Feedback Control Theory

a Computer System's Perspective

- Introduction
  - What is feedback control?
  - Why do computer systems need feedback control?
- Control design methodology
  - System modeling
  - Performance specs/metrics
  - Controller design
- Summary

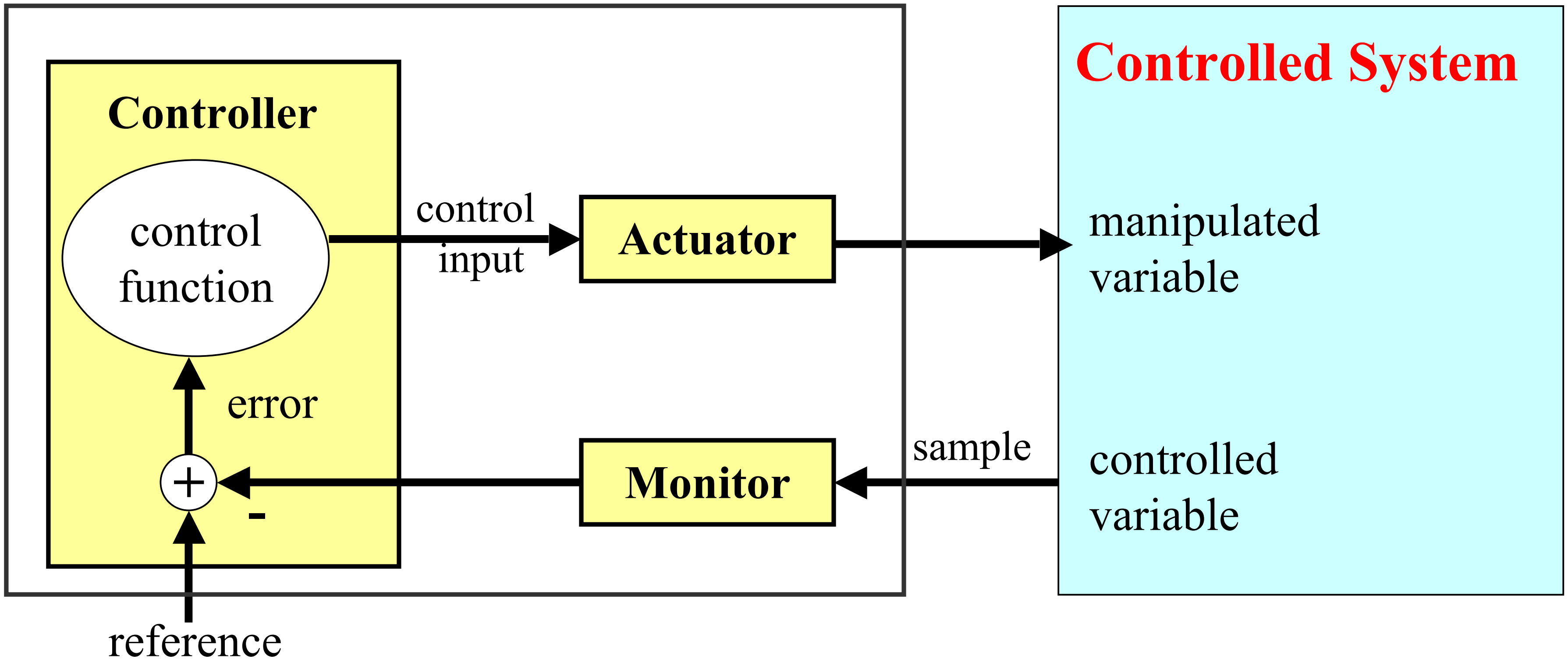
Control

- Applying input to cause system variables to conform to desired values called the reference.
  - Cruise-control car: f_engine(t)=? → speed=60 mph
  - E-commerce server: Resource allocation? → T_response=5 sec
  - Embedded networks: Flow rate? → Delay = 1 sec
  - Computer systems: QoS guarantees

Open-loop control

- Compute control input without continuous variable measurement
  - Simple
  - Need to know **EVERYTHING ACCURATELY** to work right
    - Cruise-control car: friction(t), ramp_angle(t)
    - E-commerce server: Workload (request arrival rate? resource consumption?); system (service time? failures?)
- Open-loop control fails when
  - We don’t know everything
  - We make errors in estimation/modeling
  - Things change

Feedback (close-loop) Control



Feedback (close-loop) Control

- Measure variables and use it to compute control input
  - More complicated (so we need control theory)
  - Continuously measure & correct
    - Cruise-control car: measure speed & change engine force
    - Ecommerce server: measure response time & admission control
    - Embedded network: measure collision & change backoff window
- Feedback control theory makes it possible to control well even if
  - We don’t know everything
  - We make errors in estimation/modeling
  - Things change

Why feedback control?
Open, unpredictable environments

- Deeply embedded networks: interaction with physical environments
  - Number of working nodes
  - Number of interesting events
  - Number of hops
  - Connectivity
  - Available bandwidth
  - Congested area
- Internet: E-business, on-line stock broker
- Unpredictable off-the-shelf hardware

Why feedback control?
We want QoS guarantees

- Deeply embedded networks
  - Update intruder position every 30 sec
  - Report fire <= 1 min
- E-business server
  - Purchase completion time <= 5 sec
  - Throughput >= 1000 transaction/sec
- The problem: provide QoS guarantees in open, unpredictable environments

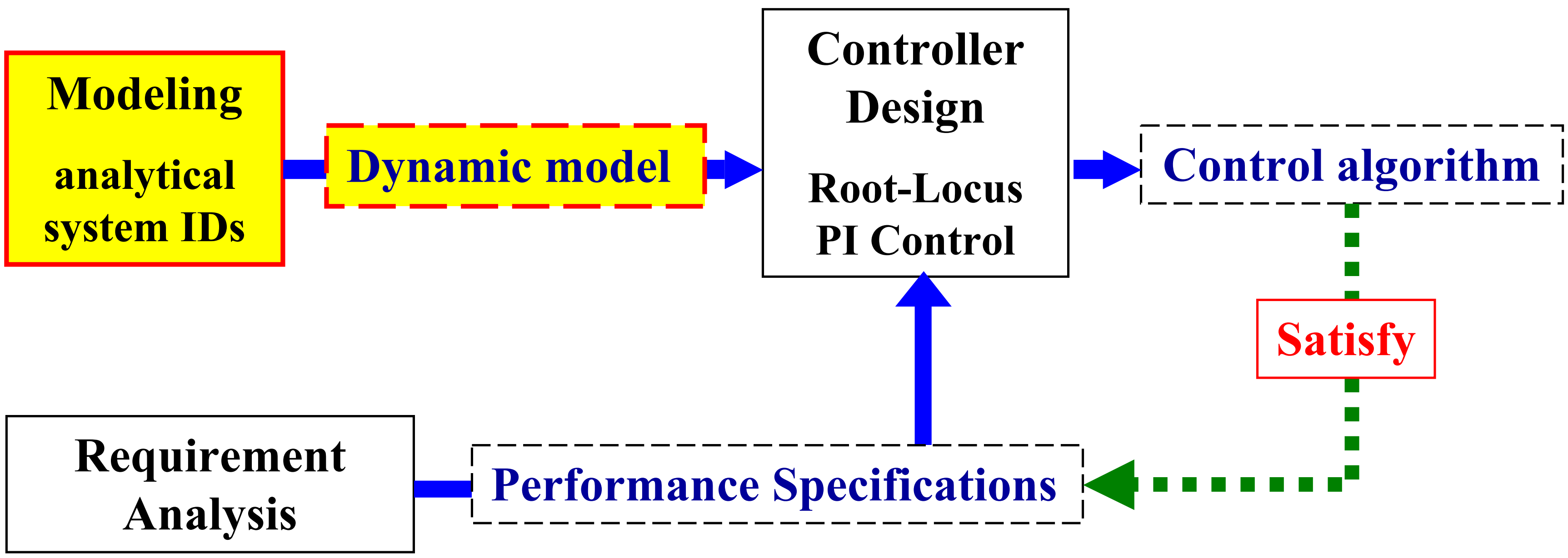
Advantage of feedback control theory

- Adaptive resource management heuristics
  - Laborious design/tuning/testing iterations
  - Not enough confidence in face of untested workload
- Queuing theory
  - Doesn’t handle feedbacks
  - Not good at characterizing transient behavior in overload
- Feedback control theory
  - Systematic theoretical approach for analysis and design
  - Predict system response and stability to input

Outline

- Introduction
  - What is feedback control?
  - Why do today's computer systems need feedback control?
- Control design methodology
  - System modeling
  - Performance specs/metrics
  - Controller design
- Summary

Control design methodology



System Models

- **Linear** vs. non-linear (differential eqns)
- **Deterministic** vs. Stochastic
- **Time-invariant** vs. Time-varying
  - Are coefficients functions of time?
- **Continuous-time** vs. Discrete-time
- System ID vs. First Principle

Dynamic Model

- Computer systems are *dynamic*
  - Current output depends on "history"
- Characterize relationships among system variables
  - Differential equations (time domain)

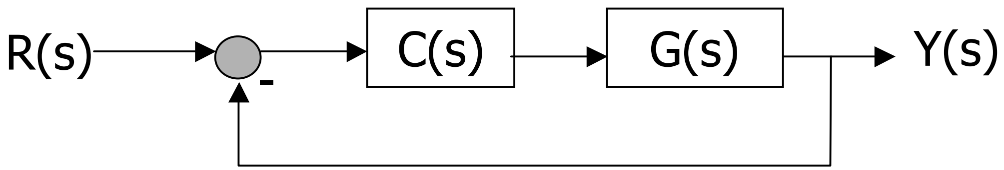
$$a_2 \ddot{y}(t) + a_1 \dot{y}(t) + a_0 y(t) = b_1 \dot{u}(t) + b_0 u(t)$$

- Transfer functions (frequency domain)

  $Y(s) = G(s)U(s)$

$$G(s) = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} = \frac{c_1}{s - p_1} + \frac{c_2}{s - p_2}$$

- Block diagram (pictorial)



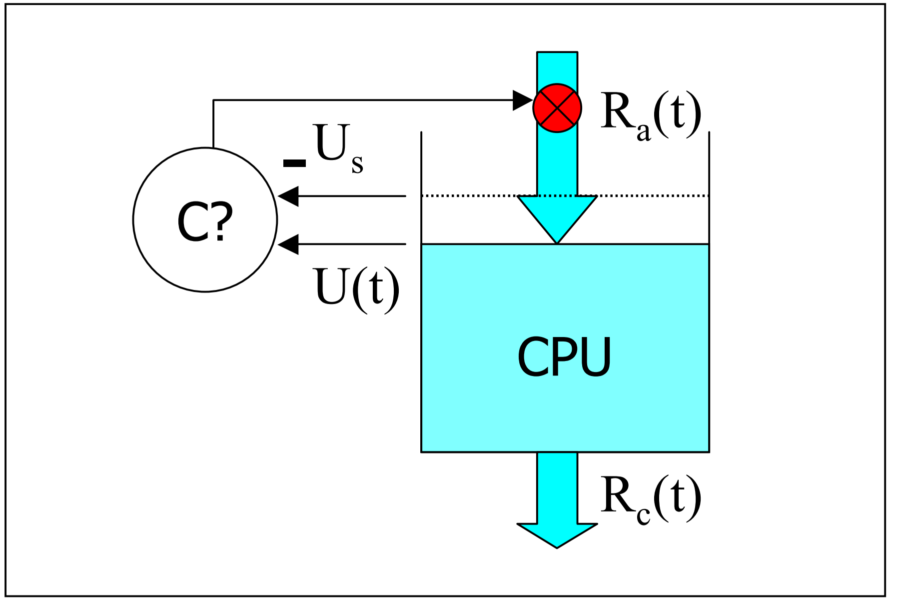
Example
Utilization control in a video server

- Periodic task T_i corresponding to each video stream i
  - c[i]: processing time, p[i]: period
  - Stream i's requested CPU utilization: u[i]=c[i]/p[i]
- Total CPU utilization: $U(t)=\Sigma_{\{k\}}u[k]$, {k} is the set of active streams
- Completion rate: $R_c(t)= (\Sigma_{\{kc\}}u[m])/\Delta t$, where {m} is the set of terminated video streams during $[t, t+\Delta t]$
  - Unknown
- Admission rate: $R_a(t)= (\Sigma_{\{ka\}}u[j])/\Delta t$, where {j} is the set of admitted streams during $[t, t+\Delta t]$
- Problem: design an admission controller to guarantee $U(t)=U_S$ regardless of $R_c(t)$

Model

Differential equation

- Error: $E(t)=U_s-U(t)$
- Model (differential equation): $$U(t) = \int_{\tau=0}^{t} (R_a(\tau) - R_c(\tau))\,d\tau$$
- Controller C? $E(t) \Rightarrow R_a(t)$



A Diversion to Math
System representations

- Three ways of system modeling

- Time domain: convolution; differential equations.

$$u(t) \longrightarrow \boxed{g(t)} \longrightarrow y(t) \qquad y(t) = g(t) * u(t) = \int_0^t g(t-\tau)u(\tau)d\tau$$

- s (frequency) domain: multiplication

$$U(s) \longrightarrow \boxed{G(s)} \longrightarrow Y(s) \qquad Y(s) = G(s)U(s)$$

- Block diagram: pictorial

s-domain is a simple & powerful “language” for control analysis

A Diversion to Math

Laplace transform

- Laplace transform of a signal f(t)

$$F(s) = L[f(t)] = \int_{0^-}^{\infty} f(t)e^{-st}dt$$

where s=σ+iω is a complex variable.

- Laplace transform is a translation from time-domain to s-domain
  - Differential equation ⇒ Polynomial function

$$a_2 \ddot{y}(t) + a_1 \dot{y}(t) + a_0 y(t) = b_1 \dot{u}(t) + b_0 u(t)$$

$$\Leftrightarrow Y(s) = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} \bullet U(s)$$

A Diversion to Math

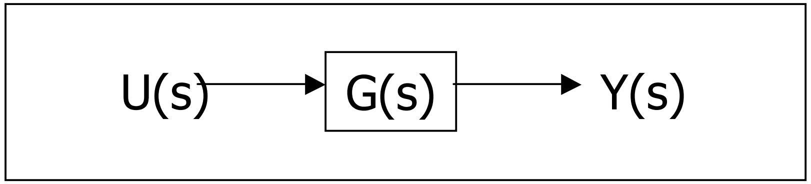
Laplace transform

- Basic translations
  - Impulse function $f(t)=\delta(t) \Leftrightarrow F(s)=1$
  - Step signal $f(t)=a \cdot 1(t) \Leftrightarrow F(s)=1/s$
  - Ramp signal $f(t)=a \cdot t \Leftrightarrow F(s)=a/s^2$
  - Exp signal $f(t)=e^{at} \Leftrightarrow F(s)=1/(s-a)$
  - Sinusoid signal $f(t)=\sin(at) \Leftrightarrow F(s)=a/(s^2+a^2)$
- Composition rules
  - Linearity $L[af(t)+bg(t)] = aL[f(t)]+bL[g(t)]$
  - Differentiation $L[df(t)/dt] = sF(s) - f(0_-)$
  - Integration $L[\int_t f(\tau)d\tau] = F(s)/s$

A Diversion to Math

Transfer function

- Modeling a linear time-invariant (LTI) system
  - $G(s) = Y(s)/U(s) \Rightarrow Y(s) = G(s)U(s)$



E.g., a second order system with *poles* p_1 and p_2

$$G(s) = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} = \frac{c_1}{s - p_1} + \frac{c_2}{s - p_2}$$

A Diversion to Math
Poles and Zeros

- The response of a linear time-invariant (LTI) system

$$F(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_0}$$

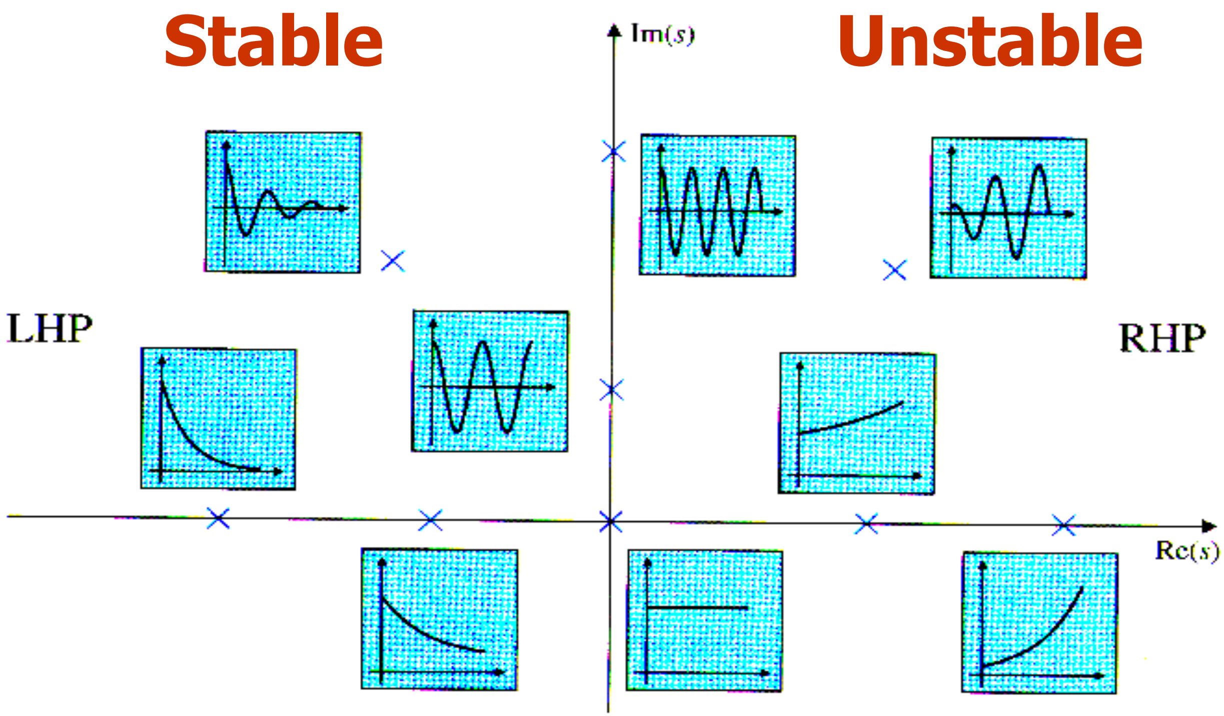
$$= K \frac{\Pi_{i=1}^{m}(s - z_i)}{\Pi_{i=1}^{n}(s - p_i)} = \frac{C_1}{s - p_1} + \frac{C_2}{s - p_2} + \ldots + \frac{C_n}{s - p_n}$$

$$\Rightarrow f(t) = \sum_{i=1}^{n} C_i e^{p_i t}$$

$\{p_i\}$ are *poles* of the function and decide the system behavior

A Diversion to Math

Time response vs. pole location

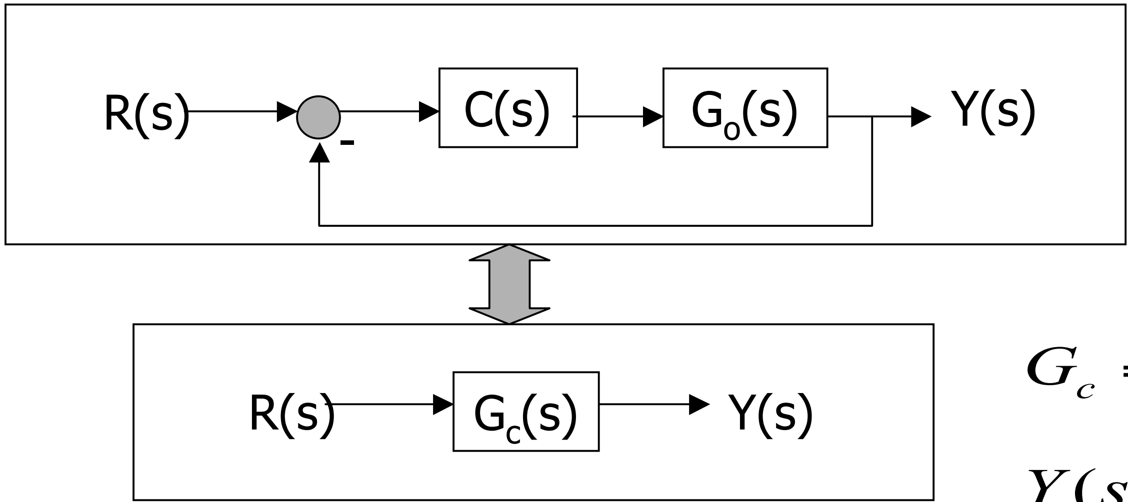


- $f'(t) = e^{pt}$, $p = a+bj$

A Diversion to Math

Block diagram

- A pictorial tool to represent a system based on transfer functions and signal flows
- Represent a feedback control system

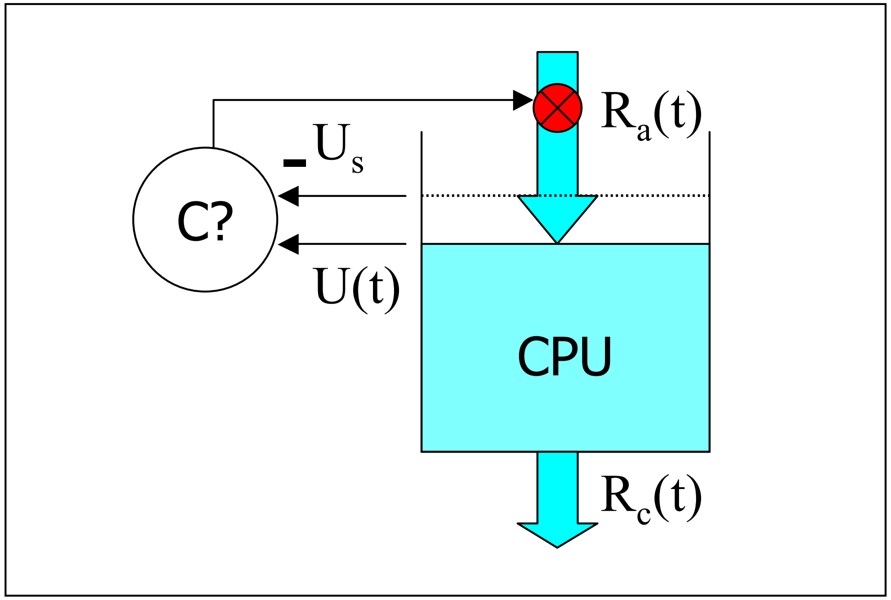


$$G_c = \frac{C(s)G_o(s)}{1 + C(s)G_o(s)}$$

$$Y(s) = G_c(s)R(s)$$

Back to
Our utilization control example

- Error: $E(t)=U_s-U(t)$
- Model (differential equation): $$U(t) = \int_{\tau=0}^{t} (R_a(\tau) - R_c(\tau))d\tau$$
- Controller C? $E(t) \Rightarrow R_a(t)$



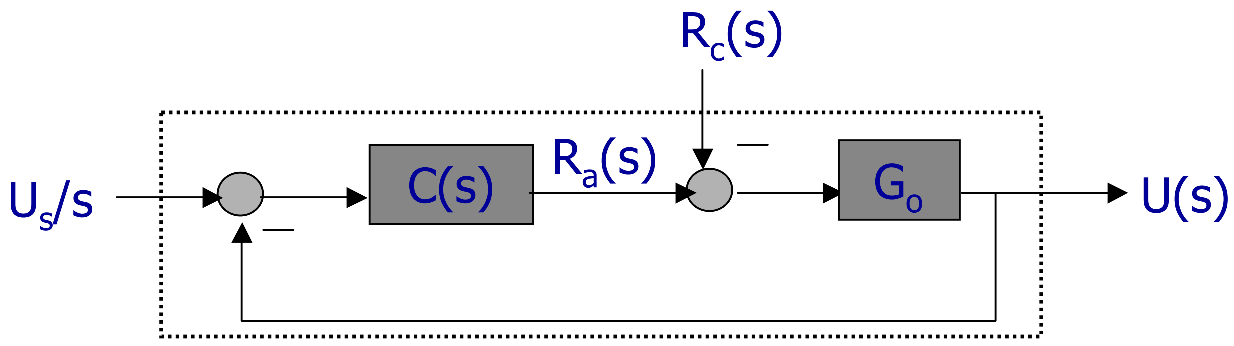
Model

Transfer func. & block diag.

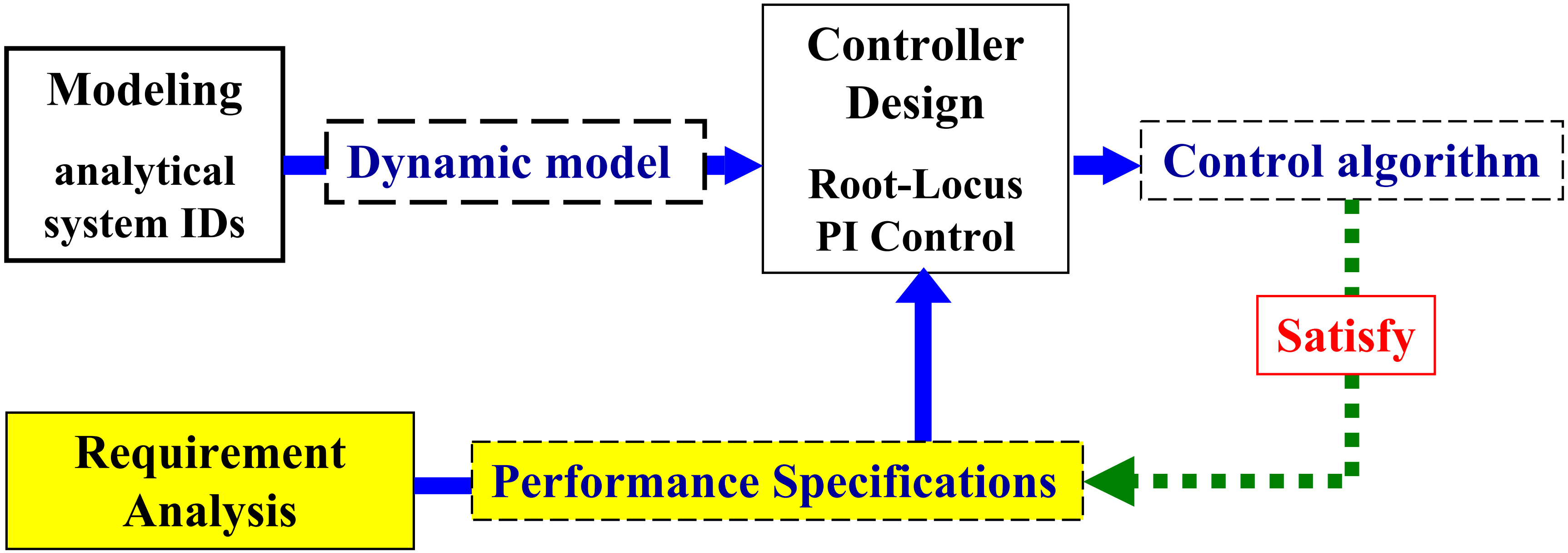
- CPU is modeled as an integrator

$$U(t) = \int_{\tau=0}^{t} (R_a(\tau) - R_c(\tau))d\tau \Leftrightarrow U(s) = \frac{R_a(s) - R_a(s)}{s} \Leftrightarrow G_o(s) = \frac{1}{s}$$

  - Inputs: reference $U_s(s) = U_s/s$; completion rate $R_c(s)$
  - Close-loop system transfer functions
    - $U_s(s)$ as input: $G_1(s) = C(s)G_o(s)/(1+C(s)G_o(s))$
    - $R_c(s)$ as input: $G_2(s) = G_o(s)/(1+C(s)G_o(s))$
  - Output: $U(s)=G_1(s)U_s/s+G_2(s)R_c(s)$



Control design methodology



Design Goals
Performance Specifications

- Stability
- Transient response
- Steady-state error
- Robustness
  - Disturbance rejection
  - Sensitivity

Performance Specs: bounded input,bounded output stability

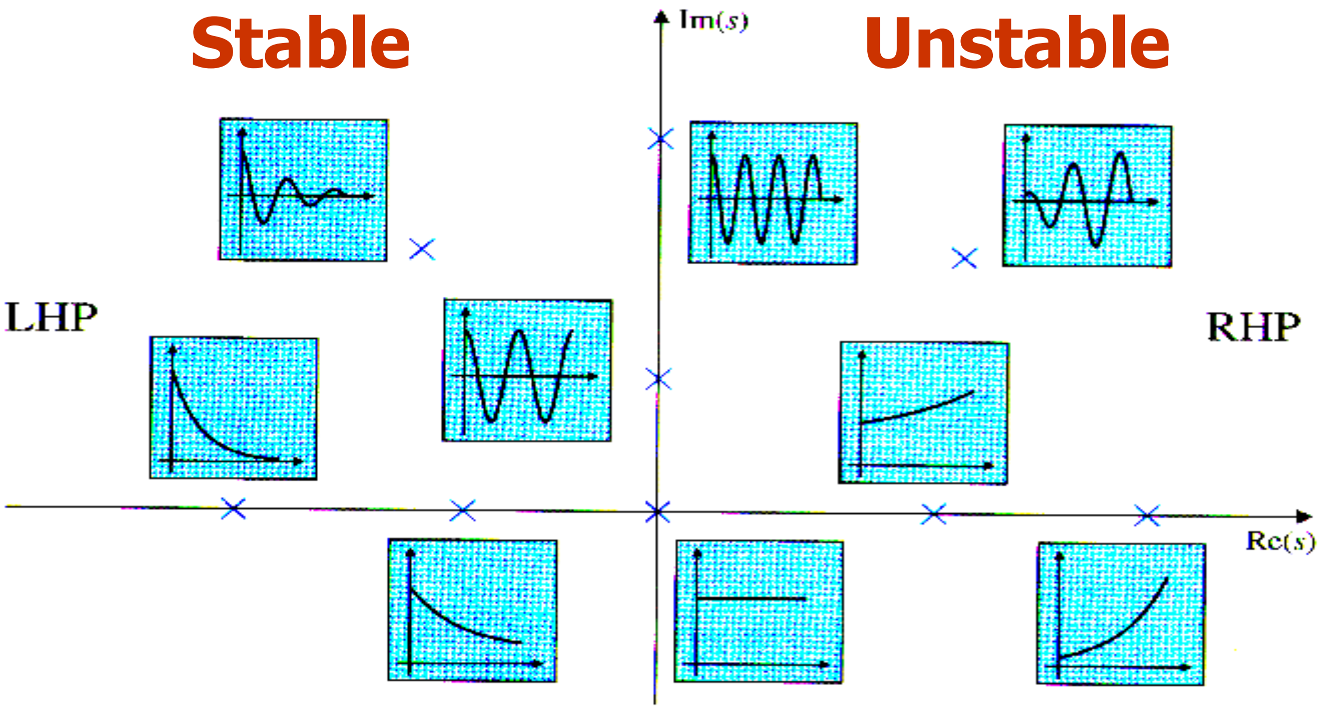
- BIBO stability: bounded input results in bounded output.
  - A LTI system is BIBO stable if all poles of its transfer function are in the LHP ($\forall p_i$, $Re[p_i]<0$).

$$Y(s) = G(s)U(s) = K\frac{\Pi_{i=1}^{m}(s-z_i)}{\Pi_{i=1}^{n}(s-p_i)} = \frac{C_1}{s-p_1} + \frac{C_2}{s-p_2} + \ldots + \frac{C_n}{s-p_n}$$

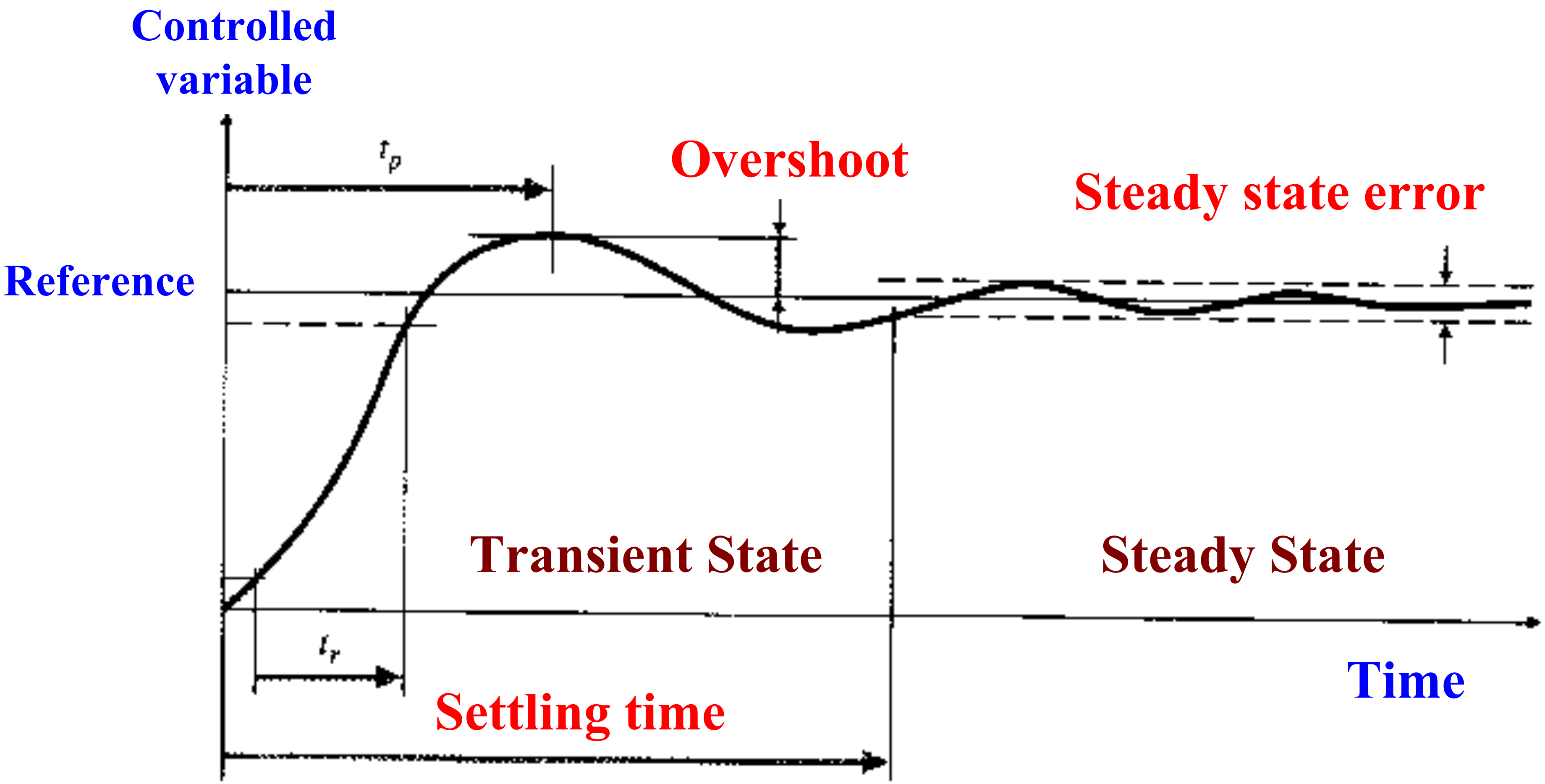
$$\Rightarrow y(t) = \sum_{i=1}^{n} C_i e^{p_i t}$$

$$Note: \quad C_i e^{p_i t} \xrightarrow{t\to\infty} \infty \qquad if \qquad \mathrm{Re}[p_i] > 0$$

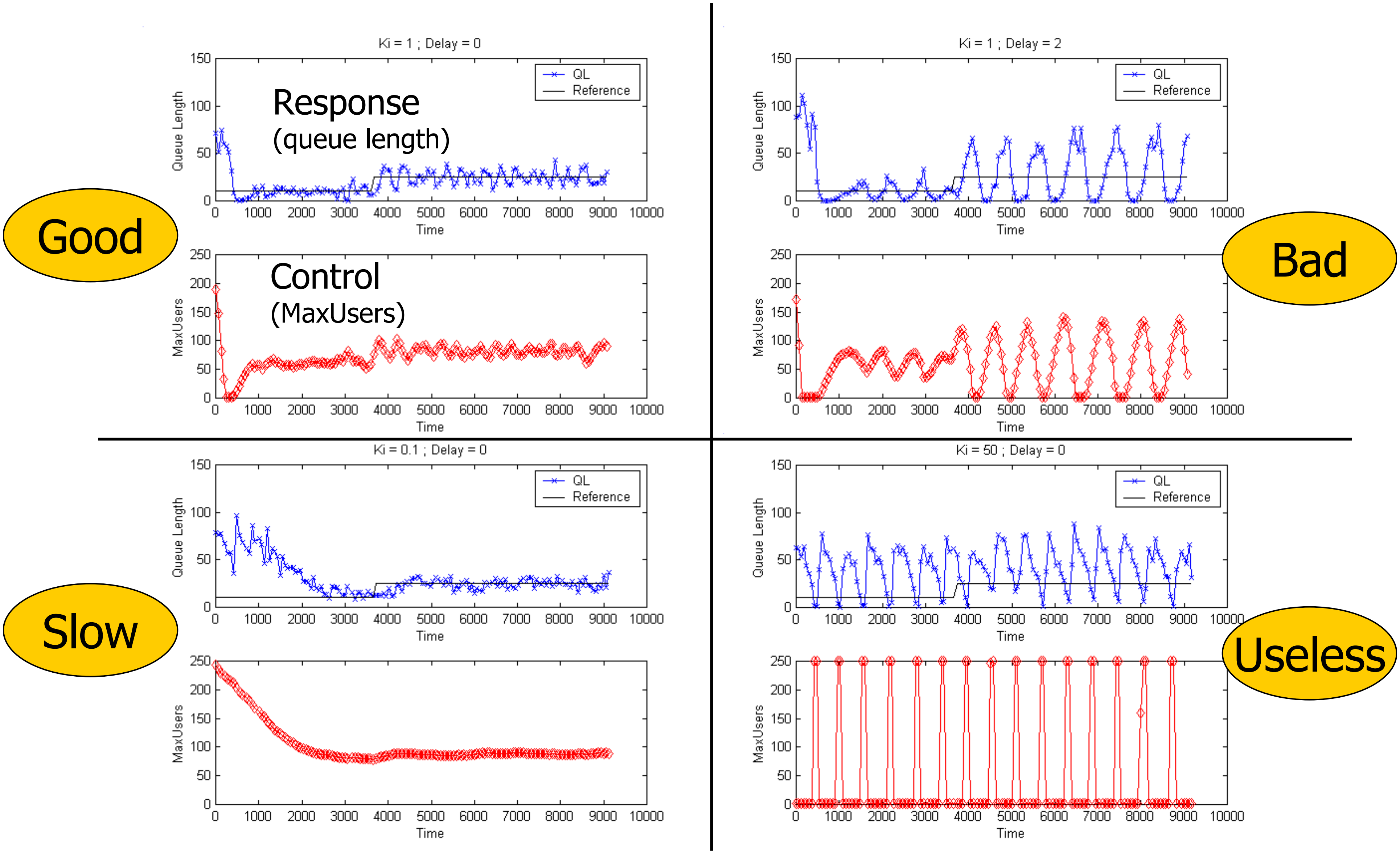
Performance Specs
Stability



Performance specifications



Example: Control & Response in an Email Server (IBM)



Performance Specs
Steady-state error

- Steady state (tracking) error of a stable system

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{t\to\infty}(r(t) - y(t))$$

r(t) is the reference input, y(t) is the system output.

- How accurately can a system achieve the desired state?
- Final value theorem: if all poles of sF(s) are in the open left-half of the s-plane, then

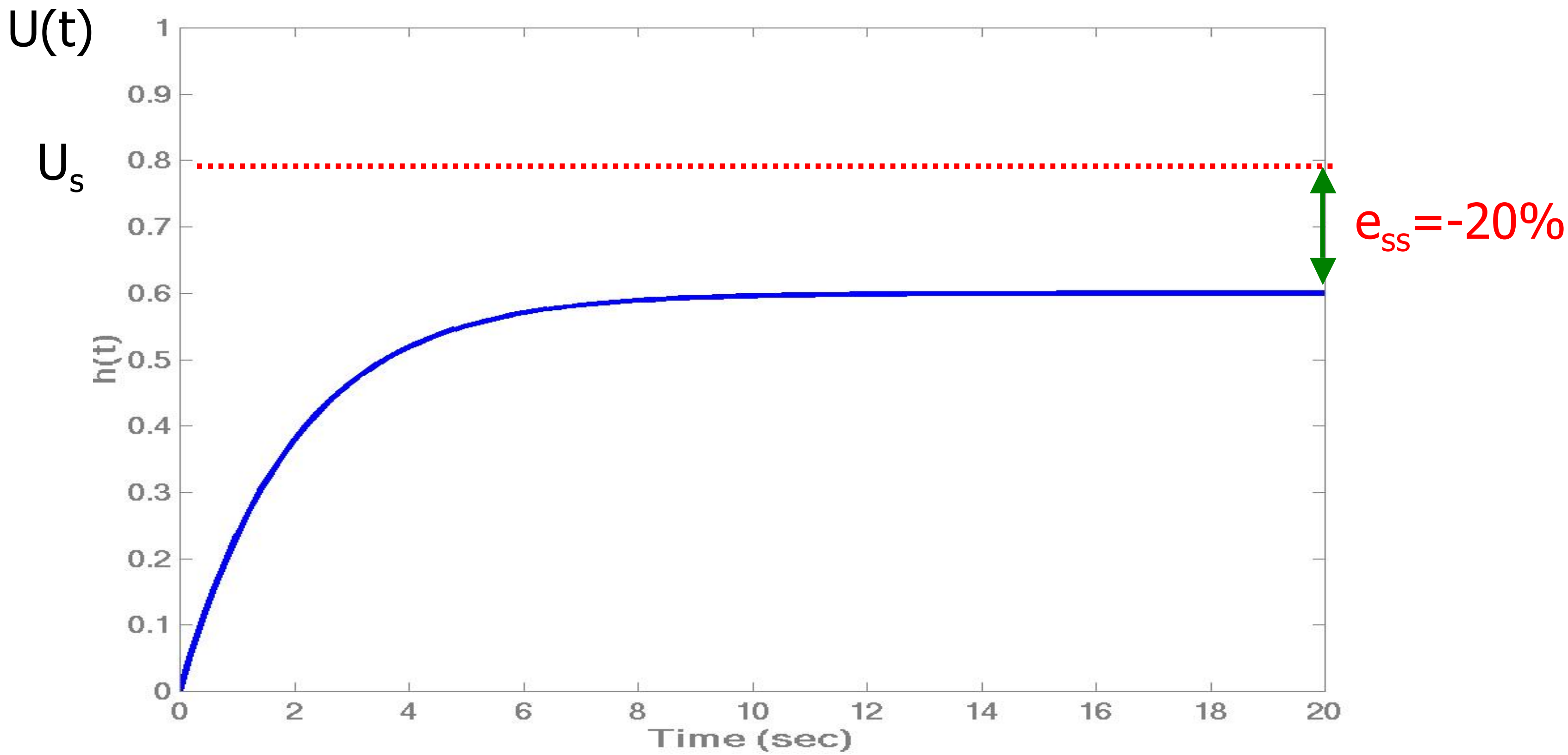
$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s)$$

- Easy to evaluate system long term behavior without solving it

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s)$$

Performance Specs
Steady-state error

Steady state error of a CPU-utilization control system



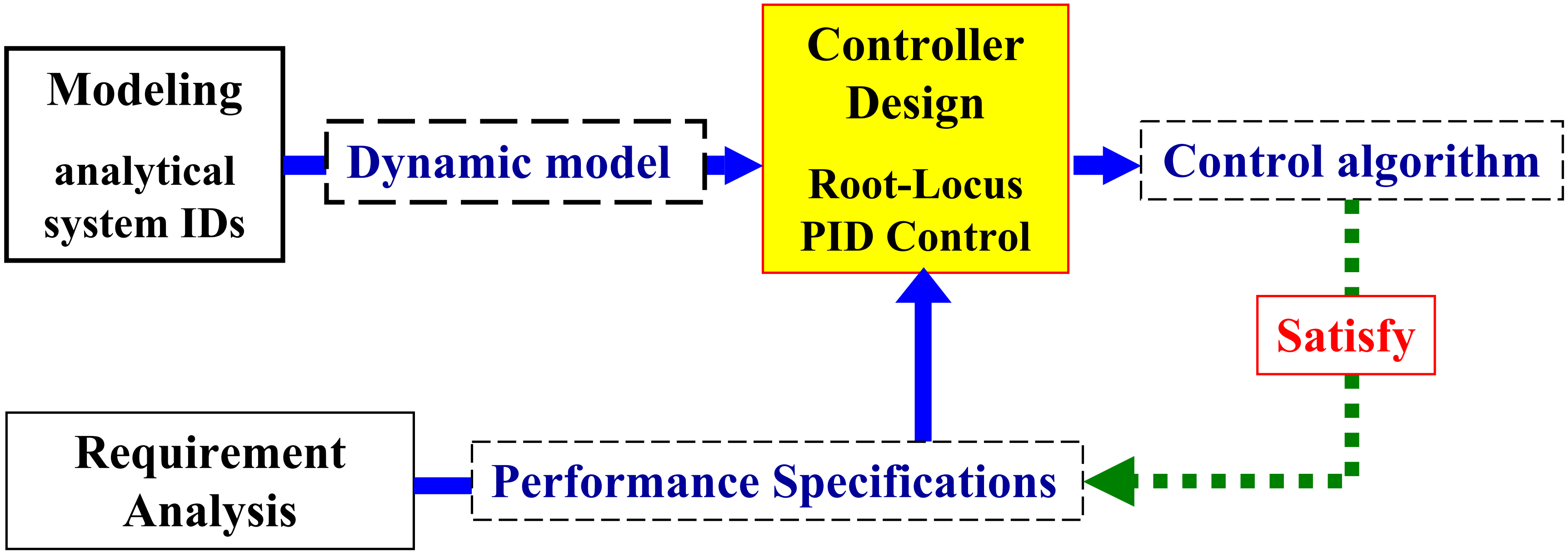
Performance Specs
Robustness

- **Disturbance rejection**: steady-state error caused by external disturbances
  - Can a system track the reference input despite of external disturbances?
  - Denial-of-service attacks
- **Sensitivity**: relative change in steady-state output divided by the relative change of a system parameter
  - Can a system track the reference input despite of variations in the system?
  - Increased task execution times
  - Device failures

Performance Specs
Goal of Feedback Control

- Guarantee stability
- Improve transient response
  - Short settling time
  - Small overshoot
- Small steady state error
- Improve robustness wrt uncertainties
  - Disturbance rejection
  - Low sensitivity

Control design methodology

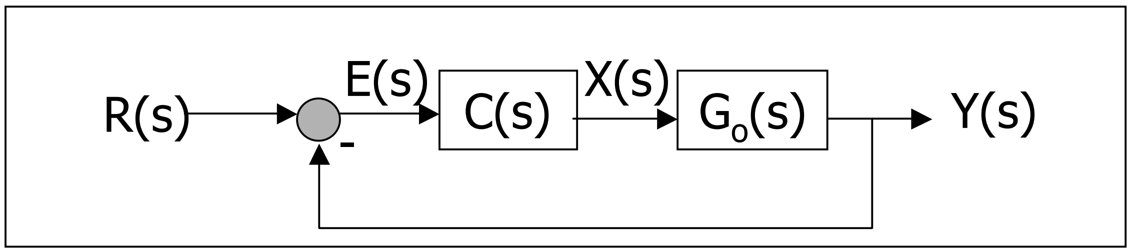


Controller Design
PID control

- Proportional-Integral-Derivative (PID) Control

- Proportional Control $x(t) = Ke(t) \quad \Leftrightarrow \quad C(s) = K$
- Integral control $x(t) = KK_i \int_0^t e(\tau)d\tau \quad \Leftrightarrow \quad C(s) = \frac{KK_i}{s}$
- Derivative control $x(t) = KK_d \dot{e}(t) \quad \Leftrightarrow \quad C(s) = KK_d s$

- Classical controllers with well-studied properties and tuning rules

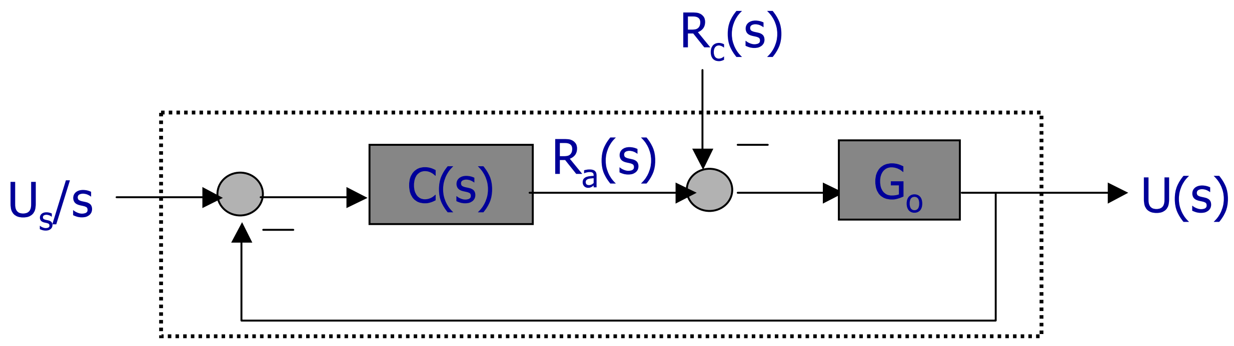


Controller Design
CPU Utilization Control

- CPU is modeled as an integrator

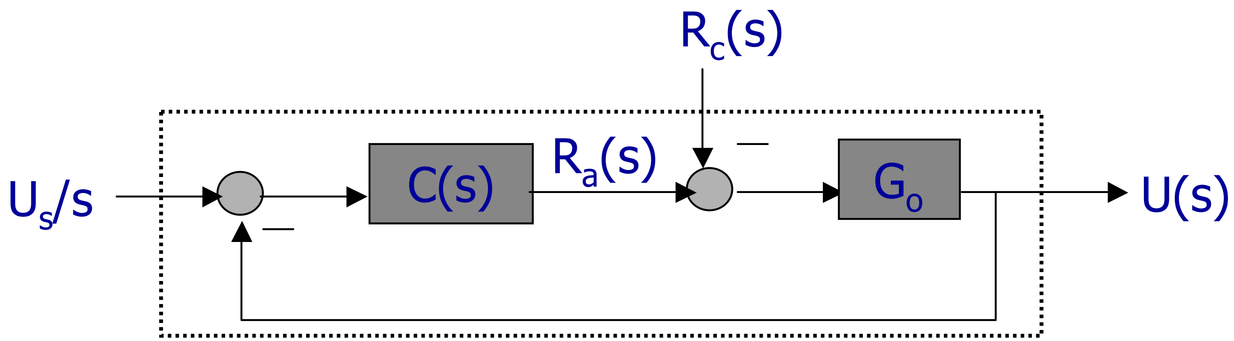
$$U(t) = \int_{\tau=0}^{t} (R_a(\tau) - R_c(\tau))d\tau \Leftrightarrow U(s) = \frac{R_a(s) - R_a(s)}{s} \Leftrightarrow G_o(s) = \frac{1}{s}$$

  - Inputs: set-point $U_s(s) = U_s/s$; task completion $R_c(s)$
  - Close-loop system transfer functions
    - $U_s(s)$ as input: $G_1(s) = C(s)G_o(s)/(1+C(s)G_o(s))$
    - $R_c(s)$ as input: $G_2(s) = G_o(s)/(1+C(s)G_o(s))$
  - C(s)=? to achieve zero steady-state error: $U(t) \rightarrow U_s$



Proportional Control
Stability

- Proportional Controller
  - $r_a(t)=Ke(t)$; $C(s) = K$
- Transfer functions
  - U_s/s as input: $G_1(s) = K/(s+K)$
  - $R_c(s)$ as input: $G_2(s) = 1/(s+K)$
- Stability
  - Pole $p_0 = -K<0 \Leftrightarrow$ System is BIBO stable *iff* $K>0$
  - Note: System may shoot to 100% if $K<0$!



Proportional Control
Steady-state error

- Assume completion rate $R_c(t)$ keeps constant for a time period longer than the settling time: $R_c(s)=R_c/s$
- System response is

$$U(s) = \frac{U_s G_1(s)}{s} + \frac{R_c G_2(s)}{s} = \frac{KU_s - R_c}{s(s+K)}$$

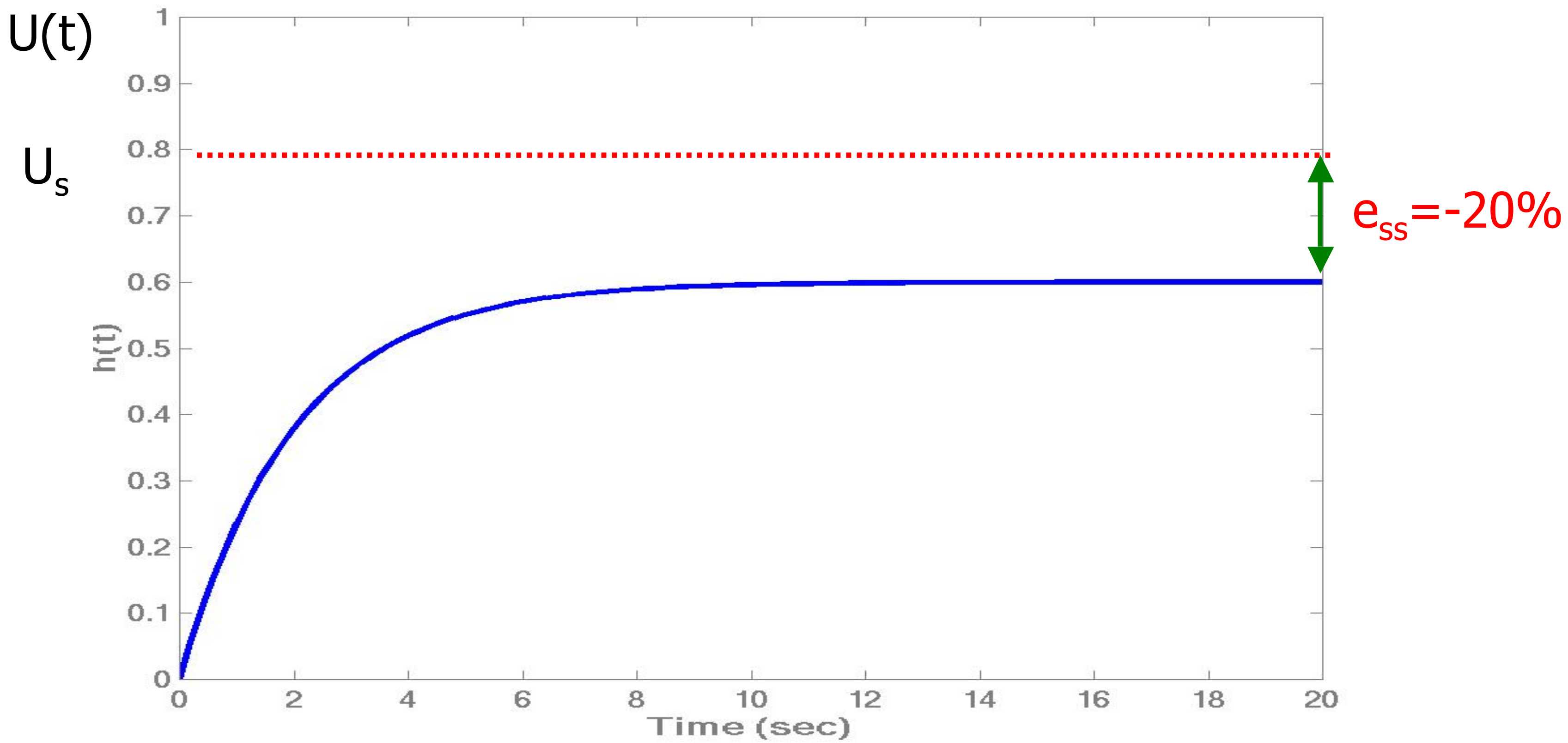
- Compute steady-state err using final value theorem,

$$\lim_{t\to\infty} U(t) = \lim_{s\to 0} sU(s) = \lim_{s\to 0} \frac{KU_s - R_c}{s+K} = U_s - \frac{R_c}{K} \qquad \Rightarrow e_{ss} = -\frac{R_c}{K} < 0$$

- P-control cannot achieve the desired CPU utilization U_s; instead it will end up lower by R_c/K Oops!
- The larger the proportional gain K is, the closer will CPU utilization approach to U_s

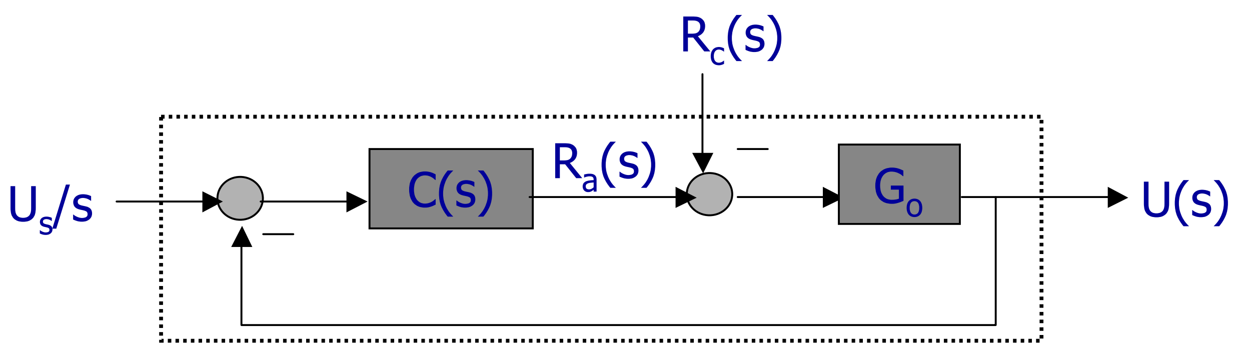
CPU Utilization

Proportional Control



Proportional-Integral Control
Stability

- Proportional Controller
  - $r_a(t)=K(e(t)+K_i \cdot \int_t e(\tau)d\tau)$ $\quad\quad C(s) = K(1+K_i/s)$
- Transfer functions
  - U_s/s as input: $\quad G_1(s) = (Ks+KK_i)/(s^2+Ks+KK_i)$
  - $R_c(s)$ as input: $\quad G_2(s) = s/(s^2+Ks+KK_i)$
- Stability
  - Poles $Re[p_0]<0$, $Re[p_0]<0$
    $\Leftrightarrow$ System is BIBO stable *iff* $K>0$ & $K_i>0$



Proportional Control

Steady-state error

- Assume completion rate $R_c(t)$ keeps constant for a time period longer than the settling time: $R_c(s)=R_c/s$
- System response is

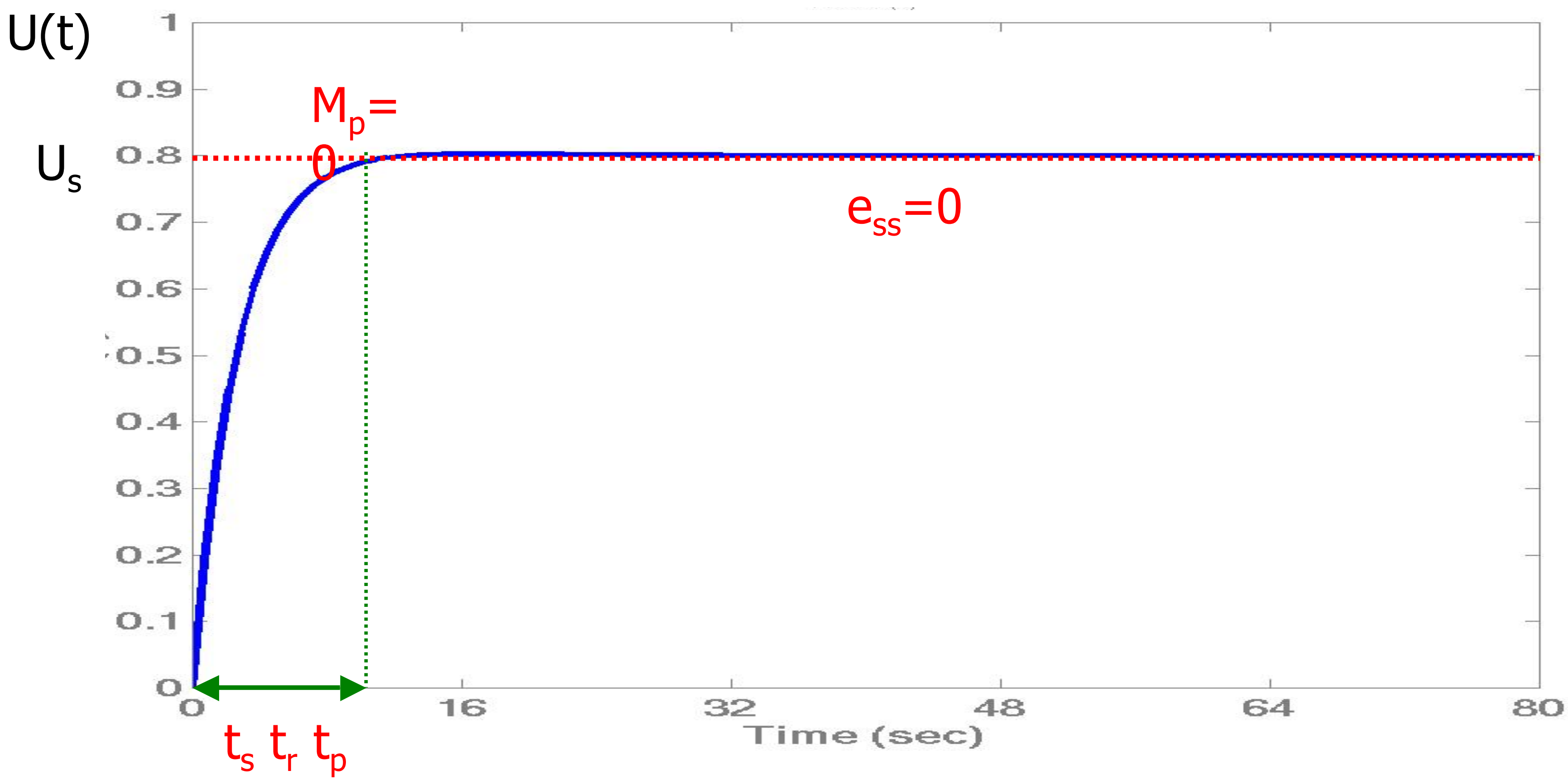
$$U(s) = \frac{U_s G_1(s)}{s} + \frac{R_c G_2(s)}{s} = \frac{(KU_s + R_c)s + KK_iU_s}{s(s^2 + Ks + KK_i)}$$

- Compute steady-state err using final value theorem,

$$\lim_{t\to\infty} U(t) = \lim_{s\to 0} sU(s) = \lim_{s\to 0} \frac{(KU_s + R_c)s + KK_iU_s}{s^2 + Ks + KK_i} = U_s \qquad \Rightarrow e_{ss} = 0$$

- PI control can accurately achieve the desired CPU utilization U_s √
- Control analysis gives design guidance

CPU Utilization
Proportional-Integral Control



Controller Design
Summary & pointers

- PID control: simple, works well in many systems
  - P control: may have non-zero steady-state error
  - I control: improves steady-state tracking
  - D control: may improve stability & transient response
- Linear continuous time control
  - Root-locus design
  - Frequency-response design
  - State-space design
  - G. F. Franklin et. al., *Feedback control of dynamic systems*

Discrete Control

- More useful for computer systems
- Time is discrete; sampled system
  - denoted k instead of t
- Main tool is z-transform

  - $f(k) \rightarrow F(z)$, where z is complex
  - Analogous to Laplace transform for s-domain

$$\mathbf{Z}[f(k)] = F(z) = \sum_{k=0}^{\infty} f(k) z^{-k}$$

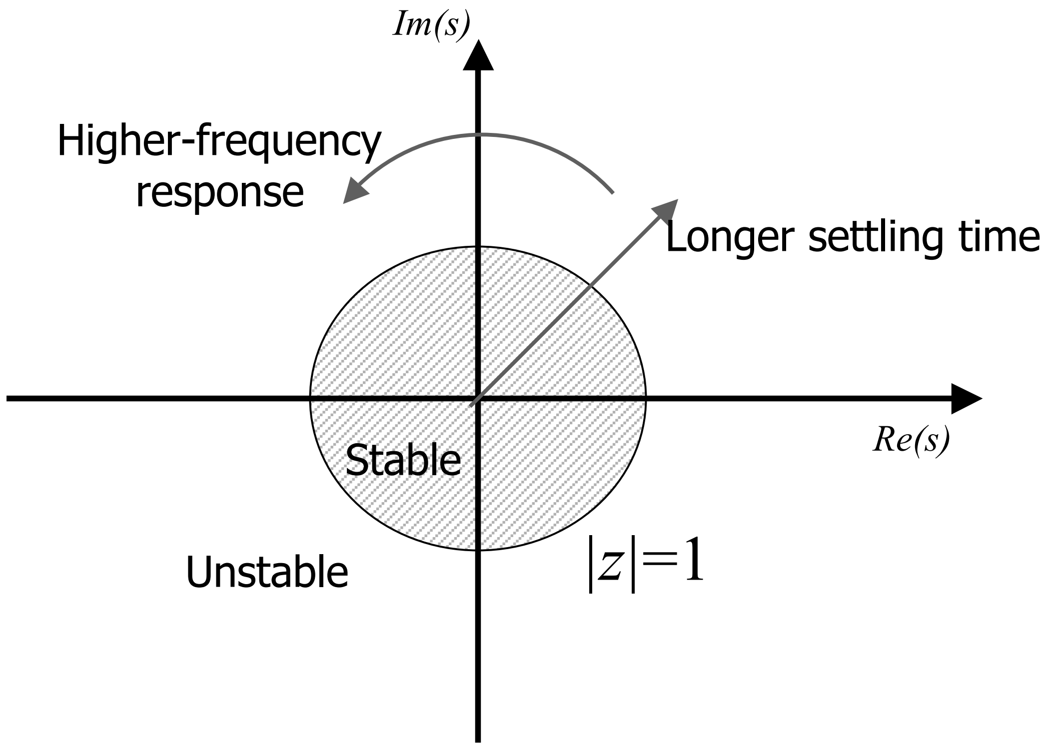
Discrete Modeling

- Difference equation
  - $V(m) = a_1 V(m\text{-}1) + a_2 V(m\text{-}2) + b_1 U(m\text{-}1) + b_2 U(m\text{-}2)$
  - z domain: $V(z) = a_1 z^{-1} V(z) + a_2 z^{-2} V(z) + b_1 z^{-1} U(z) + b_2 z^{-2} U(z)$
  - Transfer function $G(z) = (b_1 z + b_2)/(z^2 - a_1 z - a_2)$
- $V(m)$: output in m^{th} sampling window
- $U(m)$: input in m^{th} sampling window
- Order n: #sampling-periods in history affects current performance
- SP = 30 sec, and n = 2 → Current system performance depends on previous 60 sec

Root Locus analysis of Discrete Systems

- Stability boundary: $|z|=1$ (Unit circle)
- Settling time = distance from Origin
- Speed = location relative to Im axis
  - Right half = slower
  - Left half = faster

Effect of discrete poles



Intuition : $z = e^{Ts}$

Feedback control works in CS

- U.Mass: network flow controllers (TCP/IP – RED)
- IBM: Lotus Notes admission control
- UIUC: Distributed visual tracking
- UVA
  - Web Caching QoS
  - Apache Web Server QoS differentiation
  - Active queue management in networks
  - Processor thermal control
  - Online data migration in network storage (with HP)
  - Real-time embedded networking
  - Control middleware
  - Feedback control real-time scheduling

Advanced Control Topics

- Robust Control
  - Can the system tolerate noise?
- Adaptive Control
  - Controller changes over time (adapts)
- MIMO Control
  - Multiple inputs and/or outputs
- Stochastic Control
  - Controller minimizes variance
- Optimal Control
  - Controller minimizes a cost function of error and control energy
- Nonlinear systems
  - Neuro-fuzzy control
  - Challenging to derive analytic results

Issues for Computer Science

- Most systems are non-linear
  - But linear approximations may do
    - eg, fluid approximations
- First-principles modeling is difficult
  - Use empirical techniques
- Mapping control objectives to feedback control loops
  - ControlWare paper
- Deeply embedded networking
  - Massively decentralized control problem
  - Modelling
  - Node failures