**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 specifications
  - Controller design
- **Summary**

---

**Control**

- Applying input to cause system variables to conform to desired values called the reference.
  - Cruise-control car: $f_{engine}(t) \rightarrow$ speed=60 mph
  - E-commerce server: Resource allocation? $T_{response}=5$ sec
  - Embedded network: 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

---

**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

---

**Feedback (close-loop) Control**

- Measure variables and use it to compute control input
  - More complicated (as we need control theory)
  - Continuously measure & correct
    - Cruise-control car: measure speed & change engine force
    - E-commerce 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
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 specifications
  - 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)
    - Transfer functions (frequency domain)
    - Block diagram (pictorial)

\[
\begin{align*}
\dot{x}(t) &= a_2 x(t) + a_1 y(t) + a_0 y(t) = b_2 u(t) + b_1 u(t) + b_0 u(t) \\
Y(s) &= G(s) U(s) \\
\end{align*}
\]
Example
Utilization control in a video server

- Periodic task T_i corresponding to each video stream i
  - u_i: processing time, q_i: period
- Streams is requested CPU utilization: u_i(x)=q_i
- Total CPU utilization: \( U(t) = \sum_{i} u_i(x) \)
- (k) is the set of active streams
- Completion rate: \( R_i(t) = \sum_{j \in k} \delta(t-\tau_{ij}) \) \( \delta(t) \) is the set of terminated video streams during \( [t, t+\Delta] \)
- Admission rate: \( R_0(t) = \sum_{j \in k} \delta(t-\tau_{ij}) \)
- Problem: design an admission controller to guarantee \( U(t) = t^+ \)

Problem: design an admission controller to guarantee \( U(t) = t^+ \)

\[ u(t) \]

Admission rate
Completion rate
Total CPU utilization
Periodic task T

\begin{align*}
\text{A Diversion to Math} & \\
\text{System representations} & \\
\text{Time domain: convolution; differential equations.} & \\
& u(t) \rightarrow y(t) = g(t) \ast u(t) + \int_{0}^{t} g(t-\tau) u(\tau) d\tau \\
\text{s (frequency) domain: multiplication} & \\
& U(s) \rightarrow Y(s) = G(s)U(s) \\
\text{Block diagram: pictorial} & \\
\text{s-domain is a simple & powerful “language” for control analysis} &
\end{align*}

\begin{align*}
\text{A Diversion to Math} & \\
\text{Laplace transform} & \\
\text{Basic translations} & \\
\text{Impulse function} & f(t) \delta(t) = F(s) \delta(t) \\
\text{Step signal} & f(t) u(t) = F(s)u(t) \\
\text{Ramp signal} & f(t) t = F(s)t \\
\text{Sinusoidal signal} & f(t) \sin(\omega t) = F(s)\omega \sin(\omega t) \\
\text{Composition rules} & \\
\text{Linearity} & L(f(t)+g(t)) = L(f(t))+L(g(t)) \\
\text{Differentiation} & L(f(t)) = y(t) \rightarrow f(t) \rightarrow y(s) = \tau F(s) \\
\text{Integration} & L(f(t) \tau) = y(t) \rightarrow f(t) \rightarrow y(s) = \frac{1}{s} y(t) \\
\end{align*}

\begin{align*}
\text{A Diversion to Math} & \\
\text{Transfer function} & \\
\text{Modeling a linear time-invariant (LTI) system} & \\
& G(s) = Y(s)H(s) = Y(s) \rightarrow G(s)H(s) \\
& \text{E.g., a second order system with poles } p_1 \text{ 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} \\
\end{align*}
A Diversion to Math

Poles and Zeros

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

\[ F(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \ldots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_0} \]

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

\( \{p_i \} \) are poles of the function and decide the system behavior.

A Diversion to Math

Block diagram

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

- Represent a feedback control system

\[ R(s) \xrightarrow{G_c(s)} Y(s) \]

\[ R(s) \xrightarrow{G_o(s)} Y(s) \]

Model

Transfer func. & block diag.

- CPU is modeled as an integrator

\[ U(t) = \int (R_c(t) - R_a(t)) dt \]

\[ U(t) = U(s) \]

Control design methodology

- Requirement Analysis

- Dynamic model

- Modelling analytical system

- Controller Design

- Root-Locus

- PI Control

- Performance Specifications

- Control algorithm

- Satisfy
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 $\mathbb{H}_p$.

$$Y(s) = G(s)K(s) = \frac{C_p}{\mathcal{H}(s)}$$

$$\Rightarrow y(t) = \sum C_p e^{p_i}$$

Note: $C_p e^{p_i} \rightarrow \infty$ if $\text{Re}[p_i] > 0$

Performance Specs
Stability

Stable

Unstable

Time

Reference

Overshoot

Steady state error

Transient State

Steady State

Settling time

Controlled variable

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

Good

Bad

Slow

Useless

Performance Specs
Steady-state error
- Steady state (tracking) error of a stable system

$$e_s = \lim_{t \to \infty} (r(t) - y(t))$$

$r(t)$ 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 $f(s)$ are in the open left-half of the s-plane, then

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

- Easy to evaluate system long term behavior without solving it

$$e_s = \lim_{t \to \infty} s e(t) = \lim_{s \to 0} s F(s)$$
Performance Specs
Steady-state error

Steady state error of a CPU-utilization control system

\[ e_{ss} = -20\% \]

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

Control design methodology

- Modeling
  - analytical system libs
- Requirement Analysis
- Controller Design
  - Root-Locus
  - PID Control
- Controller Design
  - CPU Utilization Control
  - Classical controllers with well-studied properties and tuning rules

Performance Specs
Goal of Feedback Control

- Guarantee stability
- Improve transient response
- Small overshoot
- Small steady state error
- Disturbance rejection
- Low sensitivity
- Increased task execution times
- Device failures

Controller Design
PID control

- Proportional
  \[ x(t) = K_e(t) \rightarrow C(x) = K \]
- Integral
  \[ x(t) = K \int e(t) \, dt \rightarrow C(x) = \frac{K}{s} \]
- Derivative
  \[ x(t) = K \, \dot{e}(t) \rightarrow C(x) = K \, s \]
  - Classical controllers with well-studied properties and tuning rules

Controller Design

Performance Specifications

\[ \text{PID control: } \frac{U(s)}{R(s)} = G(s) \]

Controller Design

CPU Utilization Control

- CPU is modeled as an integrator
  \[ U(t) = \int R(t) \, dt + U_{ref} \]
  - \( R(t) \) as input
  - Compute \( U(t) \)
  - \( \frac{R(t)}{s} \) as input
  - \( u(s) \) as output
  - \( G(s) = 1 + \frac{s}{\tau} \)
  - \( C(s) \) to achieve zero steady-state error: \( U(s) \rightarrow U_{ref} \)
Compute steady-state err using final value theorem,

\[ U(t) \]

System response is the settling time:

Assume completion rate \( R_f(t) \) keeps constant for a time period longer than the settling time: \( R_f(t) \approx R_f \)

Proportional Control

Steady-state error

- Assume completion rate \( R_f(t) \) keeps constant for a time period longer than the settling time: \( R_f(t) \approx R_f \)
- System response is

\[ U(s) = \frac{G_i(s)}{s} \cdot \frac{R(s)}{s} = \frac{R_i}{s} \]

- Compute steady-state err using final value theorem,

\[ \lim_{s \to 0} U(s) = \lim_{s \to 0} s \cdot U(s) = \lim_{s \to 0} \left( \frac{R_i}{s} \cdot \frac{R_i}{s} \right) = U_s \implies \varphi_s = 0 \]

Proportional Control

Stability

- Proportional Controller
- \( r(t) \) is input; \( C(s) = K(1+K/s) \)
- Transfer functions
  - \( G_i(s) = 9(K+iK)/s^2 + K \)
  - \( R(s) \) is input: \( G_i(s) \approx 9(K+iK)/s^2 + K \)
- Stability
  - Pole \( R_p(\omega_0) > 0 \); System is BIBO stable \( \forall K>0 \)
  - Note: System may shoot to 100% if \( K<0 \)

CPU Utilization

Proportional Control

Prop: \( u(t) \)

- CPU utilization approach to \( \text{U}_i \)
- The larger the proportional gain \( K \) is, the closer will CPU utilization approach to \( \text{U}_i \)

Proportional-Integral Control

Stability

- Proportional Controller
  - \( r(t) \) is input; \( C(s) = K(1+iK/s) \)
  - Transfer functions
    - \( G_i(s) = 9(K+iK)/s^2 + K \)\( \text{a} \)
    - \( R(s) \) is input: \( G_i(s) \approx 9(K+iK)/s^2 + K \)
  - Stability
    - Pole \( R_p(\omega_0) > 0 \); Pole \( R_p(\omega_0) > 0 \)
    - System is BIBO stable \( \forall K>0 \) & \( K>0 \)

CPU Utilization

Proportional-Integral Control

Prop: \( u(t) \)

- PI control can accurately achieve the desired CPU utilization \( \text{U}_s \)
- Control analysis gives design guidance
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
- \( \mathcal{Z}[f(k)] = F(z) = \sum f(k)z^{-k} \)
- \((\mathcal{Z}(s) \rightarrow \mathcal{Z}(z)) \), where \( z \) is complex
- Analogous to Laplace transform for s-domain

Discrete Modeling
- Difference equation
  - \( V(n) = a_1 V(n-1) + a_2 V(n-2) + b_1 U(n-1) + b_2 U(n-2) \)
  - \( z \)-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 \( G(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{a_1 z^{-1} + a_2 z^{-2}} \)
- \( V(n) \): output in \( n \)-th sampling window
- \( U(n) \): input in \( m \)-th sampling window
- Order \( n \): sampling periods in history affects current performance
- SP = 30 sec, and \( n = 2 \rightarrow \) Current system performance depends on previous 60 sec

Root Locus analysis of Discrete Systems
- Stability boundary: \(|z|=1\) (Unit circle)
- Setting time = distance from Origin
- Speed = location relative to Im axis
  - Right half = slower
  - Left half = faster

Effect of discrete poles
- \( \text{higher-frequency response} \rightarrow \text{longer settling time} \)
- \( |z|=1 \)
- \( z = e^k \)

Feedback control works in CS
- UMass: network flow controllers (TCP/IP – RED)
- IBM: Lotus Notes admission control
- UIC: 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 (adaptive)
- 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
    - e.g., 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