Lecture 9 - PID Control

K. J. Åström

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

Theme: The most common controller. A glimpse of implementation.

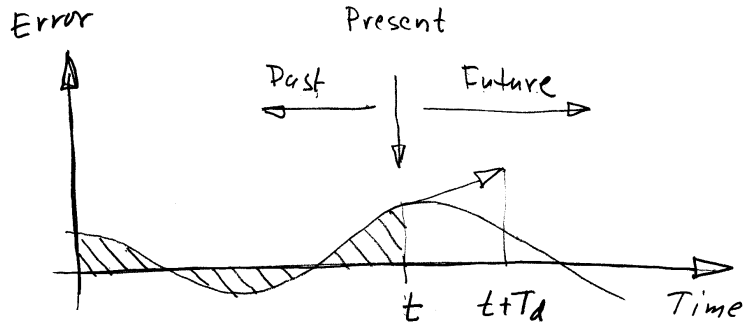
Introduction

- Why PID?
  - The most common controller
  - Widely used in all applications of control (90% of all control problems)
- An essential element of more sophisticated controllers
- Are there any research issues?
- Nonlinear features
  - Saturation and windup
  - Dead zones
  - What can be done by PID?
  - What cannot be done by PID?

Introduction

- Feedback is a very powerful concept with many useful properties
  - Reduction of effects of disturbances
  - Create robust linear relations
  - Follow command with High Fidelity
  - Robust to process variations
  - But risk for instability
- Advances in control theory have given a good insight into the design problem
- PID a simple powerful form of feedback
- Apply advances in control to PID control
- Connect with the classic tradition of Ziegler and Nichols

PID versus Advanced Control



- Advanced control - other prediction methods
- What are the benefits?

The Amazing Property of Integral Action

Consider a PI controller

$$u = ke + k_i \int_0^t e(\tau)d\tau$$

Assume that there is an equilibrium with constant $e(t) = e_0$ and constant $u(t) = u_0$. The error e_0 then must be zero. Proof: Assume $e_0 \neq 0$, then

$$u = ke_0 + k_i \int_0^t e(\tau)d\tau = ke_0 + k_i \int_0^t e_0 d\tau = ke_0 + k_i e_0 t$$

The right hand side is different from zero. Hence a contradiction unless $e_0 = 0$.

A controller with integral action will always give the correct steady state provided that a steady state exists.

A PID Algorithm

In spite of the widespread use of PID it is only given moderate attention in education. Much information among the manufacturers. PID control is much more than

$$u(t) = ke(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}$$

We have to consider

- Derivative filter
- Set point (reference) weigthing
- Integrator Windup
- Computer implementation
- Mode switches
- Bumpless parameter changes

Dealing with these issues is a good introduction to practical implementation of any control algorithm.

PID Control

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

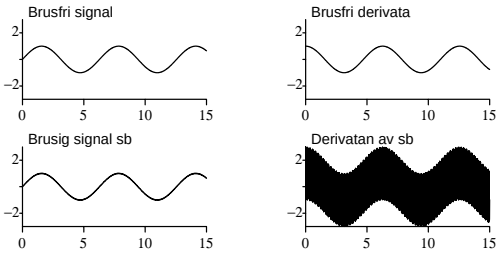
Differentiating Noisy Signals

Consider the signal

$$y(t) = \sin t + a_n \sin \omega t$$

It has the derivative

$$\frac{dy(t)}{dt} = \cos t + a_n \omega \cos \omega t$$

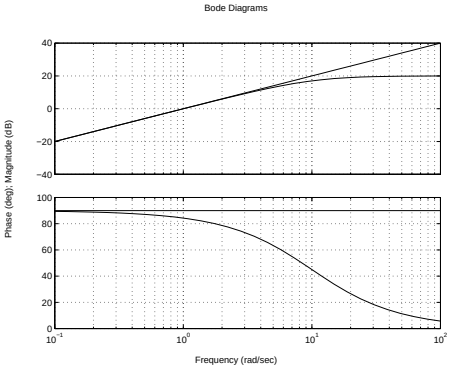


The curves are generated with $\omega = 100$, $a_n = 0.01$.
One percent error in the original signal gives 100% error in derivative!

Approximate Differentiation - High Frequency Roll-off

Replace sT by

$$G_d(s) = \frac{sT}{1 + sT/N}$$



What does it mean?

- For small s we have $G_d(s) \approx sT$.
- For large s we have $G_d(s) \approx N$.

The system $G_d(s)$ has the output $T\ dy/dt$ for low frequency signals. The gain of G_d is not greater than N.

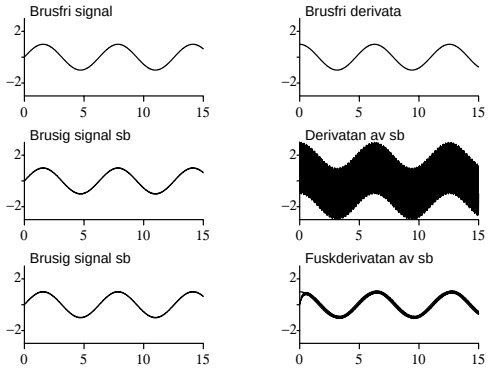
A nice illustration of use of Bode Plots!

Simulation of Approximate Derivative

$$y(t) = \sin t + a_n \sin \omega t$$

Approximate derivative

$$G_d(s) = \frac{s}{1 + s/5}$$



Different Parameterizations

Parallel form:

$$G(s) = k\left(1 + \frac{1}{sT_i} + sT_d\right) = \frac{k}{sT_i}(1 + sT_i + s^2T_iT_d)$$

Series form:

$$\tilde{G}(s) = \tilde{k}\left(1 + \frac{1}{s\tilde{T}_i}\right)(1 + s\tilde{T}_d) = \frac{\tilde{k}}{s\tilde{T}_i}(1 + s(\tilde{T}_i + \tilde{T}_d) + s^2\tilde{T}_i\tilde{T}_d)$$

Relations between coefficients

$$k = \tilde{k}\frac{\tilde{T}_i + \tilde{T}_d}{\tilde{T}_i}, \qquad T_i = \tilde{T}_i + \tilde{T}_d, \qquad T_d = \frac{\tilde{T}_i\tilde{T}_d}{\tilde{T}_i + \tilde{T}_d}$$

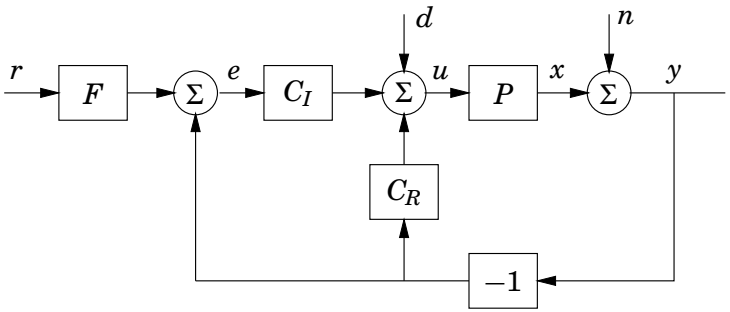
Parallel form is more general. Equivalence possible only if $T_i \geq 4T_d$. Essential for tuning to know which form is used.

PID Control

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

Set Point (Reference) Response

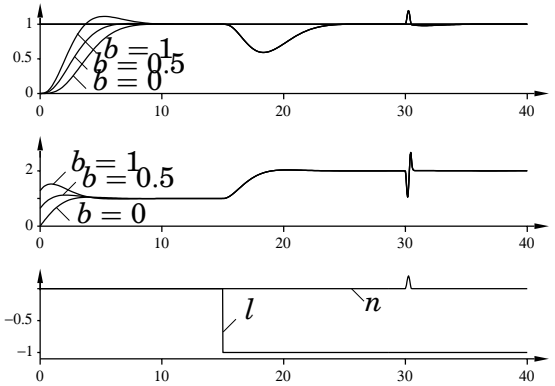
Set point weighting allows a moderate adjustment. A 2DOF structure makes set-point response independent of disturbance response.



$$U(s) = k\left(bR(s) - Y(s) + \frac{1}{T_i}(R(s) - Y(s)) - \frac{sT_d}{1 + sT_d/N}Y(s)\right)$$

Set Point (Reference) Weighting

A simple way to obtain some DOF benefits

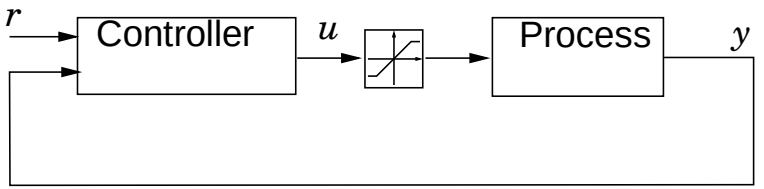


Notice different signal paths $y \rightarrow u$ and $r \rightarrow u$. Not a complete 2DOF but often a good way to separate disturbance rejection from response to reference signals.

PID Control

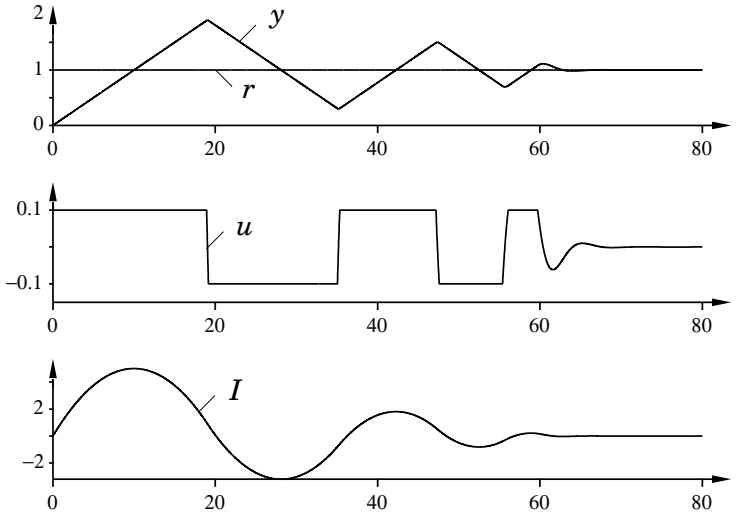
1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

Effects of Saturation

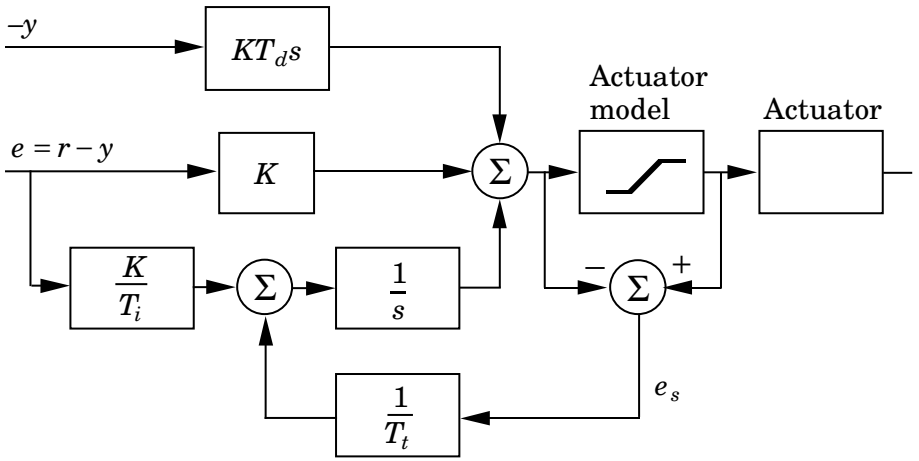


- Practically all systems have saturations in actuators
- The feedback loop is broken when saturation occurs
- Unstable modes in process and controller will grow
- An integrator is an unstable and it will *wind up*
- Windup protection is required in all controllers with integral action
- Instabilities are essential difficulites!

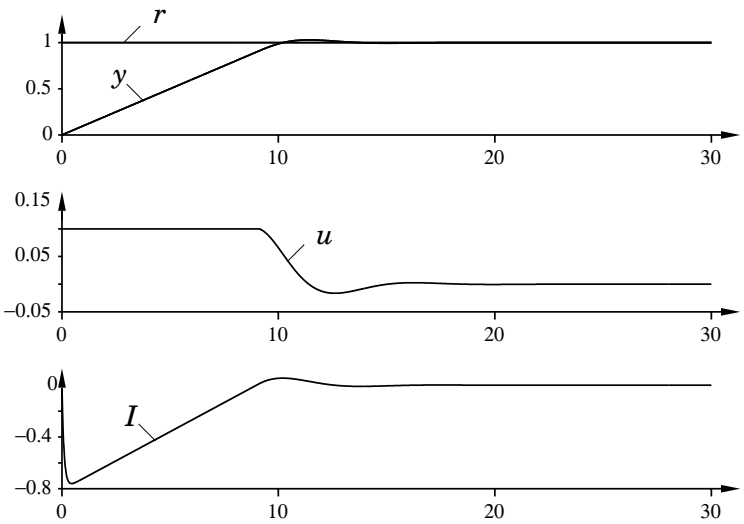
Integrator Windup



One Way to Avoid Windup

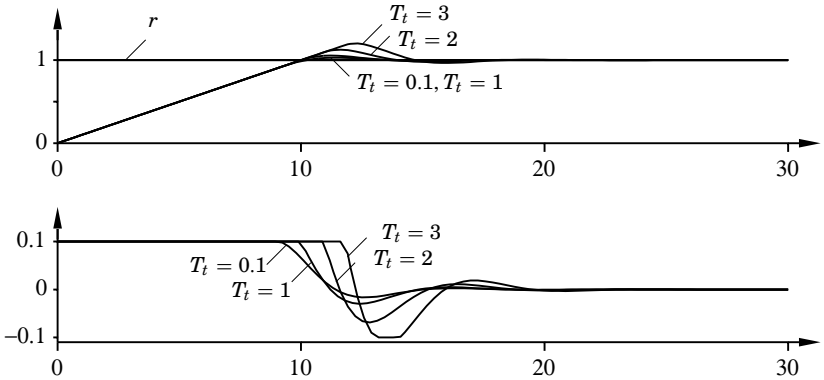


Control with Anti-Windup



Effect of Time Constant T_t

Influence of the reset time constantT_t.



Rules of thumb $T_t = 0.5T_i$ for PI control or $T_t = \sqrt{T_i T_d}$ for PID. Simulation made with PI control with $T_i = 1$.

PID Control

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

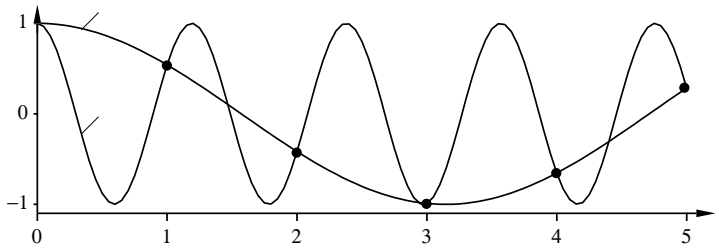
Computer Implementation

Practically all control systems are today implemented using computers. We will briefly discuss some aspects of this.

AD and DA converters are needed to connect sensors and actuators to the computer. A clock is also needed to synchronize the operations. We will discuss

- Sampling and aliasing
- A basic algorithm
- Converting differential equations to difference equations
- Wordlength issues
- Bumpless parameter changes

Sampling, Aliasing and Antialiasing Filters



- Samples of signals of different frequencies may be identical
- Nyquist frequency = (Sampling frequency)/2
- To represent a continuous signal uniquely from its samples the continuous signal cannot have frequencies above the Nyqyist frequency which which is half the Nyquist frequency
- Antialiasing fi lters that reduce the frequency content above the Nyquist frequency is essential.

A Basic Algorithm

The following operations are executed by the computer.

1. Wait for clock interrupt
2. Convert setpoint r and process output y to numbers
3. Compute control signal u
4. Convert control signal to analog value
5. Update variables in control algorithm
6. Go to step 1

Desirable to make time between 1 and 4 as short as possible. Defer as much as possible of the computations to step 5.

A Practical PID Controller

The basic equation

$$u(t) = k\big(br(t) - y(t)\big) + k_i \int_0^t \big(r(\tau) - y(\tau)\big)d\tau + k_d(-\frac{dy_f(t)}{dt}),$$

Derivative filter $\frac{T_d}{N}\frac{dy_f}{dt} + y_f = y$

Feedback $G_c(s) = k + \frac{k_i}{s} + k_d\frac{s}{1+sT_f}$

Feedforward $G_{ff}(s) = bk + \frac{k_i}{s}$

Set point weighting b

Sometimes also high frequency roll-off

$$U(s) = \frac{k}{(1+sT_f)^2}\Big(bR(s) - Y(s) + \frac{1}{sT_i}\big(R(s) - Y(s)\big) - sT_dY(s)\Big)$$

The PID Algorithm

The PID controller is described by:

$$U(s) = P(s) + I(s) + D(s)$$
$$P(s) = k\Big(bR(s) - Y(s)\Big)$$
$$I(s) = k\frac{1}{sT_i}(R(s) - Y(s))$$
$$D(s) = -k\frac{sT_d}{1 + sT_d/N}Y(s)$$

Computers can only add and multiply, it cannot integrate or take derivatives. To obtain a programmable algorithm we must approximate. There are many ways to do this.

Introduce the times t_k when the clock ticks, assume that $t_k - t_{k-1} = h$, ,where h is the sampling period.

The Proportional Part

$p(t_k) = k * (br(t_k) - y(t_k))$

No approximation required!

Integral Part

$$i(t) = \frac{k}{T_i}\int^t e(\tau)d\tau$$

Differentiate

$$\frac{di}{dt} = \frac{k}{T_i}e(t)$$

Approximate the derivative by a forward difference

$$\frac{i(t_{k+1}) - i(t_k)}{h} = \frac{ke(t_k)}{T_i}$$

This equation can be written as

$$i(t_{k+1}) = i(t_k) + \frac{kh}{T_i}e(t_k)$$

Derivative Part

$$D(s) = -k\frac{sT_d}{1+sT_d/N}Y(s)$$

Hence

$$(1+sT_d/N)D(s) = -ksT_dY(s)$$

In time domain

$$d(t) + \frac{T_d}{N}\frac{dd}{dt} = -kT_d\frac{dy}{dt}$$

Approximate derivative by backward difference

$$d(t_k) + \frac{T_d}{N}\frac{d(t_k)-d(t_{k-1})}{h} = -kT_d\frac{y(t_k)-y(t_{k-1})}{h}$$

Derivative Part Continued

$$d(t_k) + \frac{T_d}{N}\frac{d(t_k)-d(t_{k-1})}{h} = -kT_d\frac{y(t_k)-y(t_{k-1})}{h}$$

Hence

$$\left(1+\frac{T_d}{Nh}\right)d(t_k) = \frac{T_d}{Nh}d(t_{k-1}) - \frac{kT_d}{h}\left(y(t_k)-y(t_{k-1})\right)$$

or

$$d(t_k) = \frac{T_d}{T_d+Nh}d(t_{k-1}) - \frac{kT_dN}{T_d+Nh}\left(y(t_k)-y(t_{k-1})\right)$$

Notice that the algorithm works well even if T_d is small, this is not the case if forward approximations are used.

The Discrete PID Algorithm

Summarizing we find

$$\begin{aligned}
p(t_k) &= k*(br(t_k)-y(t_k))\\
e(t_k) &= r(t_k)-y(t_k)\\
d(t_k) &= \frac{T_d}{T_d+Nh}\left(d(t_{k-1}) - kN\left(y(t_k)-y(t_{k-1})\right)\right)\\
u(t_k) &= p(t_k)+i(t_k)+d(t_k)\\
i(t_{k+1}) &= i(t_k)+\frac{kh}{T_i}e(t_k)
\end{aligned}$$

Add Protection Against Windup

$$\begin{aligned}
p(t_k) &= k*(br(t_k)-y(t_k))\\
d(t_k) &= \frac{T_d}{T_d+Nh}\left(d(t_{k-1}) - kN\left(y(t_k)-y(t_{k-1})\right)\right)\\
v &= p(t_k)+i(t_k)+d(t_k)\\
u(t_k) &= \mathrm{sat}(v)\\
e(t_k) &= r(t_k)-y(t_k)\\
i(t_{k+1}) &= i(t_k)+\frac{kh}{T_i}e(t_k)+\frac{kh}{T_r}(u-v)
\end{aligned}$$

- Useful to precompute parameters
- Make sure updating is done safely
- Organize the code right

Wordlength Issues

Consider updating of the integral part

$$i(t_{k+1}) = i(t_k) + \frac{kh}{T_i} e(t_k)$$

Example

- h=0.05 s
- T_i=5000 s
- k=1
- $\frac{kh}{T_i} = 10^{-5}$

If the error has 3 digits the integral need to be updated with 8 digits (28 bits) to avoid rounding off the errors!

Bumpless Parameter Changes

A PID controller is often switched between three modes: off, manual and automatic control. It is important that there are no switching transients.

It is also important that parameter changes do not generate transients. This can be avoided by proper coding.

Example:

This implementation gives bumps

$$i = \frac{k}{T_i} \int^t e(s)ds$$

This implementation does not give bumps

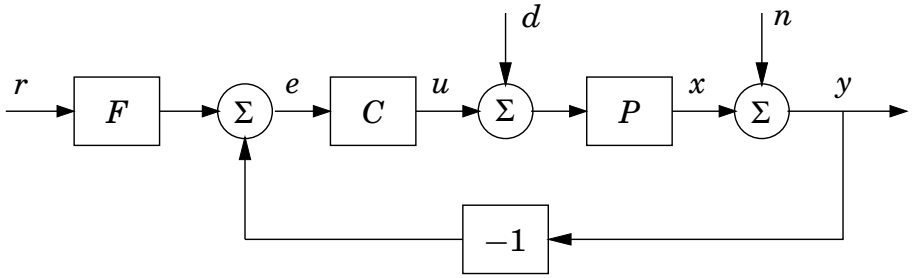
$$i = \int^t \frac{k}{T_i} e(s)ds$$

The basic issue is that multiplication with a time function does not commute with differentiation or integration.

PID Control

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

Requirements



- Reduce the effect of load disturbances
- Do not inject too much measurement noise
- Low sensitivity to process variations
- Good response to set point changes

Introduction

A wide range of methods have been developed to design and tune PID controllers

- Special methods for PID controllers
- Application of general techniques for control system design like pole placement that you have learned in the class.

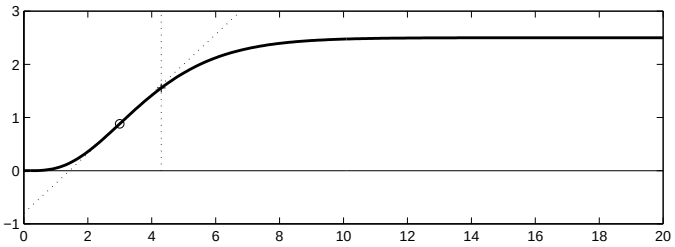
The methods differ with respect to

- Models
- Model acquisition
- Criteria
- Design techniques

We will present a selection

Ziegler-Nichols' Step Response Method

- Switch controller to manual.
- Make a step in the control variable.
- Log process output. Normalize the curve so that it corresponds to a unit step.
- Determine intercepts of tangent with steepest slope i.e. parameters a and L. The controller parameters are obtained from a table.



Ziegler-Nichols' Step Response Method

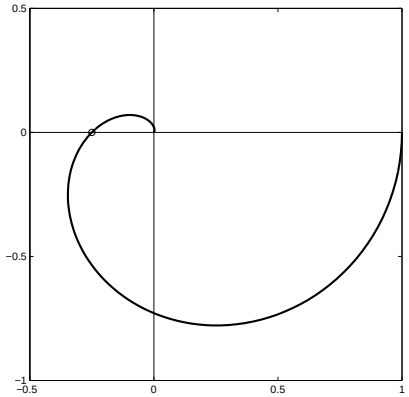
Data: apparent time delay L and intercept a. Controller parameters are given by

Controller	k	T_i	T_d	T_p
P	$1/a$			$4L$
PI	$0.9/a$	$3L$		$5.7L$
PID	$1.2/a$	$2L$	$L/2$	$3.4L$

Parameter T_p is an estimate of the response time of the closed loop system.

Ziegler-Nichols' Frequency Response Method

- Switch the controller to pure proportional.
- Adjust the gain so that the closed loop system is at the stability boundary.
- Determine the gain k_u (the ultimate gain) and the period T_u (the ultimate period) of the oscillation.
- Suitable controller parameters are obtained from a table.



Ziegler-Nichols' Frequency Response Method

Data: ultimate gain k_u and ultimate period T_u. Controller parameters given by.

Reg.	k	T_i	T_d	T_p
P	$0.5k_u$			T_u
PI	$0.4k_u$	$0.8T_u$		$1.4T_u$
PID	$0.6k_u$	$0.5T_u$	$0.125T_u$	$0.85T_u$

Parameter T_p is an estimate of the response time of the closed loop system.

Properties of Ziegler Nichols Rules

Properties

+ Easy to explain and use
+ Very common
- The closed loop system obtained too oscillatory $\zeta \approx 0.2$. Part of the criterion (quarter amplitude damping)
- Too large overshoot
- Sensitive to process variations

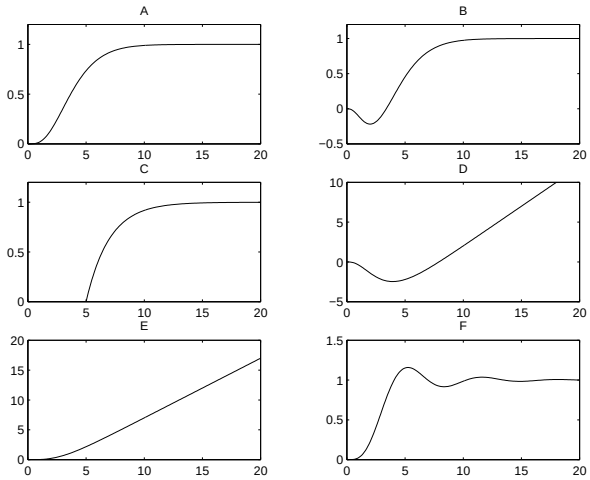
Large scope for improvements.

More process information needed.

Assessment of Ziegler-Nichols Methods

- Published in 1942 in Trans. ASME 64(1942)759–768.
- Tremendously influential
- The beginning of process control
- Slight modifications used extensively by controller manufacturers and process engineers
- Uses too little process information: only 2 parameters
- Substantial improvements can be obtained with modified rulse based on 3 parameters
- Basic design principle quarter amplitude damping is not robust, gives closed loop systems with too high sensitivity ($M_s > 3$) and too poor damping ($\zeta \approx 0.2$)

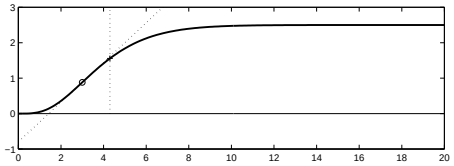
Dynamics of Processes Suitable for PID Control



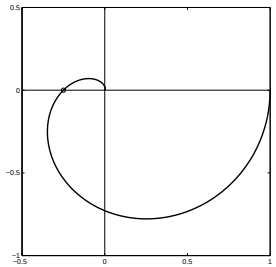
Essentially monotone step responses: $\int_0^\infty h(t)dt \Big/ \int_0^\infty |h(t)|dt \approx 1$

Characterize Dynamics by 3 Parameters

Step response method: K, L and T



Frequency response method: ω_u, $|P(\omega_u)|$ and $P(0)$



A Modified Step Response Method

Lag dominated dynamics: $L < 0.1T$

$$K = 0.3\frac{T}{K_p L}, \quad T_i = 8L$$

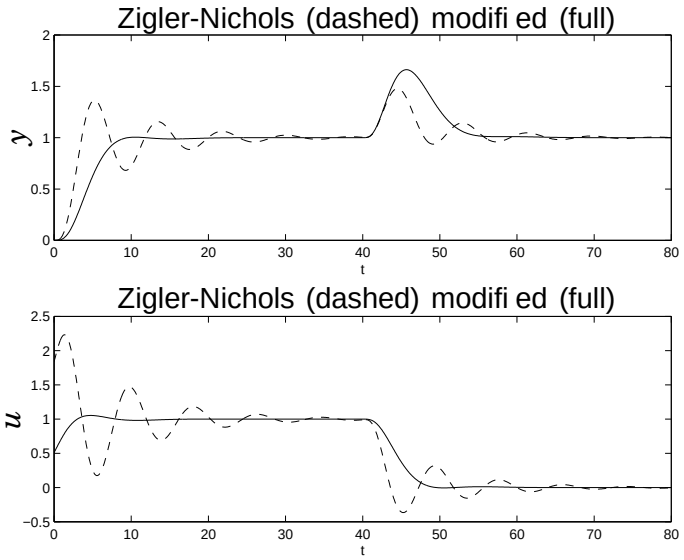
Balanced dynamics $0.1T < L < 2T$

$$K = 0.3\frac{T}{K_p L}, \quad T_i = 0.8T$$

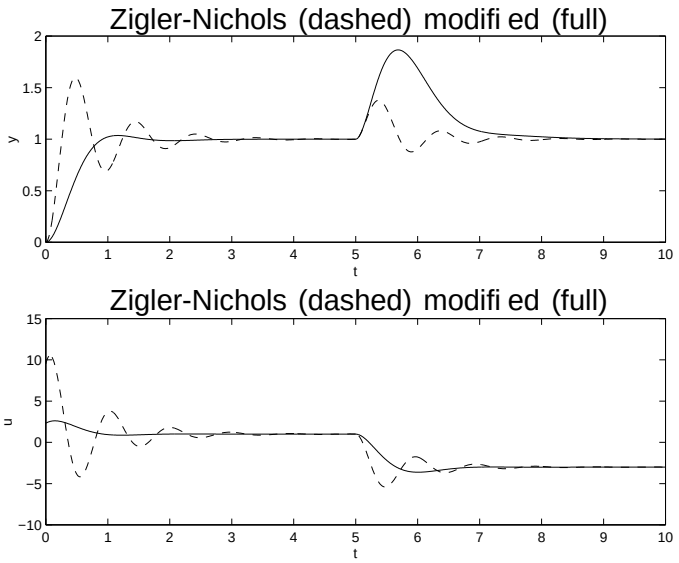
Delay dominated dynamics $L > 2T$

$$K = \frac{0.15}{K_p}, \quad T_i = 0.4L$$

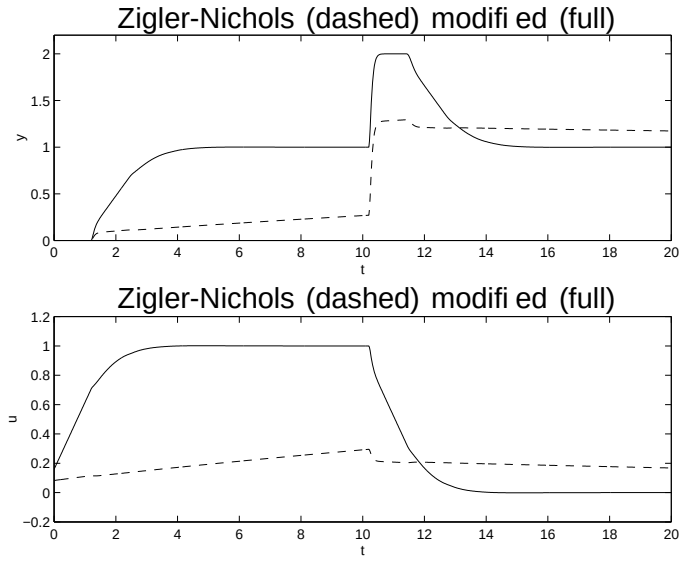
PI Balanced Process Dynamics $L \approx T$



PI Lag Dominated Dynamics $L << T$



PI Delay Domiated Dynamics $L >> T$



PID Control

1. Introduction
2. Derivative Filter
3. Set Point Weighting
4. Integrator Windup
5. Computer Implementation
6. Tuning
7. Summary

Summary

- Remember control fundamentals
  - Load disturbances and measurement noise
  - Reference signals
  - Model uncertainty
  - Six responses are needed
- Many practical and operational issues
  - Derivative filter
  - Set point weighting (2DOF)
  - Integrator windup
  - Digital control
  - Tuning
- Relevant for all control systems

Recommendations for Studies

The PI(D) controller is the most common controller. You should learn how it works and how to tune it. A laboratory is strongly recommended, you can take a lab course in the spring of 2003.

Reading suggestions:

- Study Chapter 6