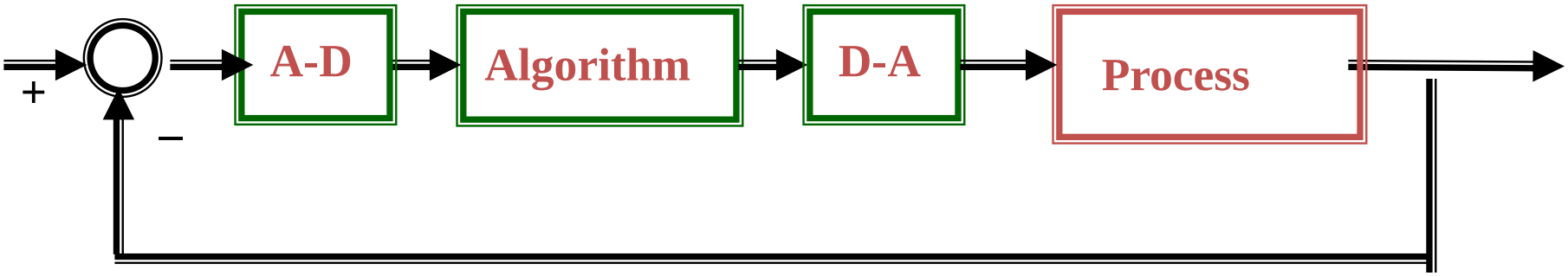
EE3304 (Part II)
Digital Control Systems
Chapter Three
PID Control

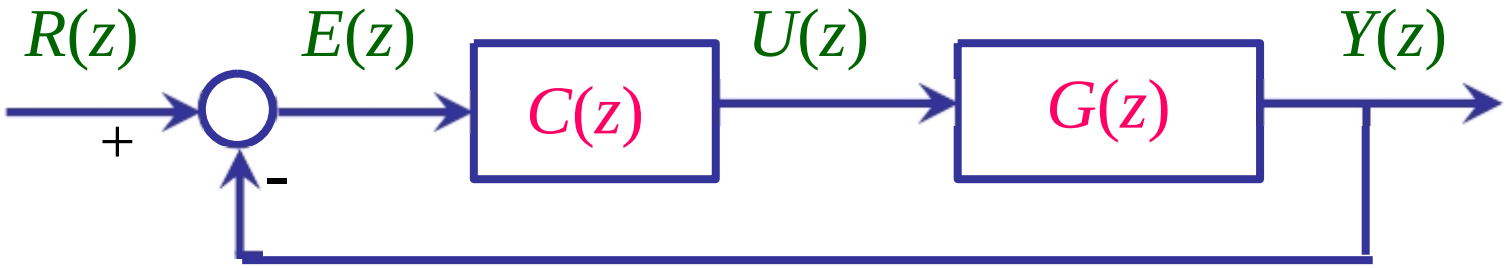
Xiang Cheng

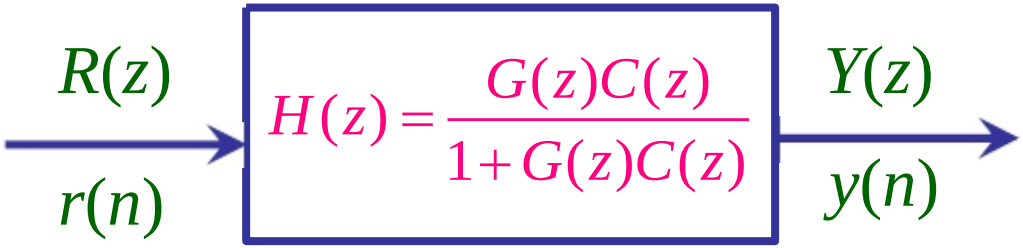
Associate Professor

Department of Electrical & Computer Engineering
The National University of Singapore

Phone: 65166210 Office: Block E4-08-07
Email: elexc@nus.edu.sg





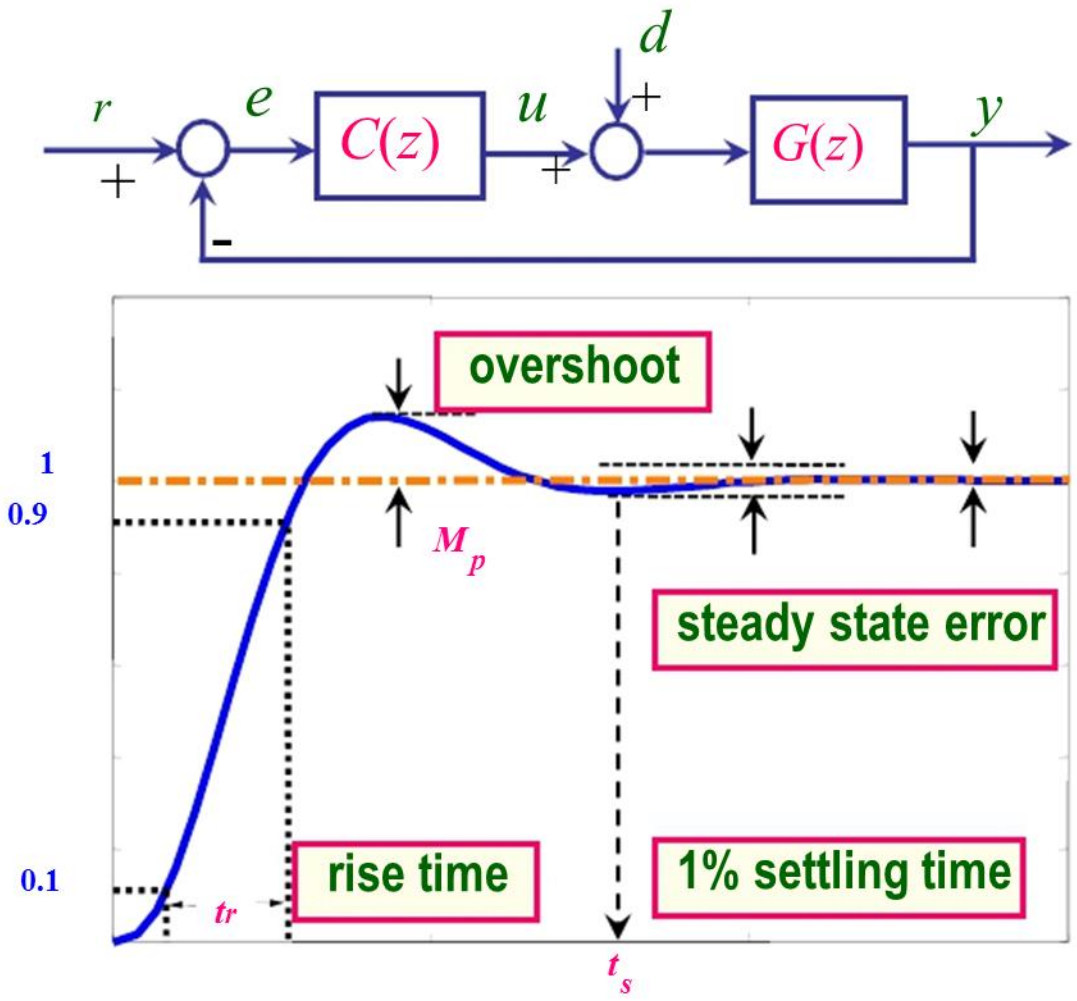


The problem becomes how to choose an appropriate controller, $C(z)$, such that the closed loop system, $H(z)$, will have desired properties.

Performance Specifications:

- stability
- steady state accuracy
- settling time
- overshoot
- rise time
- disturbance rejection
- others

Consider a unity feedback system



Often the desired CLTF is chosen to be

$$H_{desired}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

A unity desired closed-loop TF can be achieved at steady state.

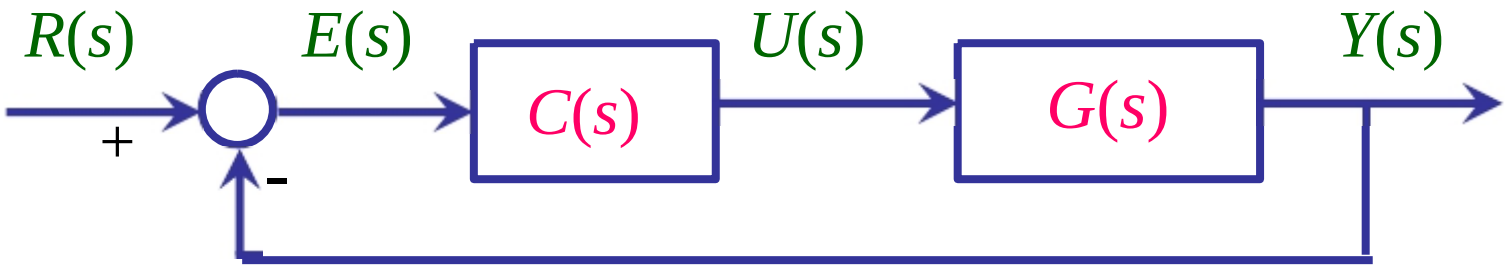
$$(t_s, M_p, t_r) \Leftrightarrow (\zeta, \omega_n) \Leftrightarrow$$ Pole location

desired pole location in *s*-plane

$z = e^{Ts}$

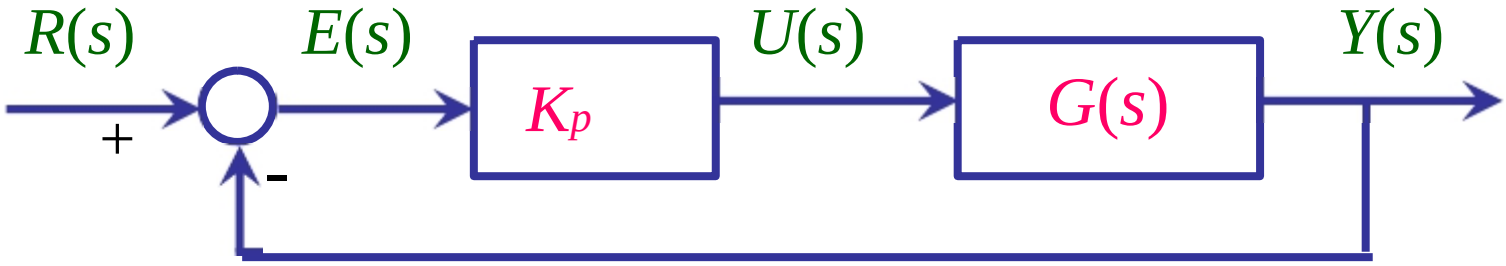
desired pole location in z-plane

From now on we will focus upon the design of the controller.



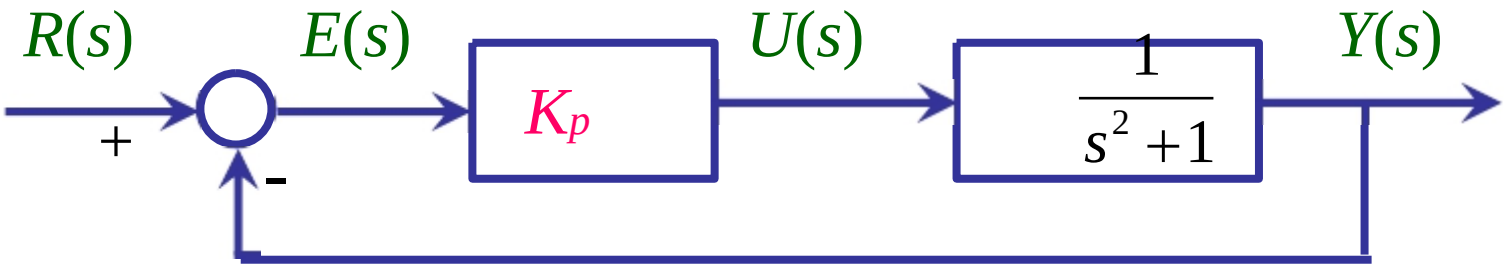
What's the simplest feedback-controller you can imagine?

Proportional Control



Proportional controller should always be the first choice.

Let's try the P control on a harmonic oscillator:



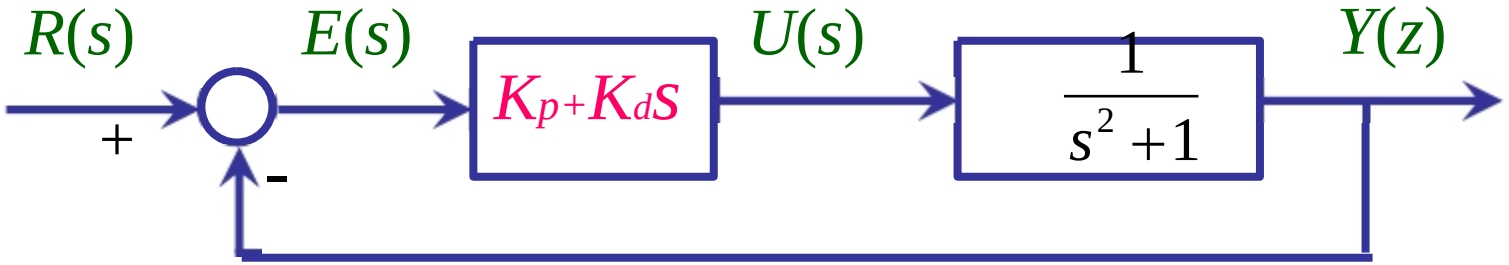
What's the closed-loop T.F.?

$$\frac{\frac{K_p}{s^2+1}}{1+\frac{K_p}{s^2+1}} = \frac{K_p}{s^2+1+K_p}$$

Is it stable?

It is only marginally stable!

Proportional and Derivative Control



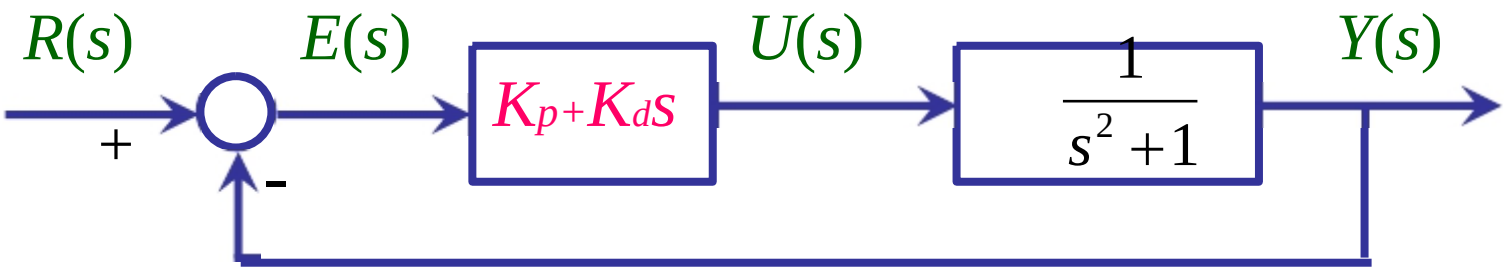
$$\frac{\frac{K_p+K_d s}{s^2+1}}{1+\frac{K_p+K_d s}{s^2+1}} = \frac{K_p+K_d s}{s^2+K_d s+1+K_p}$$

Is it stable?

Yes.

Derivative control can improve the stability of the system!

Proportional and Derivative Control

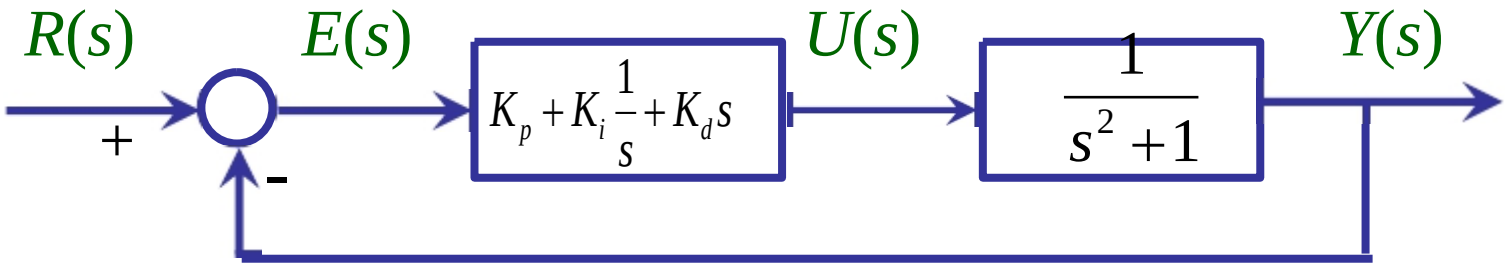


Will the error go to zero at steady state when the reference is a constant step?

No! There is no integrator. It is a type-0 system. The steady state error is not ZERO!

How can we make the steady-state error to be zero?

Proportional, Derivative and Integral (PID) Control

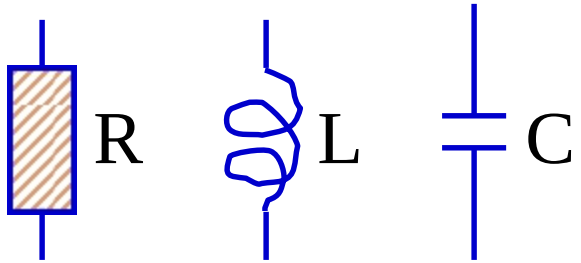


Integral control can improve the steady-state accuracy!

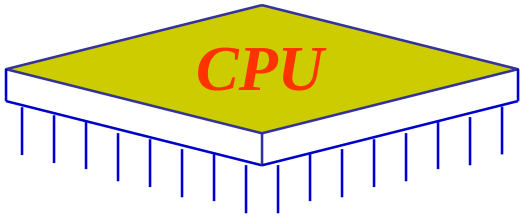
It turns out that PID control can solve 90% of the control problems in the real world!

PID control is widely used in process control and most of industrial control systems. Statistics show that more than 90% of industrial processes are actually controlled by PID type of controllers. PID control consists of three essential components, namely, **P** (proportional control), **I** (integral control) and **D** (derivative control).

Analog PID can be easily implemented by analog components – RLC circuits.



Digital PID must be implemented by – microprocessors.



Digital PID is the discretization and the approximation of – the analog PID.

Implementation of an analog PID

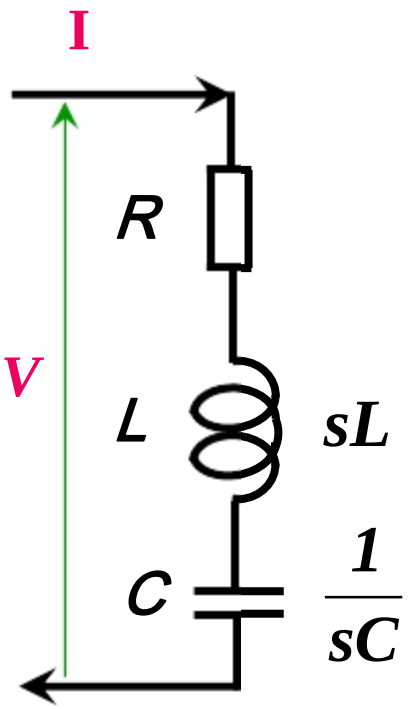
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Assume a zero initial condition

$$U(s) = K_p E(s) + K_i \frac{E(s)}{s} + s K_d E(s)$$

RLC circuit

$$V(s) = RI(s) + \frac{1}{C} \frac{I(s)}{s} + sLI(s)$$



Implementation of digital PID

Digital **P** (proportional control), **I** (integral control) and **D** (derivative control) can be derived from the continuous-time counterpart by discretization using the backward rule.

Proportional Control

The continuous-time P control is

$$u(t) = K_p e(t) \quad \Rightarrow \quad C(s) = K_p$$

What is the digital P controller C(z)?

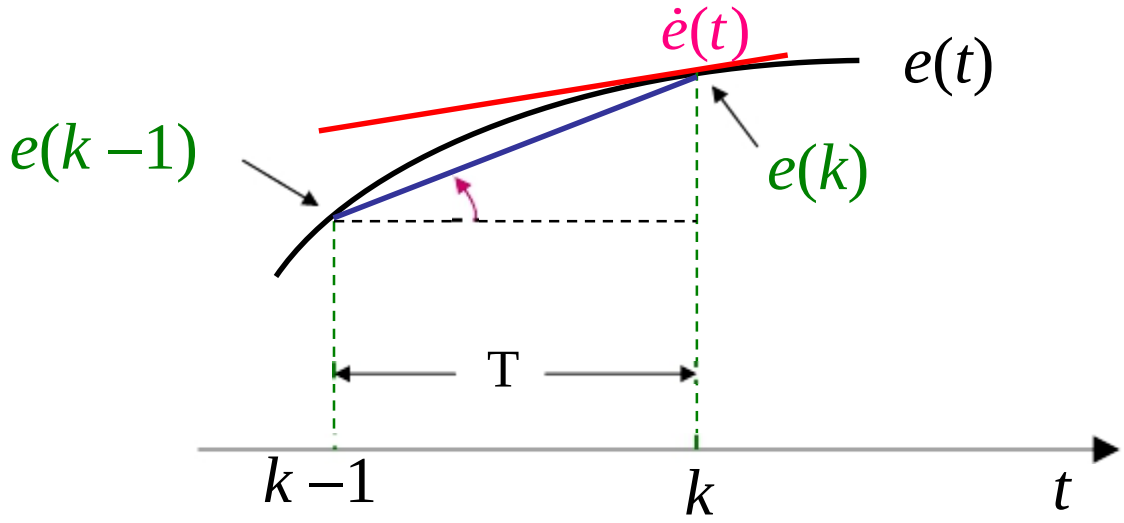
$$u(k) = K_p e(k) \quad \Rightarrow \quad C(z) = K_p$$

Derivative Control

The continuous-time version is

$$u(t) = K_d \dot{e}(t) \quad \Rightarrow \quad C(s) = K_d s$$

What is the simplest way to approximate the derivative of the error signal, $\dot{e}(t)$?



Backward rule $\dot{e}(t) \approx \dfrac{e(t) - e(t-T)}{T}$

$$u(k) = K_d \frac{e(k) - e(k-1)}{T} \quad \Rightarrow \quad U(z) = K_d \frac{E(z) - z^{-1}E(z)}{T}$$

$$\Rightarrow \frac{U(z)}{E(z)} = C(z) = K_d \frac{1 - z^{-1}}{T} = K_d \frac{z-1}{Tz}$$

Can we use forward rule here?

No. We cannot use future information!

$$\dot{e}(t) \approx \frac{e(t+T) - e(t)}{T}$$

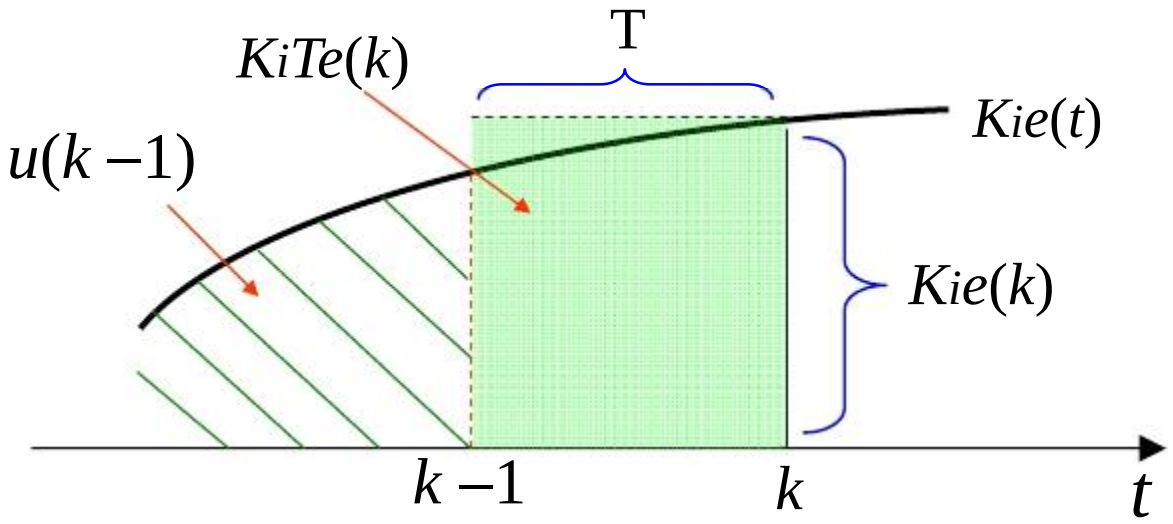
Integral Control

The continuous-time version is

$$u(t) = K_i \int_0^t e(\tau) d\tau \quad \Rightarrow \quad C(s) = K_i \frac{1}{s}$$

$$u(t) = K_i \int_0^t e(\tau) d\tau = K_i \int_0^{t-T} e(\tau) d\tau + K_i \int_{t-T}^t e(\tau) d\tau = u(t-T) + K_i \int_{t-T}^t e(\tau) d\tau$$

What is the simplest way to approximate $\int_{t-T}^t e(\tau) d\tau$? $\int_{t-T}^t e(\tau) d\tau \approx Te(t)$



The discrete-time counterpart is

$$u(k) = u(k-1) + K_i Te(k) \Rightarrow \quad u(k) - u(k-1) = K_i Te(k)$$

$$U(z)(1 - z^{-1}) = K_i TE(z) \Rightarrow \quad \frac{U(z)}{E(z)} = C(z) = K_i \frac{T}{1 - z^{-1}} = K_i \frac{zT}{z-1}$$

Digital PI Control

$$C(z) = K_p + K_i \frac{Tz}{z-1}$$

Digital PD Control

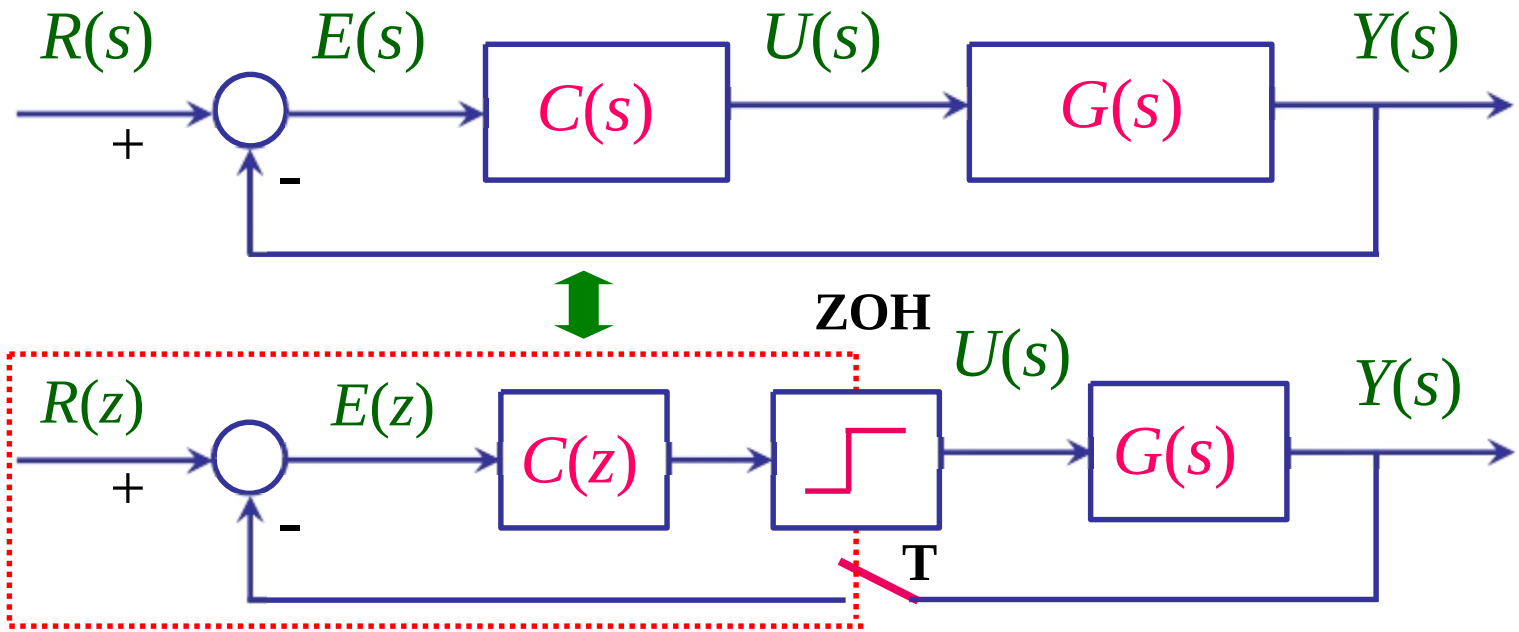
$$C(z) = K_p + K_d \frac{z-1}{Tz}$$

Digital PID Control

$$C(z) = K_p + K_i \frac{Tz}{z-1} + K_d \frac{z-1}{Tz}$$

Digital control system design using emulation

In this approach, we design a continuous-time controller that meets all design specifications and then discretize it using a bilinear transformation or other discretization technique to obtain an equivalent digital controller.



What is the condition for this_method to work as good as the analog controller?

This method works if the sampling rate is 30 times faster than the Bandwidth of the closed loop system. Further refinement is necessary for the case where the sampling rate is 6 times the bandwidth.

Emulation Design Procedure

Step 1. From time domain performance requirements, determine the desired damping ratio, ζ , and natural frequency, ω_n and hence the desired closed-loop TF, $H_d(s)$.

Step 2. Choose an appropriate type of controller, e.g. P, PI, PD, PID, etc.

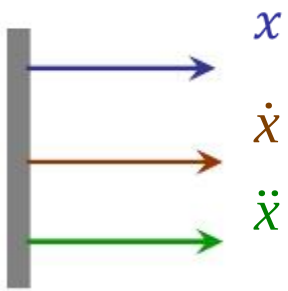
Step 3. Comparing characteristic equations of desired and actual CLTF, determine the controller parameters, and hence $C(s)$.

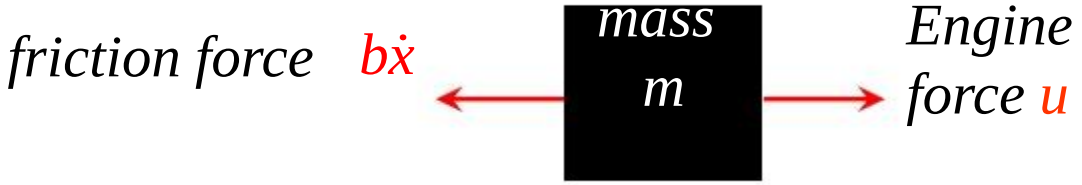
Step 4. From the system bandwidth, select a sampling period T.

Step 5. Discretize $C(s)$ to $C(z)$ via Tustin rule (replace s by $\frac{2}{T}\frac{z-1}{z+1}$).

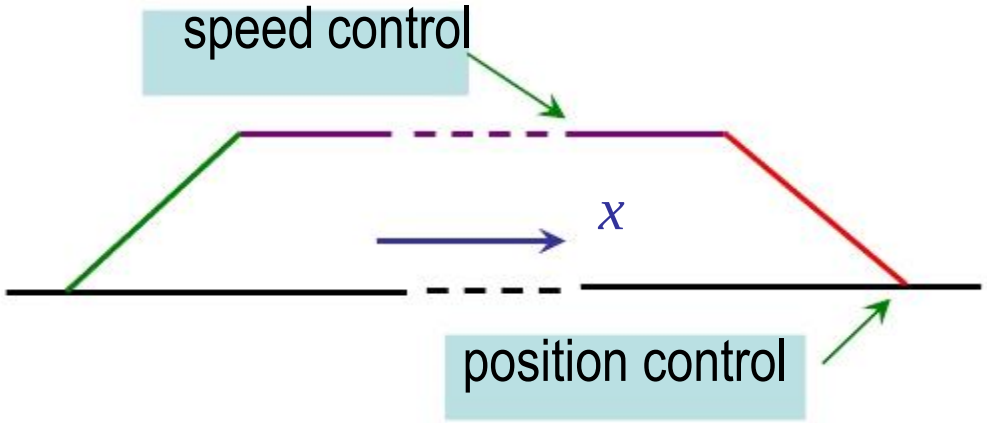
Design example 3.1 position control with P or PD

Consider a vehicle, which has a weight $m = 1000$ kg. Assuming the average friction coefficient $b = 100$, design a position control system such that the vehicle can move 100 m in 7.3 s with an overshoot less than 18%.





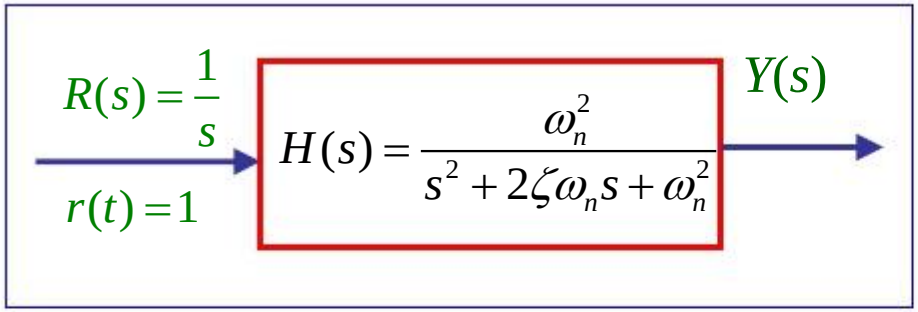
x – *displacement*
$\dot{x}$ – *speed*
$\ddot{x}$ – *acceleration*
u – *total force*

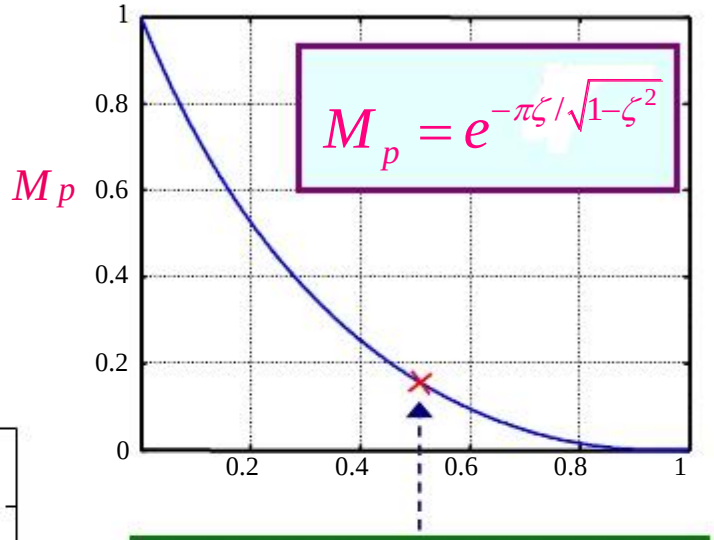


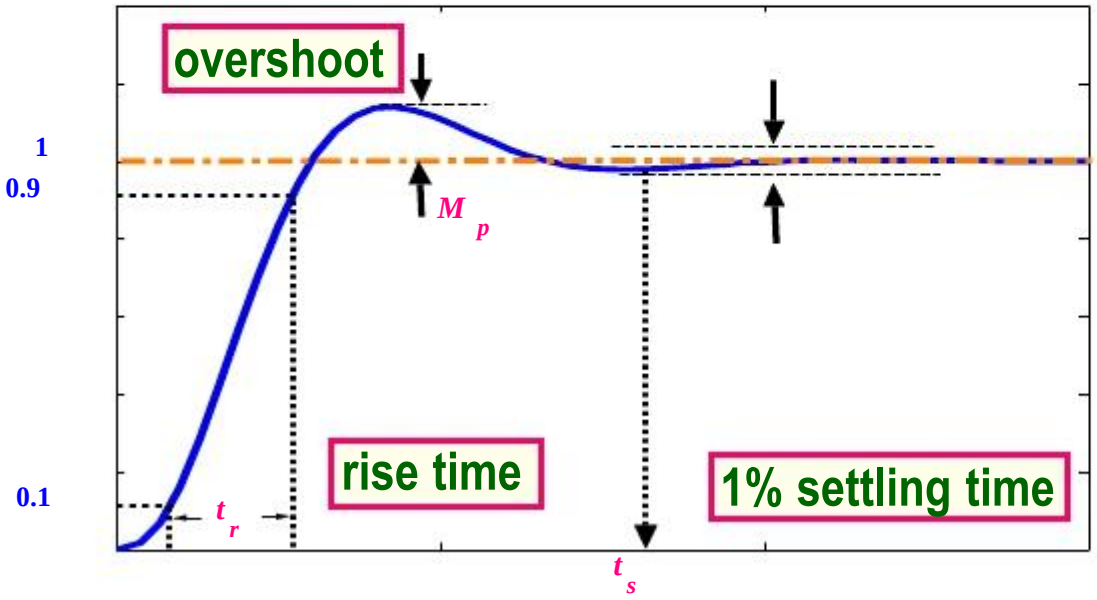
$$m\ddot{x} = u - b\dot{x} \quad \Rightarrow \quad m\ddot{x} + b\dot{x} = u \quad \Rightarrow \quad \frac{X(s)}{U(s)} = \frac{1}{ms^2 + bs}$$

Requirement: it can move 100 m in 7.3 s with an overshoot less than 18%.

Step 1. Deriving ξ and ω_n from the design specifications:







$$\zeta \geq 0.4791 \quad \Rightarrow \quad \zeta = 0.5$$

$$t_s \cong \frac{4.6}{\zeta\omega_n} \Rightarrow \omega_n \cong \frac{4.6}{t_s\zeta}$$

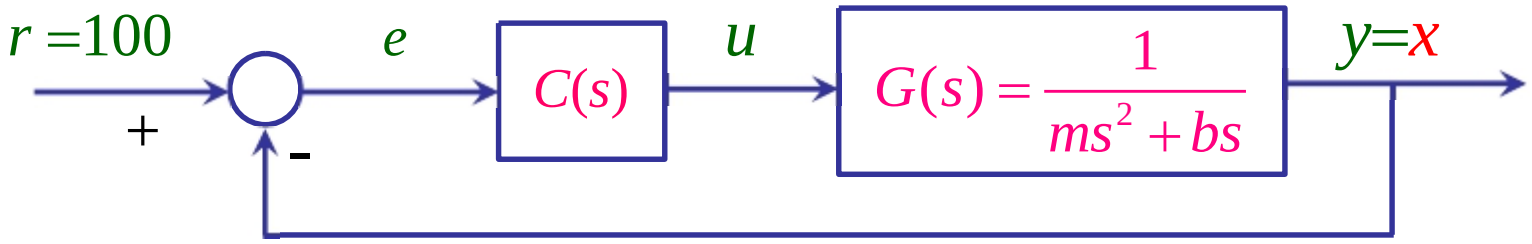
$$\Rightarrow \omega_n = \frac{4.6}{7.3\times 0.5} = 1.2603$$

Desired CLTF $\Rightarrow$ $H_d(s) = \dfrac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} = \dfrac{1.5883}{s^2+1.2603s+1.5883}$

Step 2. Choose an appropriate controller

Which controller should be our first choice?

Let us choose a P controller first, i.e., $C(s) = K_p$,



Is it possible to get zero steady state error without using integral control?

Yes. The system itself has a integrator! So it is a type-I system already.

What's the closed-loop T.F.?

$$H(s) = \frac{Y(s)}{R(s)} = \frac{G(s)C(s)}{1 + G(s)C(s)}$$

$$= \frac{0.001K_p}{s^2 + 0.1s + 0.001K_p}$$

However the desired CE and actual CE are respectively

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \quad \Rightarrow \quad s^2 + 1.2603s + 1.5883$$

$$\Rightarrow \quad s^2 + 0.1s + 0.001K_p$$

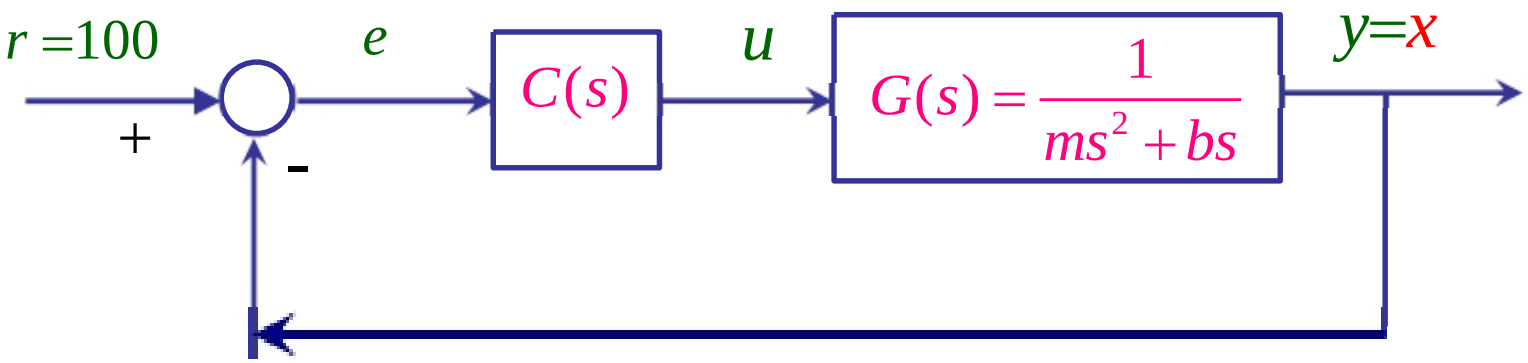
Is it possible to choose Kp such that they match each other?

Impossible!

So we need to increase the complexity of the controller.

Now let us choose a PD controller,

$$C(s) = K_p + K_d s$$



The closed-loop TF is

$$H(s) = \frac{Y(s)}{R(s)} = \frac{G(s)C(s)}{1+G(s)C(s)}$$

$$= \frac{0.001K_d s + 0.001K_p}{s^2 + (0.1 + 0.001K_d)s + 0.001K_p}$$

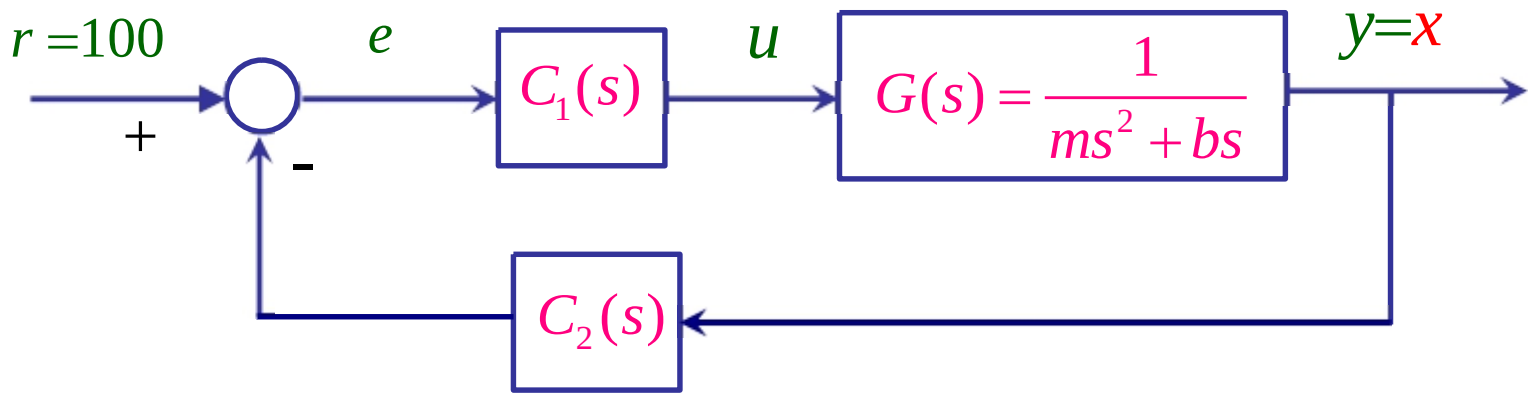
Compare with $H_d(s) = \dfrac{1.5883}{s^2 + 1.2603s + 1.5883}$

Is it possible to match the poles (the denominators)? Yes.

Is it possible to match the zeros (the numerators)? No.

Now let's modify the PD controller, with $C_1(s) = K_p$ and $C_2(s) = 1 + K_d s$

The block diagram is



What is the feedforward TF? $C_1(s)G(s)$

What is the open-loop TF? $C_1(s)G(s)C_2(s)$

The closed-loop TF is

$$H(s) = \frac{Y(s)}{R(s)} = \frac{G(s)C_1(s)}{1 + G(s)C_1(s)C_2(s)}$$

$$= \frac{0.001K_p}{s^2 + (0.1 + 0.001K_pK_d)s + 0.001K_p}$$

Step 3. Calculate controller parameters

The closed-loop transfer function of the position control system with the PD control law, i.e.

$$H(s) = \frac{0.001K_p}{s^2 + (0.1 + 0.001K_pK_d)s + 0.001K_p}$$

and the desired transfer function that produces desired performance

$$H_d(s) = \frac{1.5883}{s^2 + 1.2603s + 1.5883}$$

Can we match these two TF?

Comparing the coefficients on the denominators, we have

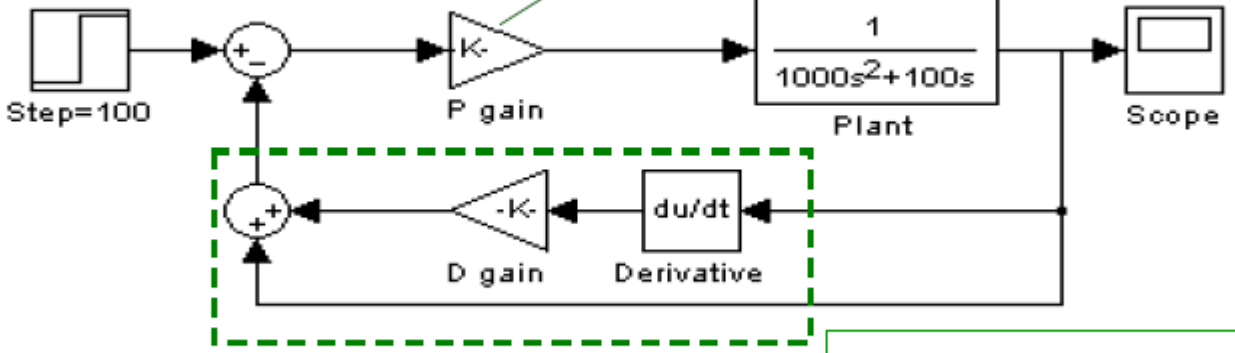
$$0.001K_p = 1.5883 \Rightarrow K_p = 1588.3$$

$$0.1 + 0.001K_pK_d = 1.2603 \Rightarrow K_d = 0.73$$

The resulting CLTF is $H(s) = H_d(s) = \frac{1.5883}{s^2 + 1.2603s + 1.5883}$

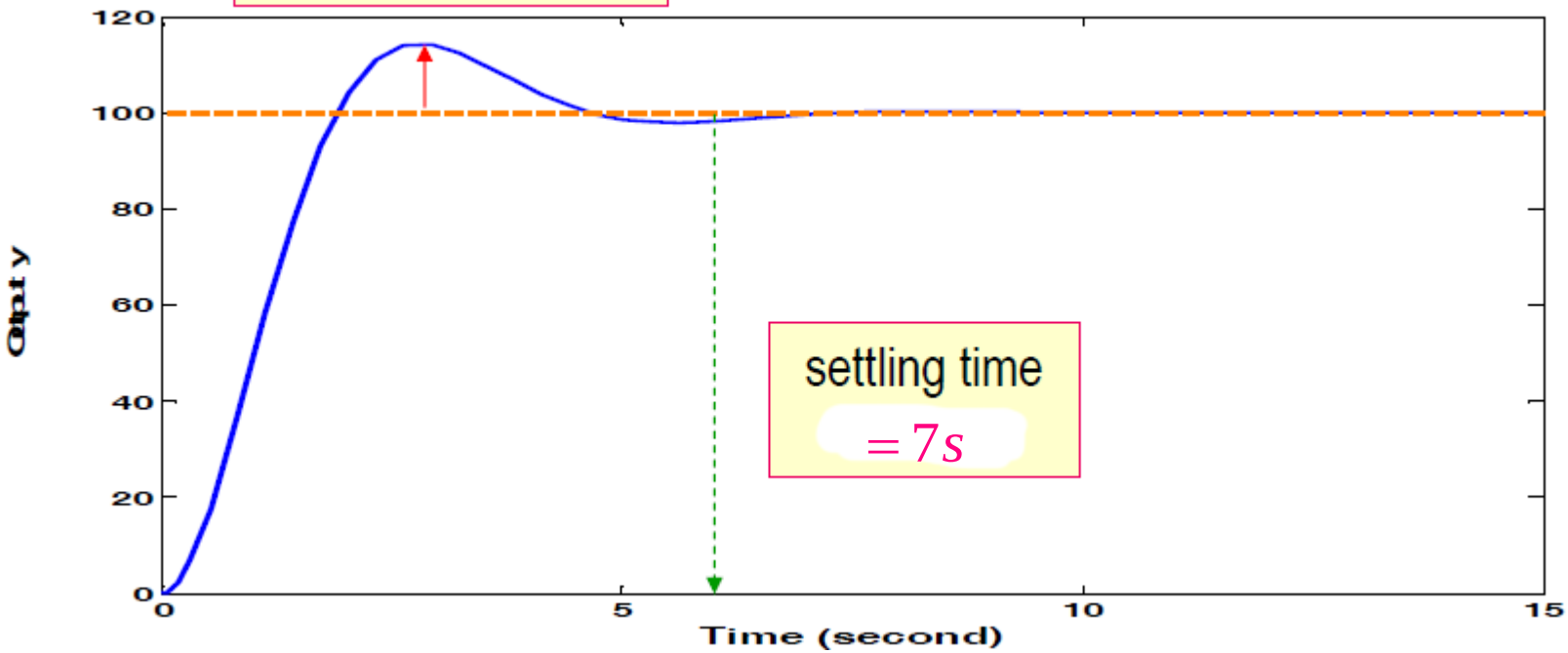
Verification through SIMULINK

$C_1(s) = 1588.3$



$C_2(s) = 0.73s + 1$

overshoot=16.2%



But that is the analog controller. We need to convert it into digital controller.
We need to decide the sampling rate first.

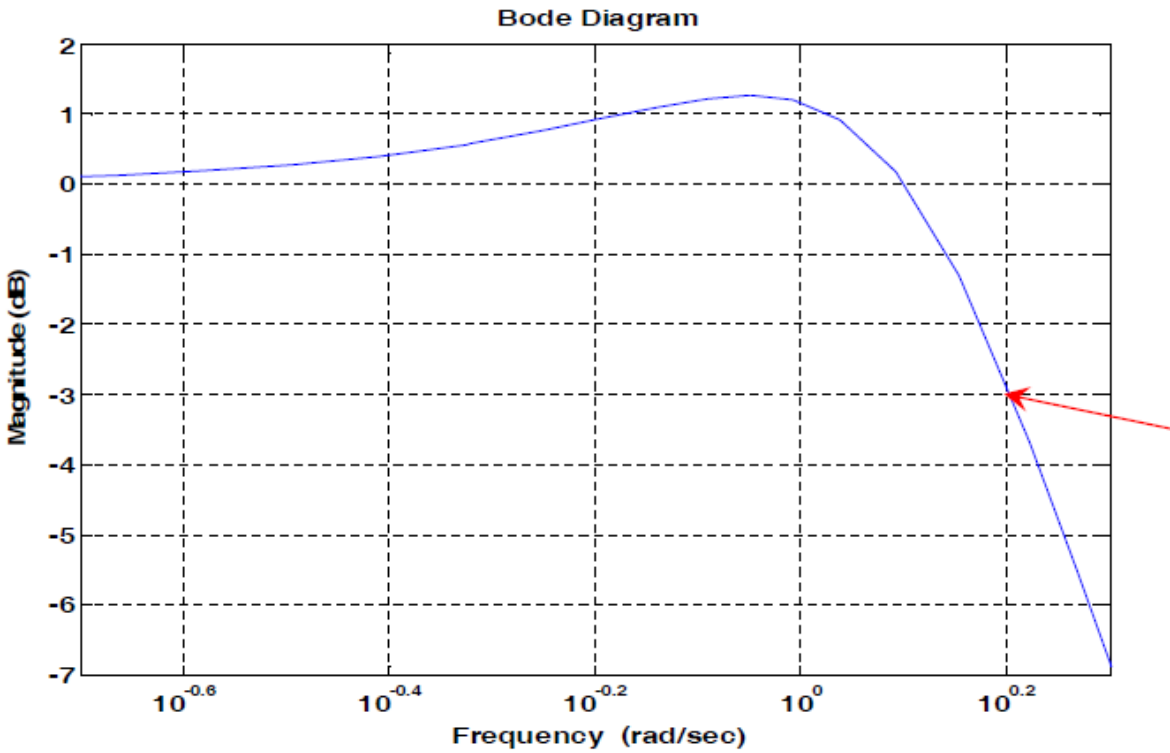
Step 4. From the system bandwidth of the closed-loop, select a sampling period T

But what is bandwidth of a system?

The bandwidth is the cut-off frequency for signal which can pass through the filter!

In other words, the frequency response is small for frequencies higher than the bandwidth.

It is defined to be the frequency where the magnitude is -3dB below the DC gain.



$$-3dB \rightarrow \frac{1}{\sqrt{2}} = 0.7$$

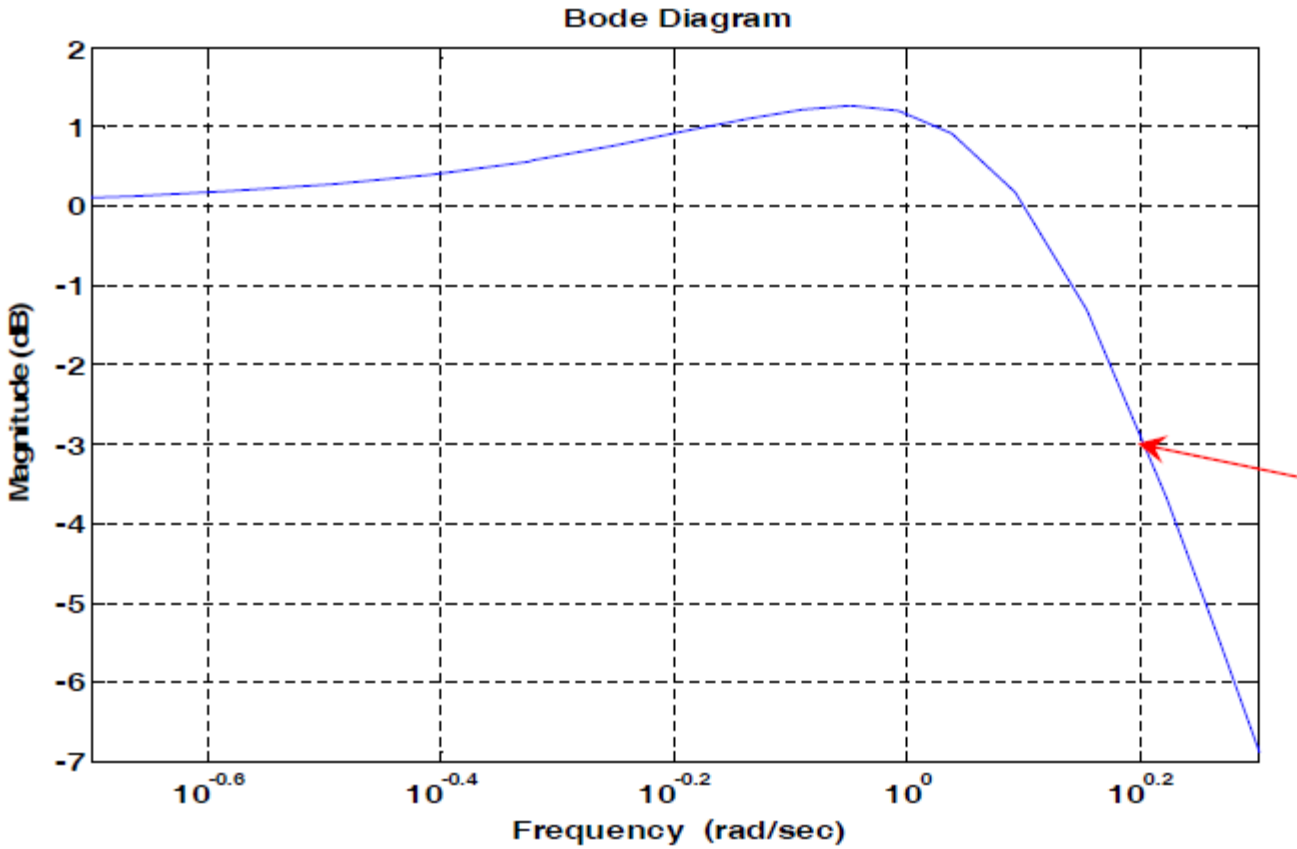
Bandwidth
= 1.6 rad/sec
= 0.2546 Hz

For second order system, the bandwidth can be approximated by the natural frequency.

$$H_d(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \frac{1.5883}{s^2 + 1.2603s + 1.5883} \Rightarrow \omega_n = 1.2603$$

From this example, you can see that the natural frequency is indeed close to the bandwidth.

Step 4. From the system bandwidth of the closed-loop, select a sampling period T



Bandwidth
= 1.6 rad/sec
= 0.2546 Hz

$$-3dB \rightarrow \frac{1}{\sqrt{2}} = 0.7$$

Let's compare two choices:

(1) choose a sampling rate at least 30 times the bandwidth

$$T = 0.1 < 1/(30 \times 0.2546) = 0.13$$

(2) choose a sampling rate at least 6 times the bandwidth

$$T = 0.5 < 1/(6 \times 0.2546) = 0.6546$$

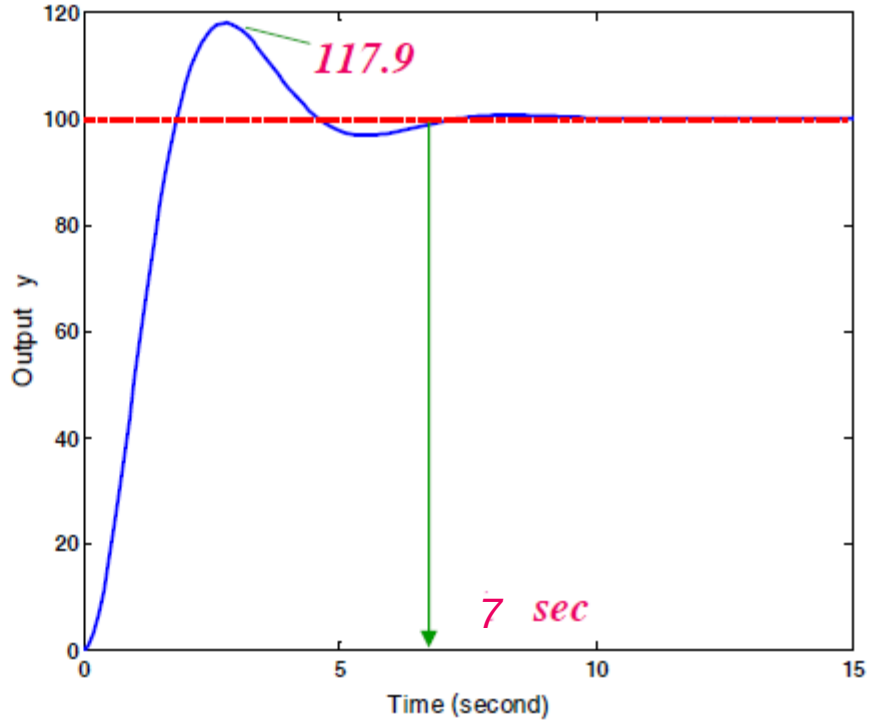
Step 5. Discretize $C(s)$ with the selected sampling period T

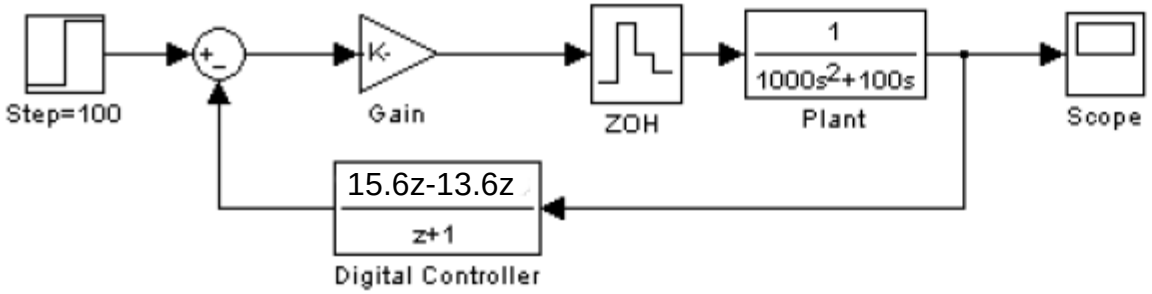
(1) Discretize the continuous-time PD control law with $T=0.1$ seconds using Tustin rule (replace s by $\frac{2}{T}\frac{z-1}{z+1}$),

$$C_2(z) = C_2(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} = 0.73s+1\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}}$$

$$= 0.73\frac{2}{0.1}\frac{z-1}{z+1}+1$$

$$= \frac{15.6z-13.6}{z+1}$$





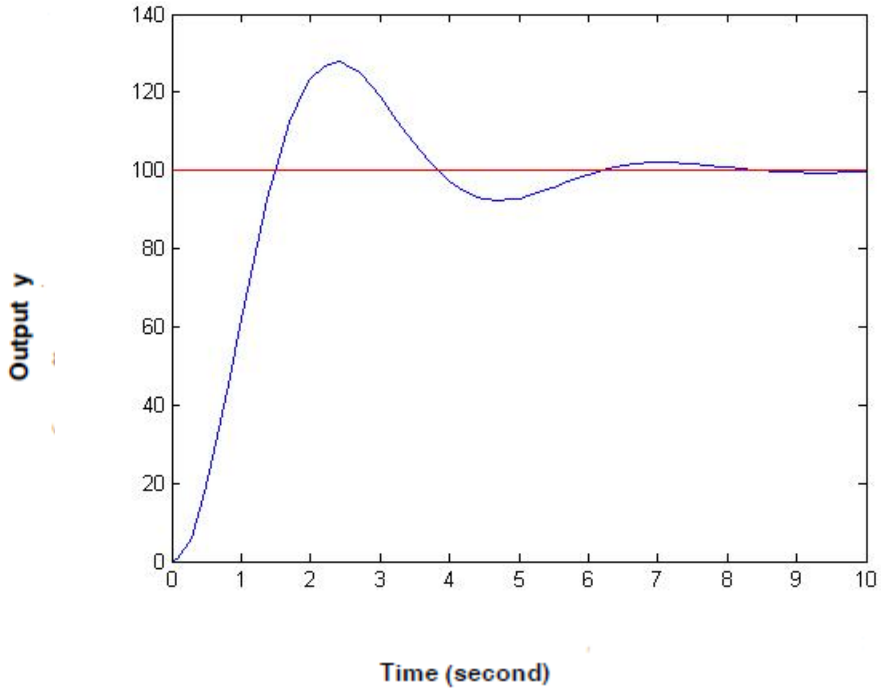
Performance still meets the time-domain specifications

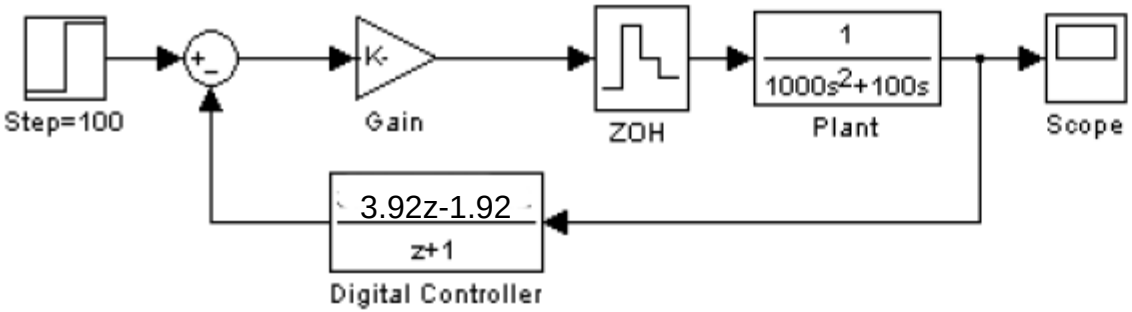
(2) Now discretize the continuous-time PD control law with $T = 0.5$ seconds using a bilinear transformation method, i.e.

$$C(z) = C_2(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} = 0.73s + 1\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}}$$

$$= 0.73\frac{2}{0.5}\frac{z-1}{z+1} + 1$$

$$= \frac{3.92z - 1.92}{z+1}$$





Performance is NOT as good as the continuous time case

Overshoot is above 20%.
The settling time is around 8s.

Notes:

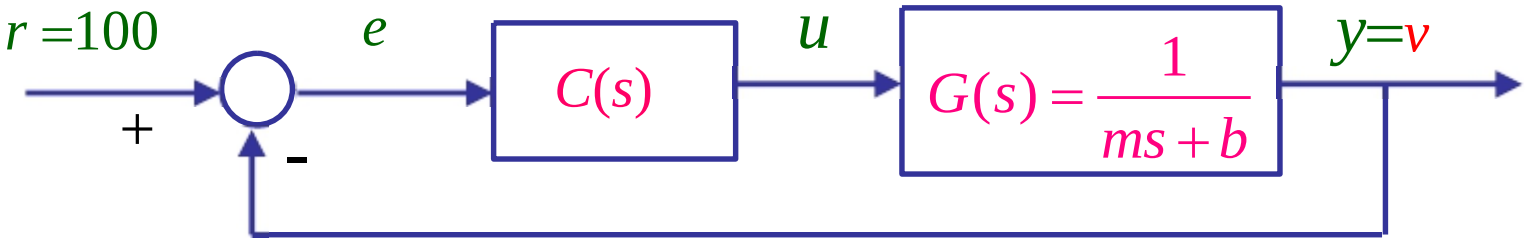
1. In practice, emulation based design can be used directly without validation when the sampling frequency is sufficiently high in comparing with the system bandwidth.

2. The system bandwidth is referred to the closed-loop system bandwidth.

3. For second order system, the bandwidth is slightly higher than (but close to) the natural frequency. If you do not know how to compute bandwidth precisely, you can use the natural frequency to approximate it.

Design Example 3.2 ***speed control with PI***

Consider the vehicle, which has a weight $m = 1000$ kg. Assuming the average friction coefficient $b = 100$, design a speed control system such that the vehicle can reach 100 km/h from 0 km/h in 8 s with an overshoot less than 5%.



Let us try a PI controller, i.e. , $C(s) = K_p + K_i \frac{1}{s}$

Why do we need the integral control this time?

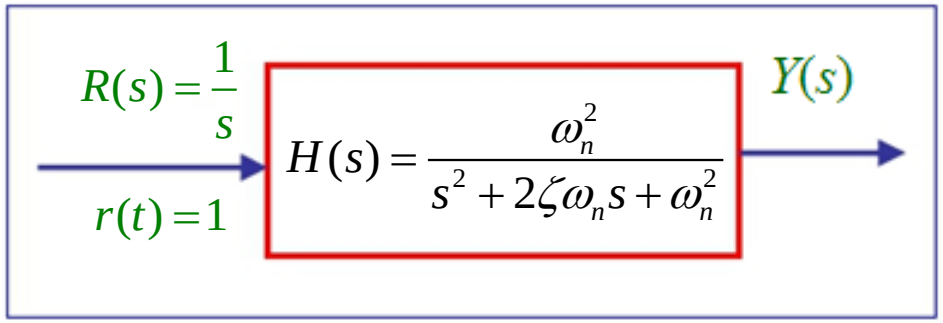
To assure the steady state accuracy!

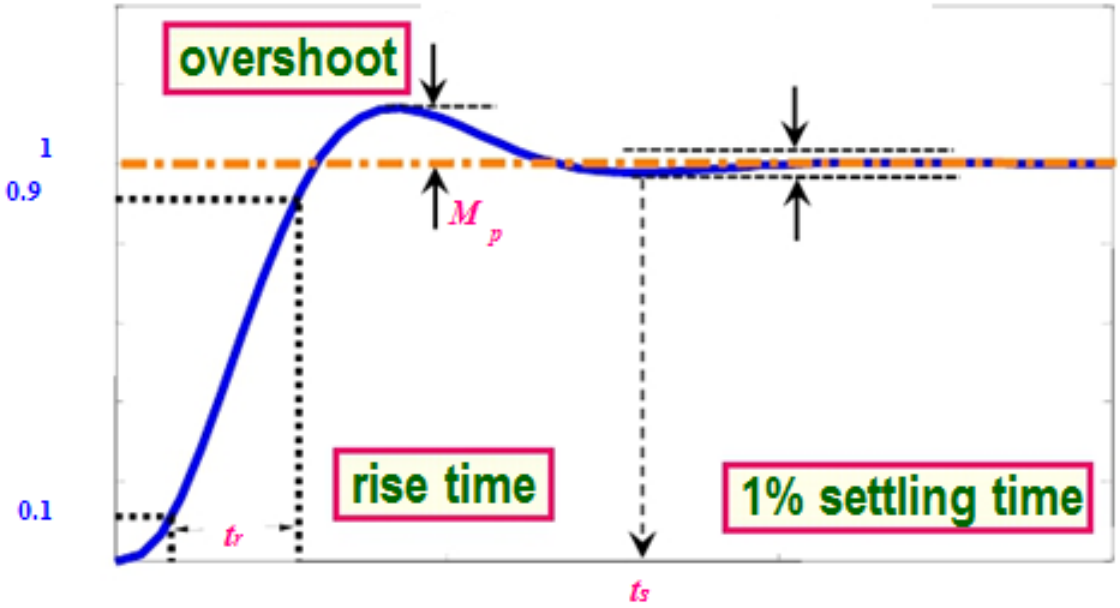
$$H(s) = \frac{Y(s)}{R(s)} = \frac{G(s)C(s)}{1+G(s)C(s)}$$

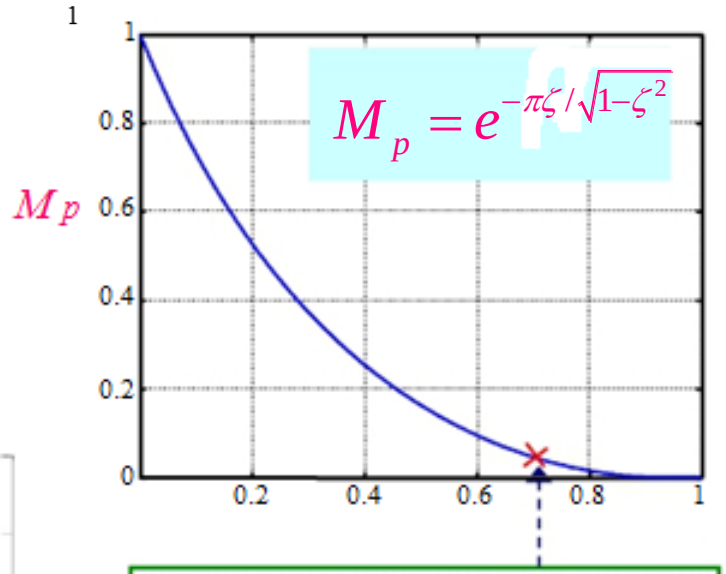
$$= \frac{0.001K_p s + 0.001K_i}{s^2 + (0.1 + 0.001K_p)s + 0.001K_i}$$

Requirement: it can reach 100 km/h from 0 km/h in 8 s with an overshoot less than 5%.

First derive ξ and ω_n from the design specifications:







$$\zeta \geq 0.6901 \Rightarrow \zeta = 0.7$$

$$t_s \cong \frac{4.6}{\zeta\omega_n} \Rightarrow \omega_n \cong \frac{4.6}{t_s\zeta}$$

$$\Rightarrow \omega_n = \frac{4.6}{8\times 0.7} = 0.82$$

Desired CLTF $\Rightarrow$ $H_d(s) = \frac{0.67}{s^2+1.15s+0.67}$

Recall the closed-loop transfer function of the cruise control system with the PI control law, i.e.

$$H(s) = \frac{0.001K_p s + 0.001K_i}{s^2 + (0.1 + 0.001K_p)s + 0.001K_i}$$

and the desired transfer function that produces desired performance

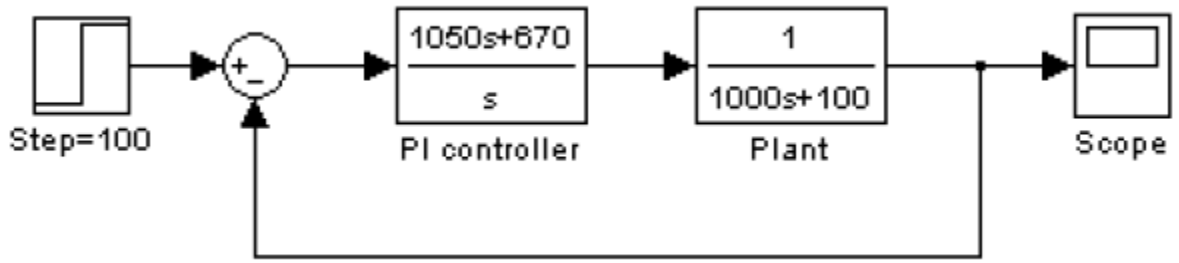
$$H_d(s) = \frac{0.67}{s^2 + 1.15s + 0.67}$$

Comparing the coefficients of the denominators, we have

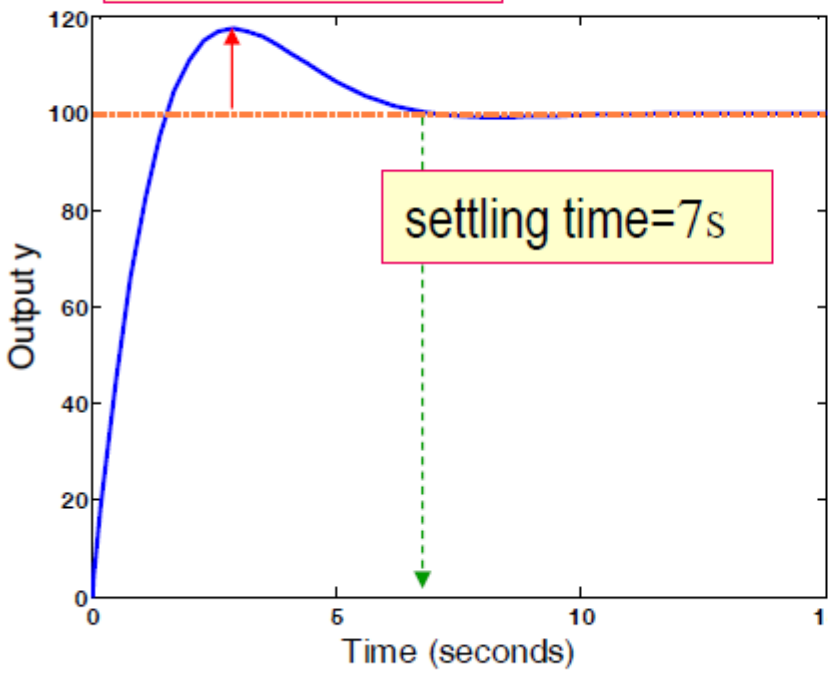
$$0.1 + 0.001K_p = 1.15 \Rightarrow K_p = 1050$$

$$0.001K_i = 0.67 \Rightarrow K_i = 670$$

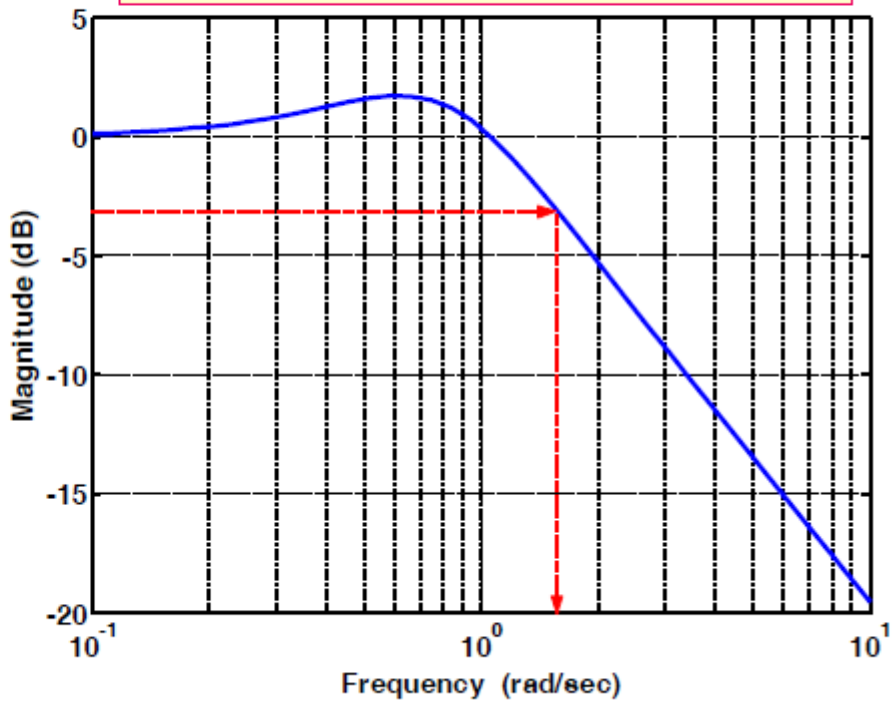
The resulting CLTF is $H(s) = \dfrac{1.05s + 0.67}{s^2 + 1.15s + 0.67}$



overshoot=18%



bandwidth=1.55 rad/sec=0.25 Hz

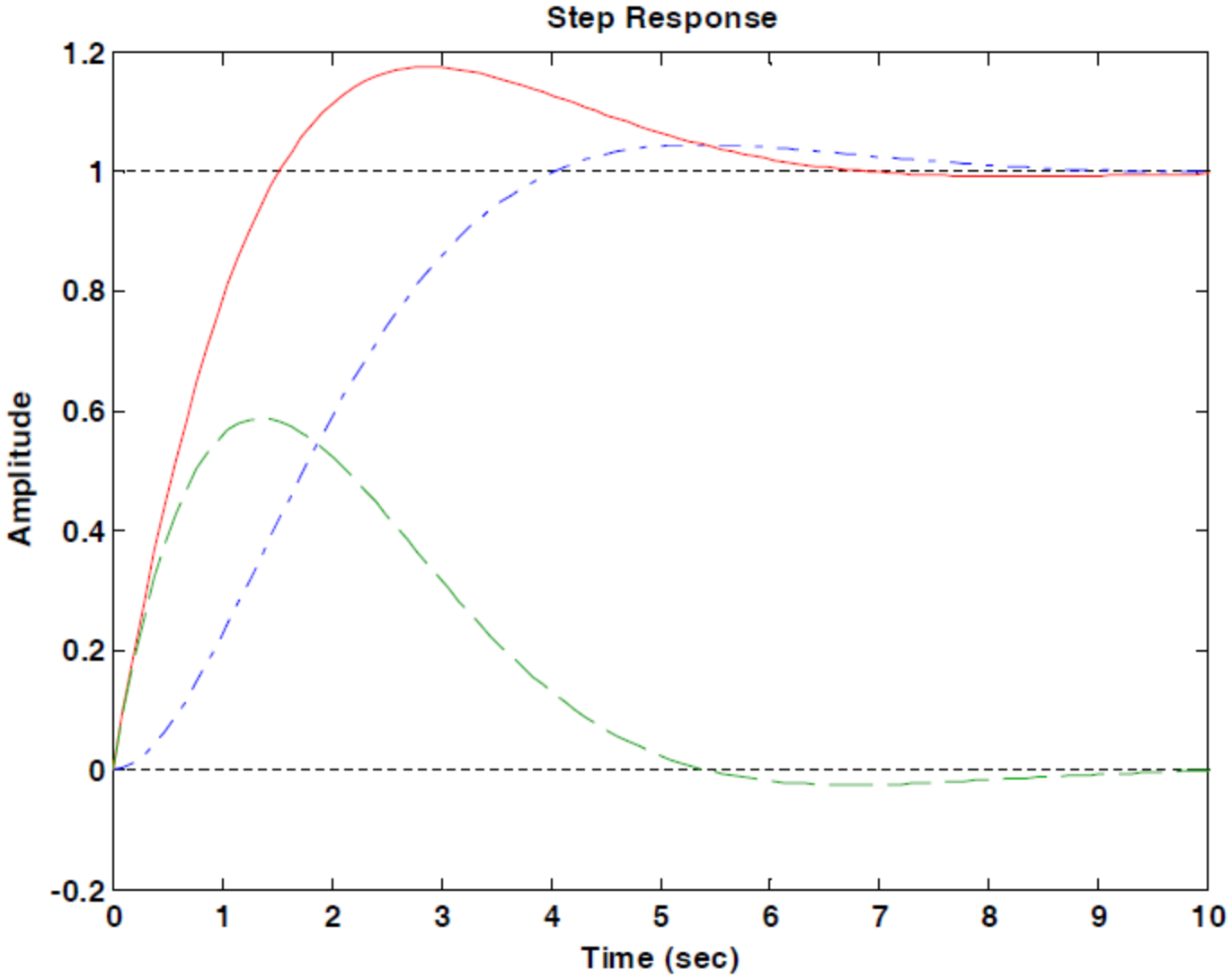


Simulation results show a rather large overshoot (why?)

We only match the poles!

The zeros of the closed loop are different from the desired one!

$$H(s) = \frac{1.05s + 0.67}{s^2 + 1.15s + 0.67} = \frac{0.67}{s^2 + 1.15s + 0.67} + \frac{1.05s}{s^2 + 1.15s + 0.67}$$



Can we also match the zeros?

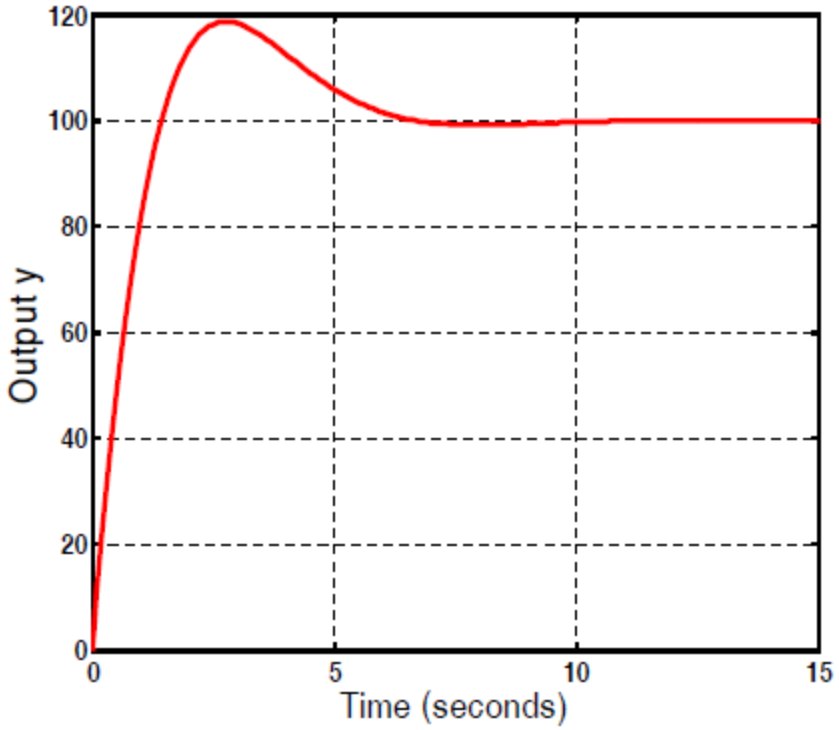
Yes, we can. But we need to use another type of controller.

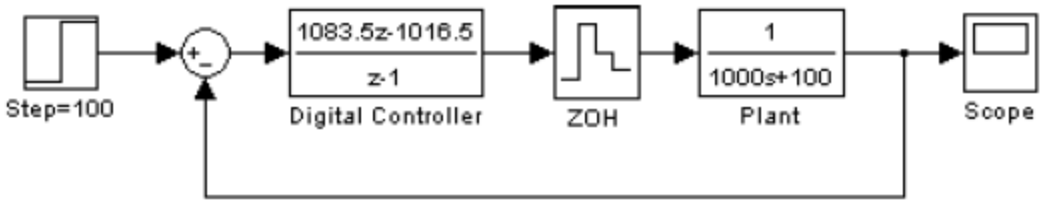
Choose a sampling rate 30 times the bandwidth. The continuous-time PI control law is discretized with $T = 0.1 < 1/(30 \times 0.25)$ using the Tustin rule,

$$C(z) = C(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} = \frac{1050s + 670}{s}\Bigg|_{s=\frac{2}{T}\frac{z-1}{z+1}}$$

$$= \frac{1050\dfrac{2}{0.1}\dfrac{z-1}{z+1} + 670}{\dfrac{2}{0.1}\dfrac{z-1}{z+1}}$$

$$= \frac{1083.5z - 1016.5}{z-1}$$





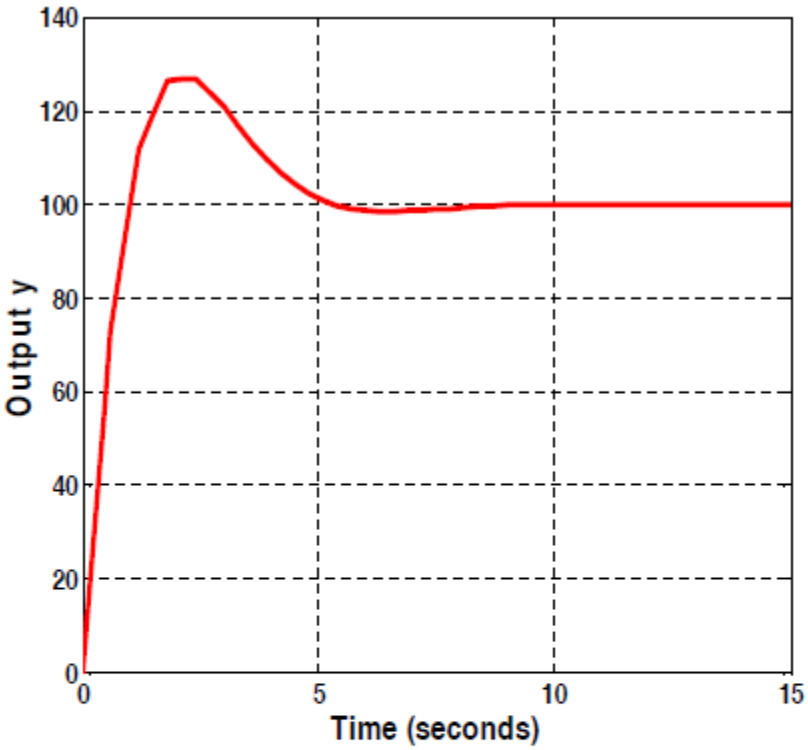
Performance is about the same as the continuous time case

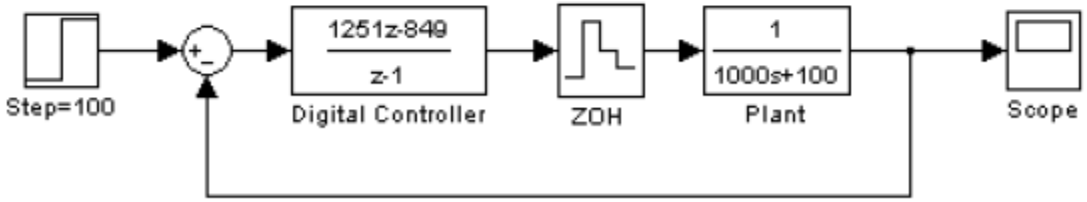
We now discretize the continuous-time PI control law with $T = 0.6 < 1/(6 \times 0.25)$ seconds using the bilinear transformation method, i.e.

$$C(z) = C(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} = \frac{1050s + 670}{s}\Bigg|_{s=\frac{2}{T}\frac{z-1}{z+1}}$$

$$= \frac{1050\dfrac{2}{0.6}\dfrac{z-1}{z+1} + 670}{\dfrac{2}{0.6}\dfrac{z-1}{z+1}}$$

$$= \frac{1251z - 849}{z - 1}$$





Performance is NOT as good as the continuous time case

We get a larger overshoot!

From the two design examples, let's summarize the emulation design procedure

Emulation Design Procedure

Step 1. Build the reference model → the perfect model which satisfies all the performance requirement.

How to do it?

From time domain performance requirements, determine the desired damping ratio, ζ , and natural frequency, ω_n and hence the desired closed-loop TF, the reference model, $H_d(s)$.

Step 2. Choose an appropriate type of controller, e.g. PID or others.
Start with simple controller first.

Step 3. Determine the control parameters and hence $C(s)$.

How to do it?

Match the closed loop TF, $H_{cl}(s)$ with the reference model $H_d(s)$.

Comparing denominator polynomials of reference model and actual CLTF, and match the coefficients!

Can we match both poles and zeros at the same time?

Not really. Sometimes we do, sometimes we do not. We are coming back to this point later.

Step 4. From the bandwidth of the closed loop system, select a sampling period T.

Step 5. Discretize $C(s)$ to $C(z)$ via Tustin rule (replace s by $\frac{2}{T}\frac{z-1}{z+1}$).

Break

State-of-the-art control systems

RHEX

Little Dog

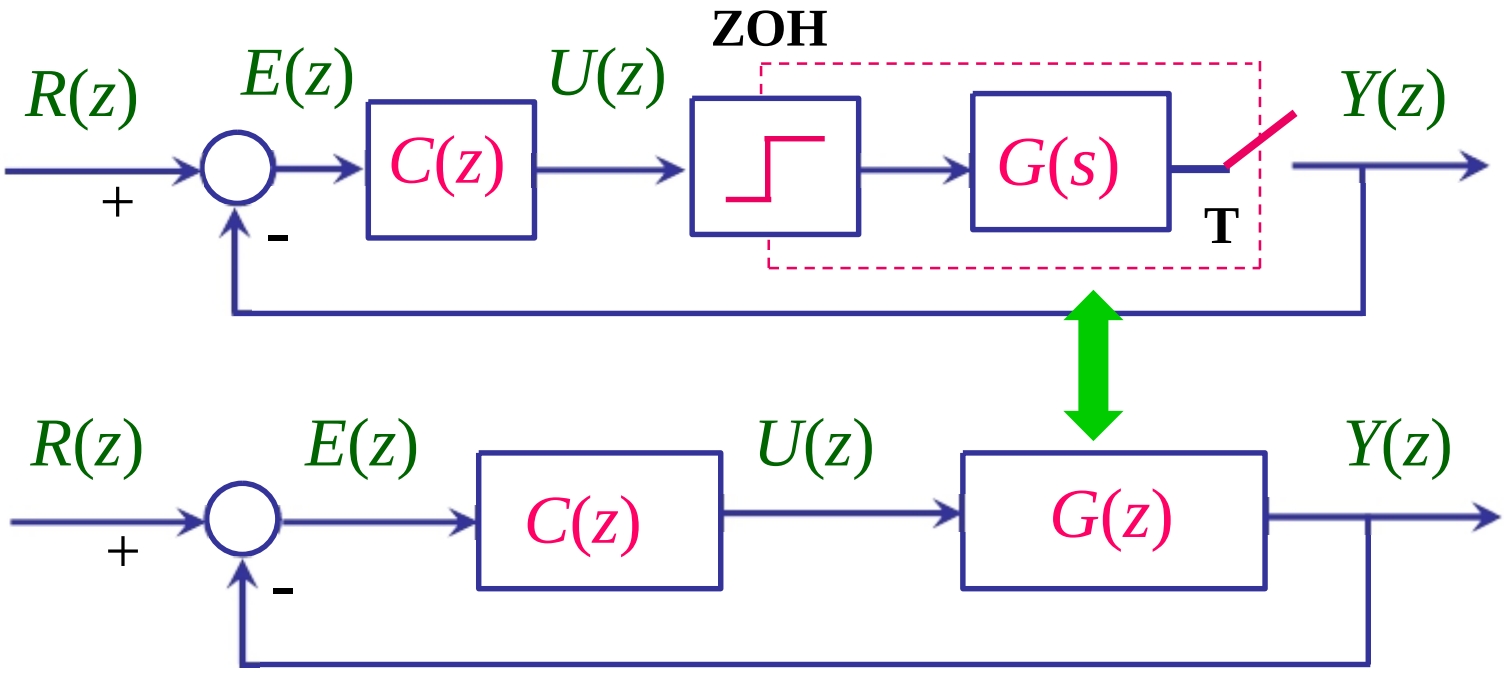
Big Dog

What is the condition for emulation design to work well?

The sampling frequency has to be sufficiently large. Therefore we need to know other ways.

Digital controller design based on discrete-time system

We can discretize the continuous-time plant first or directly work on a discrete-time plant to design a digital controller



where $G(z) = (1 - z^{-1}) Z[\frac{G(s)}{s}]$ and $C(z)$ is a digital controller, e.g. P, PI, PD, PID, etc.

Design Example 3.3* : *speed control with PI

Consider the vehicle, which has a weight $m = 1000\ \text{kg}$. Assuming the average friction coefficient $b = 100$, design a speed control system such that the vehicle can reach $100\ \text{km/h}$ from $0\ \text{km/h}$ in $8\ \text{s}$ with an overshoot less than 5%.

Assuming the sampling period $T = 0.6$ seconds, design a digital PI controller that achieves the above specifications.

Let's first try to get the discrete-time T.F. from the continuous time T.F. under ZOH. You already learned this in Part I.

$$G(z) = (1 - z^{-1}) Z[\frac{G(s)}{s}]$$

From $$G(s) = \frac{1}{ms + b} = \frac{1}{1000s + 100}$$

$$G(z) = (1 - z^{-1}) Z[\frac{G(s)}{s}] = (1 - z^{-1}) Z[\frac{0.001}{s(s + 0.1)}]$$

$$= \frac{z - 1}{z} Z[\frac{0.01}{s} - \frac{0.01}{s + 0.1}]$$

From the z-transform table

$$Z[\frac{1}{s}] = \frac{z}{z - 1} \qquad Z[\frac{1}{s + 0.1}] = \frac{z}{z - e^{-0.1 \times 0.6}}$$

$$G(z) = 0.01 \frac{z - 1}{z} [\frac{z}{z - 1} - \frac{z}{z - e^{-0.1 \times 0.6}}] = \frac{0.00058}{z - 0.942}$$

Discretized plant with T=0.6 s

This is a time-consuming procedure! There are short-cuts to this.

In MATLAB, there are commands to convert the TF from continuous-time to discrete-time:

MATLAB command
tf(num,den)
will generate a transfer function of a continuous-time system
defined by the numerator (num) and denominator (den).

sys=tf([1], [1000 100]) ➡ $G(s) = \frac{1}{1000s + 100}$

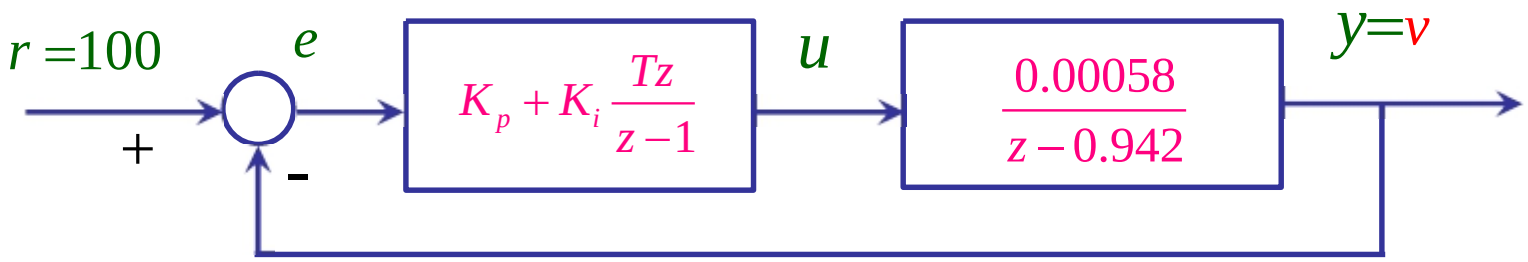
MATLAB command
c2d(sys, T)
will convert a continuous-time T.F. to discrete-time T.F. using
zero-order-hold with sampling time T.

sysd=c2d(sys, 0.6) ➡ $G(z) = \frac{0.00058}{z - 0.942}$

There is no MATLAB available in your final exam!

But you can still refer to a table 2.1 to save time!

From the following block-diagram



What is the closed-loop transfer function from r to y?

$$H(z) = \frac{G(z)C(z)}{1+G(z)C(z)} = \frac{\frac{0.00058}{z-0.942} \cdot \frac{(K_p + K_i T)z - K_p}{z-1}}{1 + \frac{0.00058}{z-0.942} \cdot \frac{(K_p + K_i T)z - K_p}{z-1}}$$

$$= \frac{0.00058(K_p + 0.6K_i)z - 0.00058K_p}{z^2 + [0.00058(K_p + 0.6K_i) - 1.942]z + (0.942 - 0.00058K_p)}$$

From the design in example 3.2, we obtained $\zeta = 0.7, \omega_n = 0.82$

$$H_d(s) = \frac{0.67}{s^2 + 1.15s + 0.67}$$

We need to get the discrete-time TF under ZOH $H_d(z)$

Do you know how to do it?

We can of course use the formula,

$$H_d(z) = (1 - z^{-1}) Z[\frac{H_d(s)}{s}]$$

But the short-cuts are using MATLAB or table 2.1 to get

$$H_d(z) = \frac{0.09521\ z + 0.07557}{z^2 - 1.331z + 0.5016}$$

Comparing the denominator of the actual closed-loop transfer function

$$H(z)=\frac{0.00058(K_p+0.6K_i)z-0.00058K_p}{z^2+[0.00058(K_p+0.6K_i)-1.942]z+(0.942-0.00058K_p)}$$

with that of the desired one

$$H_d(z)=\frac{0.09521\,z+0.07557}{z^2-1.331z+0.5016}$$

we obtain

$$\begin{cases}0.00058(K_p+0.6K_i)-1.942=-1.331\\0.942-0.00058K_p=0.5016\end{cases}$$

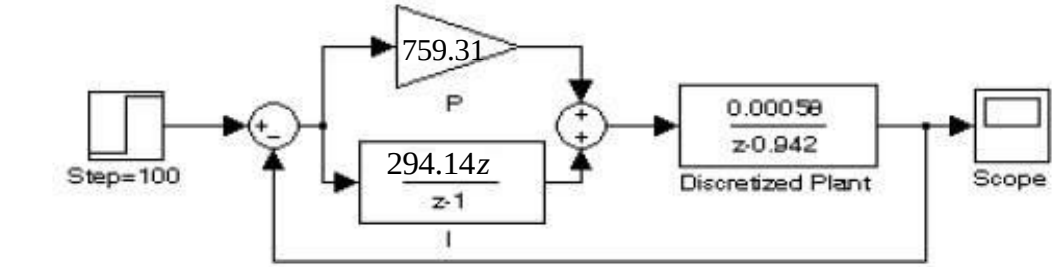
$$\Rightarrow K_p=759.31, K_i=490.23$$

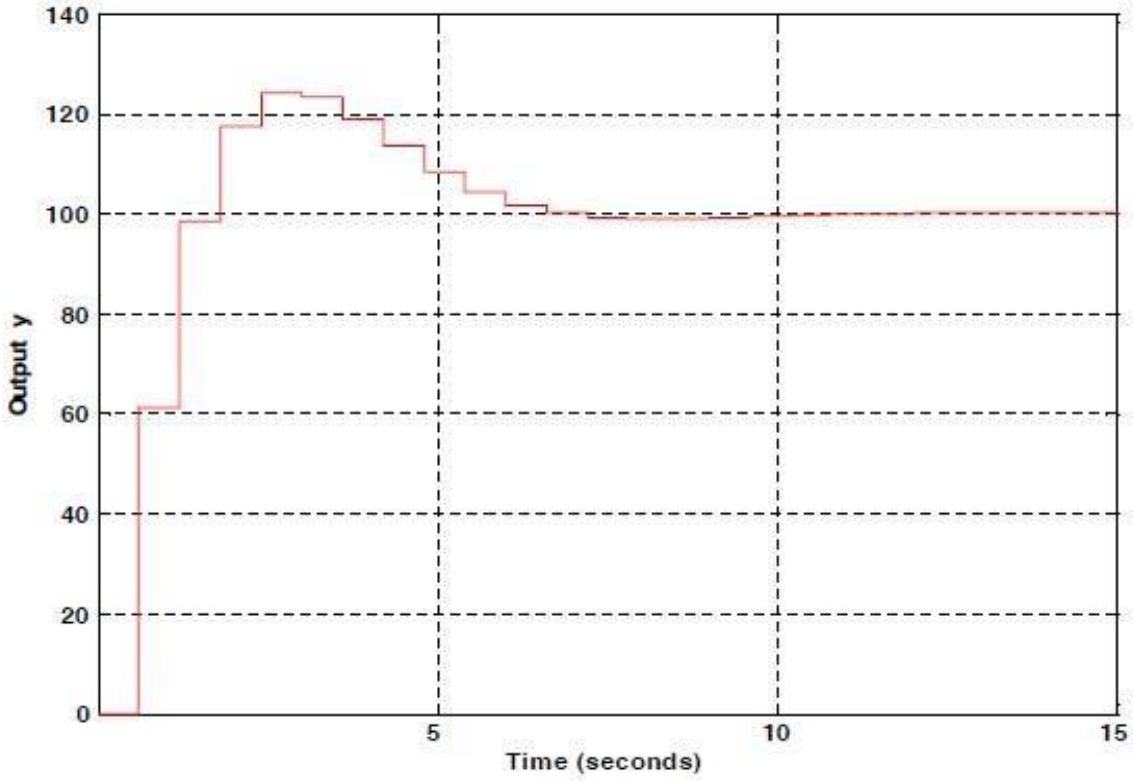
and the digital controller is

$$C(z)=759.31+\frac{294.14z}{z-1}$$

Note: we cannot do much with the numerators (zeros) of these transfer functions. They do affect the overall performance.

We simulate the digital PI control system response with the discretized plant to see whether the specifications are fulfilled in the discrete-time setting:

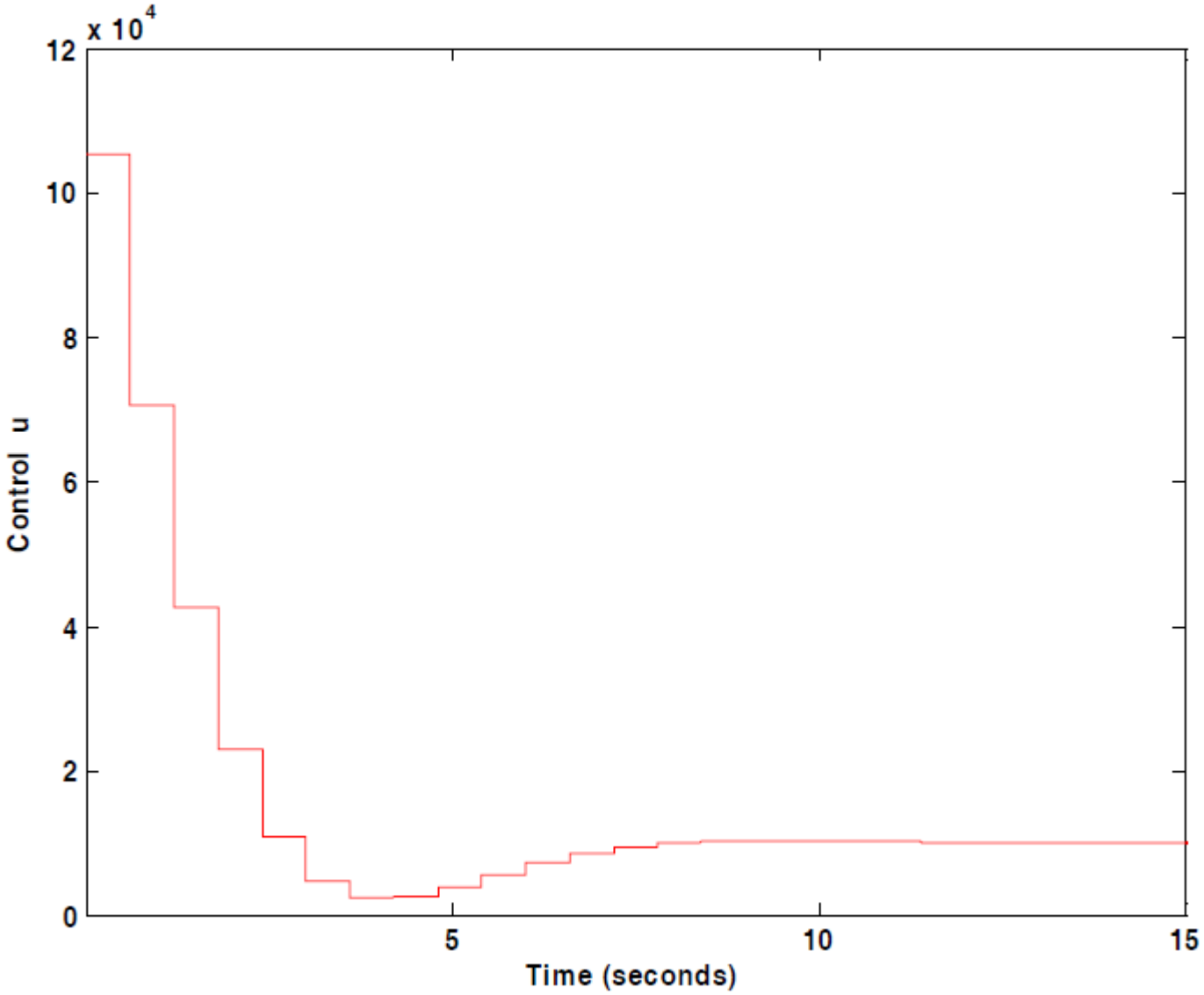




Remark: the overshoot is still larger than performance requirement (5%), but the settling time meets the specification. The system performance can be fine-tuned by re-selecting the desired pole locations in Z-plane.

Why do we still get larger overshoot?

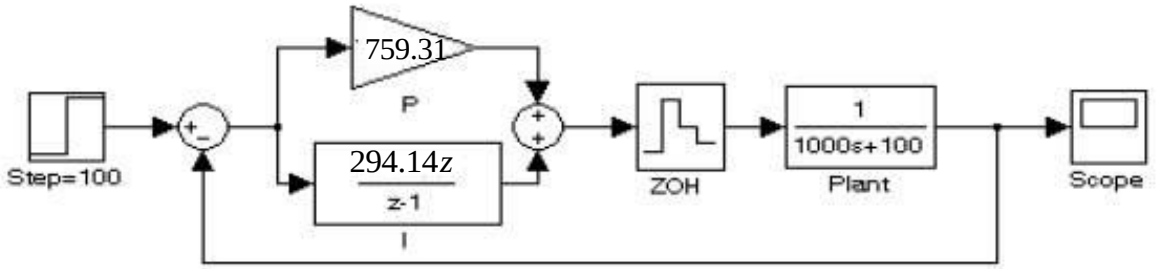
We only get the poles matched, not the zeros!

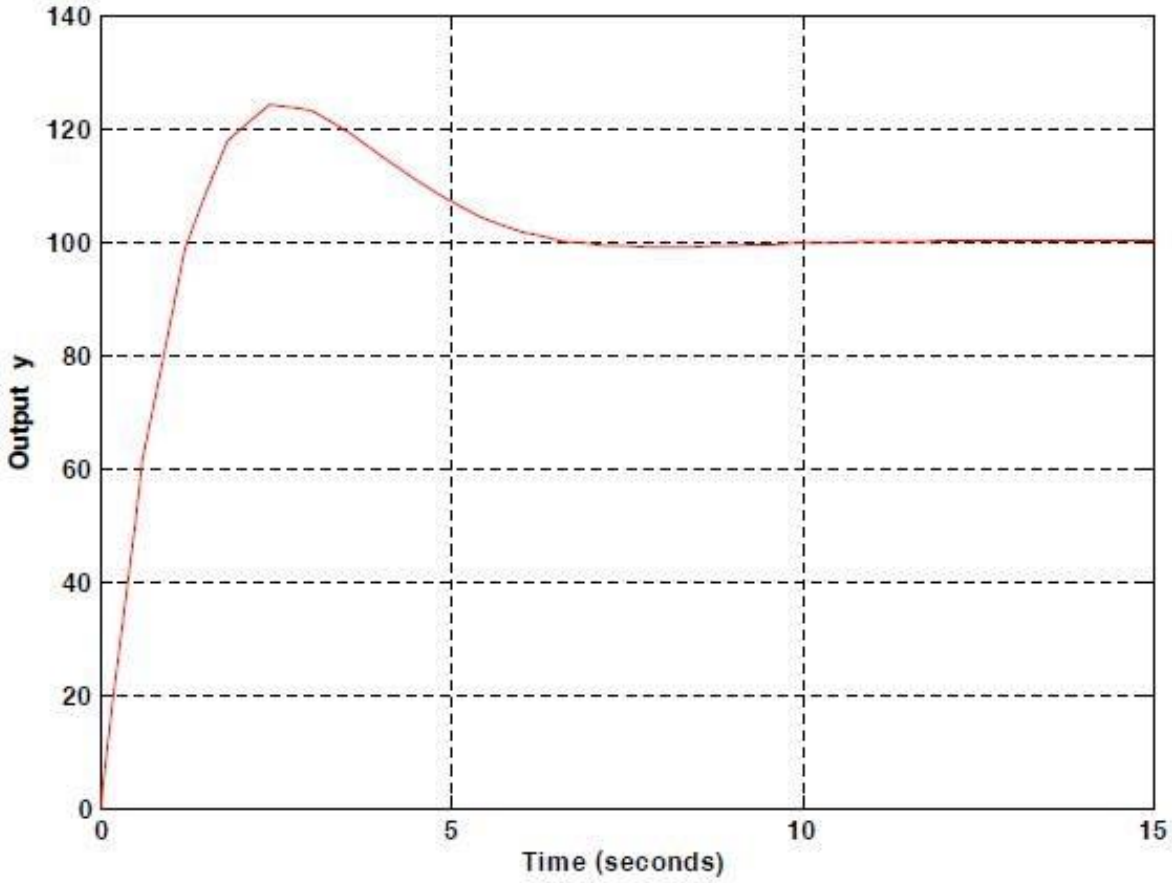


return

Verification through SIMULINK

Now simulate the digital PI control system response with actual plant



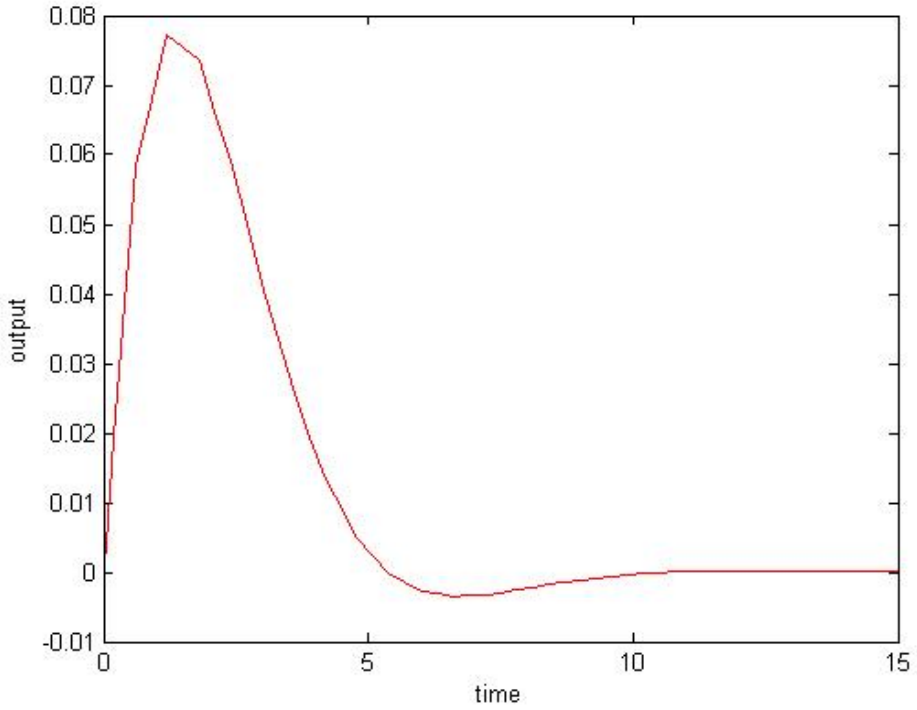


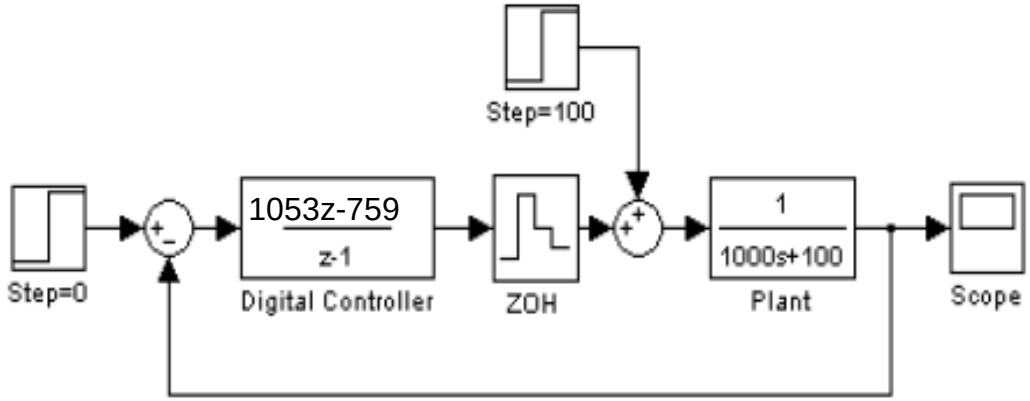
Remark: when a digital control law is implemented onto the actual continuous-time plant, the overshoot and the settling time is about the same as those obtained with the discretized systems. The response is smooth, even though the control input is updated with a sampling period of T=0.6 seconds.

Disturbance rejection

From the analysis in Chapter Two (section 2.4.6), it was shown that a step disturbance can be rejected when the controller has an integral action.

Since we are using a PI controller, step disturbance should be rejected in our design although we haven't explicitly considered such a property in our design. We verify this through simulation by injecting a step function into the plant input. To see the effect of the disturbance on the system output, we let the reference to be zero.





A disturbance with magnitude of 100 is nicely rejected from the output.

Deadbeat controller design

A unique feature of digital control is that we can design a control law such that the resulting system output is capable of following the reference input in a finite number of steps, i.e., in finite time interval, which can never be achieved in continuous-time setting. Such a control law is called deadbeat controller, which in fact places all the closed-loop system poles at the origin.

$$H_{desired}(z) = \frac{1}{z^n}$$

From $z = e^{Ts} \Rightarrow 0 = e^{Ts}$ What is the corresponding pole in s-domain?

$$s = -\infty$$

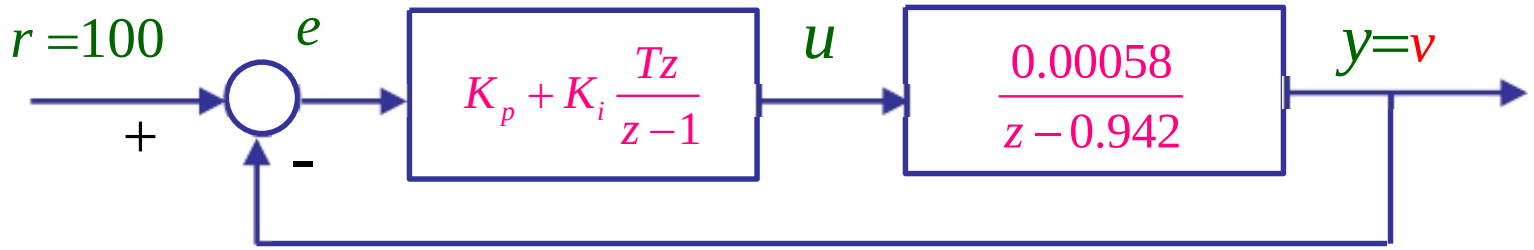
Is it possible to achieve deadbeat using analog controller?

So deadbeat cannot be done by analog controller!

But there is no free lunch!

Although the deadbeat control can guarantee the system output to reach the target reference in n steps, the resulting overshoot could be huge and the control input could be very high (which is equivalent to that the energy required to achieve such a performance is large). As such, deadbeat control is generally impractical and rarely used in practical situations.

We now design a deadbeat PI controller for the speed control system. Recall



The resulting closed-loop transfer function from r to y is given by

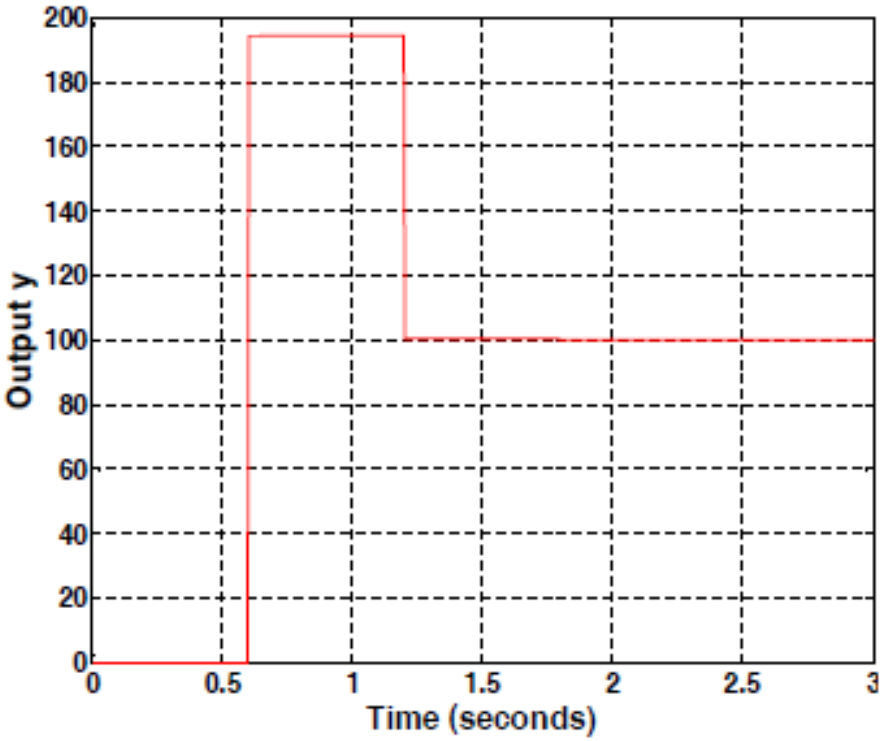
$$H(z) = \frac{0.00058(K_p + 0.6K_i)z - 0.00058K_p}{z^2 + [0.00058(K_p + 0.6K_i) - 1.942]z + (0.942 - 0.00058K_p)}$$

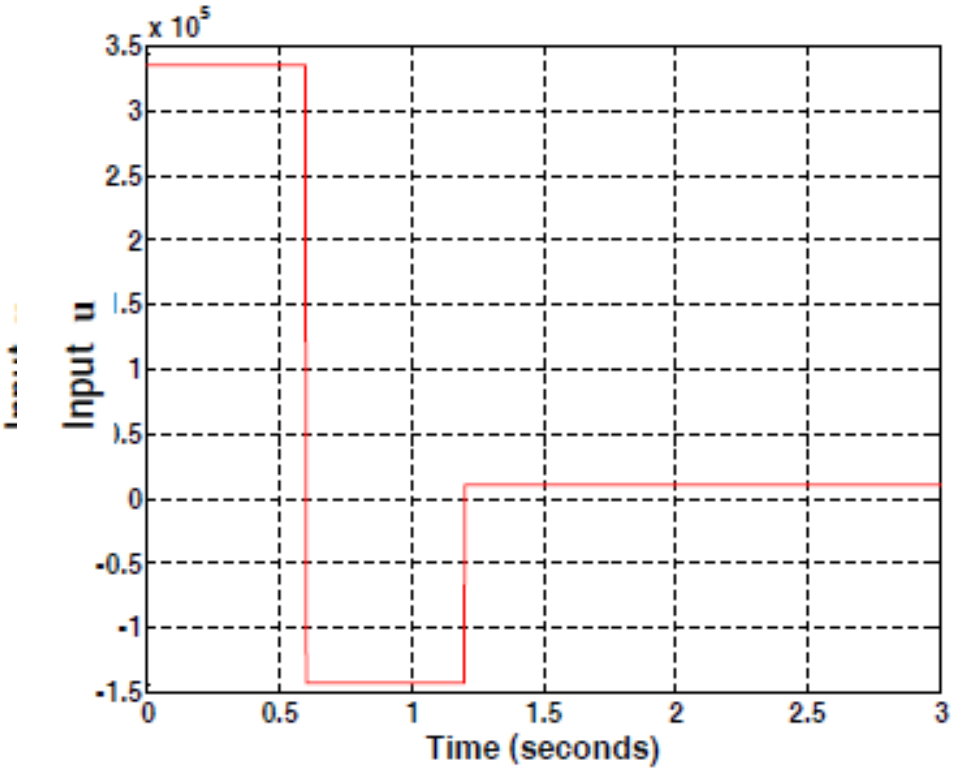
Comparing with the desired deadbeat

$$H_{desired}(z) = \frac{1}{z^2} \quad \Rightarrow \quad K_p = 1624.14, K_i = 2873.56$$

$$\Rightarrow \quad C(z) = K_p + K_i \frac{Tz}{z-1} = \frac{3348.28z - 1624.14}{z-1}$$

Simulation Results





The system output settles down to the target reference in 2 samples only, i.e., $2 \times 0.6 = 1.2$ seconds (very fast).

What is the overshoot? What is the maximal control force?

But, it has about 100% overshoot and a huge control input.

$\Leftrightarrow$ Comparison with **example 3.3**

Such a controller is very hard to be implemented in real life!
However, there could be applications (such as military applications) that deadbeat control might be desirable.

From the design examples, let's summarize the digital controller design procedure

Direct Design Procedure

Step 1. Build the reference model → the perfect model which satisfies all the performance requirement.

How to do it?

From time domain performance requirements, determine the desired damping ratio, ζ , and natural frequency, ω_n and hence the desired closed-loop TF, the reference model, $H_d(s)$.

Do not forget to convert the continuous time $H_d(s)$ to discrete time $H_d(z)$

Step 2. Get the discrete-time TF G(z) from the continuous time G(s) under ZOH.

Step 3. Choose an appropriate type of controller, e.g. PID or others.
Start with simple controller first.

Step 4. Determine the control parameters and hence *C*(*z*).

How to do it?

Match the closed loop TF, $H_{cl}(z)$ with the reference model $H_d(z)$

Comparing denominator polynomials of reference model and actual CLTF, and match the coefficients!

Can we match both poles and zeros at the same time?

Not really. Sometimes we do, sometimes we do not.
The current design does not have enough freedom to match both.
We will give you a good answer to this at the end of this course. Stay tuned!

In the previous designs, we assume that the plant model G(s) is available.

Is it possible to design the controller without the mathematical model?

The answer is YES. The best part about PID controller is that it can be model-free!

How do we design the controller parameters without the model?

Can we still try to design the controller C(z) by matching the poles?

No, we have no idea about where the poles are as the models are not given!

PID Auto-Tuning

Why auto-tuning

PID performance is determined by the PID coefficients (gains). There are three ways to tune these coefficients.

Manually tuned by experience acquired through trial and error.

time-consuming, tedious, risky, no guaranteed performance

Auto-tuned by machine algorithms derived by researchers through analysis or experience.

fast, automatic, reasonable performance 

Practical tuning: first auto-tuned by machine algorithms, then fine-tuned by engineers or experienced operators.

efficient, good performance

Dozens of PID auto-tuning methods have been proposed in the past seventy years. They can be classified as classical vs advanced, deterministic vs stochastic, time-domain vs frequency domain, model-based vs model-free.

Here we will introduce two classical methods proposed by Ziegler-Nichols in 1942: transient response method and stability limit method.
The PID gains can be determined by experiment data without any knowledge of the plant model.

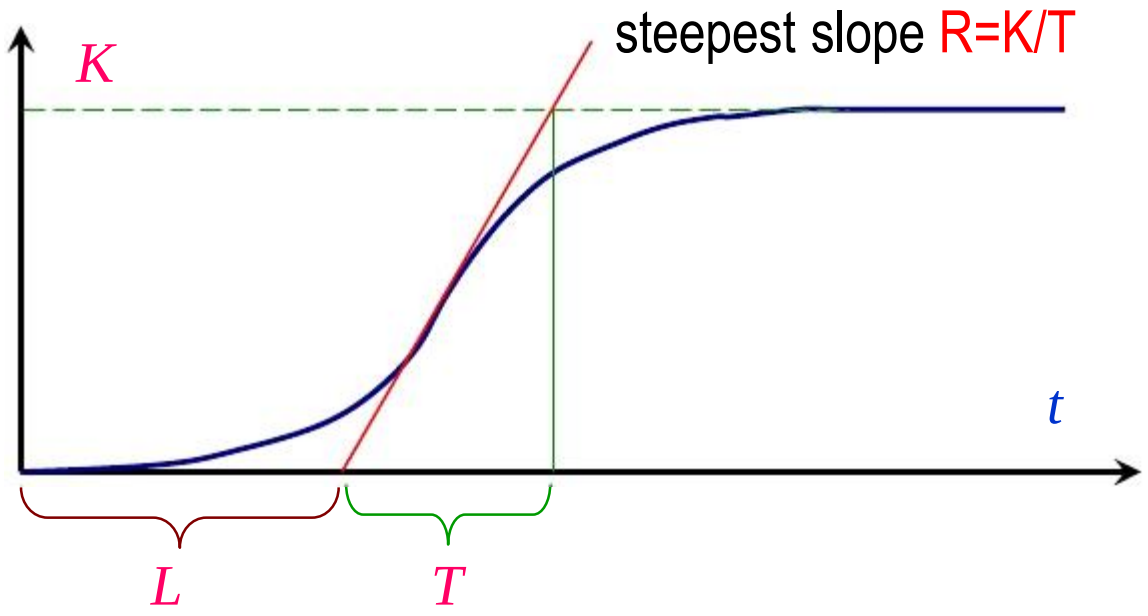
ZN auto-tuning methods aim at achieving an average response within 20% to 60% overshoot, that is a damping ratio of 0.2 or above.

1st ZN method – transient response method

In this method, a step response is conducted for the open-loop system.

If the open loop step-response is similar to the step response of first order system shown below, then we can use this method.

Three parameters are obtained from the step response: dead-time L, DC gain K, , time constant T and the steepest slope, R=K/T.



Tuning rules

	K_p	K_i	K_d
P	$\frac{1}{RL}$		
PI	$\frac{0.9}{RL}$	$\frac{0.3}{RL^2}$	
PID	$\frac{1.2}{RL}$	$\frac{0.6}{RL^2}$	$\frac{0.6}{R}$

Those heuristic rules were obtained through intensive simulations on many different plant models.

It is applicable only to processes which are TYPE 0 and open-loop stable

Why?

The second and higher order system is far more common than first order system. Normally there are oscillations in the step response. How do we tune the gains?

Classical auto-tuning – second ZN method

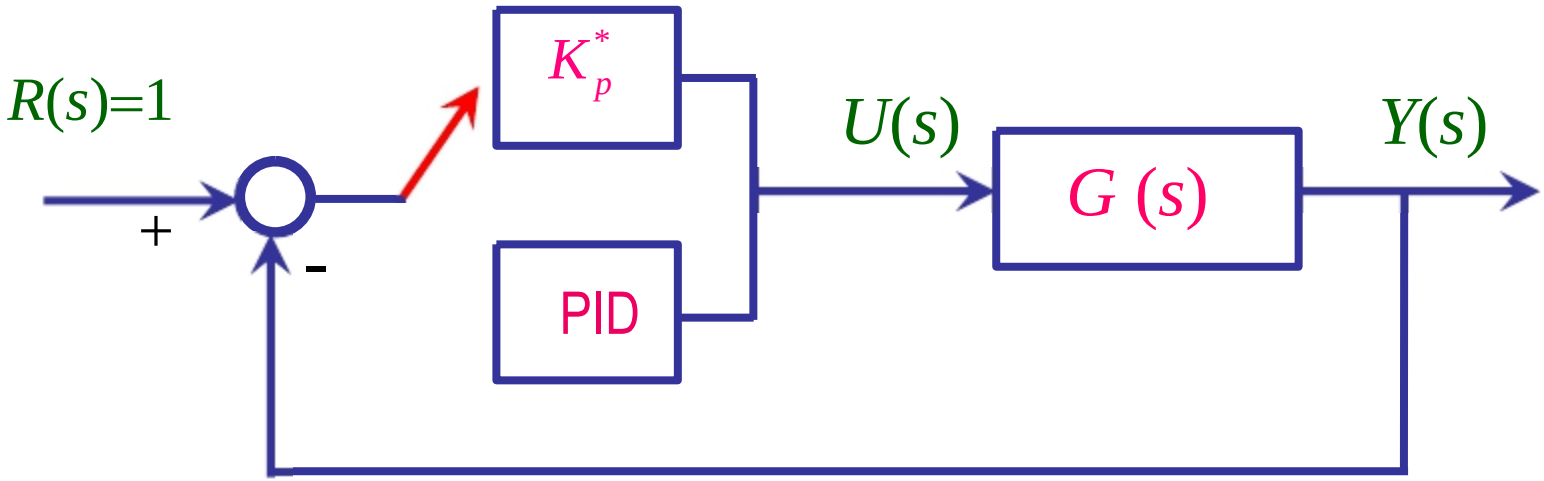
2nd ZN method – stability limit method

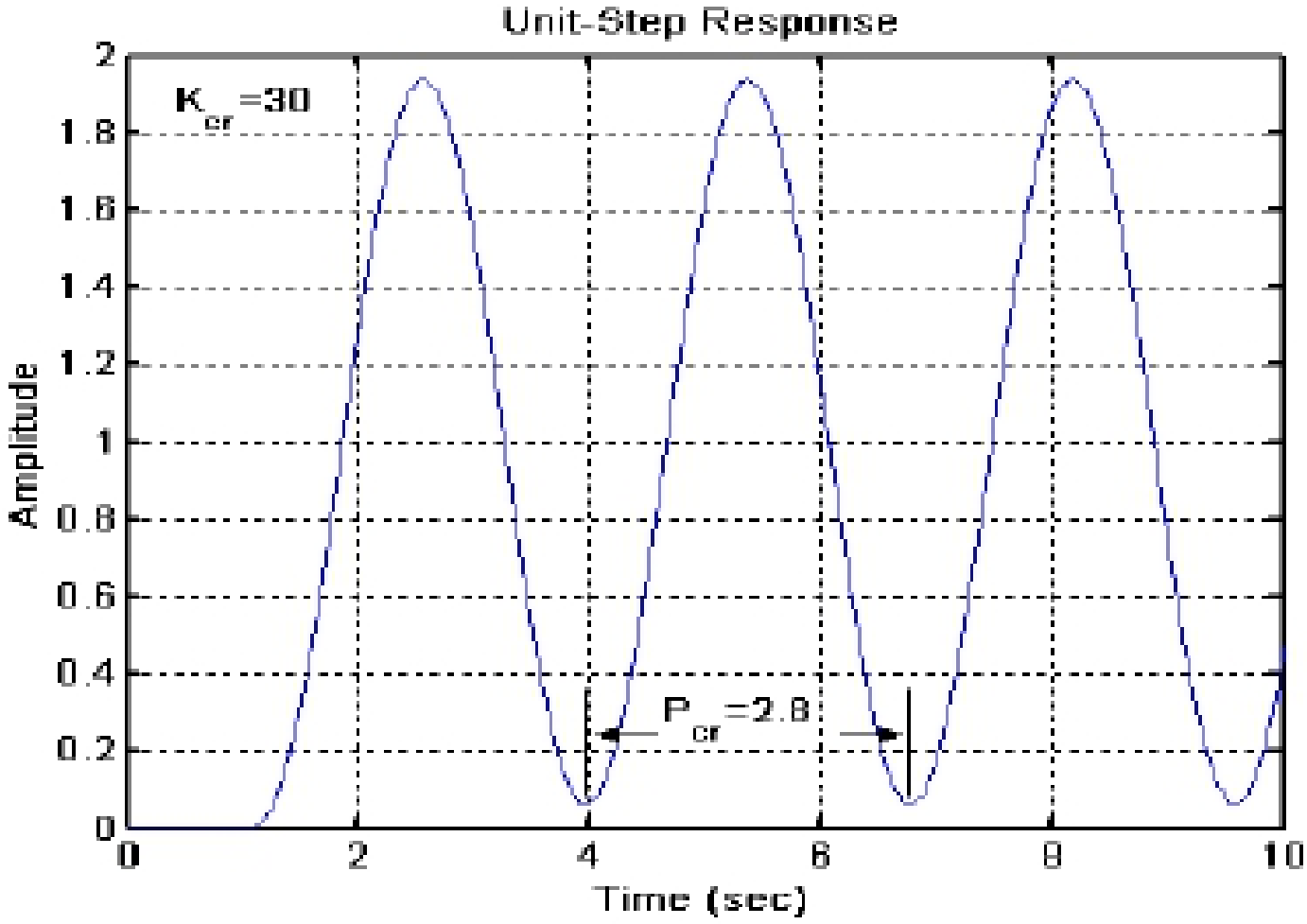
In this method, a proportional controller is first used to form a negative feedback loop. The proportional gain K_p is slowly increasing until continuous (sustained) oscillations occur. At that point, we define two parameters:

the critical gain (ultimate gain) $K_u = K_p^*$,

and the critical period (ultimate period) $P_u =$ oscillation period.

Then the P, PI and PID control gains can be determined.




Unit-Step Response
K_{cr}=30
P_{cr}=2.8
Amplitude
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
2
4
6
8
10

Tuning rules

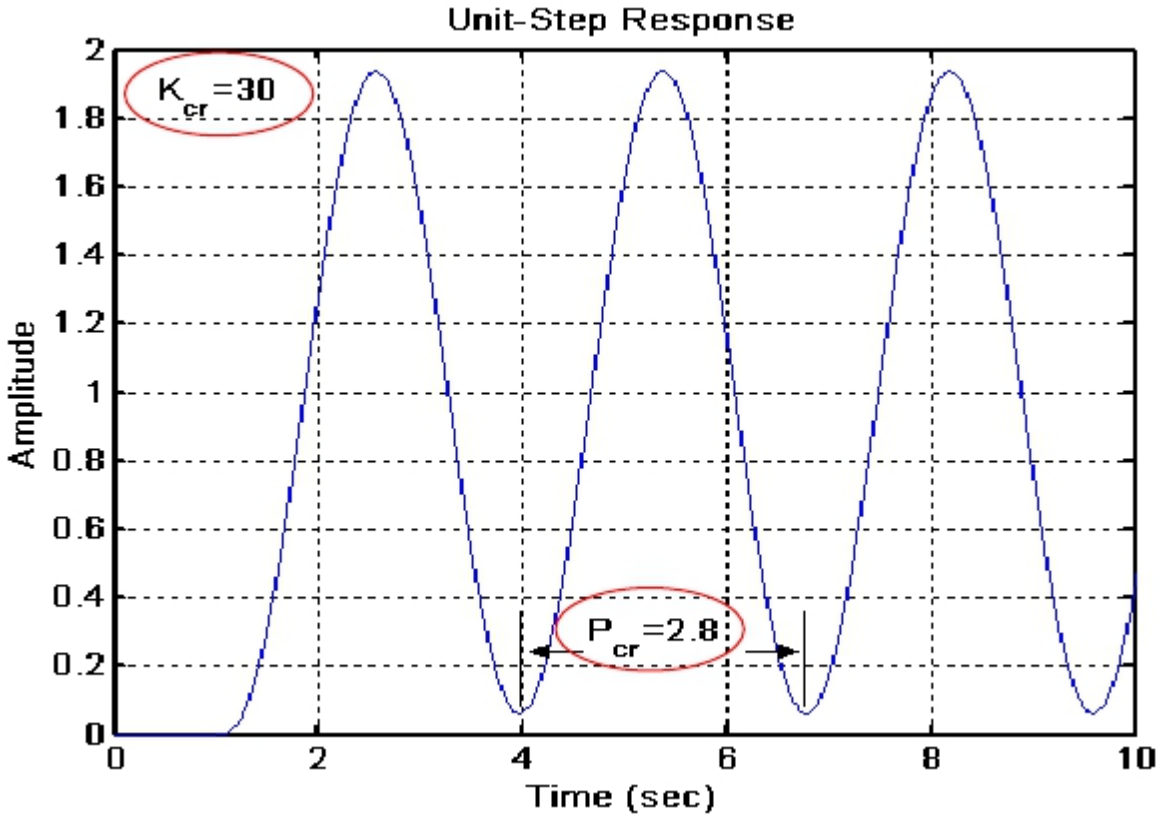
	K_p	K_i	K_d
P	$0.5K_u$		
PI	$0.45K_u$	$\frac{0.54K_u}{P_u}$	
PID	$0.6K_u$	$\frac{1.2K_u}{P_u}$	$0.075K_uP_u$

Those heuristic rules were obtained through intensive simulations on many plant models.

It pushes the process to the stability limit (at least a pair of poles on the imaginary axis). Such test may not be acceptable for a number of industrial processes such as power plant, aircraft, etc.

Design Example

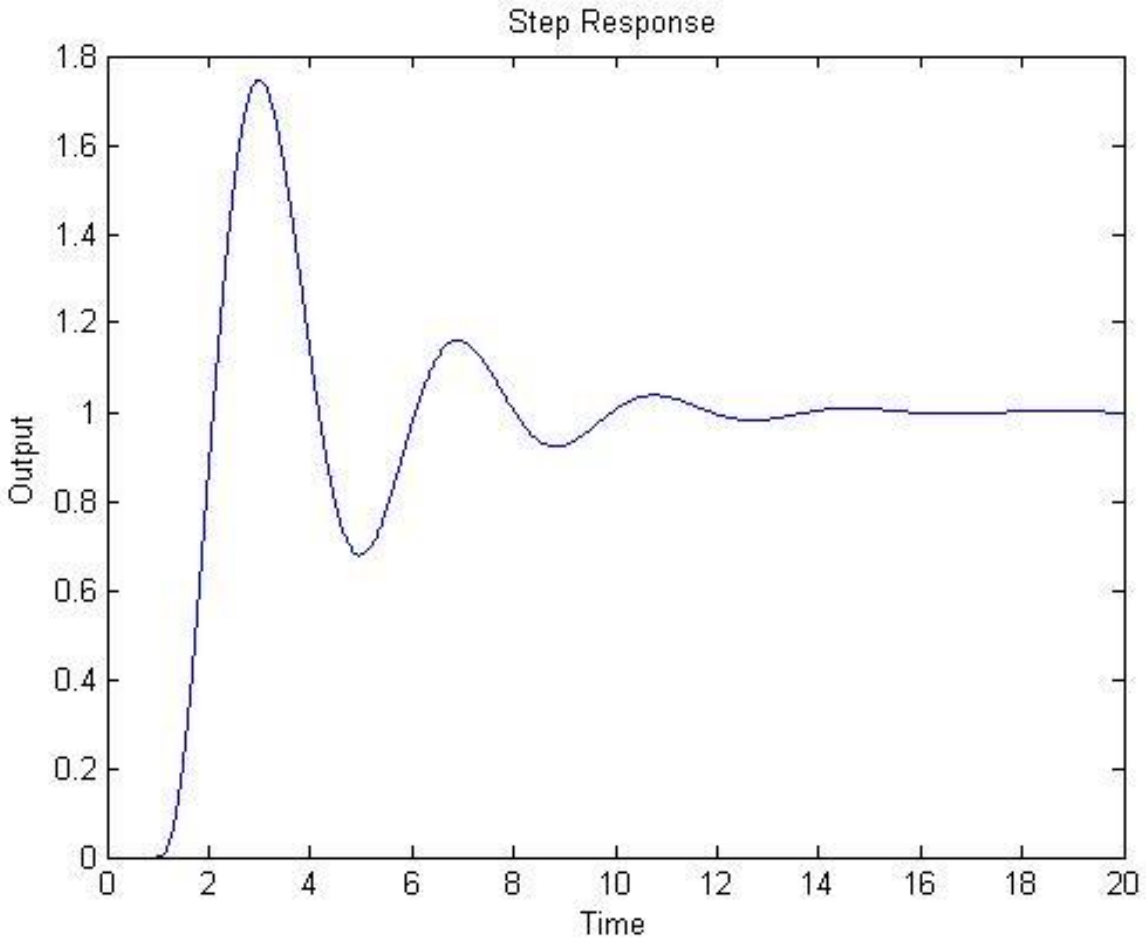
Consider a process $G(s)=\dfrac{1}{s(s+1)(s+5)}$, which has an integrator, thus only the second ZN method is applicable.



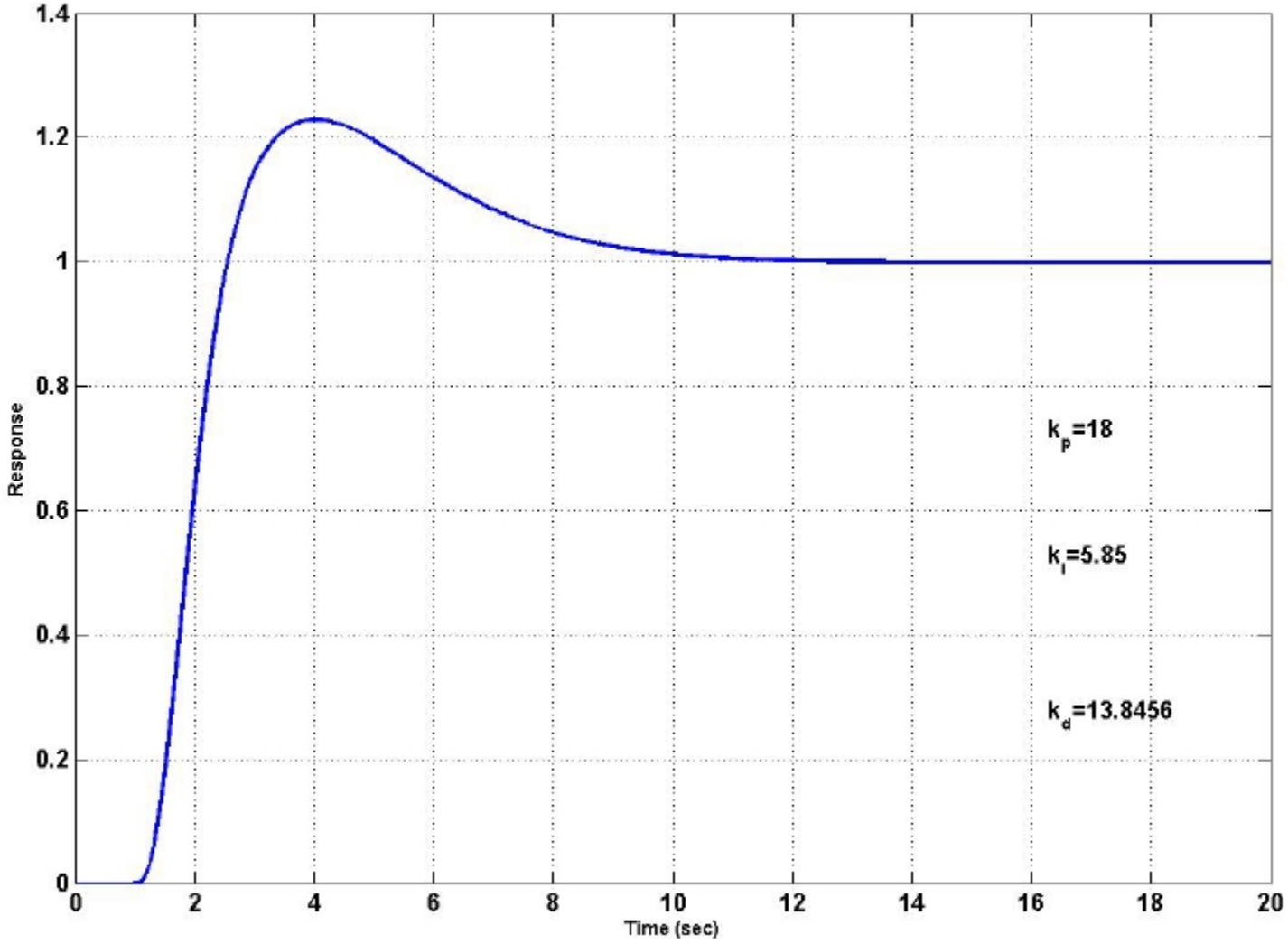
$$K_p = 0.6K_u = 18$$

$$K_i = \frac{1.2K_u}{P_u} = 12.8$$

$$K_d = 0.075K_uP_u = 6.32$$



A fine-tuned PID controller based on ZN tuning results



Notes:

The overshoot is a bit large, whereas the rise time is short. ZN methods tend to produce a fast response.

ZN methods provide primary tuning results and the starting point for further fine tuning.

Many refined ZN auto-tuning methods were developed in the past 70 years, highlighting more on overshoot and settling time.

If the model is known, then there are many model-based design techniques which are more powerful than the model free ones!

Q & A…

Entry #	Laplace Domain	Time Domain	Z Domain (t=kT)
1	1	$\delta(t)$ unit impulse	1
2	$\frac{1}{s}$	$u(t)$ unit step	$\frac{z}{z-1}$
3	$\frac{1}{s^2}$	t	$\frac{Tz}{(z-1)^2}$
4	$\frac{1}{s+a}$	e^{-at}	$\frac{z}{z-e^{-aT}}$
5		$b^k \quad (b=e^{-aT})$	$\frac{z}{z-b}$
6	$\frac{1}{(s+a)^2}$	te^{-at}	$\frac{Tze^{-aT}}{(z-e^{-aT})^2}$
7	$\frac{1}{s(s+a)}$	$\frac{1}{a}(1-e^{-at})$	$\frac{z(1-e^{-aT})}{a(z-1)(z-e^{-aT})}$
8	$\frac{b-a}{(s+a)(s+b)}$	$e^{-at}-e^{-bt}$	$\frac{z(e^{-aT}-e^{-bT})}{(z-e^{-aT})(z-e^{-bT})}$
9	$\frac{1}{s(s+a)(s+b)}$	$\frac{1}{ab}-\frac{e^{-at}}{a(b-a)}-\frac{e^{-bt}}{b(a-b)}$	
10	$\frac{1}{s(s+a)^2}$	$\frac{1}{a^2}(1-e^{-at}-ate^{-at})$	
11	$\frac{s}{(s+a)^2}$	$(1-at)e^{-at}$	
12	$\frac{b}{s^2+b^2}$	$\sin(bt)$	$\frac{z\sin(bT)}{z^2-2z\cos(bT)+1}$
13	$\frac{s}{s^2+b^2}$	$\cos(bt)$	$\frac{z(z-\cos(bT))}{z^2-2z\cos(bT)+1}$
14	$\frac{b}{(s+a)^2+b^2}$	$e^{-at}\sin(bt)$	$\frac{ze^{-aT}\sin(bT)}{z^2-2ze^{-aT}\cos(bT)+e^{-2aT}}$
15	$\frac{s+a}{(s+a)^2+b^2}$	$e^{-at}\cos(bt)$	$\frac{z^2-ze^{-aT}\cos(bT)}{z^2-2ze^{-aT}\cos(bT)+e^{-2aT}}$

return

$$H(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

$G(s)$	$H(q)$ or the coefficients in $H(q)$
$\frac{1}{s}$	$\frac{h}{q-1}$
$\frac{1}{s^2}$	$\frac{h^2(q+1)}{2(q-1)^2}$
$\frac{1}{s^m}$	$\frac{q-1}{q} \lim_{a \to 0} \frac{(-1)^m}{m!} \frac{\partial^m}{\partial a^m} \left(\frac{q}{q - e^{-ah}} \right)$
e^{-sh}	q^{-1}
$\frac{a}{s+a}$	$\frac{1 - \exp(-ah)}{q - \exp(-ah)}$
$\frac{a}{s(s+a)}$	$b_1 = \frac{1}{a}(ah - 1 + e^{-ah})$ $\quad b_2 = \frac{1}{a}(1 - e^{-ah} - ahe^{-ah})$ $a_1 = -(1 + e^{-ah})$ $\quad a_2 = e^{-ah}$
$\frac{a^2}{(s+a)^2}$	$b_1 = 1 - e^{-ah}(1 + ah)$ $\quad b_2 = e^{-ah}(e^{-ah} + ah - 1)$ $a_1 = -2e^{-ah}$ $\quad a_2 = e^{-2ah}$
$\frac{s}{(s+a)^2}$	$\frac{(q-1)he^{-ah}}{(q - e^{-ah})^2}$
$\frac{ab}{(s+a)(s+b)}$ $a \neq b$	$b_1 = \frac{b(1 - e^{-ah}) - a(1 - e^{-bh})}{b - a}$ $b_2 = \frac{a(1 - e^{-bh})e^{-ah} - b(1 - e^{-ah})e^{-bh}}{b - a}$ $a_1 = -(e^{-ah} + e^{-bh})$ $a_2 = e^{-(a+b)h}$

Return

$$H(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

return

$G(s)$	$H(q)$ or the coefficients in $H(q)$
$\frac{(s+c)}{(s+a)(s+b)}$ $a \neq b$	$b_1 = \frac{e^{-bh} - e^{-ah} + (1-e^{-bh})c/b - (1-e^{-ah})c/a}{a-b}$ $b_2 = \frac{c}{ab} e^{-(a+b)h} + \frac{b-c}{b(a-b)} e^{-ah} + \frac{c-a}{a(a-b)} e^{-bh}$ $a_1 = -e^{-ah} - e^{-bh} \qquad a_2 = e^{-(a+b)h}$
$\frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$	$b_1 = 1 - \alpha\left(\beta + \frac{\zeta\omega_0}{\omega}\gamma\right) \qquad \omega = \omega_0\sqrt{1-\zeta^2} \qquad \zeta < 1$ $b_2 = \alpha^2 + \alpha\left(\frac{\zeta\omega_0}{\omega}\gamma - \beta\right) \qquad \alpha = e^{-\zeta\omega_0 h}$ $a_1 = -2\alpha\beta \qquad \beta = \cos(\omega h)$ $a_2 = \alpha^2 \qquad \gamma = \sin(\omega h)$
$\frac{s}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$	$b_1 = \frac{1}{\omega} e^{-\zeta\omega_0 h}\sin(\omega h) \qquad b_2 = -b_1$ $a_1 = -2e^{-\zeta\omega_0 h}\cos(\omega h) \qquad a_2 = e^{-2\zeta\omega_0 h}$ $\omega = \omega_0\sqrt{1-\zeta^2}$
$\frac{a^2}{s^2 + a^2}$	$b_1 = 1 - \cos ah \qquad b_2 = 1 - \cos ah$ $a_1 = -2\cos ah \qquad a_2 = 1$
$\frac{s}{s^2 + a^2}$	$b_1 = \frac{1}{a}\sin ah \qquad b_2 = -\frac{1}{a}\sin ah$ $a_1 = -2\cos ah \qquad a_2 = 1$
$\frac{a}{s^2(s+a)}$	$b_1 = \frac{1-\alpha}{a^2} + h\left(\frac{h}{2} - \frac{1}{a}\right) \qquad \alpha = e^{-ah}$ $b_2 = (1-\alpha)\left(\frac{h^2}{2} - \frac{2}{a^2}\right) + \frac{h}{a}(1+\alpha)$ $b_3 = -\left[\frac{1}{a^2}(\alpha - 1) + \alpha h\left(\frac{h}{2} + \frac{1}{a}\right)\right]$ $a_1 = -(\alpha + 2) \qquad a_2 = 2\alpha + 1 \qquad a_3 = -\alpha$