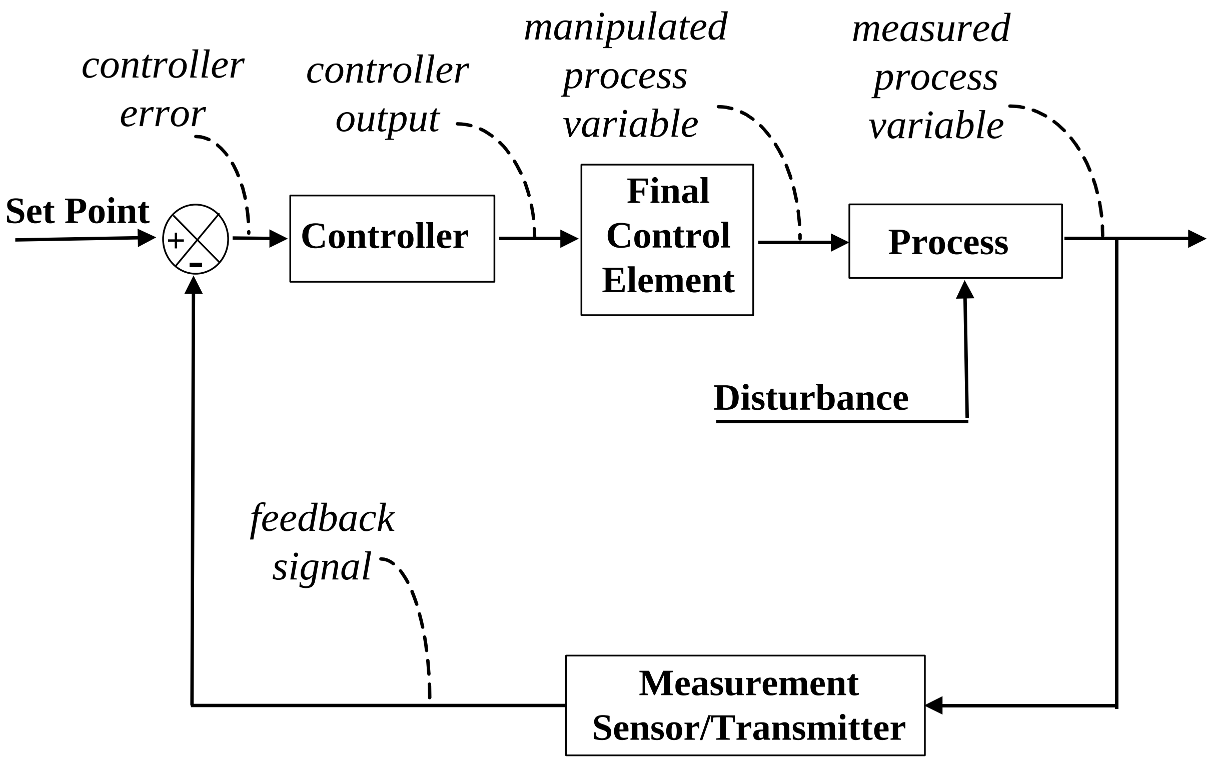
PID Controller Introduction

Proportional-Only Control

Process Control Preliminaries

The final control element, process and sensor/transmitter all have their own gain, time constant and dead time



The Process Designer’s “Process”

- For new installations, the dynamic behaviors of valves and sensors should be carefully considered prior to purchase

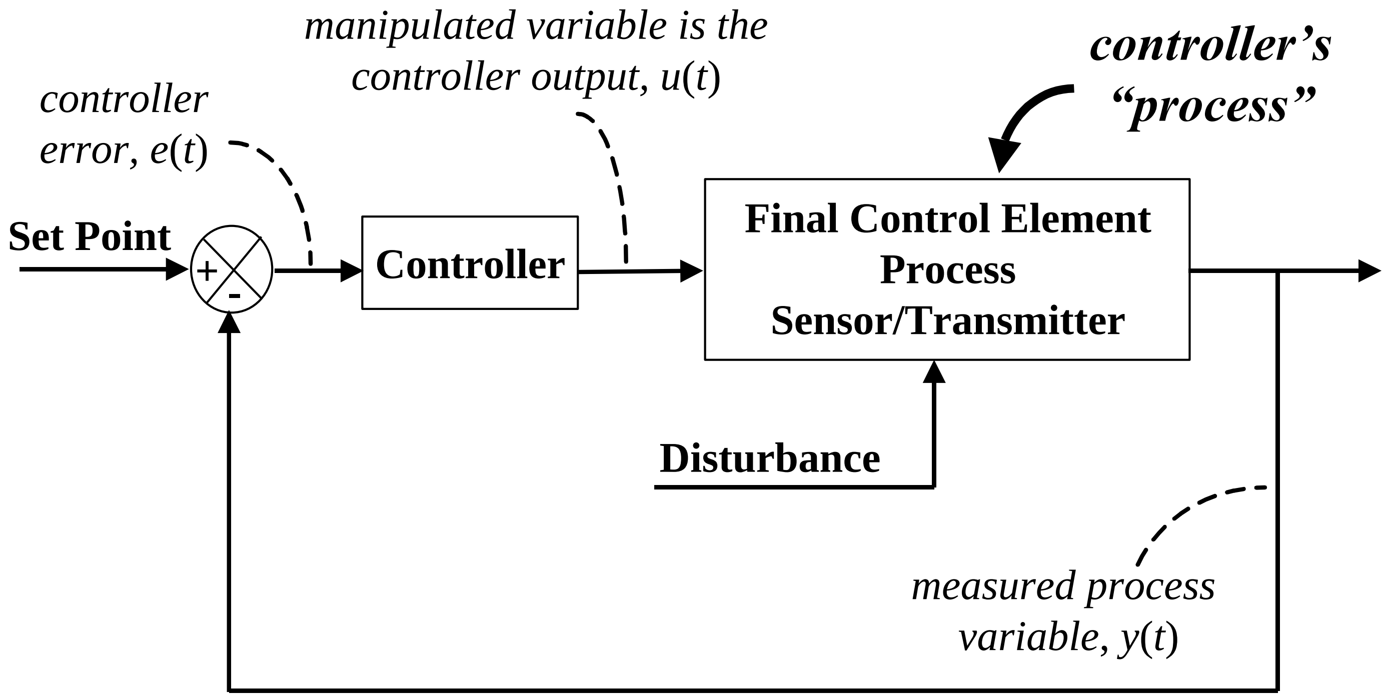
- If undesirable behavior in an existing loop can be traced to certain equipment, then consider relocation or replacement

- A final control element and sensor should start to respond quickly (add little dead time) and complete the response quickly (have a small time constant)

- The qualifier “quickly” is relative to the overall time constant of the process (a θ_P of 9 min is large relative to a τ_P of 10 min but small relative to a τ_P of 1000 min)

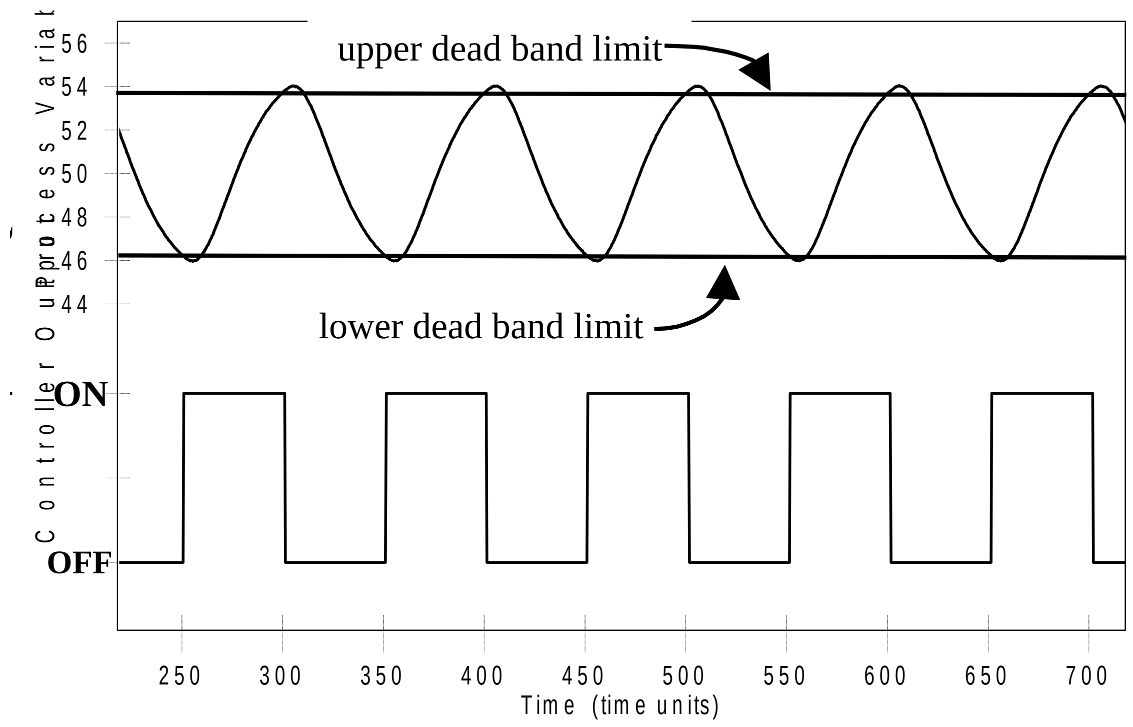
Redefining “Process” for Controller Design/Tuning

- The controller sends a signal out on one wire and receives the result as a change in measurement on another wire
- From the controller’s view, the individual gains, time constants and dead times all lump into one overall dynamic behavior
- Here, “process dynamics” refers to these combined behaviors



On/Off Control – The Simplest Controller

- For on/off control, the final control element is either completely open/on/maximum or closed/off/minimum
- To protect the final control element from wear, a *dead band* or upper and a lower set point is used
- As long as the measured variable remains between these limits, no changes in control action are made



Usefulness of On/Off Control

- On/off with dead band is useful for home appliances such as furnaces, air conditioners, ovens and refrigerators

- For most industrial applications, on/off is too limiting (think about riding in a car that has on/off cruise control)

Intermediate Value Control and PID

- Industry requires controllers that permit tighter control with less oscillation in the measured process variable
- These algorithms:
  - compute a complete range of actions between full on/off
  - require a final control element that can assume intermediate positions between full on/off
- Example final control elements include process valves, variable speed pumps and compressors, and electronic heating elements

Intermediate Value Control and PID

- Popular intermediate value controller is PID (proportional-integral-derivative) controller
- PID computes a controller output signal based on control error:

$$\underbrace{u(t) = u_{bias} + K_C\, e(t)}_{\text{Proportional term}} + \underbrace{\frac{K_C}{\tau_I}\int e(t)dt}_{\text{Integral term}} \underbrace{- K_C \tau_D \frac{dy}{dt}}_{\text{Derivative term}}$$

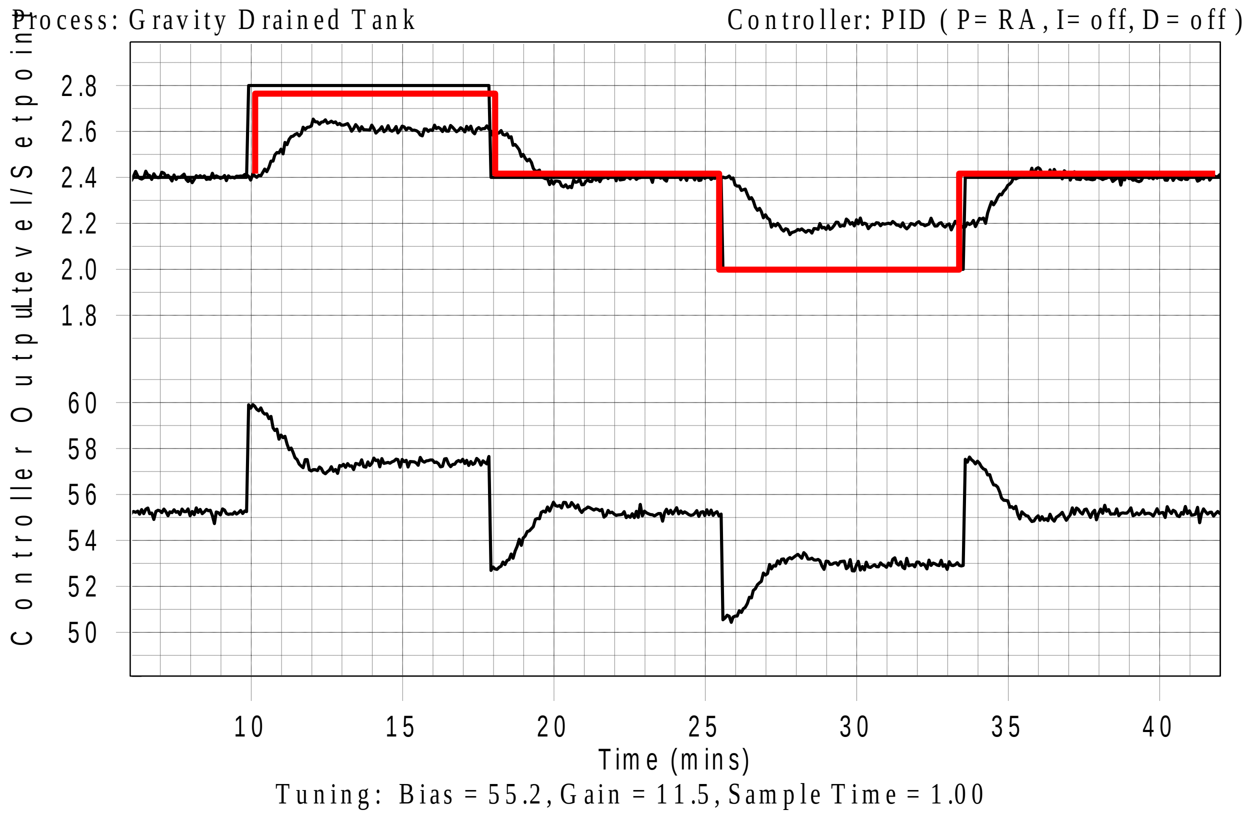
where:

$y(t)$	= measured process variable
$u(t)$	= controller output signal
u_{bias}	= controller bias or null value
$e(t)$	= controller error = $y_{\text{setpoint}} - y(t)$
K_C	= controller gain (a tuning parameter)
τ_I	= controller reset time (a tuning parameter)
τ_D	= controller derivative time (a tuning parameter)

P-Only => The Simplest PID Controller

The Proportional Controller

- The simplest PID controller is *proportional* or *P-Only* control
- It can compute intermediate control values between 0 – 100%



The Gravity Drained Tanks Control Loop

- Measurement, computation and control action repeat every loop sample time:
  - a sensor measures the liquid level in the lower tank
  - this measurement is subtracted from the set point level to determine a control error; $e(t) = y_{setpoint} – y(t)$
  - the controller computes an output based on this error and it is transmitted to the valve, causing it to move
  - this causes the liquid flow rate into the top tank to change, which ultimately changes the level in the lower tank

- The goal is to eliminate the controller error by making the measured level in the lower tank equal the set point level

P-Only Control

- The controller computes an output signal every sample time:

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

where

$u(t)$	= controller output
u_{bias}	= the controller bias
$e(t)$	= controller error = set point – measurement
	= $y_{setpoint} - y(t)$
K_C	= controller gain (a tuning parameter)

- controller gain, $K_C \neq$ steady state process gain, K_P
- a larger K_C means a more active controller
- like K_P, controller gain has a size, sign and units
  - Units of K_c are units of the controller (e.g., %) divided by the units of the measured variable, since e(t) has units of the measured variable
  - For gravity-drained tank, units of K_c are %/m
  - Remember that for this example, units of K_P are m/%

Design Level of Operation

- A controller is designed for a particular process behavior (or particular values of the FOPDT parameters K_P, τ_P and θ_P)

- Real processes are nonlinear, so their behavior changes as operating level changes

- Thus, a controller should be designed for a specific level of operation

Collect Process Data at the Design Level

- the design value for the measured process variable is where the set point will be set during normal operation

- the design values for the important disturbance variables are their typical or expected levels during normal operation

- perform the dynamic test as near practical to the design level of the measured process variable when the disturbances are quiet and near their typical values

Controller Gain, K_C, From Correlations

- P-Only control has one adjustable or *tuning parameter*, K_C

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

- K_C sets the activity of the controller to changes in error, $e(t)$
  - if K_C is small, the controller is sluggish
  - If K_C is large, the controller is aggressive

- To determine K_C, use this controller design procedure:
  - generate dynamic process data at the design level of operation
  - fit a FOPDT dynamic model to this data
  - use the FOPDT model parameters in a correlation to compute initial estimates of K_C

Controller Gain, *K_C*, From Correlations

Integral of time-weighted absolute error (ITAE) tuning correlations:

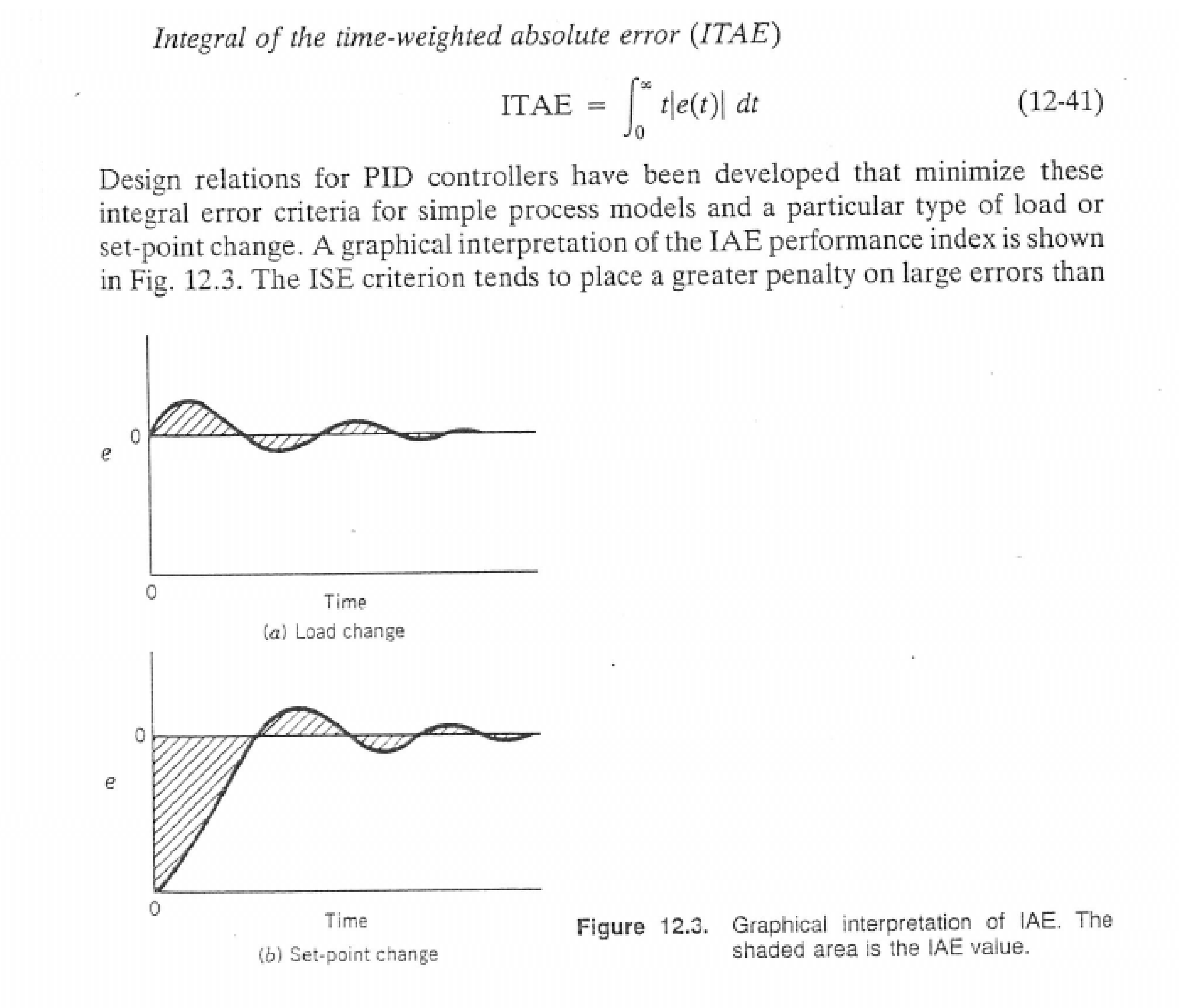
- if set point tracking (*servo* control) is the objective:

$$K_C = \frac{0.202}{K_p}(\theta_p/\tau_p)^{-1.219}$$

- if disturbance rejection (*regulatory* control) is the objective:

$$K_C = \frac{0.490}{K_p}(\theta_p/\tau_p)^{-1.084}$$

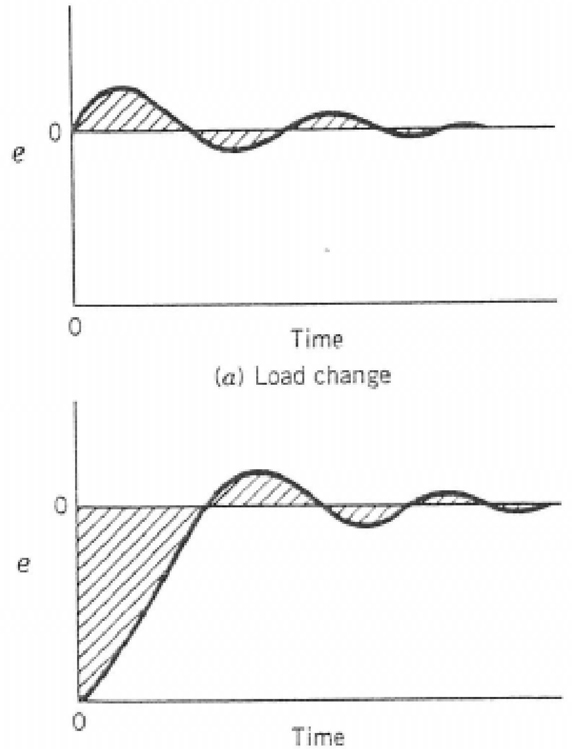
ITAE



Integral of the time-weighted absolute error (ITAE)

$$\text{ITAE} = \int_0^\infty t|e(t)|\,dt \tag{12-41}$$

Design relations for PID controllers have been developed that minimize these integral error criteria for simple process models and a particular type of load or set-point change. A graphical interpretation of the IAE performance index is shown in Fig. 12.3. The ISE criterion tends to place a greater penalty on large errors than



(*a*) Load change

(*b*) Set-point change

Figure 12.3. Graphical interpretation of IAE. The shaded area is the IAE value.

Controller Gain, K_C, From Correlations

- Correlations provide an initial guess or starting point only

- Final tuning requires online trial and error because:
  - the designer may desire performance different from that provided by the correlation
  - the FOPDT model used for tuning may not match the actual dynamic behavior of the plant
  - performance must be balanced over a range of nonlinear operation
  - performance must be balanced for set point tracking *and* disturbance rejection

- Notes:
  - the designer defines “best” control performance
  - it is conservative to start with a small K_C value

Understanding Controller Bias, u_{bias}

- Thought Experiment:
  - Consider P-Only cruise control where $u(t)$ is the flow of gas
  - Suppose velocity set point = measured velocity = 70 kph
  - Since $y(t) = y_{setpoint}$ then $e(t) = 0$ and P-Only controller is

$$u(t) = u_{bias} + 0$$

  - If u_{bias} were set to zero, then the flow of gas to the engine would be zero even though the car is going 70 kph

- If the car is going 70 kph, there clearly is a baseline flow of gas

- This baseline controller output is the bias or null value

Understanding Controller Bias, u_{bias}

- In the thought experiment, the bias is the flow of gas which, in open loop, causes the car to travel the design velocity of 70 kph when the disturbances are at their normal or expected values

- In general, u_{bias} is the value of the controller output that, in open loop, causes the measured process variable to maintain steady state at the design level of operation when the process disturbances are at their design or expected values

- Controller bias is not normally adjusted once the controller is put in automatic

Reverse Acting, Direct Acting and Control Action

- If K_P is positive and the process variable is too high, the controller decreases the controller output to correct the error

=> The controller action is the reverse of the problem

When K_P is positive, the controller is *reverse acting*

When K_P is negative, the controller is *direct acting*

- Since K_C always has the same sign as K_P , then

K_P and K_C positive	$\rightarrow$	reverse acting
K_P and K_C negative	$\rightarrow$	direct acting

Reverse Acting, Direct Acting and Control Action

- Most commercial controllers require you to enter a positive K_C
- The sign (or *action*) of the controller is specified by entering the controller as reverse or direct acting
- If the wrong control action is entered, the controller will drive the valve to full open or closed until the entry is corrected

Offset - The Big Disadvantage of P-Only Control

- Big advantage of P-Only control:

  => only one tuning parameter so it’s easy to find “best” tuning

- Big disadvantage:

  => the controller permits *offset*

Offset - The Big Disadvantage of P-Only Control

- Offset occurs under P-Only control when the set point and/or disturbances are at values other than that used as the design level of operation (that used to determine u_{bias})

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

- How can the P-Only controller compute a value for $u(t)$ that is different from u_{bias} at steady state?

- The only way is if $e(t) \neq 0$

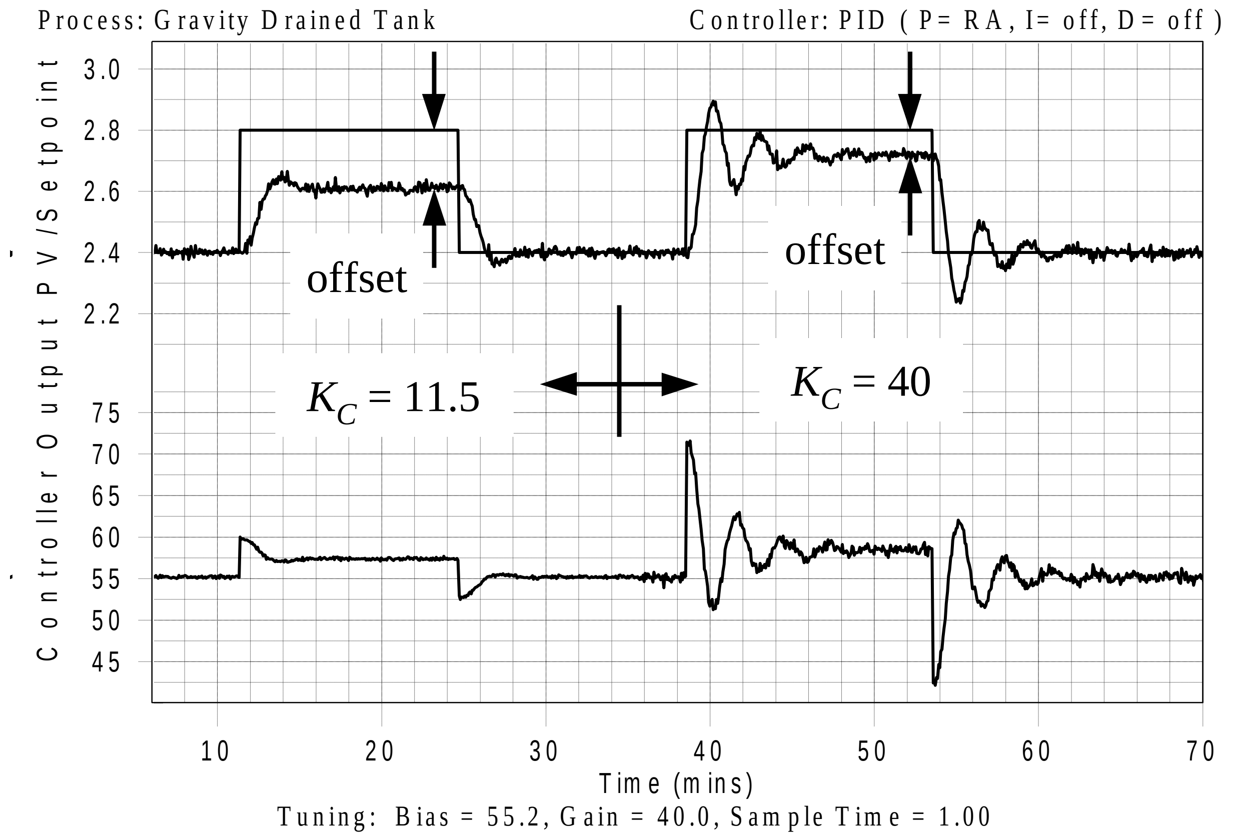
How Do You Get Rid of Offset?

- Operate at the design conditions
- Use a more advanced control algorithm
  - PI instead of P-only

Offset and K_C

As K_C increases: → Offset decreases → Oscillatory behavior increases

Impact of K_C on Offset and Oscillatory Behavior



Bumpless Transfer to Automatic

- A “bumpless transfer" achieves a smooth transition to closed loop by automatically:
  - setting u_{bias} equal to the current controller output value
  - setting the set point equal to the current measured process variable value
- So when put in automatic, there is no controller error and the bias is properly set to produce no offset
- As a result, no immediate control action is necessary that would “bump” the measured process variable