Lecture 12: PID Control, Frequency Domain Design, and Fundamental Limitations

Arman Siahvashi

Faculty of Engineering, The University of Sydney

Table of contents

This lecture is formed around Chapter 11, Chapter 12 of the textbook, and the paper "Respect the unstable" by G. Stein:

https://ieeexplore.ieee.org/document/1213600

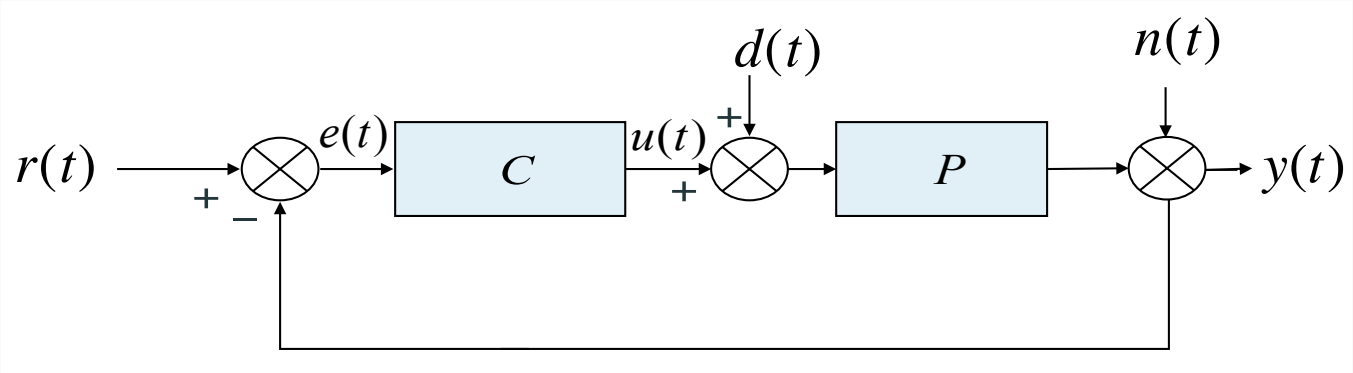
Lab report is due soon!

- Lab report submission is due on Sunday, 25 May
- Carefully read the lab report instructions on Canvas (updated with clarifications)
- Make sure you answer all the lab questions (in blue)
- Check the lab report rubric on Canvas
- Ask questions in this week's tutorial and lab sessions or on Ed Discussion

PID Control

PID controllers

Proportional + integral + derivative (PID) controllers, also called three-term controllers, are by far the most common form of controller in industry.



The control signal is calculated as:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \dot{e}(t)$$

where $e(t) = r(t) - y(t)$ is the tracking error.

P, PI, and PD controllers have only one or two terms and other gains are set to zero.

Alternative Representations

The PID controller

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \dot{e}(t)$$

has transfer function

$$C(s) = k_p + \frac{k_i}{s} + k_d s.$$

Another representation is so called "standard form":

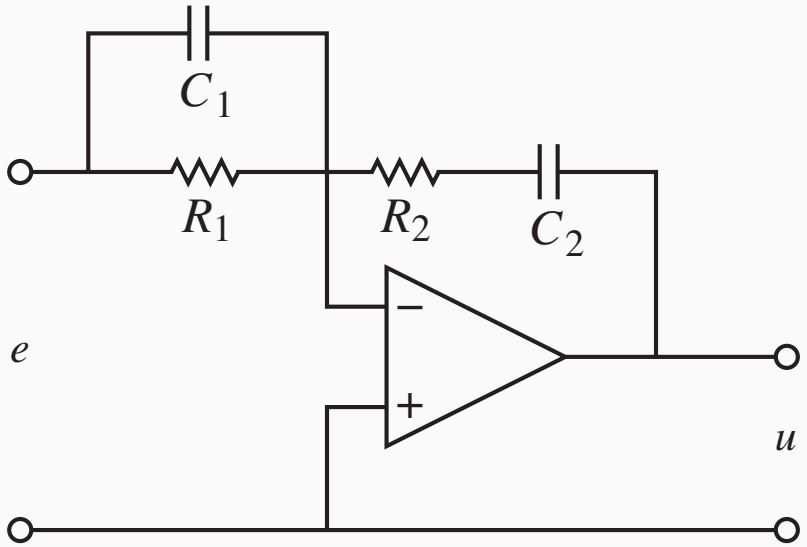
$$u(t) = k_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \dot{e}(t) \right)$$

with $T_i = \frac{k_p}{k_i}$ and $T_d = \frac{k_d}{k_p}$ interpreted as time constants.

In matlab, the commands `pid` and `pidstd` construct PID controllers in these two forms, respectively.

Hardware

Many companies produce industrial PID controllers as a standalone package. One can also be constructed with a single op-amp, two resistors and two capacitors.



(b) PID controller

$$k_p = \frac{R_1C_1 + R_2C_2}{R_1C_2}, \qquad T_i = R_1C_1 + R_2C_2, \qquad T_d = \frac{R_1R_2C_1C_2}{R_1C_1 + R_2C_2}.$$

Software

It is also straightforward to code a PID controller on a microcontroller, e.g. the following pseudocode from the textbook:

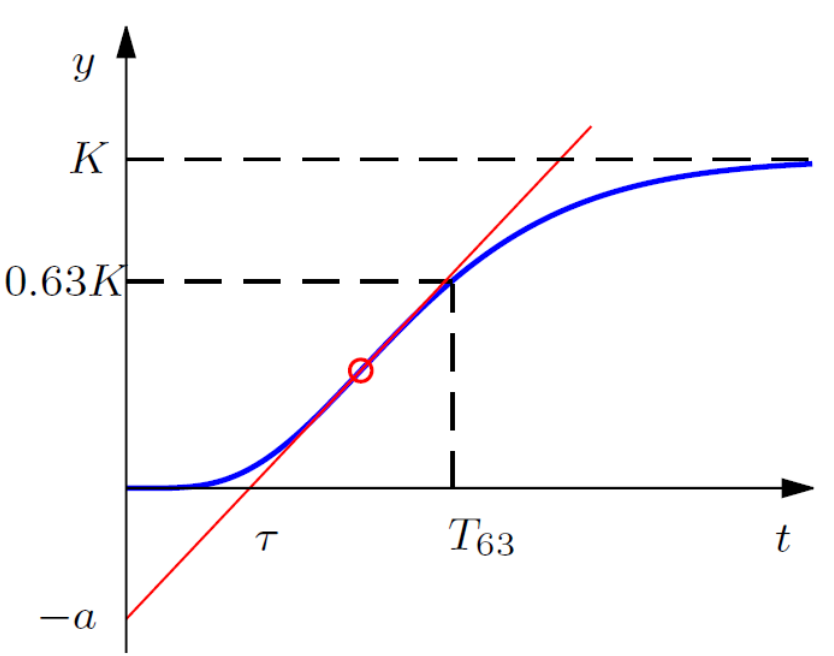
```
% Precompute controller coefficients
bi=ki*h
ad=Tf/(Tf+h)
bd=kd/(Tf+h)
br=h/Tt

% Control algorithm - main loop
while (running) {
  r=adin(ch1)                % read setpoint from ch1
  y=adin(ch2)                % read process variable from ch2
  P=kp*(b*r-y)               % compute proportional part
  D=ad*D-bd*(y-yold)         % compute derivative part
  v=P+I+D                    % compute temporary output
  u=sat(v,ulow,uhigh)        % simulate actuator saturation
  daout(ch1)                 % set analog output ch1
  I=I+bi*(r-y)+br*(u-v)      % update integral state
  yold=y                     % update derivative state
  sleep(h)                   % wait until next update interval
}
```

PID Tuning: Ziegler-Nichols Step Response (bump test)

- **Goal**: obtain a first-cut PID tuning without a detailed model.
- **Experiment**: apply a small open-loop step input and record the output $y(t)$.
- **Measure two features** from the curve (see next slide):
  - *Apparent time delay* τ : where the tangent at the steepest point intersects the time axis.
  - *Intercept* a : The tangent's intersection with the output axis.
- The pair $(\tau,\, a)$ captures how long the plant "hesitates" and how fast it eventually reacts.
- Ziegler & Nichols (1942) fitted a simple table that maps (τ, a) directly to $k_p,\, T_i,\, T_d$.

Treat this as a *starting point*; fine-tune afterwards.

From Measurement to Controller Parameters

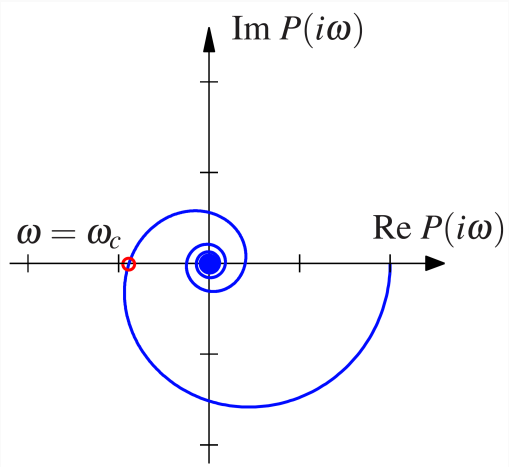


Type	k_p	T_i	T_d
P	$1/a$		
PI	$0.9/a$	$\tau/0.3$	
PID	$1.2/a$	$\tau/0.5$	0.5τ

- *Too sluggish?* $\Rightarrow$ increase k_p or shorten T_i.
- *Too oscillatory?* $\Rightarrow$ decrease k_p or increase T_d.

PID Tuning: Ziegler–Nichols Frequency-Response Method

The second approach is to start with **proportional control only**, and **increase gain until oscillation occurs**, i.e. find the gain margin. The critical gain (e.g. last P gain) is k_c, and the oscillation period is T_c (time for one full cycle). You can get T_c in time domain (e.g. time between peaks) or from crossover frequency ω_c where the oscillation period is $T_c = 2\pi/\omega_c$: Homework: can you get the value of ω_c directly from Nyquist plot?

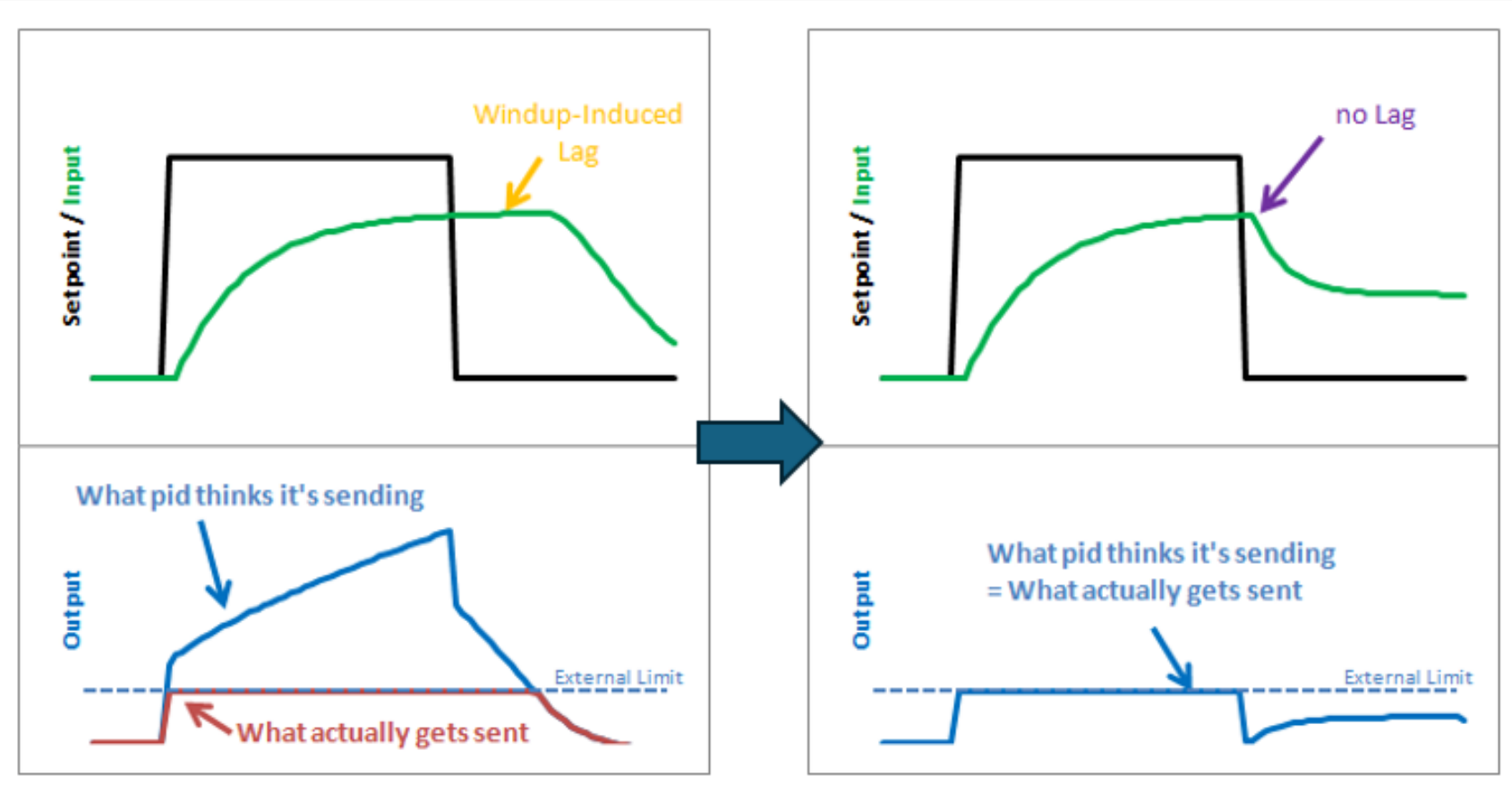


Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.45k_c$	$T_c/1.2$	
PID	$0.6k_c$	$T_c/2$	$T_c/8$

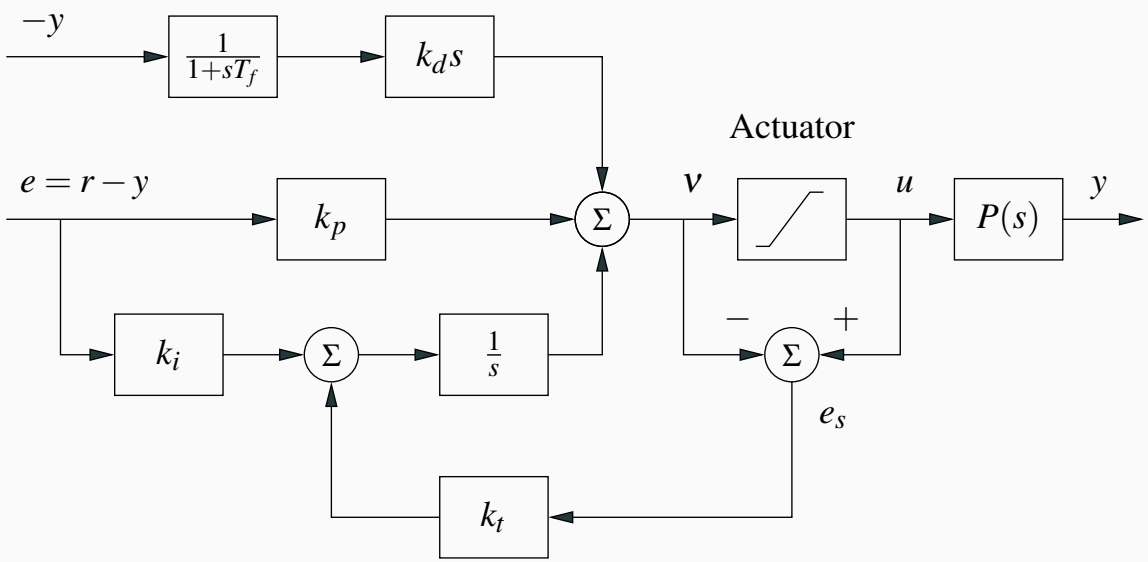
PID Challenge: Integrator Wind-up

- A very common form of nonlinearity is *saturation* or *clipping*
- Many actuators (valves, motors, pumps) can only move so far or exert so much force. Once you hit that physical limit, the controller's output simply **"clips" or "saturates"** at its maximum (or minimum) value.
- **The "I" in PID keeps summing** (integrating) any sustained error.
- If the actuator is slammed against its limit, the error often stays large (because the plant can't respond further).
- **The integrator keeps growing,** thinking "I must push harder," even though you're already at full throttle.
- The system takes *longer* to reach steady state, so error is integrated for longer than it should, and so the control signal builds up and the system overshoots.

PID Challenge: Integrator Wind-up

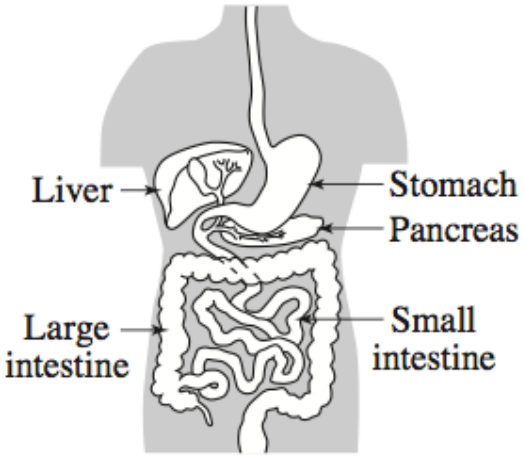


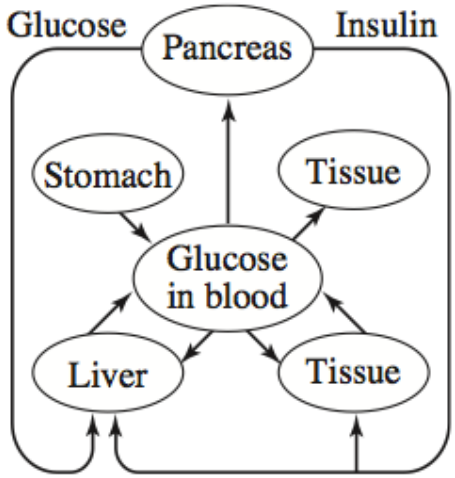
Anti-Windup by Back-Calculation

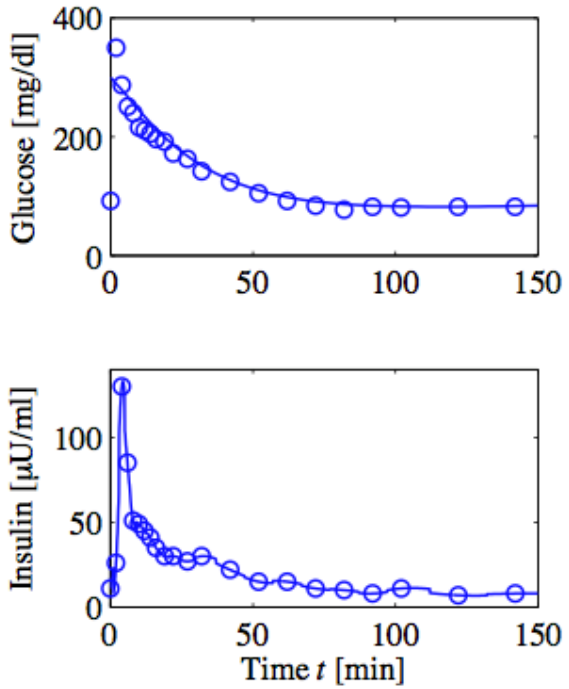


- The idea is to decrease the value of the integrator when the control signal saturates.
- When the actuator saturates, the controller output v is clipped, so $u \neq v$.
- The difference $e_s = v - u$ is the anti-windup error.
- This error is fed back to the integrator via gain k_t to prevent wind-up.
- This corrects the integrator state and allows faster recovery after saturation (subtracts the extra integral).
- Typical choice: $k_t \approx \frac{k_i}{k_p}$ or a small multiple thereof.

Insulin/Glucose Dynamics







Transfer Function for Insulin/Glucose Dynamics

I&EC research
Industrial & Engineering Chemistry Research

ACS Editors' Choice

Article

pubs.acs.org/IECR

Design and Evaluation of a Robust PID Controller for a Fully Implantable Artificial Pancreas

Lauren M. Huyett, Eyal Dassau, Howard C. Zisser, and Francis J. Doyle, III*

Department of Chemical Engineering, University of California Santa Barbara, Santa Barbara, California 93106-5080, United States

ABSTRACT: Treatment of type 1 diabetes mellitus could be greatly improved by applying a closed-loop control strategy to insulin delivery, also known as an artificial pancreas (AP). In this work, we outline the design of a fully implantable AP using intraperitoneal (IP) insulin delivery and glucose sensing. The design process utilizes the rapid glucose sensing and insulin action offered by the IP space to tune a PID controller with insulin feedback to provide safe and effective insulin delivery. The controller was tuned to meet robust performance and stability specifications. An anti-reset windup strategy was introduced to prevent dangerous undershoot toward hypoglycemia after a large meal disturbance. The final controller design achieved 78% of time within the tight glycemic range of 80−140 mg/dL, with no time spent in hypoglycemia. The next step is to test this controller design in an animal model to evaluate the *in vivo* performance.

Transfer Function for Insulin/Glucose Dynamics

Let y(t) be the blood glucose concentration (mg/dL), and $u(t)$ be the intraperitoneal insulin (U/h). Then the continuous transfer function from u to y is

$$G(s) = \frac{-12000(TDI)^{-1}}{(247s+1)(17s+1)^2} \tag{1}$$

where TDI is a constant representing the total daily insulin dose of the patient.

- This transfer function is obtained from clinical experiments first in discrete time, and then converted in continuous time.
- We can already design a PID controller for this plant.

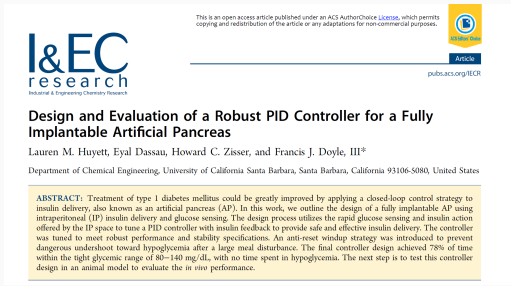
PID Controller for Artificial Pancreas

Table 2. Parameters for PID Control Using IMC Tuning for Intraperitoneal Insulin Compared to Parameters Previously Identified for PID Control Using Subcutaneous Insulin

parameter	IMC for intraperitoneal insulin	previously suggested for subcutaneous insulin[36]
K_C ([U/h]/[mg/dL])[a]	$0.023(TDI)(\tau_C + 11)^{-1}$	$0.0026(TDI)/(\text{body weight})$
τ_I (min)	273	450 (day), 150 (night)
τ_D (min)	23.5	98

[a]The units on the variables in this row are body weight (kg), TDI (U), and τ_C (min).

This controller is then implemented in discrete time and tested in the UVA / Padova metabolic simulator that mimics a diabetic patient.



The Result

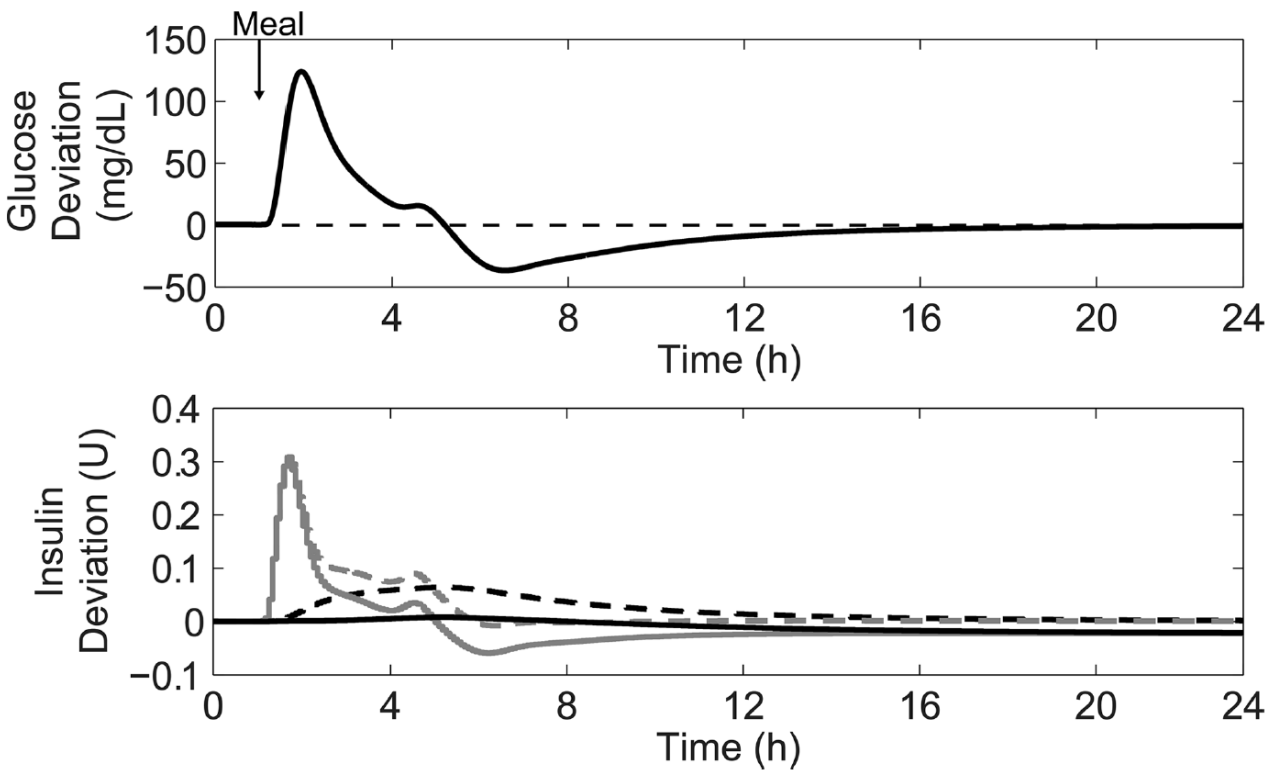
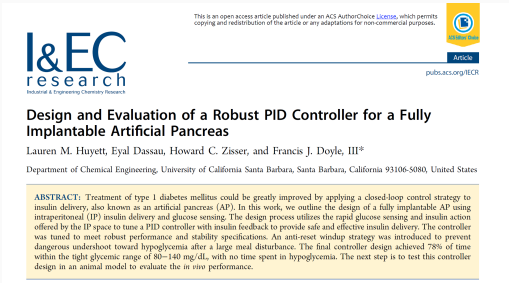


Figure 3. Demonstration of set point undershoot encountered when using integral action after a 100 g-CHO meal. The top panel shows the



I&EC research
Industrial & Engineering Chemistry Research
Article
pubs.acs.org/IECR

Design and Evaluation of a Robust PID Controller for a Fully Implantable Artificial Pancreas

Lauren M. Huyett, Eyal Dassau, Howard C. Zisser, and Francis J. Doyle, III*

Department of Chemical Engineering, University of California Santa Barbara, Santa Barbara, California 93106-5080, United States

ABSTRACT: Treatment of type 1 diabetes mellitus could be greatly improved by applying a closed-loop control strategy to insulin delivery, also known as an artificial pancreas (AP). In this work, we outline the design of a fully implantable AP using intraperitoneal (IP) insulin delivery and glucose sensing. The design process utilizes the rapid glucose sensing and insulin action offered by the IP space to tune a PID controller with insulin feedback to provide safe and effective insulin delivery. The controller was tuned to meet robust performance and stability specifications. An anti-reset windup strategy was introduced to prevent dangerous undershoot toward hypoglycemia after a large meal disturbance. The final controller design achieved 78% of time within the tight glycemic range of 80−140 mg/dL, with no time spent in hypoglycemia. The next step is to test this controller design in an animal model to evaluate the *in vivo* performance.

Sensitivity Functions and Loop Shaping (Feedback Limitations)

Setting the Stage: What Are We Trying to Control?

When designing a feedback system, what are we fighting against?

Real-world systems constantly face:

- **Disturbances** – things that push on the system from outside.
  - *Example:* A gust of wind hits a car on a cruise control.
- **Sensor Noise** – random fluctuations in measurements.
  - *Example:* A thermostat flickers due to electrical noise.
- **Tracking a Reference** – the system needs to follow a command smoothly.
  - *Example:* A drone is told to rise 1 meter and should do so quickly without overshoot.

Key idea: We can't react equally to everything.

This leads us to **sensitivity functions**, which tell us:

– How much of a disturbance or noise signal makes it to the output.

– How "sensitive" our system is to each type of input, e.g. if the system isn't highly sensitive to some inputs, you can focus your design efforts on the ones it is most sensitive to.

What Limits Feedback Control? Overview

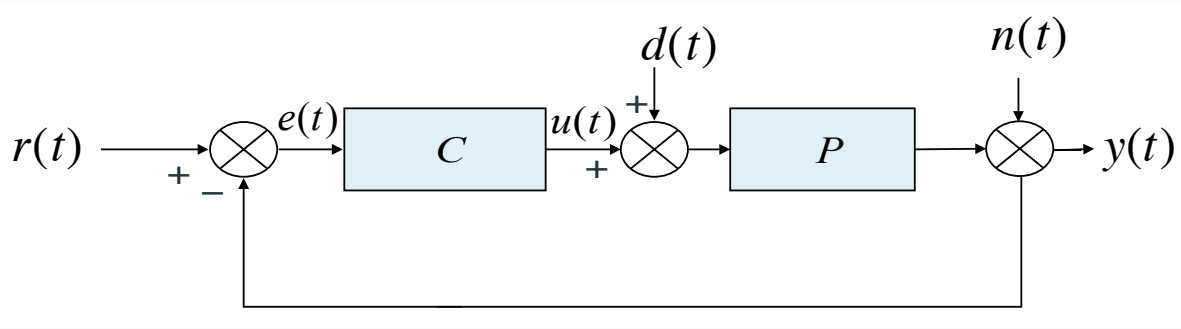
Not all control problems can be solved by clever design or tuning. Some systems are fundamentally hard to control, no matter which method you use.

In this lecture, we'll look at **4 theoretical limitations** that explain why:

1. **The Identity:** $S + T = 1$
   You cannot reduce both disturbance and noise sensitivity at the same time.
2. **Bode's Gain–Phase Relation**
   Steeper gain roll-off leads to more phase lag, hurting stability.
3. **Interpolation Constraints**
   You can't cancel unstable poles or zeros, you're stuck with them.
4. **Bode's Sensitivity Integral**
   If you push sensitivity down at some frequencies, it must pop up elsewhere.

These aren't design flaws; they're unavoidable trade-offs in feedback control.

Standard Feedback Loop



Let $\eta(t)$ be the real system output before noise. Note that $y(t)$ is the measurement with added noise $n(t)$.

We can obtain the following closed-loop transfer functions:

$$\eta = \frac{PC}{1+PC}r + \frac{P}{1+PC}d - \frac{PC}{1+PC}n \tag{2}$$

$$e = \frac{1}{1+PC}r - \frac{P}{1+PC}d - \frac{1}{1+PC}n \tag{3}$$

$$u = \frac{C}{1+PC}r - \frac{PC}{1+PC}d - \frac{C}{1+PC}n \tag{4}$$

These equations show how the output, error, and control signal respond to the three external signals.

The Sensitivity Function and Disturbance Response

For a stable plant P without control, $C = 0$, the effect of (exponential) disturbances on process output η is $\eta = P(s)d$.

With control, we have $\eta = \frac{P(s)}{1+P(s)C(s)}d = S(s)P(s)d$
Where we have defined the following transfer function

$$S(s) = \frac{1}{1 + P(s)C(s)}$$

as the **sensitivity function that tells us how much disturbance makes it through the feedback loop**, or how much of disturbance at frequency w passes through to the output:

- If $|S(j\omega)| < 1$ then the control system **attenuates disturbances** at that frequency ω;
- If $|S(j\omega)| = 0$, then disturbances at that frequency are **eliminated**. E.g. an integrator does this for zero frequency.
- But if $|S(j\omega)| > 1$ the feedback system actually **makes things worse** than they would have been in open loop.

1-First Limitation: Complementary Sensitivity Function T

Goal: We want the system to track well, reject disturbances, and ignore sensor noise.The tracking error is: $r - \eta = S \cdot r - S \cdot P \cdot d + T \cdot n$

- $S = \frac{1}{1+PC}$ is the **sensitivity function**. (e.g. model changes, tracking error, and disturbances)
- $T = \frac{PC}{1+PC}$ is the **complementary sensitivity function**. (handles what S(s) doesn't e.g. reference tracking, and sensitivity to noise)
- So: $S + T = 1$ (each affect different parts of performance and work together to balance different goals)

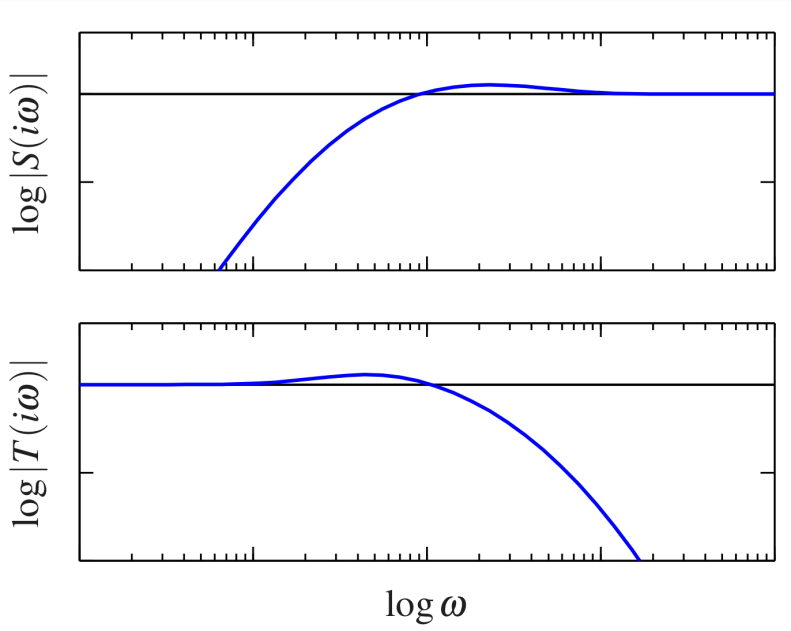
Key point: You cannot make both $|S(j\omega)|$ and $|T(j\omega)|$ small at the same time for all frequencies. Like a seesaw: one goes down, the other goes up.

- Make $|S(j\omega)|$ small at low frequencies $\rightarrow$ better tracking and disturbance rejection.
- Make $|T(j\omega)|$ small at high frequencies $\rightarrow$ better noise rejection.

You must choose where to place your trust, in disturbance rejection or in measurement accuracy.

Shaping S and T

- For many real systems, reference signals and disturbances have fairly low frequency content (i.e. change infrequently or slowly), while measurement noise tends to be high-frequency.
- For such systems, it makes sense to have $|S(j\omega)|$ small at **low frequencies** (to reject disturbances and track the reference well, and $|T(j\omega)|$ small at **high frequencies** (to suppress sensor noise).



This can be achieved by having $L(j\omega) \gg 1$ at **low frequencies**, and $L(j\omega) \ll 1$ at **high frequencies.**

2-Second Limitation: Gain Roll-off vs. Stability

We often want to make gain drop fast to block high-frequency noise. "Gain drop-off" refers to how the magnitude of the loop gain $|L(j\omega)|$ decreases as the frequency ω increases.

But Bode's rule says:

- Faster gain drop $\Rightarrow$ more phase lag.
- Too much phase lag $\Rightarrow$ lower phase margin.
- Low phase margin $\Rightarrow$ risk of instability.

So: Don't make the gain drop too sharply near crossover frequency!

Why does this matter?

- The **crossover frequency** ω_{gc} is where $|L(j\omega_{gc})| = 1$.
- This is the point where the system's **speed** and **stability** are **most sensitive.**
- If you drop gain too fast just before or after this point, phase lag increases rapidly.

Bode's Gain–Phase Relation

Bode's gain–phase relationship for minimum-phase systems is:

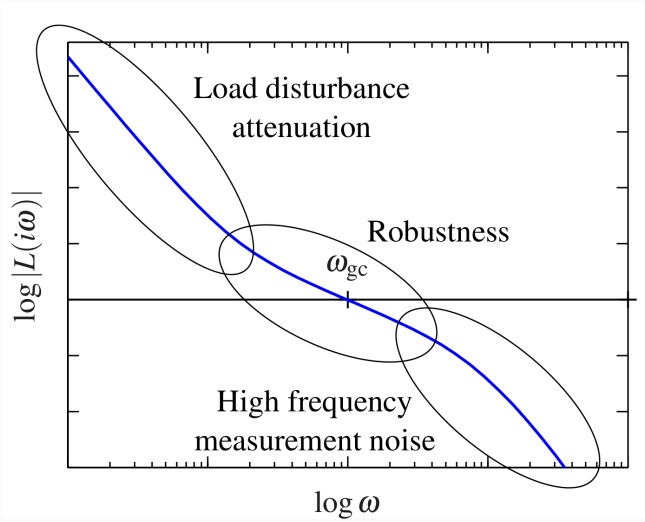
$$\angle L(j\omega_{gc}) \approx 90^\circ \times \frac{d \log |L(j\omega)|}{d \log \omega}$$

What does this mean?

- Phase is approx. proportional to gain derivative on a Bode plot.
- The faster the gain drops on a Bode plot, the more phase lag is introduced.
- A slope of -20 dB/decade leads to about -90° of phase lag.
- A slope of -40 dB/decade causes about -180° which can destroy the phase margin and cause instability.

Analogy: A **gentle slope** (–20 dB/dec) is like gently turning the steering wheel= smooth, stable control. A **steep slope** (–40 dB/dec) is like moving the wheel quickly = the car can lose control (instability).

Desired Loop Shape



The loop transfer function $L = PC$ should be shaped carefully to balance performance and stability:

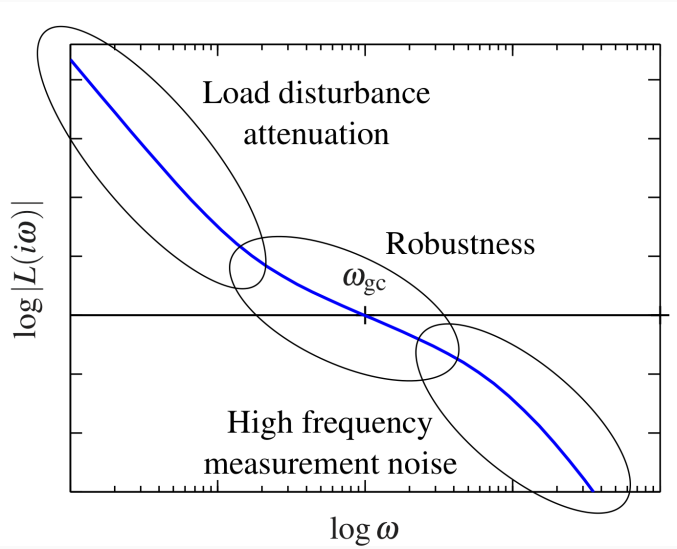
- **High gain at low frequencies** $\rightarrow$ for good reference tracking and disturbance rejection.
  (Because $S = \frac{1}{1+L} \approx 0$ when $L \gg 1$)
- **Low gain at high frequencies** $\rightarrow$ to filter out measurement noise.
  (Because $T = \frac{L}{1+L} \approx 0$ when $L \ll 1$)
- **Gentle slope near crossover** (ω_{gc}) $\rightarrow$ to keep phase margin and avoid instability.(Bode's relation: $\angle L(j\omega_{gc}) \approx 90^\circ \times \frac{d\log|L(j\omega)|}{d\log\omega}$)

Desired Loop Shape (Option 1)

To get the right loop shape, we adjust the controller C by adding poles, zeros, or gain, based on the plant P. This process is called **loop shaping** and is done using the Bode plot of $L = PC$.

Think of it like tuning an equalizer or sketching a portrait; you shape the curve iteratively until it "looks right."

- **Boost low-frequency gain** $\rightarrow$ better tracking/disturbance rejection.
- **Flatten or roll off high-frequency gain** $\rightarrow$ reduce noise sensitivity.
- **Watch the slope near crossover** $\rightarrow$ keep phase lag under control.



Loop Shaping (Option 2)

- Instead of tuning the controller $C(s)$ directly, we can **design the desired loop shape** $L(s)$ first, then compute the controller as $C(s) = \frac{L(s)}{P(s)}$.
- This is also called **loop shaping**, and it gives us direct control over how the system responds across frequencies.
- **Be careful:** Avoid canceling unstable poles or zeros (with a zero/pole in your controller) when computing $C = L/P$, this can reduce robustness and lead to hidden instability or fragile behavior.

Example: Choose $L(s) = \frac{\alpha}{s}$ (first-order integrator) with gain α which sets the desired crossover frequency. Now compute $S(s)$ and $T(s)$.

$$S(s) = \frac{1}{1+L(s)} = \frac{1}{1+\frac{\alpha}{s}} = \frac{s}{s+\alpha}, \quad T(s) = \frac{L(s)}{1+L(s)} = \frac{\frac{\alpha}{s}}{1+\frac{\alpha}{s}} = \frac{\alpha}{s+\alpha}$$

- $S(s)$ is small at low frequencies $\rightarrow$ good disturbance rejection.
- $T(s)$ is small at high frequencies $\rightarrow$ good noise rejection.

PID as Loop Shaping

Consider mass-spring damper $P(s) = \frac{1}{ms^2+ds+k}$. Let the desired loop shape be $L(s) = \frac{\alpha}{s}$. Then we compute

$$C(s) = \frac{L(s)}{P(s)} = \frac{\alpha(ms^2 + ds + k)}{s}$$

This is a PID controller with gains $k_p = \alpha d, k_i = \alpha k, k_d = \alpha m$. **You are not guessing $C(s)$, you are choosing how you want $L(s)$ to behave.**

- α sets how "aggressive" your control is, higher α means a stronger (faster-responding) loop.
- The physical parameters m, d, and k (mass, damping, and stiffness) determine how the PID gains are scaled.
- In practice, it's better to reduce gain at high frequencies to avoid noise amplification, e.g. $L(s) = \frac{\alpha}{s} \cdot \frac{\omega_b}{s+\omega_b}$ adds a low-pass filter for faster roll-off (i.e. decays quickly at high frequencies).

Example

- Consider loop-shaping PID design for a highly-resonant system:

$$P(s) = \frac{1}{s^2 + 0.01s + 1}$$

- We choose the desired loop shape as $L(s) = \frac{1}{s}$:
  - This behaves like an integrator — giving good disturbance rejection and tracking at low frequencies.
  - It's a simple and well-understood shape that's easy to implement and reason about.
- Then compute the controller:

$$C(s) = \frac{L(s)}{P(s)} = \frac{s^2 + 0.01s + 1}{s}$$

  So you've designed a PID controller to shape the loop to behave like $L(s) = \frac{1}{s}$.This corresponds to PID gains: $k_p = 0.01$, $k_i = 1$, $k_d = 1$.

* Loop shaping is sensitive to model accuracy. Wrong model = controller acts on the wrong dynamics. This leads to performance degradation, increased sensitivity, and instability, especially in highly resonant systems.

The "Gang of Four"

In a feedback system, the signals we care about (output η, error e, and control input u) are influenced by: r: the reference, d: external disturbances, and n: measurement noise. We use two key functions:

$$S = \frac{1}{1+PC} \quad \text{(sensitivity)}, \qquad T = \frac{PC}{1+PC} \quad \text{(complementary sensitivity)}$$

Using these, we get:

$$\begin{bmatrix} \eta \\ e \\ u \end{bmatrix} = \begin{bmatrix} T & +PS & -T \\ S & -PS & -S \\ CS & -T & -CS \end{bmatrix} \begin{bmatrix} r \\ d \\ n \end{bmatrix}$$

These four transfer functions—S, T, PS, and CS—describe how the loop handles tracking, disturbance rejection, noise, and control effort. Together, they are known as the "gang of four." It lets you **analyze how your controller** responds to everything that **might disturb or affect the system**.

Why You Shouldn't Cancel RHP Poles or Zeros

Imagine you design a system with simple $P(s) = \frac{s+b}{s+a}$**, zero at** $s = -b$**, pole at** $s = -a$. We want to shape the loop using:
$L(s) = \frac{\alpha}{s} \Rightarrow C(s) = \frac{L(s)}{P(s)} = \frac{\alpha(s+a)}{s(s+b)}$

S and T **look okay**, $S(s) = \frac{1}{1+L(s)} = \frac{s}{s+1}, \quad T(s) = \frac{L(s)}{1+L(s)} = \frac{1}{s+1}$

But, internal signals can still misbehave. Internal closed-loop transfer functions:

$$CS = \frac{\alpha(s+a)}{(s+b)(s+\alpha)}, \quad PS = \frac{s+b}{(s+a)(s+\alpha)}$$

What can go wrong?

- If $b < 0 \rightarrow$ the plant has a **right-half-plane zero**. Now CS becomes unstable $\rightarrow$ the control signal grows uncontrollably.
- If $a < 0 \rightarrow$ the plant has a **right-half-plane pole**. Now PS is unstable $\rightarrow$ even small disturbances cause the output to blow up.

Key Point: Aim for controller designs that stabilize the system without relying on cancelling unstable poles or zeros. Robust controllers should respect all unstable dynamics, not hide them.

3-Third Limitation: Interpolation Constraints

- No matter how clever your controller is, the sensitivity function $S(s)$ has to behave a certain way if there are RHP poles or zeros.

For internal stability, the closed-loop sensitivity $S(s)$ must satisfy:

1. $S(s)$ is finite for all s with $\text{Re}(s) \geq 0$, **You can't have infinite gain in the RHP (that would be unstable)**
2. $S(s) = 1$ at any RHP zero of $L(s)$, **That means feedback has no effect at those frequencies.**
3. $S(s) = 0$ at any RHP pole of $L(s)$, **The system is trying to force the output to 0 at that unstable frequency**

These are called **interpolation constraints** because they "pin down" the shape of $S(s)$ at certain points.**Your controller has to "interpolate" between these fixed values**, it has no choice. We cannot cancel RHP poles or zeros using feedback — we are stuck with them and must design around their effects.

Sensitivity to Plant Variations

- What happens to the closed-loop response if the plant $P(s)$ changes?
- Let's call the change in the plant dP. We want to know how much the closed-loop transfer function $T(s)$ changes.
- The answer is *:

$$\frac{dT}{T} = S \cdot \frac{dP}{P}$$

- $\frac{dP}{P}$ is the **relative change in the plant.**
- $\frac{dT}{T}$ is the **relative change in the system's behavior.**
- S is the **sensitivity function** which scales that change.
- If S is small $\rightarrow$ small change in $T \rightarrow$ **robust system**.
- If S is large $\rightarrow$ large change in $T \rightarrow$ **sensitive, fragile system**.

Takeaway: The sensitivity function measures sensitivity of the closed-loop behaviour to plant variation.

***Derivation:** Using the logarithmic derivative rule, if $f = \frac{g}{h}$ then $\frac{df}{f} = \frac{dg}{g} - \frac{dh}{h}$.
Apply this to $T = \frac{PC}{1+PC}$, with $g = PC$, $h = 1 + PC$, and C is constant:

$$\frac{dT}{T} = \frac{d(PC)}{PC} - \frac{d(1+PC)}{1+PC} = \frac{dP}{P} - \frac{PC}{1+PC} \cdot \frac{dP}{P} = \left(\frac{1}{1+PC}\right) \cdot \frac{dP}{P} = S \cdot \frac{dP}{P}$$

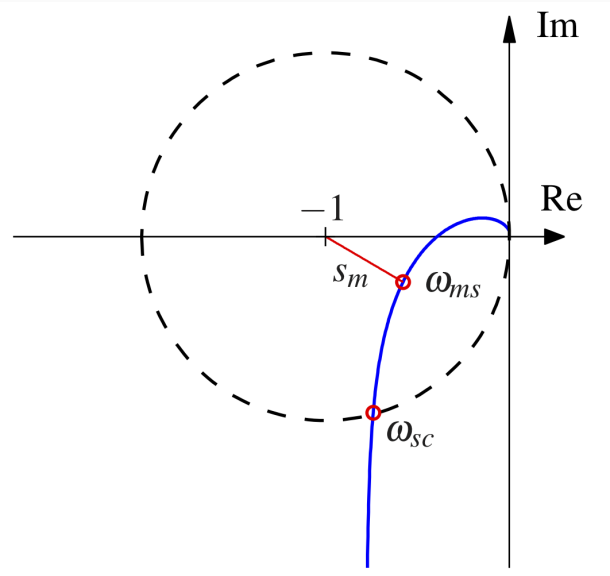
Interpreting Sensitivity: Distance to Instability

- The sensitivity function also tells us how **close we are to instability**.
- We can write:

$$|S(j\omega)| = \frac{1}{|1 + L(j\omega)|} = \frac{1}{|L(j\omega) - (-1)|}$$

- This means that **the size of $|S(j\omega)|$ is the inverse of the distance to the -1 point on the Nyquist plot (next slide)**.
- The closer $L(j\omega)$ gets to -1, the bigger $|S(j\omega)|$ becomes, and the more sensitive (fragile) the system is.

Sensitivity and the Nyquist Plot



- The blue curve is the Nyquist plot of $L(j\omega)$.
- The red line s_m is the shortest distance to the critical point -1.
- This distance determines the **maximum value of** $|S(j\omega)|$:

$$\max |S(j\omega)| = \frac{1}{s_m}$$

- ω_{ms}: frequency where $|S(j\omega)|$ is largest (max sensitivity point).
- ω_{sc}: frequency where $|S(j\omega)| = 1$ (sensitivity crossover).
- Inside the dotted circle: $|S(j\omega)| > 1 \rightarrow$ system amplifies disturbances!

Conclusion: The closer the Nyquist curve gets to -1, the less robust the system becomes.

Bode's Sensitivity Integral

4-Fourth Limitation: Bode's Sensitivity Integral

- The sensitivity function $S(j\omega)$ tells us how much the system output reacts to disturbances or model uncertainty at frequency ω.
- Can we make $|S(j\omega)|$ small at **all frequencies** for perfect disturbance rejection?
- **Answer: No.** There is a fundamental trade-off:

$$\int_0^\infty \log |S(j\omega)| \, d\omega = \pi \sum_k p_k$$

  where p_k are the unstable poles of the open-loop system $L(s) = PC$. This equation tells us that if we add up the 'log size' of the sensitivity function across all frequencies, the total area is fixed.
- This means:
  - If $|S(j\omega)|$ is small in one region (good performance),
  - Then it must be **larger elsewhere** — you can't win everywhere!
- This is known as the **"waterbed effect"**: push sensitivity down in one spot, and it pops up in another.

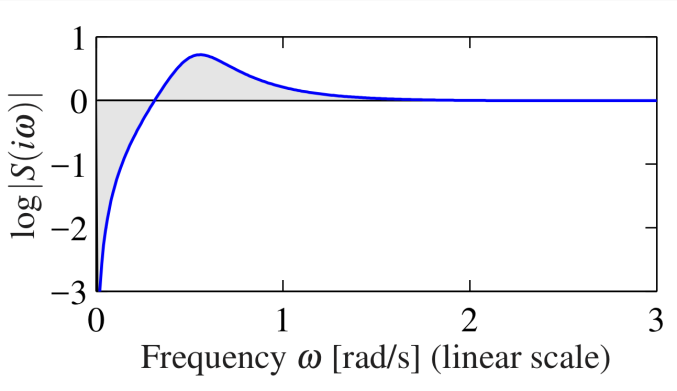
Key idea: Feedback can't eliminate all sensitivity. You must decide **where** to perform well, and where to compromise.

Interpretation: Waterbed Effect

- Bode's sensitivity integral says that for a **stable system**:

$$\int_0^\infty \log|S(j\omega)|\, d\omega = 0$$

- If $\log|S(j\omega)| < 0$ at some frequencies i.e. $|S(j\omega)| < 1$ (good disturbance rejection), then we must have $\log|S(j\omega)| > 0$, i.e. $|S(j\omega)| > 1$ for others.

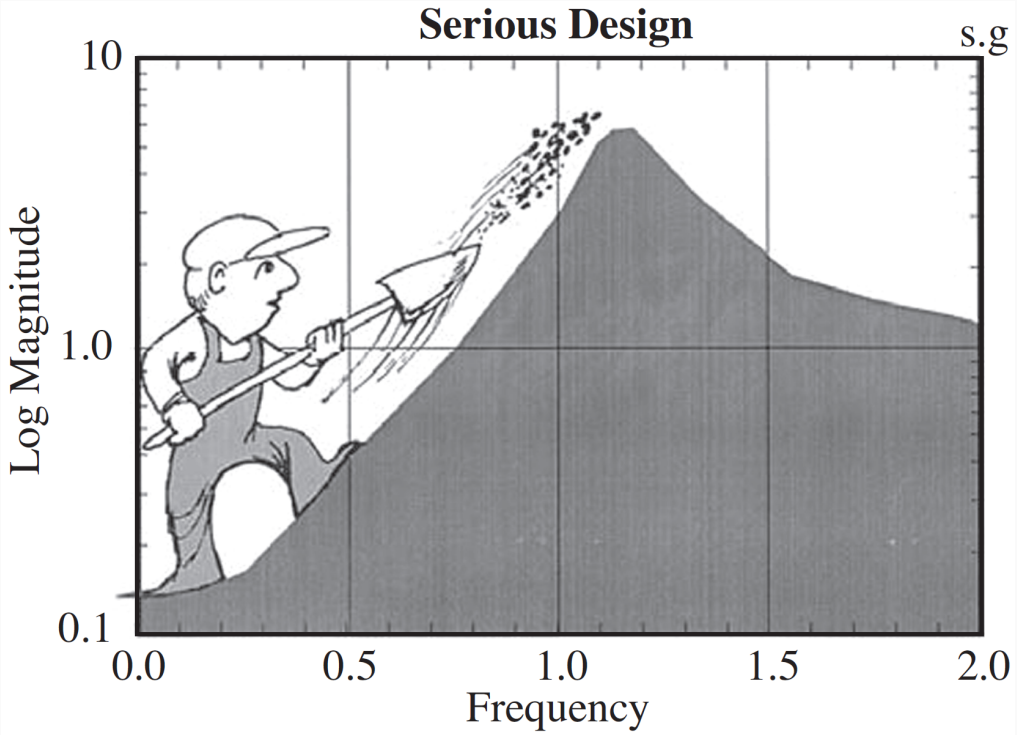


- The blue curve shows $\log|S(j\omega)|$.
- Below zero: sensitivity is low (pushed down) $\rightarrow$ good.
- Above zero: sensitivity is high (pops up) $\rightarrow$ bad.

You can't make sensitivity low everywhere — just like a waterbed, if you press down in one place, it rises in another.

Conservation Law

Dr. Gunter Stein interpreted this as a conservation law: **"conservation of dirt"**. The designer reduces sensitivity in some frequency band, but is always adding sensitivity in another. There is a risk in auto-tuning control design that you don't know where the dirt is being piled up.



Broomstick Balancing

Consider the dynamics of balancing a broomstick upright:

$$ml^2\ddot{\theta} - mgl\sin\theta = u$$

Linearising for small angles $\theta \approx 0$:

$$\ddot{\theta} - \frac{g}{l}\theta = \frac{1}{ml^2}u$$

- The transfer function becomes:

$$P(s) = \frac{1/ml^2}{s^2 - \frac{g}{l}} = \frac{1/ml^2}{(s+\sqrt{g/l})(s-\sqrt{g/l})}$$

- There is an **unstable pole** at $s = \sqrt{g/l}$, no matter how we design the controller:

$$\int_0^\infty \log|S(j\omega)|\,d\omega = \pi\sqrt{g/l}$$

Since $\sqrt{g/l}$ increases as l decreases, shorter broomsticks are harder to balance.

NASA X-29 Forward-Swept-Wing Research Aircraft

- Experimental aircraft with forward-swept wings — naturally **unstable** in flight.
- Had **right-half-plane poles**; would diverge in seconds without feedback.
- Early in the design, modern control tools masked how unstable the aircraft really was. The issue was discovered late in the design phase.
- **Unflyable by a human alone**; stability depends entirely on control.
- Real-world proof of Gunter Stein's lesson: *"Respect the unstable."*

X-29 Example: A Real-World Control Challenge

- A simplified model for the X-29 aircraft dynamics:

$$P(s) = \frac{1}{(s-6)(s+6)}$$

  $\rightarrow$ Has an **unstable pole at +6** — fast divergence without control.

- This is like balancing a broomstick of length:

$$\sqrt{g/l} = 6 \Rightarrow l = \frac{9.8}{36} \approx 0.27\,\text{m}$$

  $\rightarrow$ About the length of a standard school ruler — very difficult to stabilize.

- Maximum available control bandwidth is limited by hardware:
  - **Sensors:** 120 rad/s, **Computer:** 40 rad/s, **Actuators:** 70 rad/s, **Aerodynamics:** 100 rad/s, **Airframe structure:** 40 rad/s
- So despite fast instability, the control system is constrained to only **40 rad/s** bandwidth.

Designing feedback for unstable systems like the X-29 requires careful trade-offs; instability cannot be canceled, only managed within bandwidth limits.

Designing Available Bandwidth (Rule of Thumb)

If we take the sum of the unstable poles, we need a control bandwidth **about 12 times greater in rad/s,** or equivalently about double in Hz. **This is a useful rule of thumb for component selection and system design.**

F-14 Landing

$$P(s) = \frac{0.072(s+23)(s^2+0.05s+0.04)}{(s+1.7)(s-0.7)(s^2+0.08s+0.04)}$$

Note the pole at $+0.7$. An available bandwidth of 40rad/s here would be plenty, since $12 \times 0.7 = 8.4 \ll 40$.

Summary

- A good controller needs robustness against **disturbances and noises.** PID controllers are useful enough to provide simple, robust control solutions.
- Interpreting control design in the **frequency domain** provides insight to many important issues.
- Understanding "good" **loop transfer function shapes** allows straightforward design of controllers for relatively simple systems.
- **The sensitivity function,** and more generally the "gang of four", provide insight into the effects of external signals and model uncertainty.
- The (1) fundamental limitations of $S + T = 1$, (2) Bode's gain-phase relation, (3)the interpolation constraints, and (4) Bode's sensitivity integral help us understand which systems are difficult to control no matter what design method is used.