

Tutorial: Cascade vs. Feed Forward for Improved Disturbance Rejection

Doug Cooper (speaker), Robert Rice, Jeff Arbogast
Chemical Engineering Dept, unit 3222
University of Connecticut
191 Auditorium Rd
Storrs, CT 06269-3222

KEYWORDS

Cascade control, feed forward control, disturbance rejection, load disturbance, PID control, software

ABSTRACT

The most popular architectures for improved regulatory performance are cascade control and feed forward with feedback trim. Both architectures trade off additional complexity in the form of instrumentation and engineering time in return for a controller better able to reject the impact of disturbances on the measured process variable. Neither architecture benefits nor detracts from set point tracking performance. This paper compares and contrasts the two architectures. A comparative example is presented using a jacketed reactor simulation.

FEED FORWARD ARCHITECTURE

The feed forward with feedback trim architecture requires an additional sensor and the programming of a dynamic model. The sensor is installed to directly measure changes in the disturbance variable. The model receives the disturbance measurement, computes control actions to counter its impending impact on the measured process variable, and transmits the result to the controller for execution. Because of this architecture, feed forward is useful when one specific disturbance variable is responsible for costly disruptions to stable operation. It is also useful when a secondary measured process variable cannot be identified to construct a control cascade.

Consider that many disturbances originate in some other part of the plant. A measurable series of events occur that cause that “distant” event to impact your process. A feed forward controller gains advantage by using a sensor to directly measure the disturbance *before* it reaches the process. As shown in Fig. 1, a feed forward element receives the disturbance measurement signal and uses it to compute and schedule preemptive control actions that will counter the impact of the disturbance just as it reaches the measured process variable.

A feed forward implementation requires the purchase and installation of a sensor and the construction of a feed forward model element. This element is comprised of a disturbance model and a process model. Both models are linear in form. The computation performed by the feed forward element may be thought of as a two step procedure:

- The *disturbance* model receives disturbance measurement, $d(t)$, and predicts an “impact profile,” or when and by how much the measured process variable, $y(t)$, will be impacted.
- Given this predicted sequence of disruption to $y(t)$, the *process* model then back calculates a series of control actions, $u_{\text{feedforward}}(t)$, that will exactly counteract the disturbance as it arrives so the measured process variable remains constant at set point.

Implementation requires that these linear models be programmed into the control computer. Linear models never exactly describe real process behavior. So although a feed forward element can dramatically reduce the impact of a disturbance, it will not succeed in providing perfect control.

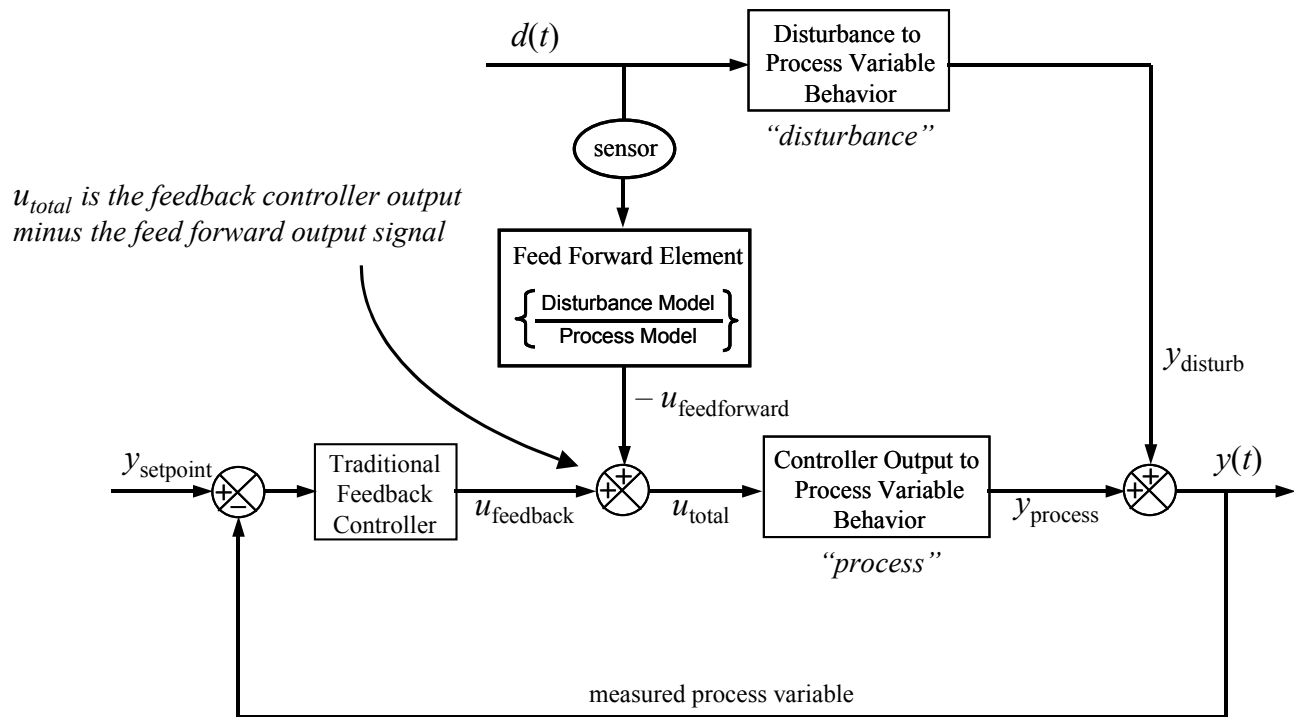


Figure 1 - Architecture of a feed forward controller with feedback trim

Feed forward is combined with traditional feedback control trim, $u_{\text{feedback}}(t)$, to create a total controller output, $u_{\text{total}}(t)$. Feedback trim: rejects those portion of the measured disturbance that make it past the feed forward element, works to reject all other unmeasured disturbances affecting plant operation, and provides set point tracking capabilities as needed.

The computed feed forward control action, $u_{\text{feedforward}}$, is subtracted from the feedback output signal to create a total controller output:

$$u_{\text{total}}(t) = u_{\text{feedback}}(t) - u_{\text{feedforward}}(t) \quad (1)$$

CASCADE ARCHITECTURE

Cascade control implementation is a familiar task because the architecture is comprised of two ordinary controllers from the PID family. Cascade is specifically designed for improved disturbance rejection.

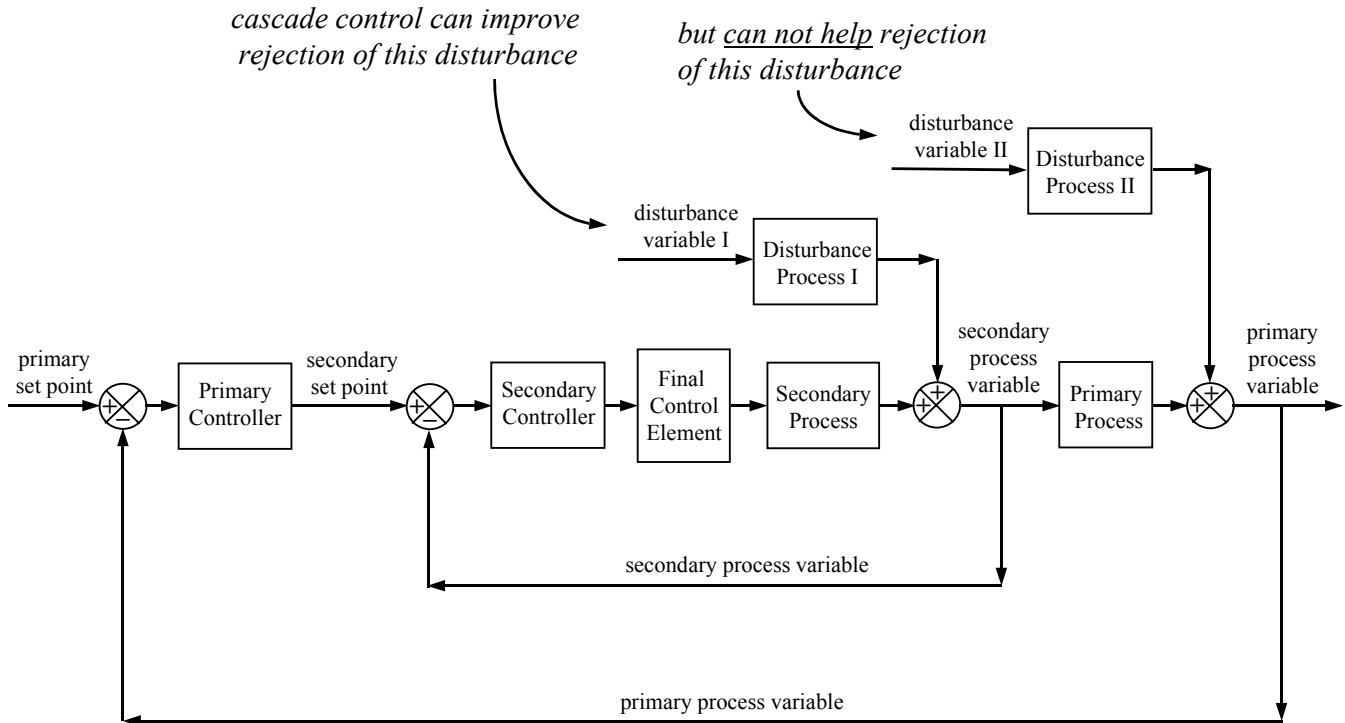


Figure 2 – Block diagram of the cascade architecture

In a traditional feedback loop, a controller adjusts a manipulated variable so the measured process variable remains at set point. The cascade design requires that you identify a *secondary* process variable (call the main process variable associated with original control objective the *primary* variable). This secondary process variable must meet certain criteria:

- it must be measurable with a sensor,
- the same valve used to manipulate the primary variable must also manipulate the secondary variable,
- the same disturbances that disrupt the primary variable must also disrupt the secondary variable,
- the secondary variable must be *inside* the primary process variable, which means it responds well before the primary variable to disturbances and final control element manipulations.

With a secondary process variable identified, a cascade is constructed as shown in Fig. 2. The block diagram shows how both Disturbance I and the final control element impact the secondary variable before they affect the primary variable. Notice that the secondary loop has a traditional feedback control structure, except here it is literally nested inside the primary loop.

A cascade requires two sensors and two controllers but only one final control element because the output of the primary controller, rather than going to a valve, becomes the set point of the secondary controller.

With this nested architecture, *success in a cascade implementation requires that the settling time of the (inner) secondary loop is significantly faster than the settling time of the primary (outer) loop.*

Cascade loop tuning uses the same skills as tuning a regular PID controller:

- begin with both the primary and secondary controllers in manual mode.
- select a P-Only controller for the inner secondary loop (integral action increases settling time and offset is rarely an issue for the secondary process variable).
- tune the secondary P-Only controller using a set point tracking criterion (its main job is to respond to set point commands from the primary controller). Test it to ensure a satisfactory performance.
- leave the secondary controller in automatic; it now literally becomes part of the primary process.
- select and tune a controller with integrating action for the primary loop (PI or PID)
- with both controllers in automatic, tuning of the cascade is complete.

EXAMPLE PROCESS

A jacketed reactor simulation from controlstation.com (Cooper, 2004) is used to contrast and compare the cascade and feed forward architectures. The reactor is a continuously stirred vessel in which an exothermic (heat producing) reaction occurs. Residence time is constant in this well mixed reactor, so the conversion of feed to desired product can be inferred from the temperature of the reactor exit stream. The process graphic for the single loop and cascade controller architecture is shown in Fig. 3.

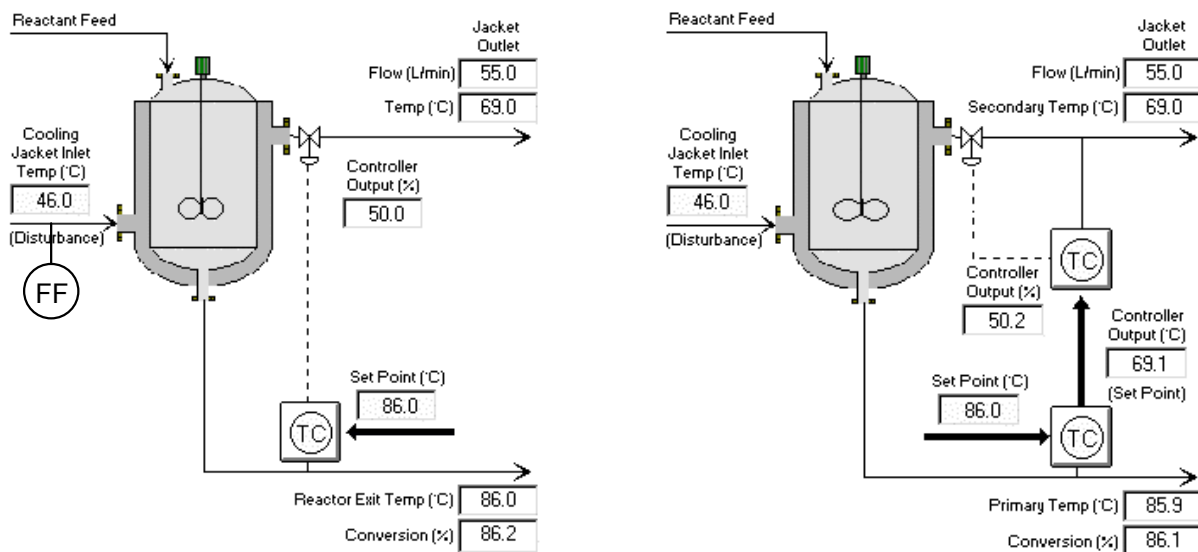


Figure 3 – Jacketed reactor process with single loop (left) and cascade control (right) architecture

To control exit stream temperature, the vessel is enclosed in a cooling jacket. The controller manipulates a valve to adjust the cooling liquid flow rate. If the exit stream temperature (and thus conversion) is too high, the controller opens the valve. This increases cooling liquid flow rate, which cools off the reactor and causes the heat producing reaction to slow. Ultimately, the measured temperature of the stream

exiting the reactor drops in response to the control action. As shown in Fig. 3, the disturbance variable of interest for this process is the temperature of cooling liquid entering the jacket.

The control objective is to maintain the reactor exit stream temperature at set point by rejecting disturbances caused by changes in the cooling jacket inlet temperature. The design level of operation is a reactor exit stream temperature (measured process variable) of 86°C. The cooling jacket inlet temperature (disturbance) is normally at its design value of 46°C, but our concern is that it is known to unexpectedly spike as low as 40°C. An open loop study establishes that a controller output of 50% causes the reactor to steady at the design measured exit stream temperature of 86°C when the cooling jacket inlet temperature is at its expected or design value of 46°C.

SINGLE LOOP DISTURBANCE REJECTION

Figure 4 shows a dynamic test where the controller output is stepped from the design value of 50% up to 53%, then down to 47%, and finally back to 50%, forcing the measured reactor exit stream temperature to exhibit a clear response after each controller output step that dominates any measurement noise.

A FOPDT (first order plus dead time) fit of the dynamic process data is computed by controlstation.com software to yield the model parameters:

Process Gain, $K_p = -0.36 \text{ } ^\circ\text{C}/\%$

Time Constant, $\tau_p = 1.58 \text{ min}$

Dead Time, $\theta_p = 0.88 \text{ min}$

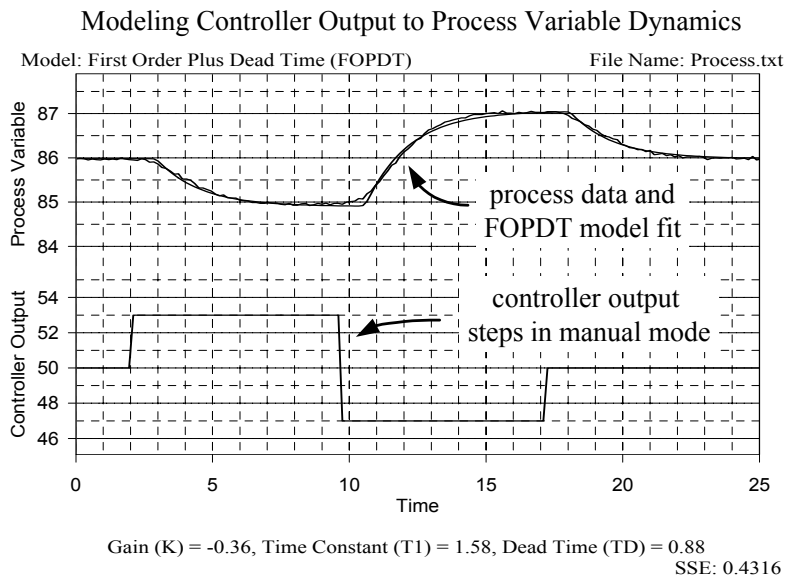


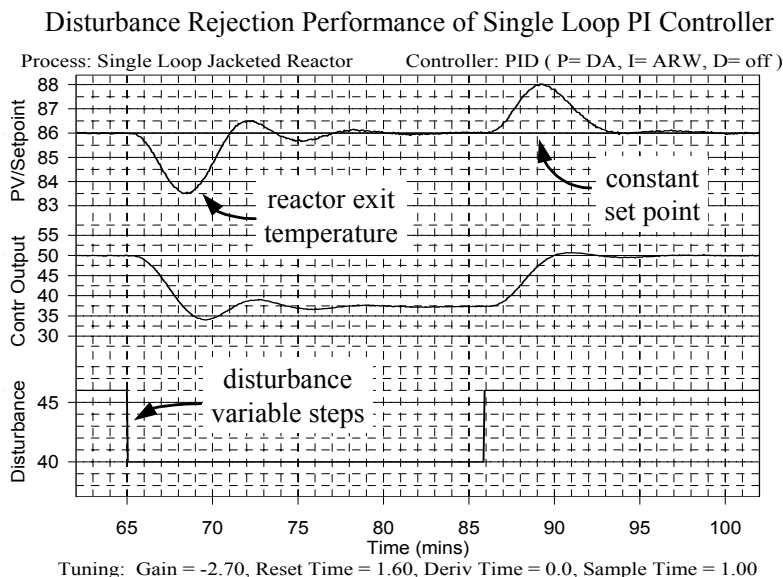
Figure 4 – FOPDT model fit of single loop controller output to measured process variable data

These parameters are used in IMC (internal model control) tuning correlations to obtain the PI tuning:

Controller Gain, $K_C = -2.7 \text{ } \%/^\circ\text{C}$

Reset Time, $\tau_I = 1.6 \text{ min}$

The disturbance rejection performance of the single loop PI controller is shown in Fig. 5. The measured reactor exit stream temperature is initially steady at the design set point value of 86°C. The cooling jacket inlet temperature (disturbance) is stepped from 46°C down to 40°C and back again. As shown, the single loop PI controller is able to maintain reactor exit stream temperature near the set point of 86°C, with deviations ranging as high as 2.5°C during the event. This will serve as a base case for the study.



*Figure 5 – Disturbance rejection in the jacketed reactor under single loop PI control
Process variable deviations reach 2.5°C*

CASCADE CONTROL

For cascade design, the primary process variable is the reactor exit stream temperature. To construct a cascade, we need to identify a secondary process variable. As shown in Fig. 3, the temperature out of the cooling jacket is used. As required for a cascade design:

- cooling jacket outlet temperature is measurable with a sensor,
- the same valve used to manipulate reactor exit stream temperature (the primary variable) also manipulates the cooling jacket outlet temperature,
- changes in cooling jacket inlet temperature that disturb the reactor exit stream temperature also impact the cooling jacket outlet temperature, and
- the cooling jacket outlet temperature is *inside* the reactor exit temperature in that it responds first to changes in valve position and changes in the cooling jacket inlet temperature.

Like all cascades, there are two measurements, two controllers and one final control element; the same final control element as in the single loop case. The primary measured process variable is the reactor exit stream temperature. The output of the primary controller is the set point of the secondary controller. The manipulated variable of the secondary loop is the cooling jacket liquid flow rate and the secondary measured process variable is the cooling jacket outlet temperature.

The secondary controller is tuned while the primary controller is in manual mode. Based on Fig. 3, the design operating conditions are a cooling jacket inlet temperature of 46°C while the secondary measured process variable steadies at 69°C. Hence, for the secondary controller: $y_{\text{setpoint}} = 69^\circ\text{C}$.

The bias is the value of the controller output that, in open loop, causes the measured process variable to steady at its design condition when the disturbances are at their design or expected value. So for the secondary P-Only controller, a controller output of 50% causes the reactor exit stream temperature to steady at the design value of 86°C, hence $u_{\text{bias}} = 50\%$.

Starting from steady state at the design level of operation, a doublet is used to generate controller output to secondary process variable dynamic data as shown in Fig. 6. The controller output is stepped from the design value of 50% up to 55%, down to 45%, and back to 50%. After each control action, the secondary process variable displays a clear response that dominates measurement noise. A FOPDT dynamic model fit to the data as shown in Fig. 6 to yield the secondary control loop model parameters.

Process Gain, $K_P = -0.37^\circ\text{C}/\%$
 Time Constant, $\tau_P = 1.9 \text{ min}$
 Dead Time, $\theta_P = 0.25 \text{ min}$

Although disturbance rejection is the overall objective, the inner secondary loop is designed to track set point changes computed by the primary controller. Using these FOPDT model parameters in the ITAE for set point tracking correlation yields the P-Only tuning parameter: Controller Gain, $K_C = -6.4 \%/^\circ\text{C}$.

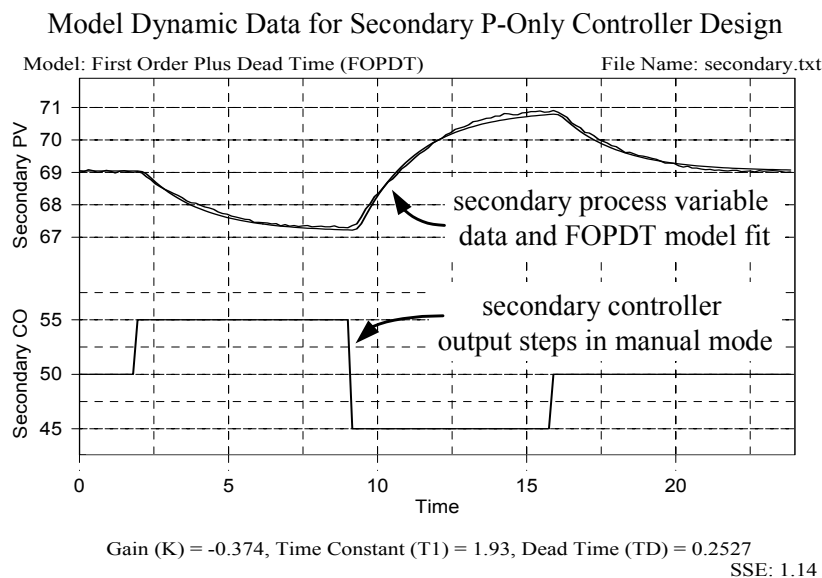


Figure 6 – FOPDT fit of controller output to secondary process variable dynamic data

P-Only set point tracking performance for the inner (secondary) loop is shown in Fig. 7. The primary loop is still in manual mode at this point. As expected for a P-Only controller, offset exists when the set point is not at the design value. The secondary process variable responds quickly and settles rapidly to set point changes so we consider the design of the secondary loop to be complete. The secondary loop is left in automatic and literally becomes part of the primary process. We now tune the primary controller.

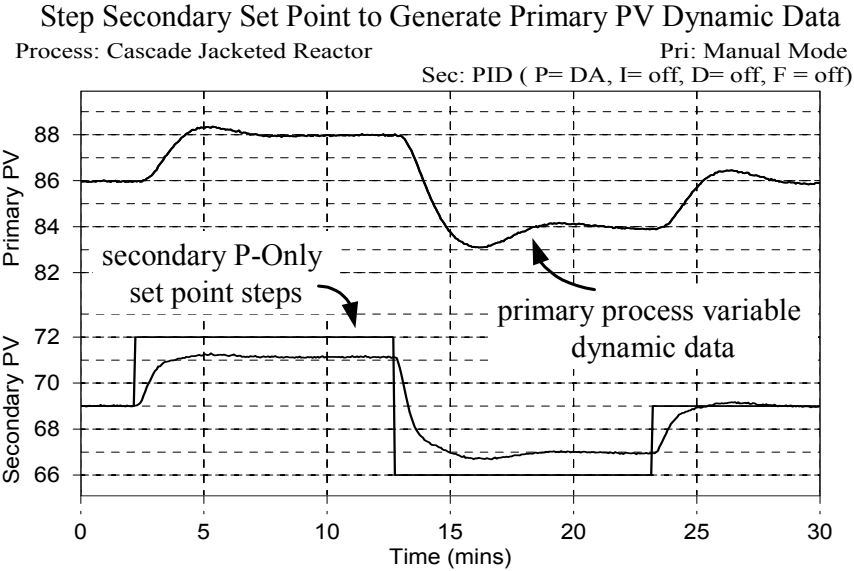


Figure 7 – Set point tracking performance of the secondary loop under P-Only control

In the cascade architecture, the controller output of the primary loop is the set point of the secondary controller. So to design the primary controller, we must step, pulse or otherwise perturb the secondary set point and then fit a model to the corresponding response in the primary measured process variable. Figure 7 contains such data and this is used for the design of the primary PI controller.

The set point of the secondary P-Only controller, as shown in Fig. 8, is stepped in a doublet from its design value of 69°C up to 72°C, down to 66°C, and back to 69°C. The measured reactor exit stream temperature displays a clear response after each change that dominates the measurement noise. A FOPDT fit of the dynamic data, also shown in Fig. 8, yields the following primary control loop model parameters.

$$\text{Process Gain, } K_P = 0.70 \frac{^{\circ}\text{C of reactor exit stream}}{^{\circ}\text{C of cooling jacket outlet stream}}$$

$$\text{Time Constant, } \tau_P = 0.55 \text{ min}$$

$$\text{Dead Time, } \theta_P = 0.71 \text{ min}$$

We first compute the closed loop time constant, or:

$$\tau_C = \text{larger of } 0.1\tau_P \text{ or } 0.8\theta_P = \text{larger of } 0.1(0.55) \text{ or } 0.8(0.71) = 0.57 \text{ min.}$$

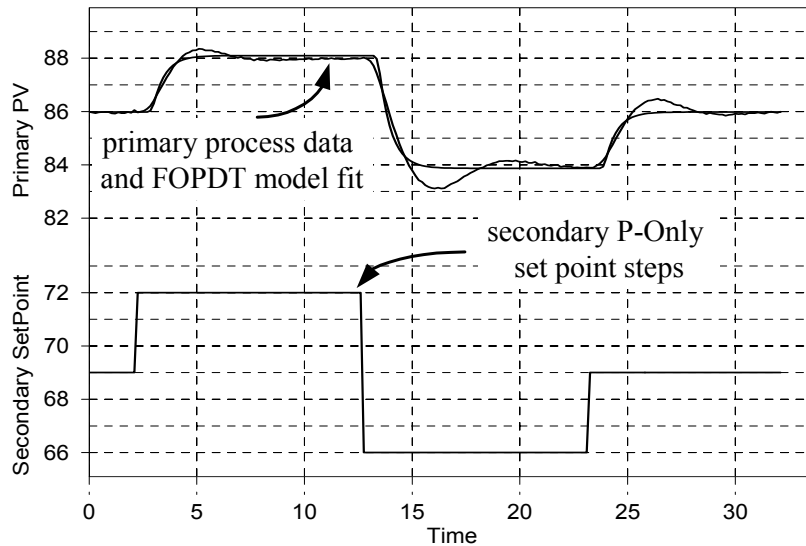
Substituting this closed loop time constant and the above FOPDT model parameters into the IMC correlations for PI control yields the following tuning values:

$$\text{Controller Gain, } K_C = 0.61 \frac{^{\circ}\text{C of cooling jacket outlet stream}}{^{\circ}\text{C of reactor exit stream}}$$

$$\text{Reset Time, } \tau_I = 0.55 \text{ min}$$

FOPDT Model Fit for Design of Primary Controller

Model: First Order Plus Dead Time (FOPDT) File Name: secondary.txt
 Sec: Manual Mode

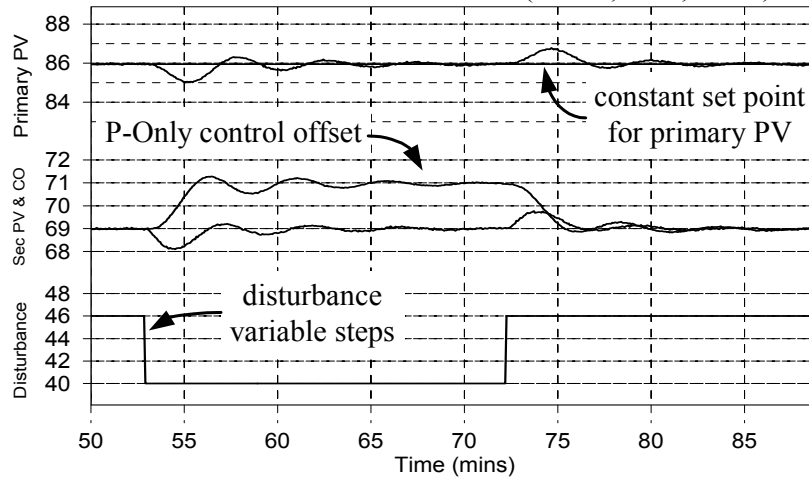


Gain (K) = 0.7049, Time Constant (T1) = 0.5497, Dead Time (TD) = 0.7135
 SSE: 13.78

Figure 8 – FOPDT model fit of dynamic data from a primary loop doublet test

Disturbance Rejection Performance of Cascade Architecture

Process: Cascade Jacketed Reactor Pri: PID (P= RA, I= ARW, D= off, F= off)
 Sec: PID (P= DA, I= off, D= off, F= off)



Tuning: Gain = 0.61, Reset Time = 0.55, Sample Time = 1.00

Tuning: Gain = -6.40, Sample Time = 1.00

Figure 9 – Disturbance rejection in the jacketed reactor with a P-Only to PI cascade
 Process variable deviations reach 1.0°C

These tuning values are implemented on the primary loop and the design of the cascade is complete. Figure 9 shows the disturbance rejection performance of the cascade using these tuning parameters for the primary PI controller while the secondary loop remains under P-Only control as previously described. To test the controller, the cooling jacket inlet temperature is again stepped from its design value of 46°C down to 40°C and back again.

One interesting outcome of this cascade implementation is the offset that occurs in the secondary P-Only loop when the process moves away from the design level of operation. Of more importance, however, is that the cascade shows improved performance over the single loop case in maintaining the reactor product temperature near the constant set point of 86°C.

Reactor exit temperature should have improved disturbance rejection capability. Temperature deviations for the single loop PI controller range as high as 2.5°C during the event (see Fig. 5) while here the cascade limits the maximum deviation to about 1.0°C. This improved performance is not free, however, as the cascade architecture requires an additional sensor, controller and tuning effort.

FEED FORWARD CONTROL

To construct a feed forward controller, the cooling jacket inlet temperature (the disturbance) is measured as shown in the left reactor diagram of Fig. 3. The signal from this disturbance temperature sensor is sent to a feed forward element comprised of a process model and disturbance model.

Generating disturbance driven data can be problematic on real processes if the disturbance variable cannot be manipulated at will. In this case we can perform disturbance manipulations. As shown in Fig. 10, the cooling jacket inlet temperature is stepped from the design value of 46°C up to 49°C, down to 43°C and back to 46°C. The measured reactor exit stream temperature exhibits a clear response after each step that dominates measurement noise.

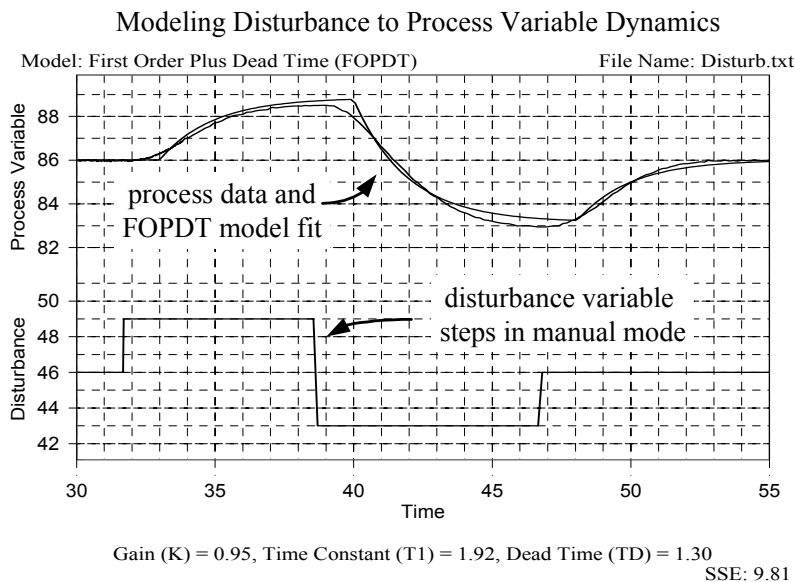


Figure 10 – FOPDT model fit of disturbance to measured process variable data

A FOPDT model reasonably describes the disturbance to measured process variable dynamics as shown in Fig. 10. The disturbance model parameters used to construct the disturbance model of the feed forward element are thus:

$$\text{Disturbance Gain, } K_D = 0.95 \frac{\text{°C of reactor exit stream temperature}}{\text{°C of cooling jacket inlet temperature}}$$

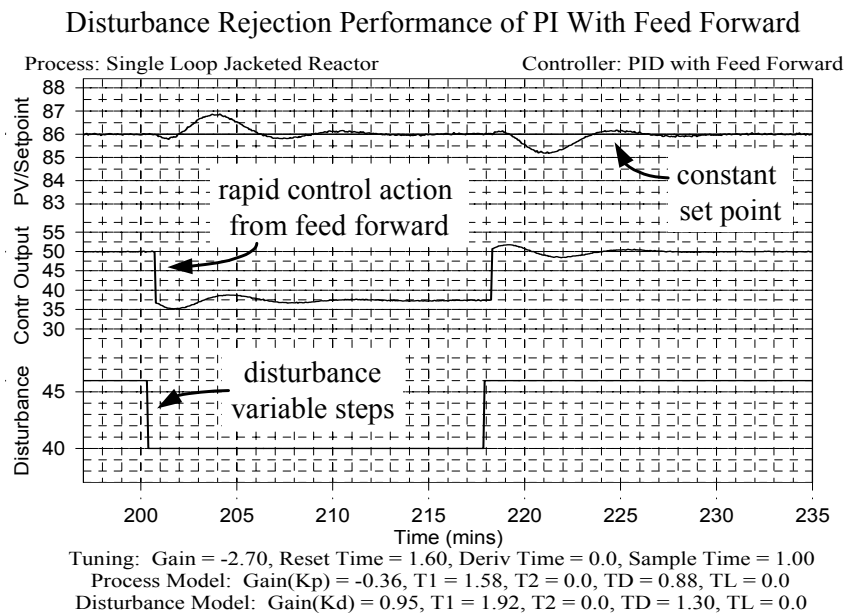
$$\text{Disturbance Time Constant, } \tau_D = 1.92 \text{ min}$$

$$\text{Disturbance Dead Time, } \theta_D = 1.30 \text{ min}$$

Using the process and disturbance models, the feed forward element is constructed:

$$U_{\text{feedforward}}(s) = \left\{ \left(\frac{0.95}{-0.36} \right) \left[\frac{(1.58s + 1)}{(1.92s + 1)} \right] e^{-(1.30 - 0.88)s} \right\} D(s)$$

This feed forward element is combined with the PI feedback controller used in the single loop base case study to yield a feed forward with feedback trim architecture. Figure 11 shows the disturbance rejection performance of this controller. The measured process variable (reactor exit stream temperature) is initialized at the design set point value of 86°C. To test the controller, the cooling jacket inlet temperature is stepped from its design value of 46°C down to 40°C and back again.



*Figure 11 – Disturbance rejection in jacketed reactor under feed forward with feedback trim
 Process variable deviations reach 1.0°C*

The feed controller with feedback trim architecture limits the maximum deviation to less than 1.0°C. This is similar in improved performance to the Cascade architecture. The improved performance did not come free, however, as an additional sensor, controller and tuning effort were required.

CONCLUSIONS

Cascade and feed forward control both provide for improved disturbance rejection. Additional instrumentation and engineering time are required to obtain these benefits.

Cascade control is preferable if a secondary process variable can be identified. It is capable of providing benefit from multiple disturbances that disrupt the secondary primary variable before disrupting the primary variable. It employs traditional controllers from the PID family, tuning of the controllers follows traditional methods and no special programming or other skills are required for implementation.

Feed forward is useful if the secondary process variable required for cascade implementation cannot be identified. It may also show benefit if one particular disturbance is causing risk to safety or profitability. This is because a feed forward element is designed to specifically address one particular disturbance. Hence, it can be tailored specifically to that problem.

LITERATURE CITED

Cooper, Douglas, "Practical Process Control Using Control Station," Published by Control Station LLC, Storrs, CT (2004).