set Point" box on the process graphic. Step the set point from 3.3 m up to 4.3 m and back to 3.3 m. Observe the performance of your controller as it attempts to track the changes. Do you notice offset when the set point is different from the design value?

   Now explore how the magnitude of K_C impacts offset and the oscillatory nature of the controller response. Click the level controller (the LC in the circle on the graphic) and in the controller design menu, change your controller gain. If you think your controller is too aggressive because of large response oscillations, cut your K_C in half. If your controller is rather slow or sluggish in its response, double your K_C.

   Now make the same set point step as before. Keep changing your controller gain to search for a performance that balances offset against oscillatory behavior. Call this your "best" K_C tuning value.

Workshop 2: P-Only Control of Tank Level (cont.)

6) Using your "best" K_C, put the set point at 3.3 m and let the measured liquid level steady out. Now, step the set point from 3.3 m up to 4.3 m, then up to 5.3 m, then 6.3 m and finally 7.3 m. Let the measured level steady out after each set point step. You may need to use the "Change Strip Chart History" icon on the tool bar to increase the strip chart display to 80 minutes.

   How does offset and oscillatory nature of the response change as the set point moves further from the design value? Does the controller output ever become constrained (hit a maximum or minimum limit)? If so, how does this impact the set point response?

7) Study disturbance rejection performance of a P-Only controller. Using your "best" K_C, put the set point back at 3.3 m and let the measured level steady out. Use the "Change Strip Chart History" icon from the tool bar and return the strip chart display to 30 minutes. Wait for previous changes to scroll off of the moving strip charts and when they show essentially straight lines, expand the plots using the "Rescale Vertical Axis" icon from the tool bar.

   Step the disturbance flow rate from 2.0 L/min up to 5 L/min by clicking on the Pumped Flow Disturbance box located to the lower left of the process graphic. The set point is at its design value, so why does the controller now show offset? Step the disturbance back and forth between 2.0 L/min and 5 L/min using different values of K_C. How does controller gain impact offset and oscillatory behavior for P-Only disturbance rejection? Is your best K_C for disturbance rejection the same as that for set point tracking?

(optional)

8) Design and test a P-Only controller for the Heat Exchanger process. The design level of operation is a measured exit stream temperature of 144 ^oC. The warm oil flow disturbance stream is expected to be about 10 L/min during typical operation. Test the controller for set point tracking from 144 ^oC up to 150 ^oC and back again. Also test the controller in rejecting warm oil flow stream disturbance changes from 10 L/min up to 15 L/min and back again.

Workshop 3: The Hazard of Tuning PI Controllers by Trial and Error

Objective: To build intuition about PI controller performance and tuning when the control objective is disturbance rejection. Also, to experience the drawbacks of trial and error tuning.

Reference: "Practical Process Control" Chapters 6-9

1) Here we explore the tuning of a PI controller by trial and error when disturbance rejection is the control objective. The process will be an ideal linear simulation designed using *Custom Process*. Start by clicking on the *Custom Process* button on Control Station's main screen and then choosing Single Loop Process from the list.

   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).

   To create the simulation, click the Process button on the graphic. This opens a "Construct Process and Disturbance Models" menu. First specify the controller output to measured process variable dynamic behavior. Click the Process Model tab (it should already be active if you have not done any exploring) and select the Overdamped Linear Model from the list of models available. Enter the process gain, three time constants and dead time that define the controller output to measured process variable dynamic behavior of this overdamped linear third order plus dead time model:

Process Gain, K_P	=	1.0
First Time Constant, τ_{P1}	=	10.0
Second Time Constant, τ_{P2}	=	1.0
Third Time Constant, τ_{P3}	=	1.0
Lead Time, τ_{PL}	=	0
Dead Time, θ_P	=	1.0

   Next specify the behavior of the disturbance to measured process variable dynamics. Click the Disturbance tab at the top of the form, ensure that Overdamped Linear Model is selected, and enter:

Disturbance Gain, K_D	=	1.5
First Time Constant, τ_{D1}	=	15.0
Second Time Constant, τ_{D2}	=	1.5
Third Time Constant, τ_{D3}	=	1.5
Lead Time, τ_{DL}	=	0
Dead Time, θ_D	=	1.5

   Click Done at the bottom of the form to start the simulation.

2) Controller output and disturbance variable changes differ in their impact on the measured process variable because the dynamic models defined above are different. Verify this by stepping the controller output from its default value of 50% up to 60% and when the response is substantially complete, step it back to 50%. Next, step the disturbance variable from its default value of 50% up to 60% and when the response is substantially complete, step it back to 50%.

   Compare the two responses side by side (you may need to use the Change Strip Chart History icon on the toolbar to see both responses). Does a step in the disturbance variable show a larger but slightly slower response associated with a larger gain and longer overall time constant?

Workshop 3: Tuning PI Controllers by Trial and Error (cont.)

3) Click the Controller icon on the graphic and select PID from the controller design menu. Specify a very conservative (sluggish) PI controller by entering the tuning values:

Set Point	=	50
Controller Gain	=	1.0
Reset Time	=	50

Choose the Integral with Anti-Reset Windup mode and be sure you are using a PI controller by leaving Derivative Mode off. Click Done to put the controller in automatic.

4) Trial and error tuning can use up significant production time and create expensive off-spec product. To track how inefficient a trial and error approach can be, zero the simulation clock by clicking on the Reset Process Clock to Zero on the upper left of the button tool bar. From this point forward, do not reset the clock or pause process execution until instructed to do so.

5) Start tuning for disturbance rejection by considering the current performance of the controller. Click the white "Disturbance, D" box on the graphic, step the disturbance variable from 50% up to 60% and after the response, step it back to 50%.

Now try and improve controller performance by adjusting K_C and/or τ_I on the controller design menu. The PI controller tuning map in Chapter 8 of the Practical Process Control book is for set point tracking but may assist your thought process as you search for your best tuning values. Test your new values by again stepping the disturbance from 50% up to 60% and back again. Repeat this procedure as necessary to arrive at "best" tuning values for disturbance rejection. Use your own experience and opinion to define "best."

When you have determined best tuning values for disturbance rejection, pause the simulation and record them below. Also record the elapsed time displayed on the process clock at the lower left of the screen.

$K_C =$ $\tau_I =$ Elapsed Time = minutes

(optional)

6) Repeat the above exercise with the objective of controller tuning for set point tracking. Start with your best disturbance rejection tuning values above and determine by trial and error whether they are best when the control objective is tracking set point steps from 50% up to 60% and back again. Record your results:

$K_C =$ $\tau_I =$ Elapsed Time = minutes

Is your best tuning for set point tracking the same as your best tuning for disturbance rejection?

Workshop 4: PI Control of Heat Exchanger Temperature

Objective: *To learn about the performance of a PI controller and to explore the interaction of the two tuning parameters on controller performance. Also, to learn how* Design Tools *can automate the dynamic modeling and controller design task.*

Reference: *"Practical Process Control" Chapters 6-9*

1) A formal or systematic procedure for designing and tuning a controller is detailed in the "Practical Process Control" book and will be used in this study. This procedure is important because it is the one used in the plant, the lab and with Control Station simulations:

   (Aside: A summary of the procedure we will follow in this workshop is:

   a) Move the process to the design level of operation and when it reaches steady state, generate and record controller output to measured process variable dynamic data.
   b) Fit a FOPDT model to this data. *Design Tools* is well-suited for this modeling task, though graphical methods explored in earlier workshops can be used.
   c) Use the resulting FOPDT model parameters in a correlation to compute initial PID controller tuning values.
   d) Implement your controller on the actual process and perform final tuning by trial and error until control objectives are satisfied.)

   Begin this study by starting the heat exchanger case study in Control Station. Consider a case where the design temperature of the exit stream is 134 ^{o}C and where the warm oil disturbance flow rate is expected to be about 30 L/min during typical operation. Record your design set point here:

   $$y_{\text{setpoint}} =$$

   Your first goal is to move the process to the design level of operation. Click the disturbance box shown on the graphic and set the warm oil flow rate to its expected value of 30 L/min. Now search for the value of the controller output (that manipulates the cooling flow rate) that causes the measured exit temperature to steady at the design value of 134 ^{o}C.

   When you have moved your process to the design state, wait until all of the dynamics have scrolled off screen on the data plots (wait until the plots show essentially straight lines) and expand the plots using the "Rescale Vertical Axis" icon from the tool bar.

2) Just as for the P-Only controller design, we start a PI controller design with experiments to generate dynamic process data around the design exit stream temperature of 134 ^{o}C. Since a controller output of 42% causes the measured exit temperature to steady at 134 ^{o}C, make your changes starting at this controller output value.

   (Aside: *Design Tools* is not limited to simple step tests. In this study, we make a quick pulse up followed by a quick pulse down in the controller output. Such a *doublet* generates data both above and below our design level of operation, a desirable result when the process has a nonlinear character. This method also returns the process to the initial design value relatively quickly, which is a popular notion with operations staff.)

Workshop 4: PI Control of Heat Exchanger Temperature (cont.)

Before starting the experiment, turn file storage on with the "Save Data to File" icon on the tool bar. You must enter a file name before data storage begins. After entering a file name, a pop-up menu asks for a data storage rate and starting time. Accept the default values by clicking "OK."

Storage of process data to file is now in progress. Step the controller output from 42% up to 47%. When the measured temperature has shown a clear response to this change (you need not wait for the response to reach steady state), step the controller output from 47% down to 37%. And after a clear response to this change, step the controller output back to its original value of 42%. As soon as the final response is complete, use the "Stop Saving Data" icon on the tool bar to turn file storage off.

3) Now use *Design Tools* to automatically fit a FOPDT model to this data. First pause execution of the heat exchanger simulation using the "Pause" icon on the tool bar. Then, use the "Navigate" icon on the tool bar to open *Design Tools.*

   From the *Design Tools* main screen, click the "Open Data File" icon to the left on the toolbar. In the Open File pop up menu, click the file name where you saved the process data, click "OK," and import your dynamic process data. After the file is read, it will be displayed on your screen with the Time, Manipulated Variable and Process Variable columns of data already properly labeled. If you agree that the data labels are correct, click "OK" to accept the column labels.

   (Aside: You can use the "Edit" button located at the upper right of this Label Data screen and read the column labels if you are not sure which data each column contains.)

4) You should be back to the *Design Tools* main screen now. Make sure the "First Order Plus Dead Time (FOPDT)" model is displayed in the equation box at the bottom of the screen. If not, use the "Select Model" icon on the tool bar to select it from the library of models.

   Click the "Start Fitting" icon on the tool bar to start fitting the FOPDT model to the process data. If the fit is successful, *Design Tools* will display a plot. Use your judgment to determine whether the model reasonably describes the data you collected. Hopefully, the FOPDT model plot line (shown in yellow) will substantially approximate the response of the measured process variable data (shown in white). If the model does not describe the data reasonably well, then the tuning values computed from the model parameters will be suspect.

   If the model describes the data to your satisfaction, close the plot screen and note the FOPDT model parameters displayed on *Design Tools'* main screen.

   $K_P =$ $\tau_P =$ $\theta_P =$

5) The main *Design Tools* screen will also display tuning parameter values for the P-Only, PI and PID controller. Click the "PI" tab to display the tuning for that controller.

Workshop 4: PI Control of Heat Exchanger Temperature (cont.)

One of the choices displayed is the PI controller gain, K_C, and reset time, τ_I, computed using the IMC (Internal Model Control) correlation. The correlations are shown below, but you need not perform the calculation as Control Station has done it for you. Record the IMC tuning values for the PI controller:

IMC Controller Gain; $K_C = \dfrac{1}{K_p}\dfrac{\tau_p}{(\theta_p + \tau_c)}$ IMC Reset Time; $\tau_I = \tau_P$

$K_C =$ $\tau_I =$

6) After recording the PI controller tuning values, close *Design Tools* to return to the Heat Exchanger process. Click the "Continue" icon on the tool bar to restart the simulation. (Note: if you had closed the heat exchanger simulation rather than navigating from it, you must again enter the warm oil disturbance flow rate of 30 L/min and a controller output value of 42% to move the process back to the design conditions.)

   Click the temperature controller (the TC circle on the graphic) to open the controller design menu. Click "Manual Mode" at the top of the menu and from the list of controllers, choose PID. The default design is for a PI controller with anti-reset windup logic.

   (Aside: Note that when the integral mode is on, the controller bias need not be entered. This is because, when the controller is put in automatic, the bias will be automatically set to the current value of the controller output. This provides a smooth or *bumpless* transfer to automatic.)

   On the design menu, click "Set Point," then "Controller Gain" and "Reset Time" boxes and enter the values you recorded above. Because this process has a negative gain, you must select the "Proportional - Direct Acting" mode and enter K_C as a positive number. Leave derivative time as zero, thus making the PID controller into a PI controller. When you are finished, click "Done" at the bottom of the menu to put your controller in automatic.

7) We first want to study the performance of the controller in tracking set point steps. Starting with a set point temperature of 134 ^{o}C each time, explore the performance of the controller to step increases in set point from 134 up to 144 ^{o}C and back again (It is most convenient to make step changes in set point by clicking on the set point box on the heat exchanger graphic).

   Use trial and error to determine a "best" tuning by adjusting K_C and τ_I until the set point tracking performance balances a quick response with a modest overshoot and rapid decay. Use the PI controller tuning map toward the end of Chapter 8 of the Practical Process Control book as a guide to arrive at your desired tuning.

Workshop 4: PI Control of Heat Exchanger Temperature (cont.)

8) Study how nonlinear behavior impacts controller performance. Using your best K_C and τ_I from step 7, make a set point step from 134 down to 124 ^{o}C and back again. How does the nonlinear behavior of this process impact the set point response performance when compared to step set point increases?

9) Explore how each tuning parameter impacts the performance of a PI controller when tracking set point steps. Double your K_C while leaving τ_I at its best value and step the set point from 134 ^{o}C up to 144 ^{o}C and back. Now return K_C to its best value, double your τ_I and again step the set point from 134 ^{o}C up to 144 ^{o}C and back. Try other tuning combinations to build your intuition on how these parameters impact controller performance.

10) Investigate the disturbance rejection capability of the PI controller. Return to your best K_C and τ_I values from step 7. Click the warm oil flow rate box and step this disturbance flow rate from 30 L/min up to 40 L/min. After the response is complete, step it back to 30 L/min. Determine through trial and error the best K_C and τ_I for disturbance rejection. Is it the same as your best tuning for set point tracking?

    Now step the disturbance from 30 down to 20 liters/min and back. Does the nonlinear character of this process impact disturbance rejection performance?

11) Briefly explore the hazards of reset windup (windup occurs from improper programming of the integral term of the PI algorithm. The result is decreased performance whenever the final control element reaches a maximum or minimum constraint).

    With the set point at 134 ^{o}C, the warm oil disturbance flow rate at 30 L/min and the process steady, step the set point up to 180 ^{o}C. This set point is higher than the process can reach. In fact, the valve hits its low limit as the temperature approaches 170 ^{o}C. After the process has steadied at its extreme value, step the set point back to 134 ^{o}C.

    On the controller design menu, switch the integral term to Integral with Windup, repeat the set point step up to 180 ^{o}C, and after the response is complete, back to 134 ^{o}C. Do you see a difference between the two algorithms when the set point is stepped back down? Specifically, note the delay that occurs before the process starts responding to the return set point step for the reset windup algorithm. The integral term must first "unwind" before the response can begin.

    Now compare Integral with Windup against Integral with Anti-Reset Windup logic for set point steps from 134 ^{o}C up to 144 ^{o}C and back again. Do you notice that as long as the controller output signal does not reach a maximum or minimum constraint, controller performance is unaffected by choice of algorithm?

(optional)

12) Now that you have learned a formal or systematic method of tuning PI controllers, return to exercise 3 and repeat the workshop using your new skills. Compare the systematic method to the trial and error approach and make some decisions about the strengths and weaknesses of each method.

Workshop 5: PID Control of Heat Exchanger Temperature

Objective: To learn about the tuning and performance capabilities of a PID controller and to explore the interaction of the three tuning parameters on controller performance.

Reference: "Practical Process Control" Chapter 10

1) The systematic controller design procedure presented in workshop 4 can be applied to full, three mode PID control. For the heat exchanger process, consider a case where the design or expected operating temperature of the exit stream is 137 ^oC when the warm oil disturbance flow rate is at its expected value of about 10 L/min.

   Perform the necessary open loop dynamic modeling studies near this design value to determine a first order plus dead time (FOPDT) model that describes process operation near the design operating conditions. Use *Design Tools* as needed throughout this study. If you are uncertain how to do this step, follow the procedures detailed in Workshop 4.

2) Use your K_P, τ_P and θ_P in the IMC (Internal Model Control) tuning correlations and compute the K_C , τ_I and τ_D tuning parameters for a PID controller. Return to the heat exchanger process and using these tuning values implement a PID controller. Select the Integral with Anti-Reset Windup mode and the Derivative on Measurement mode on the controller design menu.

   Test the set point tracking capability of this controller by exploring the response of the process to steps in set point from 137 ^oC up to 147 ^oC and back again.

3) Determine a "best" tuning for tracking set point steps from 137 ^oC up to 147 ^oC. We will study how noise in the measured process variable impacts derivative action later. For now, set the measurement noise to zero (a convenient way to do this is by clicking the right mouse key).

   Using trial and error, adjust K_C , τ_I and τ_D until the set point tracking performance balances a quick response with a modest overshoot and rapid decay. Record your "best" tuning here.

   $K_C =$ $\tau_I =$ $\tau_D =$

   When you have determined a "best" tuning, compare set point steps from 137 ^oC up to 147 ^oC against set point steps from 137 ^oC down to 127 ^oC. How does the nonlinear behavior of this process impact the set point tracking performance?

4) Build your intuition on the interaction that K_C , τ_I and τ_D have with each other. Starting from your best tuning values, double and then halve each of the tuning parameters in turn while leaving the other two fixed at their best value. Note how each change affects controller tracking performance for set point steps from 137 ^oC up to 147 ^oC. Do your experiments show that derivative action works to dampen oscillations?

Workshop 5: PID Control of Heat Exchanger Temperature (cont.)

5) Explore the impact of measurement noise on derivative action. Starting with your best tuning parameter values, set the signal noise in the measured process variable to 0.5 (click the right mouse key) and step the set point from 137 ^oC up to 147 ^oC and back again. Now double the noise to 1.0 and repeat the set point steps. Double the noise to 2.0 and repeat the set point steps. Does derivative action lead to degraded controller performance when the measurement process variable signal is corrupted with significant noise?

6) To verify that it is the derivative action that is causing the poor set point tracking performance in the presence of significant noise, return to the controller design menu, turn derivative mode "Off" and click Done. Does the set point tracking improve under what is now PI control? Do the oscillations in the measure process variable increase because derivative action is no longer damping them?

   Compare the controller output signal when derivative action was present to the way it looks now under PI control. To do this, you may need to click the View and Print Plot icon on the tool bar and use the fixed plot options to view back to where this data is located. Can you see that derivative action amplifies process variable noise in the controller output signal?

7) Explore the difference between derivative on measurement and derivative on error. Set the measurement noise to zero, the derivative mode to Derivative on Measurement, and enter your best tuning values from step 3. Now change the derivative time on the controller design menu to a value four times larger than your best value. This makes derivative action very significant. Make steps in set point from 137 ^oC up to 147 ^oC and back again.

   Now switch the derivative mode to Derivative on Error while leaving τ_D at the very large value. Again make steps in set point from 137 ^oC up to 147 ^oC and back. View a full screen plot and focus on the controller output signals right after the set point step for the two cases. Describe the "derivative kick" that you observe. Speculate on the problems this might cause in a production environment.

8) Study the disturbance rejection capability of the PID controller. Leave the measurement noise at zero, return the derivative mode to Derivative on Measurement, and set the controller parameters to their best tuning values. Test the ability of the controller to reject steps in the warm oil disturbance flow from 10 L/min up to 20 L/min and back again. Determine the "best" tunings for disturbance rejection. Are there different "best" tunings for disturbance rejection as compared to set point tracking?

   Return the warm oil disturbance flow to 10 L/min and let the process steady. Step the disturbance flow from 10 L/min up to 20 L/min. When the process steadies, step the flow up to 30 L/min, and then 40 L/min. Is the nonlinear character of the process evident through changes in disturbance rejection performance?

Workshop 6: Modeling Process Dynamics For Linear Simulation

Objective: To learn how Design Tools and Custom Process can be used to model, simulate and explore controller design of real operating processes.

Reference: "Practical Process Control" Chapters 6-7

1) In this exercise, we learn how *Design Tools* and *Custom Process* can be used to analyze real processes. We will use the Jacketed Reactor in this study so we can test our results. The procedure detailed here is important, however, because it is the same one you would follow in the plant or lab.

   (Aside: A summary of the procedure we will follow in this workshop is:

   a) At the design level of operation, collect controller output to measured process variable dynamic data. Also, for the dominant disturbance, collect disturbance to measured process variable dynamic data.
   b) Use *Design Tools* to fit a first order plus dead time (FOPDT) model to the controller output data set. Use the resulting model parameters to compute controller tuning values.
   c) Use *Design Tools* to fit second order models to both controller output and disturbance data sets. The resulting models are implemented in *Custom Process* to simulate the process.
   d) Implement a controller in *Custom Process* and test it in tracking set points and/or rejecting disturbances. Refine the design and tuning as needed.
   e) When satisfied with the controller performance in *Custom Process*, return to the actual process for final implementation.)

2) To begin, start the Jacketed Reactor (<u>not</u> the cascade) process. This will serve as our "real" process for the study. The first step outlined above is to collect data at the design level of operation, which in this study is a reactor exit stream temperature (the measured process variable) of 92 ^{o}C. The cooling jacket inlet temperature (the disturbance) is expected to be about 50 ^{o}C during normal operation but may occasionally spike as high as 60 ^{o}C. Rejecting the influence of this disturbance spike so the reactor exit stream temperature remains as close as possible to 92 ^{o}C will be the focus of our design.

   (Aside: The temperature of the reactor exit stream is directly related to percent conversion of reactants to products and thus can be used to infer product purity. Though product purity is our true control objective, reactor temperature is an inexpensive and reliable measured variable substitute.)

   With the reactor at steady state at the design level of operation, note the values of the controller output, measured process variable and disturbance. We will need these values for the simulation:

   $u_{\text{Design}} =$ $\qquad\qquad$ $y_{\text{Design}} =$ $\qquad\qquad$ $d_{\text{Design}} =$

   Begin saving data to a file (called, say, PROCESS.TXT). Step the controller output from the design level of 42% up to 47%, then down to 37%, and back to 42%. Let the process show a clear response between steps. Stop saving data after the process variable returns to steady state.

   Again begin saving data to a file (called, say, DISTURB.TXT) and step the cooling jacket inlet disturbance temperature from 50 ^{o}C up to 60 ^{o}C, and after the process responds, back to 50 ^{o}C. Stop saving data after the measured process variable steadies.

3) Exit the reactor process and return to the main Control Station screen. At this point we pretend that we have just returned from the plant with dynamic process data on disk.

   Start *Custom Process* and choose Single Loop Process from the list. Once the simulation has started, click Pause to suspend activity and then navigate to *Design Tools.* Read in file PROCESS.TXT, fit a FOPDT model to the data and use the model parameters to compute a PI controller gain, K_C, and reset time, τ_I, using the IMC (Internal Model Control) correlation. Record the results. Also, record the FOPDT SSE for this fit in the space provided in step 4 below.

   $K_C =$ $\tau_I =$

   (Aside: The sum of squared errors (SSE) shown at the bottom of the Model Parameters table indicates how well the linear model describes the process data contained in the file. A smaller SSE indicates a better goodness of fit. Thus, SSE is useful for comparing how accurately different models describe the data.)

4) Next determine a second order model that best describes the open loop dynamic process behavior. The open loop behavior of the jacketed reactor, like a great many chemical processes, is well described with a second order plus dead time (SOPDT) model.

   (Aside: The SOPDT with Lead model available in *Design Tools* is also useful in describing the open loop dynamic behavior of certain processes and we will investigate this model form in later workshops. We do not consider the second order underdamped model in any workshops because very few chemical processes require its inherent oscillating nature to describe their open loop dynamic behavior.)

   Click on the Select Model button on the Design Tools tool bar and choose the SOPDT model form. Fit the data with the SOPDT model and record the SSE below:

   FOPDT SSE = SOPDT SSE =

   Hopefully, the SOPDT model yields a smaller SSE value, indicating that it best describes the process data. Record the SOPDT model values for use in the *Custom Process* simulation.

   $K_P =$ $\tau_{P,1} =$ $\tau_{P,2} =$ $\theta_P =$

5) Repeat step 4 above for the data in file DISTURB.TXT. **Note:** after reading in a data file, *Design Tools* requires that you label the time, manipulated variable and process variable data columns. In previous exercises, the default labels have been correct. Recall that for this file, it was the cooling jacket inlet temperature that was manipulated, not the controller output signal. Thus, you must click on the last column of data in the DISTURB.TXT file and press the "m" key to properly label this column as the manipulated data. If you are unsure about this, scroll down the data to verify that this column contains the step disturbance manipulation between 50 ^oC and 60 ^oC.

Fit the disturbance data with the FOPDT and SOPDT model and record the SSE for each fit:

FOPDT SSE = SOPDT SSE =

Again the SOPDT model (hopefully) shows improved ability to describe the process data. Note the values for the disturbance driven SOPDT model:

$K_D =$ $\tau_{D,1} =$ $\tau_{D,2} =$ $\theta_D =$

6) The next step in the procedure is to implement the process and disturbance model parameters in *Custom Process* to create a jacketed reactor control simulation. Exit *Design Tools* and return to the single loop *Custom Process*. The graphic to the right of the scrolling plots contains a Process button, Disturb(ance) button and Controller button (the C in the white circle).

To create the simulation, click on the Process button to open a Construct Process and Disturbance Models menu. The Process Model tab will be active. The SOPDT model is an overdamped linear model so be sure that model is selected on the menu. Enter the process gain, two time constants and dead time that define the controller output driven data as determined in step 4. Take care to enter each parameter in its proper box. Parameters not used in the model should read zero.

Now click on the Disturbance Model tab and enter the process gain, two time constants and dead time that *Design Tools* computed to describe the disturbance driven data in step 5. This also is a overdamped linear model form. Parameters not used in the model should read zero.

Custom Process assumes as a default that the controller output, process variable and disturbance variable have a minimum value of 0, a maximum value of 100, and a design level startup value of 50. As in this study, these values are not always appropriate for a simulation. In step 2 we noted the design level start up values. The minimum and maximum values along with these start up values are listed in the table below.

(Aside: The minimum and maximum values for the reactor process actually have a larger span than that listed below. Recall that we are approximating a “real” nonlinear process with simplifying linear models. The farther we move from our design conditions, the more suspect the linear models become in describing actual process behavior. Hence, we deliberately narrow this span to a range where we believe the linear models will be reasonably accurate.)

Click on the Zeros and Spans tab and enter the maximum, minimum and start up values for each variable of the process as listed in the table:

	Controller Output	Process Variable	Disturbance
Minimum Value	0	60	40
Maximum Value	100	110	60
Start up Value	42	92	50

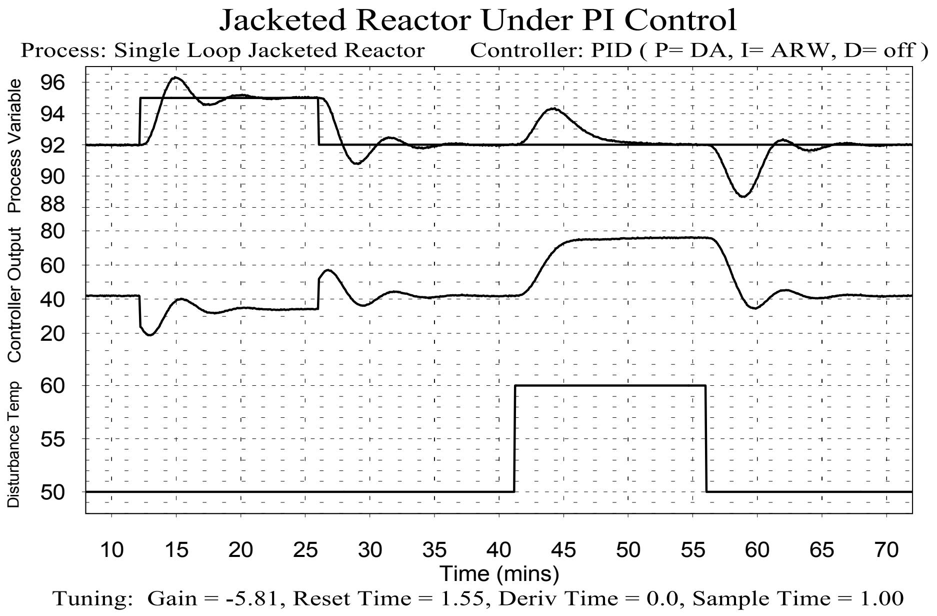
Workshop 6: Modeling Process Dynamics For Linear Simulation (cont.)

When finished, click on Done to start the simulation. You now are simulating the Jacketed Reactor process. Test the open loop dynamic behavior by stepping the controller output and disturbance variable and observing the measured process variable response. Return the process to the design conditions when you accept the simulation as appropriate.

7) Click on the controller icon on the graphic, select a PID controller, and enter the controller gain and reset time determined in step 3 (leave the derivative time as zero). Since the controller gain is negative, be sure you specify that the sense of the controller is direct acting. The integral mode should have anti-reset windup logic.

Test the controller performance in tracking a set point step from 92 ^oC up to 95 ^oC and back to 92 ^0C. Then test the controller in rejecting disturbance steps from 50 ^oC up to 60 ^oC and back to 50 ^oC.

Do your responses look similar to that of the actual Jacketed Reactor process shown below? If so, you have successfully used Control Station to simulate this "real" process. Why do you think there are differences between your simulation and the behavior of the actual reactor process as shown?



Set point tracking and disturbance rejection performance of the Jacketed Reactor process

8) From the main Control Station screen, start the Jacketed Reactor (<u>not</u> the cascade) process. Click on the controller icon on the graphic and enter your PI controller tuning values just as in step 7 above. Repeat the set point tracking and disturbance rejection test and verify the above plot is accurate.

Does your disturbance rejection performance look like that shown in the plot? It is especially important to verify this because Workshop 7 and 8 focus on using cascade control and feed forward control for improved disturbance rejection in the jacketed reactor. The above plot will be used as a baseline to establish benefits for these advanced architectures.

Workshop 7: Cascade Control of the Jacketed Reactor

Objective: *To explore cascade controller design and implementation. Also, to compare disturbance rejection performance of a cascade architecture with that of a lone PI controller for the jacketed reactor process.*

Reference: *"Practical Process Control" Chapter* 11

1) Select the cascade jacketed reactor in *Case Studies* and start the simulation. Like all cascade architectures, the process graphic shows two measured variables, two controllers, but only one manipulated variable (one valve). The upper strip chart on the screen shows the primary process variable (the reactor exit temperature) while the lower strip chart shows the secondary process variable (the cooling jacket outlet temperature). To see controller output signals, use the "View and Print Plot" icon on the tool bar to generate a fixed plot, then use the "Plot Options" icon on the plot tool bar to display controller output traces.

   The design level of operation is the same as in Workshop 6. The control objective is to maintain reactor exit temperature at 92 ^{o}C. Reactor exit temperature indicates percent conversion of reactants to products so this exit temperature is the primary or outer loop process variable. The cooling jacket inlet temperature (the disturbance) is expected to be 50 ^{o}C during normal operation but occasionally may spike as high as 60 ^{o}C. Our controller design will focus on rejecting this disturbance. Begin this study with the process steady at the design level of operation.

   Offset is unacceptable for reactor temperature control because off-spec product means lost profit. Hence, a PI controller will be used on the primary loop. The secondary or inner controller maintains jacket exit temperature at the design value of 75.6 ^{o}C. A fast and simple controller is desired here. Since we do not sell cooling jacket water, offset does not present a problem and a P-Only controller will serve our purpose.

   Consider the discussion above and the cascade jacketed reactor graphic on your screen, then record this design information for later:

Secondary (inner) P-Only Controller:	$T_{setpoint}$ =	u_{bias} =
Primary (outer) PI Controller:	$T_{setpoint}$ =	

2) Cascade controller design begins with the secondary controller (the cooling jacket exit temperature control loop). This will be P-Only controller designed to track set point updates sent from the primary controller.

   Controller tuning follows the same procedure presented in earlier workshops. Begin saving data to file (say, SECOND.TXT) and generate dynamic data by stepping the controller output from 42% down to 37%, then up to 47%, and finally back to 42%. Let the response become clearly established before making each change. Turn file storage off when the response is complete.

3) Pause the reactor simulation, navigate to *Design Tools* and read in SECOND.TXT. You will notice the Label Data Columns screen displays eight columns of data. This is because the file contains data for both the secondary and primary loop. It is essential in this (and any) study that the columns be properly labeled for model fitting and controller tuning.

Workshop 7: Cascade Control of the Jacketed Reactor (cont.)

If you are unsure which columns of data in the file are for the secondary control loop, click on the Edit Data button in the upper right corner of the Label Data Columns screen and read the headings contained in the file. For the cascade reactor, the second column will hold the secondary controller output data and the third column will hold the secondary measured process variable data. Be sure these columns are labeled accordingly.

Fit a FOPDT model to the data and if the fit seems reasonable, record the P-Only controller gain from the ITAE for Set Point Tracking correlation:

Secondary controller P-Only controller gain, K_C =

Remember that if the model does not reasonably approximate the data, the controller tuning values computed from the model parameters will be suspect.

4) Close *Design Tools* and return to the cascade reactor. Click on the secondary controller icon on the reactor graphic (the controller for the cooling jacket exit temperature) and implement a P-Only controller (select a PID controller and turn the integral and derivative action off). Enter your set point, bias and controller gain as recorded above and click Done. Make sure you have the control sense (whether it is direct or reverse acting) properly selected.

   Test the controller in tracking set point steps near the design conditions. When you are satisfied that the set point tracking performance balances a rapid response with a short settling time, leave the secondary controller in automatic and proceed to designing the primary controller.

5) The primary controller output is the set point of the secondary controller. Thus, stepping the set point of the secondary controller is the same as stepping the controller output of the primary controller. Note that because of this cascade architecture, the output of the primary controller has units of ^{o}C rather than the more traditional units of %.

   Begin saving data to file (say, PRIMARY.TXT) and generate dynamic data by stepping the secondary set point from 75.6 ^{o}C down to 72.6 ^{o}C, then up to 78.6 ^{o}C, and finally back to 75.6 ^{o}C. Allow the process to respond before making each change. When the data has been collected, turn data file storage off.

6) Again use *Design Tools* to tune the controller. Take care to label the proper manipulated and measured process variable data columns (The default labels are not correct. Click on the Edit Data button in the upper right corner of the Label Data Columns screen and read the headings contained in the file).

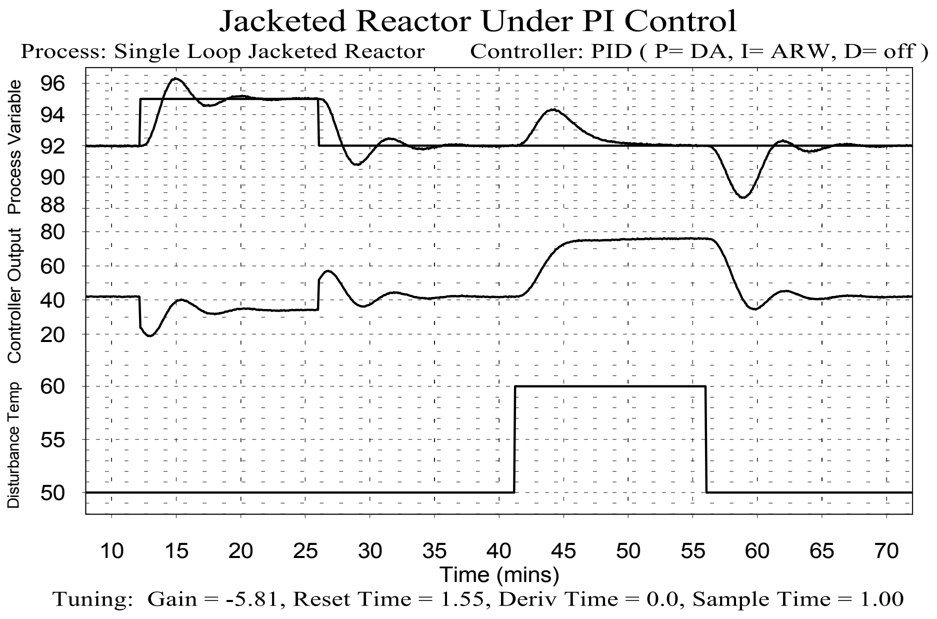
   If the model fit is reasonable, compute PI controller tuning parameters using the IMC (Internal Model Control) tuning correlation:

   Primary controller PI tuning parameters: K_C = τ_I =

Workshop 7: Cascade Control of the Jacketed Reactor (cont.)

7) Return to the jacketed reactor simulation, click on the outer primary controller and implement your PI controller. With both controllers in automatic, test the disturbance rejection capabilities of the cascade architecture by making step changes in the cooling jacket inlet temperature from 50 ^oC up to 60 ^oC and back to 50 ^oC again. Compare the performance of this cascade architecture in rejecting the disturbance to that of the single loop controller shown below.

   As shown below, the reactor exit temperature for the single loop case ranges during the disturbance from about 88.5 ^oC to about 94.5 ^oC



8) Step the set point from 92 ^oC up to 95 ^oC and back again and compare the results to the above single loop case. Does the cascade architecture offer any benefit in set point tracking performance?

9) Explore how P-Only controller tuning affects disturbance rejection performance. What happens when you increase and decrease the P-Only controller gain of the secondary controller? How does disturbance rejection performance change if you put a PI controller on this inner loop?

   Restore the P-Only controller gain to its design value and explore how tuning adjustments to the outer primary PI controller affect disturbance rejection performance. Does derivative action show benefit?

Workshop 8: Feed Forward Control of the Jacketed Reactor

Objective: To explore feed forward controller design and implementation. Also, to compare disturbance rejection performance of a feed forward controller with that of a lone PI controller for the jacketed reactor process.

Reference: "Practical Process Control" Chapter 12

1) Start the Jacketed Reactor (not the cascade) process. As in the previous workshops, the design level of operation is a reactor exit temperature of 92 ^oC. Reactor exit temperature indicates percent conversion of reactants to products so maintaining this exit temperature is our control objective. The cooling jacket inlet temperature (the disturbance) is expected to be 50 ^oC during normal operation but occasionally may pulse as high as 60 ^oC. Our controller design will focus on rejecting the influence of this disturbance. Begin this study with the process steady at the design level of operation and record the design set point.

$T_{\text{setpoint}} =$

2) A feed forward control element will be used in combination with a traditional feedback controller in this study (called feed forward with feedback trim). The feed forward element is comprised of a process model that describes controller output to process variable dynamic behavior and a disturbance model that describes the disturbance to process variable dynamic behavior.

   (Aside: Feed forward, like cascade control, is used for improved disturbance rejection. Feed forward control is appropriate when one particular disturbance causes a significant disruption to process operation. For feed forward implementation, a sensor must be installed to measure this disturbance. A feed forward model must be programmed to compute a sequence of control actions that will just counter the impact of each disturbance event on the measured process variable.)

   As always, tuning requires a controller output to process variable dynamic data set. For this study, this same data will be used to fit the process model required in the design of the feed forward element. Start saving data to file (say, PROCESS.TXT) and perform an open loop test by stepping the controller output from 42% down to 37% then up to 47% and finally back to 42%. Let the response become clearly established before making each change. Turn file storage off when the response is complete.

3) Disturbance to process variable dynamic data is needed for the disturbance model in the feed forward element. With the process at steady state, start saving data to another file (say, DISTURB.TXT). Step the disturbance (the cooling jacket inlet temperature) from 50 ^oC up to 60 ^oC and after the process has responded, back to 50 ^oC. When the response is complete, stop saving data.

4) Pause the simulation and navigate to *Design Tools.* Open the PROCESS.TXT file and verify that the first, second and third data columns are labeled as time, manipulated variable and process variable, respectively. Fit a first order plus dead time (FOPDT) model to the data and record the PI controller tuning values. Choose the IMC (Internal Model Control) correlation to compute a PI controller gain, K_C, and reset time, τ_I.

$K_C =$ $\tau_I =$

Workshop 8: Feed Forward Control of the Jacketed Reactor (cont.)

Also record the FOPDT model parameters for use in constructing the process model portion of the feed forward element. Include the SSE to gauge of goodness of model fit.

$K_P =$	$\tau_P =$	$\theta_P =$	SSE =

Later we will compare the benefit of different models on feed forward performance. Select a second order plus dead time model (SOPDT) and fit it to the process data. Record the results:

$K_P =$	$\tau_{P,1} =$	$\tau_{P,2} =$	$\theta_P =$	SSE =

Based on the SSE, does the SOPDT model show improved capability in describing process behavior over the FOPDT model?

5) Now open DISTURB.TXT in *Design Tools* and as before, label the first and third data columns as time and process variable. For this data set, recall that it was the disturbance variable that was manipulated during the dynamic experiment. Scroll down the data to verify that the last column in the file contains this disturbance step between 50 and 60 ^oC. You can also edit the file to confirm this. After you have verified the disturbance data, click on the column and press M to label it as the manipulated variable for model fitting.

   Fit a first order plus dead time (FOPDT) model to the data and record the FOPDT model parameters for use in constructing the disturbance model portion of the feed forward element:

$K_D =$	$\tau_D =$	$\theta_D =$	SSE =

   Also select a second order plus dead time model (SOPDT) and fit it to the disturbance data. Record the model parameters and SSE:

$K_D =$	$\tau_{D,1} =$	$\tau_{D,2} =$	$\theta_D =$	SSE =

   Based on the SSE, does the SOPDT model show improved capability in describing the disturbance behavior over the FOPDT model?

6) Close *Design Tools* and return to the jacketed reactor. Click on the controller icon on the reactor graphic. At the top of the controller design menu, select the PID with Feed Forward controller.

   On the left side of the menu, enter the set point from step 1 and PI controller K_C and τ_I from step 4. On the right side of the menu, click on the Process tab and enter your FOPDT process model parameters. Then click on the Disturb(ance) tab and enter your FOPDT disturbance model parameters. Leave parameters not included in the model as zero. Click Done to put the controller in automatic.

   (Aside: For the feed forward model to be physically realizable (mathematically solvable), the dead time of the disturbance model must be greater than or equal to the dead time of the process model. Otherwise, the math would say that the feed forward corrective actions must start before the disturbance even occurs! This is not a limitation of Control Station but a physical and mathematical reality of all feed forward elements. If your model fits from steps 4 and 5 do not meet this criterion, you must arbitrarily change the disturbance or process model dead time to make them equal. Unfortunately, such a change will reduce model accuracy and degrade feed forward performance.)

Workshop 8: Feed Forward Control of the Jacketed Reactor (cont.)

7) Test the disturbance rejection performance of this feed forward with feedback trim implementation. Make step changes in the disturbance from 50 up to 60 ^{o}C and back to 50 ^{o}C again. When the response is complete, pause the simulation.

   To determine the benefits of the feed forward element, click on the controller icon and from the controller menu select the regular PID controller (without feed forward). Start the simulation and repeat the step changes in the disturbance from 50 up to 60 ^{o}C and back to 50 ^{o}C.

   How do the two controllers compare? Does the feed forward element show clear benefit? Is the PI tuning you used for the non-feed forward study correct or to make the comparison fair, do you need to adjust the tuning values?

8) Return the controller to PID with Feed Forward and on the right side of the menu, enter your SOPDT model parameters for the process and disturbance. Leave parameters not included in the model as zero. Click Done to put the controller in automatic. Remember that for the feed forward model to be physically realizable (mathematically solvable), the dead time of the disturbance model must be greater than or equal to the dead time of the process model. If need be, shorten the process dead time to equal the disturbance dead time and test the feed forward disturbance rejection performance. Make step changes in the disturbance from 50 ^{o}C up to 60 ^{o}C and back to 50 ^{o}C again.

   The SOPDT models (hopefully) had lower SSE values than the FOPDT models, implying better model fits of the data. Does the SOPDT models show clearly improved feed forward disturbance rejection performance? Why or why not?

9) Compare the disturbance rejection performance of static feed forward to dynamic feed forward. Open the controller design menu and on the model entry portion of the form, click the box to switch to static feed forward.

   (Aside: Static feed forward only uses the process and disturbance gains in the feed forward calculation. Thus, the dead time model parameters that account for the delay that occurs before a disturbance reaches the measured process variable, and the time constant model parameters that account for the speed at which each disturbance disrupts the measured process variable, are not considered in the static feed forward calculation. With static feed forward, each feed forward action is implemented immediately and completely upon detection of a disturbance.

   The benefit of static feed forward control is that it requires no programming. Often, static feed forward can be implemented with conventional control loop instrumentation.)

   Test the static feed forward disturbance rejection performance by making step changes in the disturbance from 50 ^{o}C up to 60 ^{o}C and back to 50 ^{o}C again. Is there any clear change in disturbance rejection performance compared to the fully dynamic model form?

Workshop 8: Feed Forward Control of the Jacketed Reactor (cont.)

10) Return to the dynamic FOPDT Feed Forward model parameters (Step 6) and perform a sensitivity study to determine how each feed forward model parameter impacts the disturbance rejection performance. To do this, first double and then halve the process model gain and perform a disturbance rejection experiment after each change. Keep all the other model parameters at their original design values. Repeat this procedure for the process model time constant and finally the process model dead time. Do this study a second time for the disturbance model parameters.

    Which of the model parameters most impacts disturbance rejection performance? For each model parameter, determine if it is more conservative (less aggressive) to have it be too large or too small.

11) Explore the benefit of feed forward control on set point tracking performance. Step the set point from 92 ^{o}C up to 95 ^{o}C and back again. Now use the controller design menu to change the controller to traditional PID (no feed forward). Keep the PI controller tunings unchanged and repeat the set point step. Does the feed forward element offer any benefit in set point tracking performance?

Workshop 9: Advanced Feed Forward Control of Heat Exchanger

Objective: *To explore advanced modeling issues and the impact of plant-model mismatch on feed forward controller performance.*

Reference: *"Practical Process Control" Chapter* 12

1) The design objective for this heat exchanger study is to maintain a constant measured exit stream temperature of 135 ^{o}C. During normal operation, the warm liquid disturbance flow rate is steady at about 10 L/min, but up stream operation occasionally causes the disturbance stream flow rate to jump as high as 20 L/min. Rejecting the impact of this disturbance is our design objective.

   Start the heat exchanger and design, implement and validate a PI controller. That is,
   - move the process to the design operating level,
   - generate controller output to measured process variable dynamic data around this design level and save the data to file called, say, PROCESS.TXT,
   - fit a first order plus dead time (FOPDT) model to the data,
   - use the resulting parameters in a disturbance rejection tuning correlation to obtain initial values for PI controller gain, K_C, and reset time, τ_I,

   If the FOPDT fit appears satisfactory, record the model parameters before leaving *Design Tools*:

   $K_P =$ $\tau_P =$ $\theta_P =$ SSE =

   Test the controller in rejecting step changes in the warm liquid disturbance flow from 10 L/min up to 20 L/min and back again. Recall that the heat exchanger has a strong nonlinear character. It is likely that you will have to adjust the parameters by trial and error to achieve a best tuning for disturbance rejection. Record this best tuning:

   $K_C =$ $\tau_I =$

2) A feed forward control element is comprised of a process model that describes controller output to process variable dynamic behavior and a disturbance model that describes the disturbance to process variable dynamic behavior. In step 1, you already computed the required process model.

   Generate the dynamic data needed to determine the disturbance model. With the process at steady state, start saving data to another file (say, DISTURB.TXT). Step the warm oil disturbance flow rate from 10 L/min up to 20 L/min. Because of the complicated models we will use to fit this data, let the measured exit temperature pattern evolve all the way to a clear steady state. When the response is complete, step the disturbance flow rate back to 10 L/min and again wait for the response to complete. Stop saving data.

   Why does the heat exchanger show this inverse response to a disturbance variable step?

3) Pause the simulation and navigate to *Design Tools.* Open DISTURB.TXT and label the first and third data columns as time and process variable. It was the disturbance variable that was manipulated during the experiment. Verify that the last data column in the file contains this disturbance step between 10 and 20 L/min. You can edit the file if need be to confirm this. Then, click on the column and press M to label it as the manipulated variable for model fitting.

Workshop 9: Advanced Feed Forward Control of Heat Exchanger (cont.)

Fit a first order plus dead time (FOPDT) model to the disturbance data and record the model parameters for use in constructing the disturbance model portion of the feed forward element:

$K_D =$	$\tau_D =$	$\theta_D =$	SSE =

Fit a second order plus dead time model (SOPDT) to the disturbance data. Record the model parameters and SSE:

$K_D =$	$\tau_{D,1} =$	$\tau_{D,2} =$	$\theta_D =$	SSE =

Fit a second order plus dead time with lead time model (SOPDT w/ Lead) to the disturbance data. Record the model parameters and SSE:

$K_D =$	$\tau_{D,1} =$	$\tau_{D,2} =$	$\tau_{D,L} =$	$\theta_D =$	SSE =

Based on the SSE and your observations of the goodness of model fit, does the SOPDT model show improved capability in describing the disturbance behavior over the FOPDT model? How about the SOPDT w/ Lead model?

(Aside: *Design Tools* must search for five model parameters that minimize the SSE when fitting a SOPDT w/ Lead model. This is quite a challenge for any optimization tool. The fit routine works well as long as the data contains a lead term influence. To learn about this, go to the single loop *Custom Process* and construct a second order plus dead time with lead time model. First put in a large positive lead term, say twice the size of the largest time constant, and perform an open loop test by stepping the controller output. Repeat with a large negative lead term. Repeat a third time with a zero lead term. The lead term influence should be clear. If a process has a zero lead term influence, *Design Tools* may converge on a fit that looks reasonable but beware of nonsense results. For example, the lowest SSE can sometimes be achieved mathematically for such data with a fantastically large τ_L and a K_P near zero. It is your job not only to visually accept model fits, but also to ask if the model parameters make sense.)

4) Close *Design Tools* and return to the heat exchanger. Select the PID controller and enter your best PI tuning values from step 1. Keep these values constant for all tests hereafter. Create a baseline graph showing the performance of a lone feedback controller with no feed forward element in rejecting a disturbance step from 10 L/min to 20 L/min and back again. Pause the simulation when the response is complete.

   Select the PID with feed forward controller. Click on the Process tab of the feed forward model and enter the FOPDT process model parameters from step 1. Leave parameters not included in the model as zero.

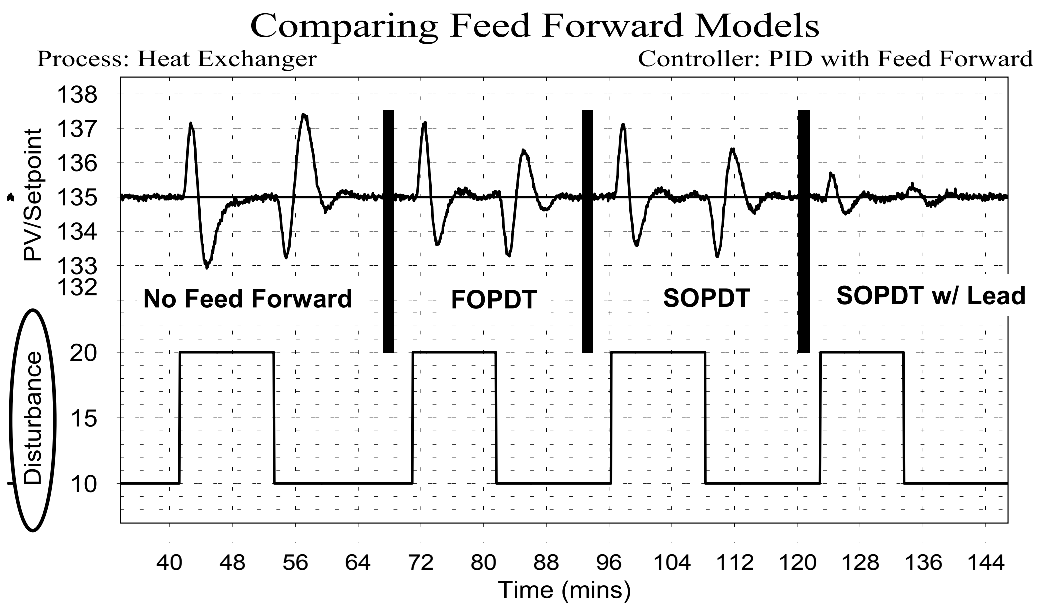
   Click on the Disturb(ance) tab of the feed forward model and enter the FOPDT disturbance model parameters from step 3. Click Done to put the controller in automatic. Show the performance of a feed forward controller with feedback trim in rejecting a disturbance step from 10 L/min to 20 L/min and back again. Pause the simulation when the response is complete.

Workshop 9: Advanced Feed Forward Control of Heat Exchanger (cont.)

(Aside: Recall that for the feed forward model to be physically realizable, the dead time of the disturbance model must be greater than or equal to that of the process model. If your models do not meet this criterion, you must change one of the model dead times to make them equal.)

5) Return to the controller design menu, click on the Disturb(ance) tab of the feed forward model, enter the SOPDT disturbance model parameters and click Done to put the controller in automatic. Show the performance of the controller in rejecting a disturbance step from 10 L/min to 20 L/min and back again. Pause the simulation when the response is complete.

   Repeat one more time for the SOPDT w/ Lead disturbance model. Pause the simulation when the response is complete and click on View and Print Plot to view a fixed plot of the investigation. Use the Plot Options icon to create a plot similar to that shown below.



Tuning: Gain = -1.80, Reset Time = 1.10, Deriv Time = 0.0, Sample Time = 1.00
Process Model: Gain(Kp) = -0.39, T1 = 0.96, T2 = 0.0, TD = 0.80, TL = 0.0
Disturbance Model: Gain(Kd) = -0.146, T1 = 0.735, T2 = 0.735, TD = 0.822, TL = -3.41

6) Why does feed forward with a FOPDT disturbance model only show modest benefit over the disturbance rejection performance of a feedback controller with no feed forward element at all?

   Why does a SOPDT disturbance model show no benefit in disturbance rejection performance compared to that of the FOPDT model?

   Why does the SOPDT w/ Lead disturbance model not show perfect disturbance rejection performance?

9) Compare the disturbance rejection performance of static feed forward to the performance of dynamic feed forward with the SOPDT w/ Lead model form. Step the disturbance from 50 ^{o}C up to 60 ^{o}C and back to 50 ^{o}C again. Pause the simulation after the response has completed.

   Open the controller design menu and on the model entry portion of the form, click the box to switch to static feed forward.

   (Aside: Static feed forward only uses the process and disturbance gains in the feed forward calculation. Thus, the dead time model parameters that account for the delay that occurs before a disturbance reaches the measured process variable, and the time constant model parameters that account for the speed at which each disturbance disrupts the measured process variable, are not considered in the static feed forward calculation. With static feed forward, each feed forward action is implemented immediately and completely upon detection of a disturbance.

   The benefit of static feed forward control is that it requires no programming. Often, static feed forward can be implemented with conventional control loop instrumentation.)

   Test the static feed forward disturbance rejection performance by again making step changes in the disturbance from 50 ^{o}C up to 60 ^{o}C and back to 50 ^{o}C again. Is there any clear change in disturbance rejection performance compared to the fully dynamic model form?

7) Compare the set point tracking performance of a feedback controller (no feed forward element) to that of your best feed forward with feedback trim design. Use the same PI tuning parameters for both set point tracking experiments. Does the feed forward element offer any benefit in set point tracking performance?

Workshop 10: Multivariable Decouplers and Distillation Control

Objective: To observe control loop interaction on a multi-loop process and to design and explore model-based decouplers that minimize such interaction.

Reference: "Practical Process Control" Chapter 13

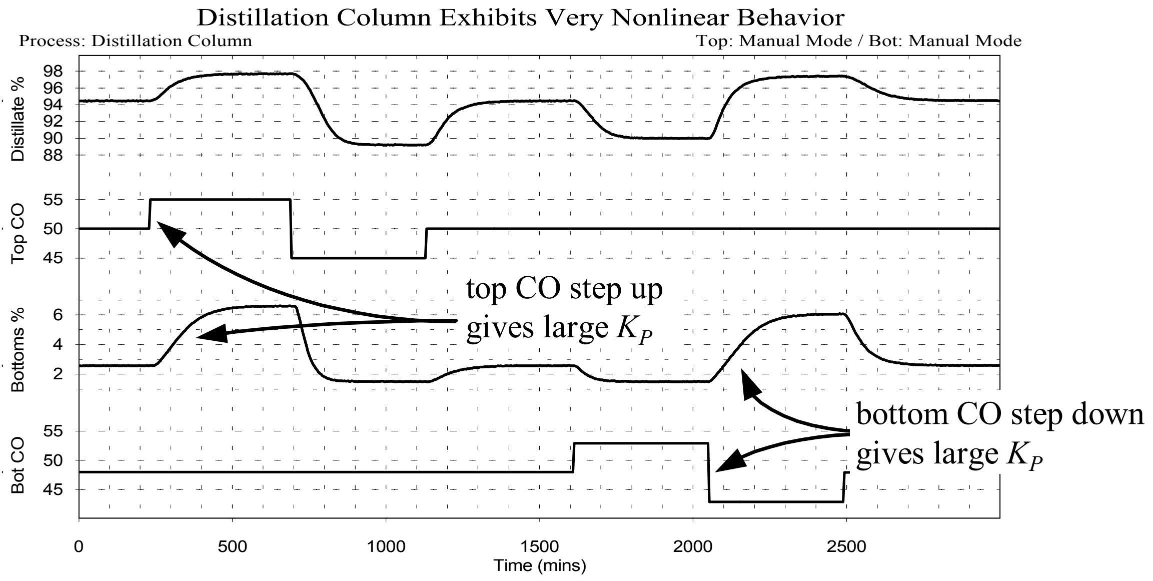
1) Start the distillation column simulation and observe the process graphic. This column separates a mixture of benzene and toluene into a top and bottom stream. As shown on the graphic, there is a controller at the top of the column that manipulates the reflux flow rate to control the distillate composition. A second controller at the bottom of the column that manipulates the steam rate to control the bottoms composition. The design level of operation for this study is:

   - distillate composition of 94.5% benzene
   - bottoms composition of 2.6% benzene (97.4% toluene)
   - total feed flow rate to column of 596 Kg/min

   Record the design set point values for the top distillate controller and bottoms controllers:

   $y_{\text{setpoint}}^{\text{top}} =$ % benzene $\qquad$ $y_{\text{setpoint}}^{\text{bottom}} =$ % benzene

2) Before designing the controllers, explore the dynamics of this process. Step the top controller output (CO) from 50% up to 55%, down to 45% and back to 50%. Repeat this experiment with the bottom controller, stepping it from 47.9% up to 52.9%, down to 42.9% and back to 47.9%. Compare your results to the plot below and verify that it represents the dynamic behavior of the distillation column.



   This plot shows that the top composition loop is only mildly nonlinear.

   The bottoms composition loop, however, is extremely nonlinear. In particular, the process gain of the bottom loop is *three times larger* when the bottoms composition is increasing from the design level of operation compared to when it is decreasing, regardless of whether it is the top or bottom controller forcing the response.

(Aside: The extreme nonlinear behavior makes this process quite challenging to control. To meet this challenge, we take advantage of the information in the above plot to design the dynamic tests. Rather than the standard doublet, we will pulse each controller output in one direction only, the direction that yields parameters leading to the most stable 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 values of K_C. Exploiting the information contained in the above figure, we achieve the largest values of K_P by pulsing the top controller up and bottom controller down when generating our dynamic test data.)

3) Remember that both loops should remain in manual mode throughout the dynamic tests.

   For the top loop, start saving data to file (say, TOP.TXT) and pulse the top distillate controller output from 50% up to 55% and back to 50%. Let the response become well established before making the second step. Turn file storage off when the response is complete.

   Let the column settle out to steady state. For the bottom loop, start saving data to file (say, BOTTOM.TXT). Step the bottom controller output from 47.9% down to 42.9% and back to 47.9%. Again wait for the response to complete before making the return step. Turn file storage off when the test is complete.

4) Use *Design Tools* to compute PI tuning values for the top distillate controller. Pause the distillation simulation, navigate to *Design Tools* and read in TOP.TXT. The Label Data Columns screen will display eight columns of data because the file contains data for both the distillate and bottoms loop. It is essential in this (and any) study that the columns be properly labeled before starting a model fit. If you are unsure which columns of data are for the top distillate loop, click the Edit Data button in the upper right corner of the Label Data Columns screen and read the headings contained in the file.

   For this study, the second column holds the top distillate controller output data and the third column holds the top distillate measured process variable data. Be sure these are labeled as such before starting the model fit. Fit a FOPDT model to the data and if the fit seems reasonable, record the PI controller tuning values from the IMC tuning correlation:

   Top distillate controller PI tuning parameters: $K_C =$ $\tau_I =$

   Remember that if the model does not reasonably approximate the data, the controller tuning values computed from the model parameters will be suspect.

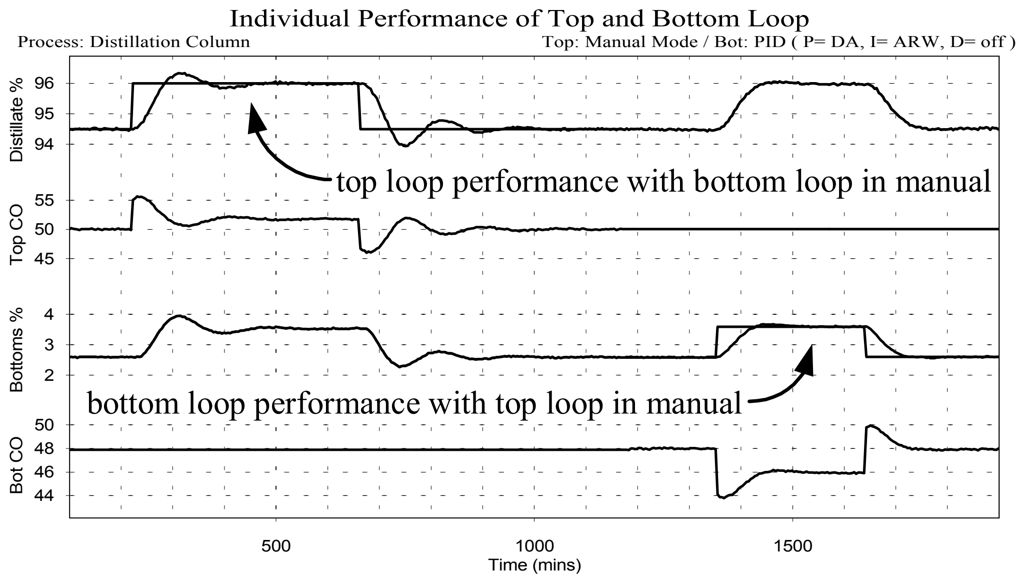
   Repeat the process to tune a PI controller for the bottom loop. Read BOTTOM.TXT into *Design Tools.* The fifth and sixth columns hold the controller output and measured process variable data for the bottom loop. Label these columns accordingly. Fit a FOPDT model to the data and check that the fit seems reasonable. Record the PI controller tuning values from the IMC tuning correlation:

   Bottom controller PI tuning parameters: $K_C =$ $\tau_I =$

5) Exit *Design Tools* and return to the distillation simulation. Implement your PI controller design for the top distillate loop while leaving the bottom loop in manual. Test the top controller by stepping the distillate composition set point from 94.5% benzene up to 96% and back again.

   Put the top loop back in manual and implement your PI controller design for the bottom loop. Test this controller by stepping the bottoms composition set point from 2.6% benzene up to 3.6% and back again.

   Verify that each of your controllers performs similarly to that shown below.



6) Now put <u>both</u> PI controllers in automatic, step the top distillate composition set point from 94.5% benzene up to 96% and watch the loop interaction.

   (Aside: As the top controller sends more cold reflux down the column to increase the purity of the distillate, it has the undesirable consequence of causing the bottom of the column to cool. This moves the bottoms composition from set point. The bottom controller responds by increasing the steam rate, which heats up the bottom of the column as desired. It results, however, in hot vapors traveling up the vessel and heating the top of the column at the same time the top controller is trying to cool it. Over time, you will see both the reflux rate and the steam rate increase more and more as each controller "fights" for its own measured process variable. The result is a dismal overall performance.)

   When you have established the negative impact of loop interaction, return the set point to 94.5% and let the process reach steady state.

Workshop 10: Multivariable Decouplers and Distillation Control (cont.)

7) (Aside: A multivariable decoupler is a feed forward element where the measured disturbance is the output signal of the cross-loop controller. To construct a decoupler, we need to identify a process and cross-loop disturbance model for both the top and bottom loop (four models total). No additional process testing is necessary to create these models because the tests just completed contain the information we need.

When the top controller output signal changes (thus changing the reflux rate), it not only causes a response in the top distillate composition so we can fit a process model, but it also acts as a disturbance to the bottoms composition. Thus, the cross-loop disturbance data we need for the bottom decoupler is already contained in file TOP.TXT.

And when the bottom controller output signal changes (thus changing steam rate), it not only causes a response in the bottoms composition so we can fit a process model, but it also acts as a disturbance to the top distillate composition. Hence, the cross-loop disturbance data we need for the top decoupler is already contained in file BOTTOM.TXT.)

To design the top distillate decoupler, the process and disturbance models needed are:
- Process model: $G_{DT}(s)$ = <u>d</u>istillate composition response to <u>t</u>op controller output model
- Disturbance model: $G_{DB}(s)$ = <u>d</u>istillate composition response to <u>b</u>ottom controller output model

For the bottom decoupler, the process and disturbance model parameters needed are:
- Process model: $G_{BB}(s)$ = <u>b</u>ottoms composition response to <u>b</u>ottom controller output
- Disturbance model: $G_{BT}(s)$ = <u>b</u>ottoms composition response to <u>t</u>op controller output

To obtain the process model for the top decoupler, $G_{DT}(s)$, read TOP.TXT into *Design Tools*. The second column holds the top controller output data. The third column holds the top distillate process variable data (you can verify this with the Edit Data button). Label these columns as such and fit a FOPDT model. If the fit is acceptable, record the model below as the distillate decoupler process model.

To obtain the cross-loop disturbance model for the top decoupler, $G_{DB}(s)$, read BOTTOM.TXT into *Design Tools*. The fifth column holds the bottom controller output data. The third column holds the top distillate process variable data. Label these columns as such and fit a FOPDT model. If the fit is acceptable, record the model below as the distillate decoupler cross-loop disturbance model.

<u>Top Distillate Decoupler:</u>

Process Model (distillate composition response to top controller output)

$K_P =$ $\tau_P =$ $\theta_P =$

Cross-Loop Disturbance Model (distillate composition response to bottom controller output)

$K_D =$ $\tau_D =$ $\theta_D =$

In absolute value, if $|K_D| > |K_P|$, then the decoupler assigns more influence over the process variable to the cross-loop disturbance than it does to the controller. This can lead to unstable control. If your fits result in this situation, lower your K_D above so $|K_D| = |K_P|$.

Workshop 10: Multivariable Decouplers and Distillation Control (cont.)

8) To obtain the process model for the bottom decoupler, $G_{BB}(s)$, read BOTTOM.TXT into *Design Tools*. The fifth column holds the bottom controller output data. The sixth column holds the bottom process variable data. Label these columns as such and fit a FOPDT model. If the fit is acceptable, record the model below as the bottoms decoupler process model.

   To obtain the cross-loop disturbance model for the bottom decoupler, $G_{BT}(s)$, read TOP.TXT into *Design Tools*. The second column holds the top controller output data. The sixth column holds the bottoms process variable data. Label these columns as such and fit a FOPDT model. If the fit is acceptable, record the model below as the distillate decoupler disturbance model.

   Bottoms Decoupler:

   Process Model (bottoms composition response to bottom controller output)

   $K_P =$ $\tau_P =$ $\theta_P =$

   Cross-Loop Disturbance Model (bottoms composition response to top controller output)

   $K_D =$ $\tau_D =$ $\theta_D =$

   In absolute value, if $|K_D| > |K_P|$, then the decoupler assigns more influence over the process variable to the cross-loop disturbance than it does to the controller. This can lead to unstable control. If your fits result in this situation, lower your K_D above so $|K_D| = |K_P|$.

9) Return to the distillation simulation, select the top controller, choose PID with Decoupler from the menu list, enter your process and disturbance models for the distillate decoupler, and click Done. Repeat this for the bottoms controller. For both loops, leave your original PI tuning values unchanged.

   With both the top and bottom loop under automatic PI Control with Decoupler, test the process by stepping the top distillate composition set point from 94.5% benzene up to 96% and back again.

   Is your controller stable? Do the decouplers decrease loop interaction? Is the set point tracking performance similar to when only the top loop was in automatic (step 5 above)?

10) Now test the benefit of decouplers on disturbance rejection. With the process at steady state, step the column feed rate from 596 Kg/min up to 650 Kg/min and after the response is complete, step it back again.

   Do this both under PI with Decouplers control, and again under just PI control. Does decoupling help with disturbance rejection?

Workshop 11: Dead Time Compensation Using the Smith Predictor

Objective: To design, implement and explore the capabilities of the model-based Smith predictor on processes with significant dead time.

Reference: "Practical Process Control" Chapter 14

1) Begin this study by constructing a linear process simulation using *Custom Process*. Click on the *Custom Process* button on Control Station's main screen and then choose Single Loop Process from the list. When the simulation starts, click the Process button on the graphic. This opens a "Construct Process and Disturbance Models" menu.

   Click the Process Model tab (it should already be active if you have not done any exploring) and select the Overdamped Linear Model from the list of models available. Enter these process parameters to create a third order without dead time process:

Process Gain, K_P	=	0.8
First Time Constant, τ_{P1}	=	70.0
Second Time Constant, τ_{P2}	=	50.0
Third Time Constant, τ_{P3}	=	10.0
Lead Time, τ_{PL}	=	0
Dead Time, θ_P	=	0

   We will use the default Zeros and Spans in this study, but you should click that tab to confirm that all parameters range from 0-100 with startup values equal to 50. We will not specify a model describing the disturbance behavior right now so leave the default values in place.

   When you are finished, click Done at the bottom of the menu to begin the simulation of your third order without dead time process.

2) While saving data to file, perform a dynamic test at the startup operating conditions of the process (CO = 50 and PV = 50). When the test is complete, navigate to *Design Tools*, fit a FOPDT model to the data and record the PI tuning values using the IMC (Internal Model Control) tuning correlation:

   PI tuning values: K_C = τ_I =

   When you fit a FOPDT model to this third order without dead time process, does the model fit produce a positive value for dead time? Why?

3) Return to *Custom Process* and test the set point tracking performance of this controller on your third order without dead time process. Starting from steady state, make a set point step from 50 up to 55. When the response is complete, click View and Print Plot. Using the plot options as needed to help you visualize the response, compute and record the rise time and peak overshoot ratio for this base case:

   t_{rise} = POR =

   We will call this peak overshoot ratio and rise time our design or base case performance (your POR should be in the range of 15-25% and the measured process variable should settle completely to the new set point in roughly one cycle of the measured process variable. If this is not true, you should recheck your design.)

4) Explore the impact of dead time on controller performance. Click on the Process block on the control loop graphic and add a dead time of 25 to your existing third order process. Click Done to start the simulation. *Do Not* change the controller tuning.

Make a set point step from 50 up to 55 and from the screen plot, determine the rise time and peak overshoot ratio for your controller:

$t_{\text{rise}} =$ POR =

Comment on the change in these values due to the presence of dead time.

5) Put the controller in manual mode and use *Design Tools* to design a PI controller for this third order plus dead time process using the IMC tuning correlations. Record both the PI tuning values and the FOPDT model parameters computed by *Design Tools*. Also fit a SOPDT model that describes the dynamic behavior of this process and record those parameters:

PI tuning values:	$K_C =$	$\tau_I =$		
FOPDT model:	$K_P =$	$\tau_P =$	$\theta_P =$	
SOPDT model:	$K_P =$	$\tau_{P1} =$	$\tau_{P2} =$	$\theta_P =$

6) Return to Custom Process and implement your updated tuning values. Make a set point step from 50 up to 55. If the peak overshoot ratio is not similar to the base case (the no dead time process of step 3), adjust tuning by trial and error until it is. Record the peak overshoot ratio and rise time for this third order with dead time process:

$t_{\text{rise}} =$ POR =

Comment on how the dead time has impacted the performance on this properly designed controller.

7) From the controller design menu, select the PID with Smith Predictor controller. A process model form will open. Enter the FOPDT model parameters from step 5 as the predictive process model for the Smith architecture.

For PI controller tuning, use the IMC tuning correlations. You should use the FOPDT K_P and τ_P from step 5, but since a Smith predictor can theoretically eliminate dead time, use $\theta_P = 0.1\tau_P$ for the dead time in the correlation.

(Aside: There are no tuning correlations for a PI controller used in a Smith predictor architecture. Since the Smith predictor theoretically eliminates the impact of dead time on control performance, we use a theoretical minimum dead time in the correlations to compute PI tuning values. For commercial equipment, the theoretical minimum dead time is the controller sample time. That is, one sample time passes between a control action and when the controller receives the next measurement. Since sample time, T, should be less than or equal to $0.1\tau_P$, use $\theta_P = 0.1\tau_P$ and the process gain and time constant from the FOPDT fit of the process test data in the IMC tuning correlations.)

Workshop 11: Dead Time Compensation Using the Smith Predictor

Recall the IMC tuning correlation for the PI controller:

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

$$K_C = \qquad \tau_I =$$

With these tuning values and the FOPDT process model implemented as the Smith predictive model, make a set point step from 50 up to 55.

How does the rise time and peak overshoot ratio compare with the values you achieved in step 3.

$t_{\text{rise}} =$ POR =

Don't forget that even in the perfect case, the rise time must increase by 25 because of the dead time added to the process.

8) Perform a sensitivity study to determine which of the three FOPDT model parameters most impact set point tracking performance. To do this, change first the model gain, then the time constant and finally the dead time implemented in your Smith predictor model while keeping the other two constant at the proper values.

   After each change, perform a set point tracking experiment. Which of the model parameters most impacts performance? What is its effect?

9) Replace the FOPDT Smith model with the SOPDT model from step 5. Use the base case tuning values from step 2 (no dead time) for the PI controller. Repeat the set point tracking experiment.

   Does the second order model provide any improvement to control performance? Is there any impact on the trace of the controller output signal? Is trial and error tuning required to achieve the design rise time and peak overshoot ratio?

Optional

10) Click the Disturbance button on the control loop graphic to open the “Construct Process and Disturbance Models” menu. Enter as a disturbance simulation the same model used for the process from step 1. Use the PI tuning (no Smith predictor) from step 2 and test the disturbance rejection performance of the controller by stepping the disturbance variable from 50 to 55.

    Click on the Disturbance block on the control loop graphic and add a dead time of 25 to your existing third order process. Test the disturbance rejection performance of the controller by stepping the disturbance variable from 50 to 55. Compare the result to that above. Does disturbance dead time impact disturbance rejection performance?

    Select the Smith predictor controller and input the FOPDT process model from step 5 and PI tuning values from step 7. Repeat the disturbance test and draw a conclusion on the benefit of the Smith predictor when used for disturbance rejection applications.