Recap, Intuition, Implementation and a Complete Example

Automatic Control, Basic Course, Lecture 9.5 + 10

Fredrik Bagge Carlson

December 13, 2016

Lund University, Department of Automatic Control

Content

Recap

What is control?

Making a measured or estimated signal follow a desired reference.

- Position of 3D-printer head.
- Blood glucose level in diabetes patient.
- Fuel-to-air ratio in combustion engine[1].
- Temperature in building, oven, 3D-printer.
- Amount of force applied by the scalpel when a surgical robot operates on a patients brain.

[1] If VW had hired better control engineers, they might not have had to cheat during the emission tests, who knows?

What is feedback?

- Acting in response to a measurement

- Acting in response to a measurement

Feedback is essentially the act of observing something and acting in response to it. Humans rely heavily on feedback for most things we do. Imagine the difficulties that would arise if we had no senses, no eyes, ears or sense of touch.

Why do we need feedback?

To reduce the influence of *Uncertainty*

- There are always unknown disturbances
- There are always model errors
- Get linear behavior from nonlinear components.

To reduce the influence of *Uncertainty*

- There are always unknown disturbances
- There are always model errors
- Get linear behavior from nonlinear components.

Feedback is essential to reduce the influence of disturbances and model errors. There is typically uncertainty related to both phenomena and without feedback, there would be no knowledge of the fact that something is wrong; there is a control error.

You probably do not own many electrical gadgets that do not involve feedback. Feedback is the essential principle that allows for linear amplifiers from nonlinear components, e.g., transistors.

Where does the process model come from?

- Physics often lead to differential equations $\dot{y} = -ay + bu$.
- Diff. eqs. can be transformed to a transfer function or a state-space model $sY(s) = -aY(s) + bU(s) \Leftrightarrow Y(s) = \dfrac{b}{s+a}U(s)$.
- Nonlinear models can be linearized.
- Models with known structure but unknown parameters (gray-box models).
- Generic models with parameters estimated from data (black-box models), e.g., neural networks.

Where does the process model come from?

- Physics often lead to differential equations $\ddot{y} = -ay + bu$.
- Diff. eqs. can be transformed to a transfer function or a state-space model $sY(s) = -aY(s) + bU(s) \Leftrightarrow Y(s) = \frac{b}{s+a}U(s)$.
- Nonlinear models can be linearized.
- Models with known structure but unknown parameters (gray-box models).
- Generic models with parameters estimated from data (black-box models), e.g., neural networks.

Black-box models are useful if we have no idea how to model the system. We use a model which is very flexible and used data from experiments to find parameters of the model that makes the output of the model agree with the observed data. This is called system identification and is treated in the course *System Identification* FRT041

Why do the poles determine stability?

- A rational transfer function corresponds to a differential equation. If the poles have positive real part, the solution to the differential equation blows up.

$$\frac{1}{s+a} \leftrightarrow y(t) \sim e^{-at}$$

- A state-space model is already a differential equation. In the scalar case, the scalar a is the only eigenvalue of A.

Why are most systems of lowpass character?

- Most naturally occurring systems are of lowpass character.
- Most systems can store energy, i.e, thermal, kinetic, potential.
- The stored quantity always *follows* or lags the input.
- Velocity follows force, thermal energy follows power.

- Most naturally occurring systems are of lowpass character.
- Most systems can store energy, i.e, thermal, kinetic, potential.
- The stored quantity always *follows* or lags the input.
- Velocity follows force, thermal energy follows power.

Consider the ocean being warmed up by the surrounding air. Thermal energy will be transfered from the warmer body to the colder body until they are in equilibrium. During a warm day, the air is hot. The ocean will be warmed up by the air during the entire time the air is warmer than the ocean. Let's say temp peaks at noon and starts decreasing in the afternoon. Even though temp is decreasing, the air is still hotter than the ocean, which is still increasing in temp. Not until late in the evening when the air finally gets colder than the ocean, the ocean temp has its peak, several hours after the peak in temp. This is typical of a lowpass system. The phase of the output (ocean temp) lags the input (air temp). If the air temp would vary with a much higher frequency, there would be no time for the ocean to change in temp and the output would be more or less constant. This high frequency of variation thus does not pass through the system, which only lets low frequencies pass.

What is a disturbance? - Examples

- The slope of the road changes.
- Wind blowing on airplane.
- Surrounding temperature changes.
- Diabetes patient eats carbs.
- Hackers are DDOSing the server.
- Someone pushes the robot.

What is a model error? - Origins

- Linearization of nonlinear process $\Rightarrow$ gain is wrong everywhere except for at equilibrium.
- Model estimated from noisy data $\Rightarrow$ all model parameters deviate from the true parameters.
- Model is *always* a simplification of a more complex model.

- Linearization of nonlinear process ⇒ gain is wrong everywhere except for at equilibrium.
- Model estimated from noisy data ⇒ all model parameters deviate from the true parameters.
- Model is *always* a simplification of a more complex model.

Models are *always* simplifications. Everything we observe is the outcome of a vast amount of interactions on the atomic level which we can never hope to model accurately, let alone simulate.

For example, simple friction models such as the Coulomb model are empirically derived and easy to use, but careful study shows that friction is an extremely complicated phenomenon arising due to electric forces between particles.

Why do we need a robust design?

Robust design implies small maximum value of the sensitivity function, large gain margin and large phase margin.

Once again, *uncertainties* are the villain.

- The model is wrong. If the gain is wrong, we might run into instability if we are close to the gain margin.
- If the time constant is wrong, we might not get the desired dampening of oscillations.

A robust design can tolerate large model errors!

What is feedforward?

- Feedback requires something to happen before a reaction occurs, i.e., an error has to arise.
- If a known event is about to occur, and this event is known to create an error if not cared for, we may use feedforward to negate the effect of the event.

Example - Diabetes

Eating carbs is known to create a rise in blood glucose. A diabetes patient (or an automatic glucose controller) may use this knowledge and take insulin at the same time as the food, eliminating the rise in blood glucose[2].

[2] The patient needs a good model of the blood glucose response from carbs as well as the response of glucose to insulin for the elimination to be complete.

What is feedforward?

- Feedback requires something to happen before a reaction occurs, i.e., an error has to arise.
- If a known event is about to occur, and this event is known to create an error if not cared for, we may use feedforward to negate the effect of the event.

Example - Diabetes

Eating carbs is known to create a rise in blood glucose. A diabetes patient (or an automatic glucose controller) may use this knowledge and take insulin at the same time as the food, eliminating the rise in blood glucose[2].

[2] The patient needs a good model of the blood glucose response from carbs as well as the response of glucose to insulin for the elimination to be complete.

Humans also rely heavily on feedforward. An action such as throwing a ball requires a very fast movement, during which the limited bandwidth in the human motor control system is insufficient to achieve the desired motion. Instead, we rely on a carefully tuned internal model of our muscles and kinematics and make use of feedforward from this model to accomplish the task. Feedforward models in humans take long time to train, throwing a ball like a professional probably takes several years of training to master.

Prerequisites for feedforward

For feedforward to be effective:

- Knowledge of the disturbance.
- (Inverse) Model of the dynamics involved. The effect of the disturbance is determined by the dynamics.

For feedforward to be effective:

- Knowledge of the disturbance.
- (Inverse) Model of the dynamics involved. The effect of the disturbance is determined by the dynamics.

Feedforward obviously requires accurate knowledge of the disturbance we want to account for. If we see changes in the reference value as a disturbance, this is fulfilled. In this case we only need an accurate dynamics (inverse) model to eliminate the effect of the disturbance. If the disturbance is an auxiliary signal, we need an accurate measurement of this.

A typical linear model can be written like $Y(s) = G(s)U(s)$. For feedforward, we require the **inverse model** $U(s) = G^{-1}(s)Y(s)$, i.e., a model of the input, given the output. If we know the output Y we want, the inverse model tells us what input, or control signal U, to apply to get the desired output.

Implementation

How do we implement a controller?

This lecture

- We work in continuous-time domain, the computer works in discrete time.
- Continuous-time signals must be *sampled* by the computer.
- Continuous-time models and controllers must be *discretized*.

This lecture

- We work in continuous-time domain, the computer works in discrete time.
- Continuous-time signals must be *sampled* by the computer.
- Continuous-time models and controllers must be *discretized*.

The subjects of sampling and discretization are treated in great detail in the course FRTN01 *Real-Time Systems*.

Sampling

- The computer uses an analog-to-digital (AD) converter to measure the value of a signal $y(t)$ at discrete points in time.
- This typically occurs at uniform intervals, called the sample interval h.
- The result is a sequence of numbers $y_k = y(kh)$.

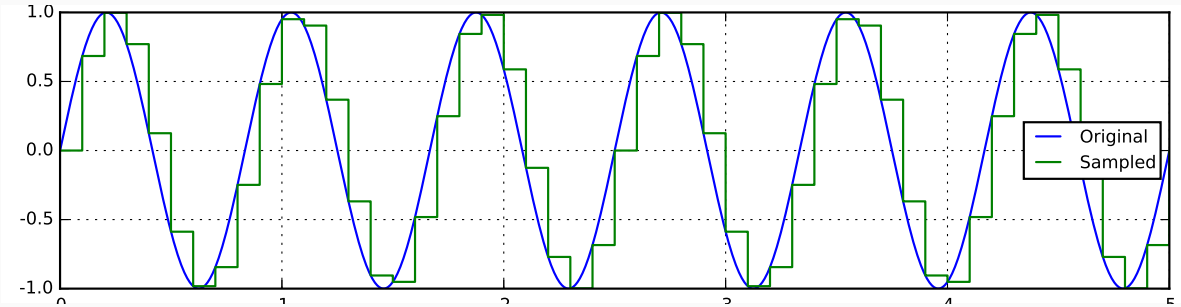


Figure 1: An example of a sampled signal. The information between the samples is lost.

Aliasing

- The sample frequency $f_s = 1/h$ determines the highest frequency we can reconstruct.
- The Nyquist[3] frequency $f_N = f_s/2$ is the highest frequency we can reconstruct.
- Frequencies $f > f_N$ will be *aliased*.
- All measured signals must be lowpass filtered before sampling (anti-alias filter)[4].

[3] Harry Nyquist, Swedish born electronic engineer who made important contributions to control and communication theory.

[4] These filters are often implemented with analog circuits.

Aliasing - Example

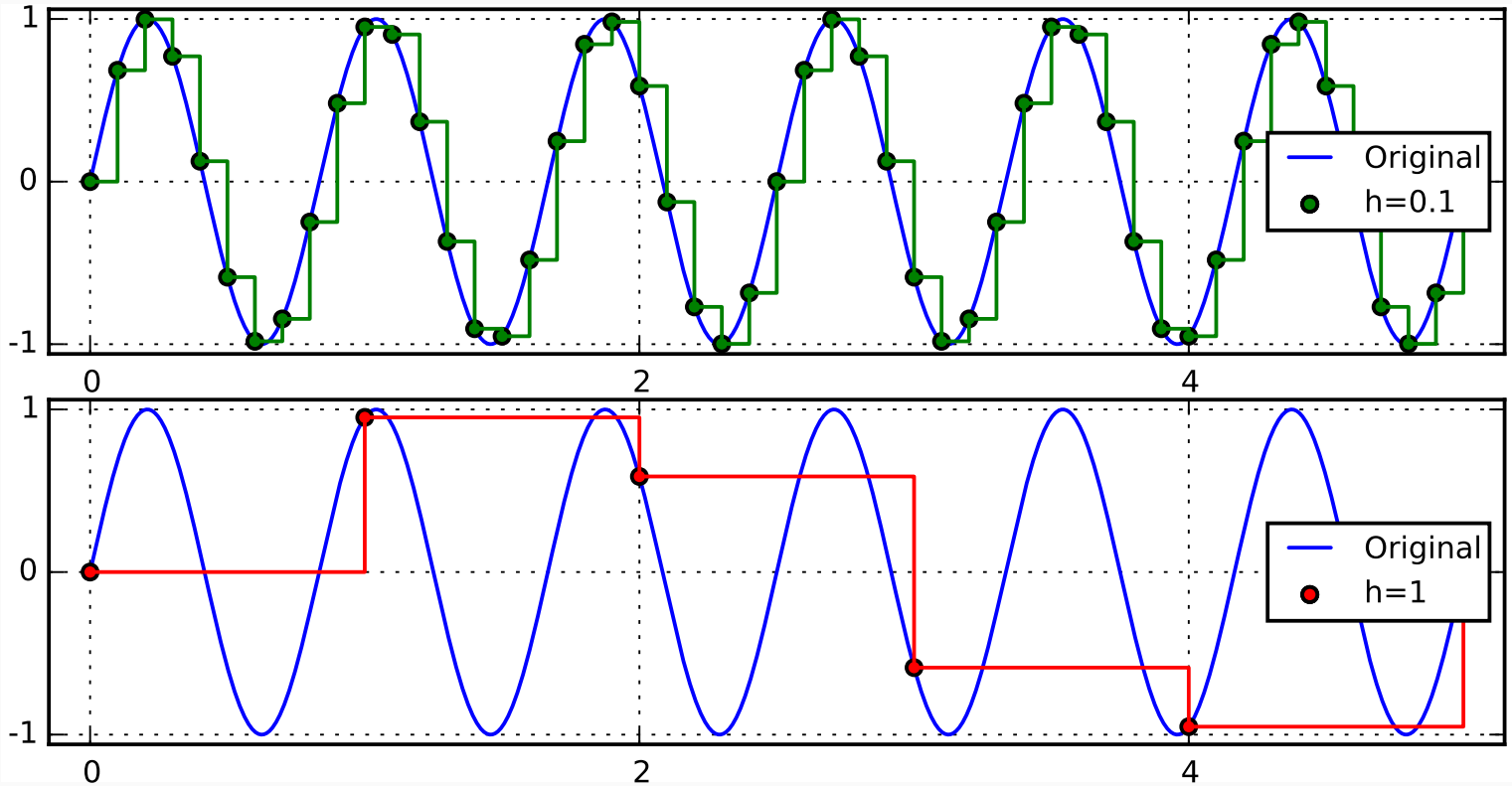


Figure 2: An example of a signal sampled with a frequency above and below the Nyquist frequency.

Aliasing - Example

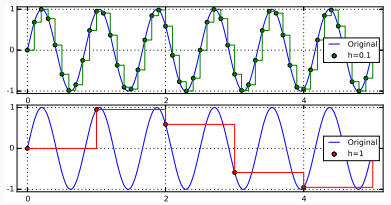


Figure 2: An example of a signal sampled with a frequency above and below the Nyquist frequency.

Alias: Noun - a false or assumed identity

The term aliasing refers to the fact that a signal with a frequency $f > f_N$ appears as a lower frequency f_a in the sampled signal. The lower frequency f_a is called an alias of the original frequency f.

Note how the act of sampling with zero-order-hold introduces a phase lag for high frequencies (the sampled signal appears to lag slightly behind the original signal). This topic is treated in FRTN01 *Real-Time Systems*

Aliasing - Example

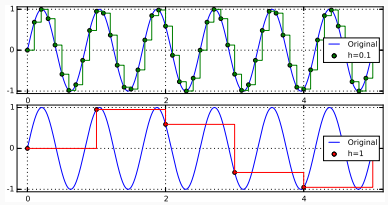


Figure 2: An example of a signal sampled with a frequency above and below the Nyquist frequency.

Julia code for aliasing fig

```
using Plots
t = 0:0.001:5
t1 = 0:0.1:5
t2 = 0:5
f = 1.2
y = sin(2pi*f*t);
y1 = sin(2pi*f*t1);
y2 = sin(2pi*f*t2);
plot(t,y, lab="Original", layout=(2,1), subplot=1, c=:blue)
scatter!(t1,y1, lab="h=0.1", c=:green)
plot!(t1,y1, l=:step, lab="", c=:green, subplot=1)
plot!(t,y, lab="Original", subplot=2, c=:blue)
scatter!(t2,y2, lab="h=1", c=:red, subplot=2)
plot!(t2,y2, l=:step, lab="", c=:red, subplot=2)
```

Discretization of the PID controller

The PID controller with practical modifications:

$$U = K\left((bR - Y) + \frac{1}{sT_i}E - \frac{sT_d}{1 + sT_d/N}Y\right)$$

Implemented in the computer as the sum of three terms[5]:

$$u_k = P_k + I_k + D_k$$

[5]Subindex k refers to the time index. $u_k = u(kh) = u(t)|_{t=kh}$

Discretization of the PID controller - P

$$u_k^p = K(br_k - y_k)$$

The P-part does not contain any dynamics and is thus straightforward to implement

```
P = K*(b*r - y)
```

Discretization of the PID controller - I

$$u_k^i = \frac{K}{T_i} \int_0^{t=kh} e(t)dt$$

The I-part is an integral, we approximate it with a sum

Naive implementation:

```
I = K/Ti*h*sum(e[1:k])
```

Smarter implementation:

```
I = I + K/Ti*h*e
```

Discretization of the PID controller - I

$$u_k^i = \frac{K}{T_i}\int_0^{t=kh} e(t)dt$$

The I-part is an integral, we approximate it with a sum

Naive implementation:

```
I = K/Ti*h*sum(e[1:k])
```

Smarter implementation:

```
I = I + K/Ti*h*e
```

The sample time h enters into the discrete approximation. This is easy to understand since we approximate the integral, which is the area under a curve, as the sum of small rectangular areas. The width of each rectangular area is h and the height is e_k.

Discretization of the PID controller - D

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

$$u^d + \frac{T_d\dot{u}^d}{N} = -KT_d\dot{y}$$

$$u_k^d + \frac{T_d}{N}\frac{u_k^d - u_{k-1}^d}{h} = -KT_d\frac{y_k - y_{k-1}}{h}$$

$$u_k^d = \frac{T_d}{T_d + Nh}u_{k-1}^d - \frac{KT_dN}{T_d + Nh}(y_k - y_{k-1})$$

Implementation

```
D = T_d/(T_d + Nh)*D -(K*T_d*N)/(T_d + N*h)*(y_k - y_k1)
y_k1 = y_k
```

Discretization of the PID controller - D

$$D(s) = -K\frac{sT_d}{1+sT_d/N}Y(s)$$
$$u^d + \frac{T_d\dot{u}^d}{N} = -KT_d\dot{y}$$
$$u_k^d + \frac{T_d}{N}\frac{u_k^d - u_{k-1}^d}{h} = -KT_d\frac{y_k - y_{k-1}}{h}$$
$$u_k^d = \frac{T_d}{T_d+Nh}u_{k-1}^d - \frac{KT_dN}{T_d+Nh}(y_k - y_{k-1})$$

Implementation

```
D = T_d/(T_d + Nh)*D -(K*T_d*N)/(T_d + N*h)*(y_k - y_k1)
y_k1 = y_k
```

If we define the derivative as

$$f'(x) = \lim_{h\to 0}\frac{f(x+h) - f(x)}{h}$$

we immediately see a way of approximating the derivative; instead of the limit, let it suffice with h being small (the sample time). Instead of a derivative $\frac{f(x+h) - f(x)}{h}$ is called a *finite difference* approximation of the derivative. If $f = y$ and $x = kh$, we get the approximation on the previous slide.

How well does the discrete PID perform?

If the sample time is high enough, there is little performance lost in discretization.

Rule of thumb, allow for ten samples during the dominant time-constant of the process or closed-loop system, whichever is faster.

Full PID controller

```
y = yIn.get();
e = yref - y;
D = ad * D - bd * (y - yold);
v = K*(beta*yref - y) + I + D;
if (mode == auto) u = sat(v,umax,umin)
else u = sat(uman,umax,umin);
uOut.put(u);
I = I + (K*h/Ti)*e + (h/Tr)*(u - v);
if (increment)
uinc = 1;
else if (decrement)
uinc = -1;
else uinc = 0;
uman = uman + (h/Tm) * uinc + (h/Tr) * (u - uman);
yold = y;
```

ad and bd are precalculated parameters given by the backward difference approximation of the D-term.

Full PID controller

```
y = yIn.get();
e = yref - y;
D = ad * D - bd * (y - yold);
v = K*(beta*yref - y) + I + D;
if (mode == auto) u = sat(v,umax,umin)
else u = sat(uman,umax,umin);
uOut.put(u);
I = I + (K*h/Ti)*e + (h/Tr)*(u - v);
if (increment)
uinc = 1;
else if (decrement)
uinc = -1;
else uinc = 0;
uman = uman + (h/Tm) * uinc + (h/Tr) * (u - uman);
yold = y;
```

ad and bd are precalculated parameters given by the backward difference approximation of the D-term.

The code above is included as an example of how a practical implementation of a PID controller in JAVA might look. It includes anti-windup, filtering, saturation, bumpless-transfer and minimizes delay between measurement and control signal output. All these topics are treated in FRTN01 *Real-Time Systems*.

Ball and beam

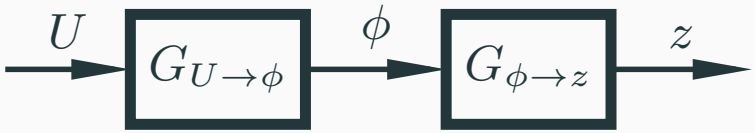
Ball and beam - Contents

- Modeling - Intuition
- Control design - Cascade PID
- Other approaches

Two transfer functions

Beam angle and ball position measurable. We can divide the model into two parts.

- $G_{U\to\phi}$ From input voltage to beam angle.
- $G_{\phi\to z}$ From beam angle to ball position.



$G_{U \to \phi}$

- A voltage over a motor induces a current through the motor.
- The current gives rise to a torque, τ.
- Newtons second law says $\tau = J\dot{\omega}$. Compare with linear case: $f = ma$.
- Newtons second law implies an integrator between torque and angular velocity. Due to back-EMF and friction, this pole (*) is moved into the left half-plane.
- There is one pure integrator between ω and ϕ ($\dot{\phi} = \omega$).
- Due to an internal controller, the time-constant (*) can be considered very fast (disregarded). Experiments show that this assumption is valid up to $\omega \approx 10\text{rad/s}$.
- We are left with one integrator and an unknown gain $\frac{K}{s}$.

$G_{U \to \phi}$ - Gain

The gain can be determined experimentally, e.g., through a step-response experiment or frequency-response experiment.

One such experiment on the real process establishes $K \approx 4.5$

The book derives this model from first principles.

We will derive it from intuition.

$G_{\phi \to z}$

- An angle $\phi \neq 0$ causes the ball to accelerate $\phi \to \ddot{z}$.
- Acceleration is proportional to $\sin \phi$, for small ϕ, $\sin \phi \approx \phi$
- $\phi \sim \ddot{z} \xrightarrow{\mathcal{L}} \Phi(s) \sim s^2 Z(s)$
- We can therefore conclude $Z(s) = \frac{K}{s^2}\Phi(s)$ for some gain K.
- In lecture notes, K is *almost* derived from first principles. They still lack the conversion factor between measured voltage and position $\to$ have to make an experiment.
- One could equally well determine the entire model from the same experiment, with the simple reasoning above.

$G_{\phi \to z}$ - Gain

$Z(s) = \frac{K}{s^2}\Phi(s)$ The gain K is determined from an experiment, $K \approx 10$.

The experiment further shows that the model is valid up to about
$\omega \approx 5\text{rad/s}$

Full model

The full model is given by

$$Z(s) = G_{\phi \to z} G_{U \to \phi} U(s) \tag{1}$$

$$Z(s) = \frac{10}{s^2} \cdot \frac{4.5}{s} U(s) \tag{2}$$

$$Z(s) = \frac{45}{s^3} U(s) \tag{3}$$

$$\tag{4}$$

The full model is thus a triple integrator, phase is -270°everywhere!

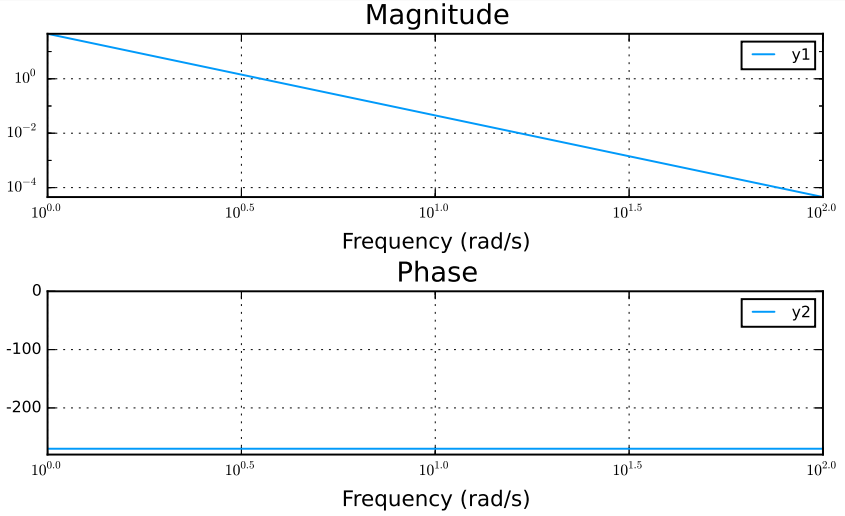


Figure 3: Bode plot for the process

Model limitations

Important to keep in mind!

- Ball jumps if acceleration > 9.8
- Centrifugal forces not considered
- Model validity ranges $\omega < 10$rad/s, $\omega < 5$rad/s

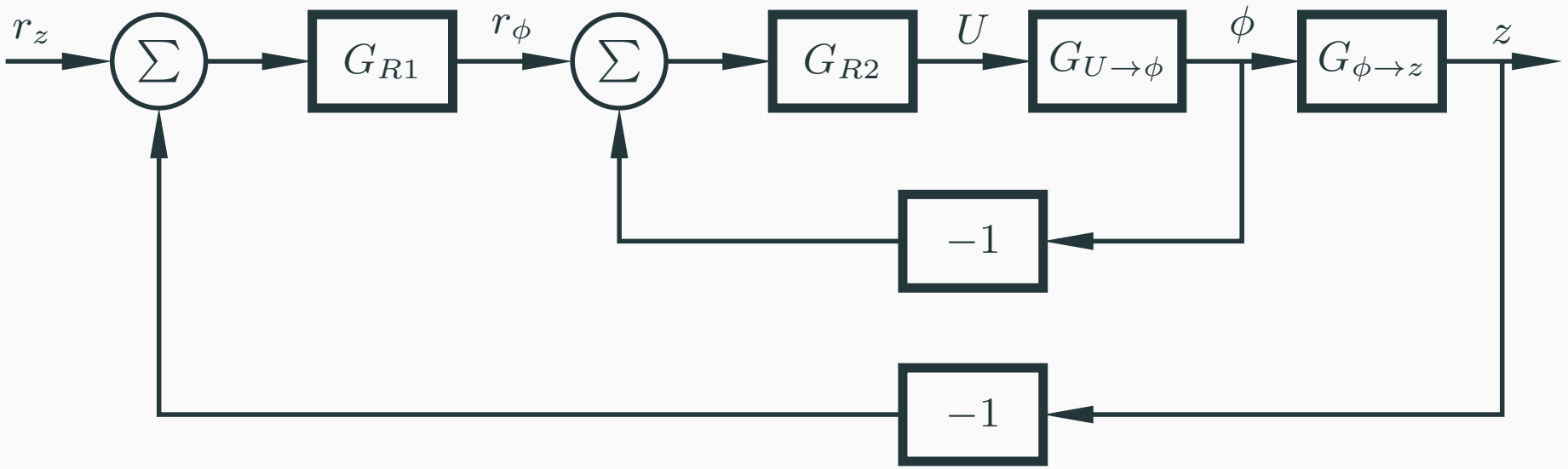
Control design - Methodology

In practice, one typically chooses the methodology one is *comfortable* with.

- In this case, phase is -270°everywhere.
- PID controller can only lift the phase by $< 90°$.
- Something more sophisticated is needed.

Cascaded PID

Since we have a measurement of the beam angle, we can design a controller for the beam angle.



This controller will lift the phase for low frequencies by 90°!

Angle controller

The process is given by $G_{U\to\phi} = \dfrac{4.5}{s}$

A simple P-controller $G_{R2} = K_2$ will do the trick!

$$G_2 = \frac{4.5K_2}{s + 4.5K_2} = \frac{1}{1 + sT_2}$$

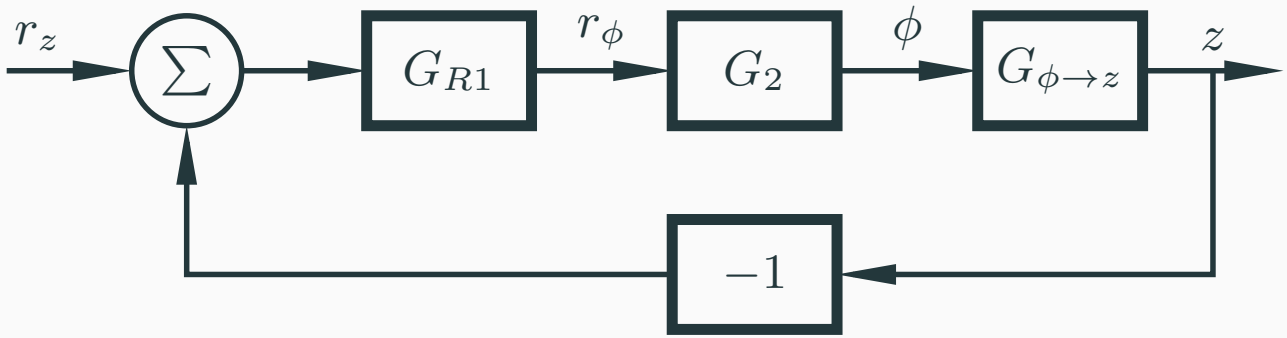
We choose T_2 with the model limitation in mind!

- Model $G_{U\to\phi}$ is valid up to $\omega \approx 10$rad/s.
- Choose $\omega_c = 10$ rad/s

$$\Rightarrow K_2 = 10/4.5 \approx 2.2$$

Simplified block diagram

With the angle controller, the outer controller sees the following system



$$G_{P1} = \frac{1}{1 + sT_2} \cdot \frac{10}{s^2}$$

Simplified bode plot

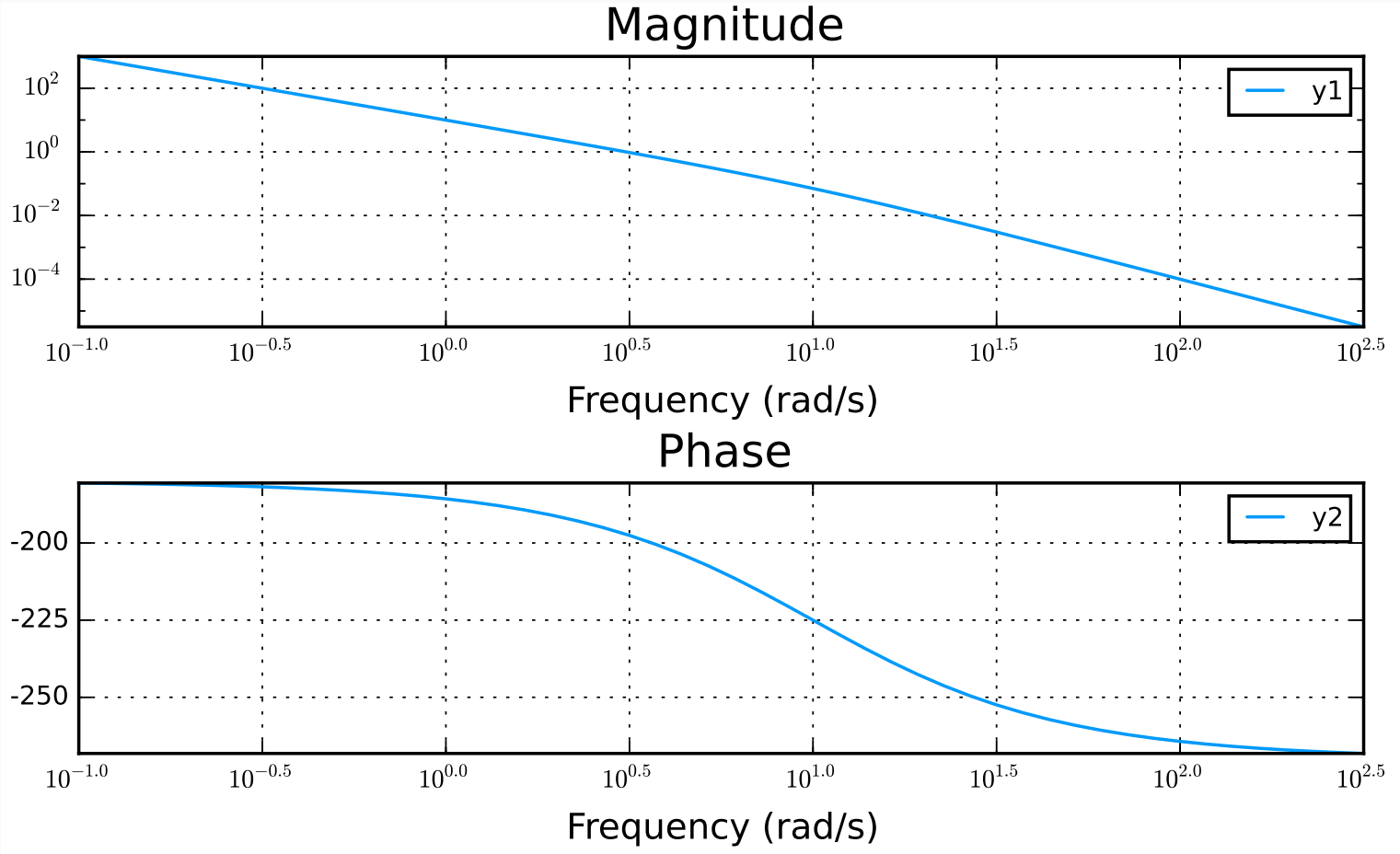


Figure 4: Bode plot for the process with internal angle controller, $G_{P1} = \frac{1}{1+sT_2} \cdot \frac{10}{s^2}$

Design of position controller

$$G_{P1} = \frac{1}{1 + sT_2} \cdot \frac{10}{s^2}$$

Process has third order dynamics $\Rightarrow$ poles can not be placed arbitrarily.

Write controller on series form

$$G_{R1} = K \left(1 + \frac{1}{sT_i}\right) \frac{1 + sT_d}{1 + sT_d/10}$$

It is composed of a lead link and a lag link!

Design of position controller - Lead link Strategy

- $N = 10$ is given $\rightarrow$ 55°phase advance at ω_c.
- Lag link will lower phase at ω_c by 6°.
- Controller phase at $\omega_c = 55 - 6 = 49$.
- Process phase at $\omega_c = -180 - \arctan \omega_c T_2 = -180 - \arctan 0.1\omega_c$
- Phase margin $\varphi_m = 180+$ process phase $+$ controller phase.
  $\varphi_m = 180 + 49 - 180 - \arctan 0.1\omega_c = 49 - \arctan 0.1\omega_c$
- Never better phase margin than 49°!
- $\omega_c = 2\text{rad/s} \Rightarrow \varphi_m = 38$

Design of position controller - Lead link

The value of ω_c determines T_d and K

- $b\sqrt{N} = \omega_c \Rightarrow T_d = \sqrt{10}/2 \approx 1.6$
- Choose K to make $|G_0(i\omega_c)| = 1$

  $|G_0(i\omega_c)| = |G_{R1}(i\omega_c)G_{P1}(i\omega_c)| = K\sqrt{N}\dfrac{1}{\sqrt{1+\omega_c^2 T_2^2}}\dfrac{10}{\omega_c^2}$

  $K \approx 0.13$

Design of position controller - Lag link

Choose the integral time T_i such that the phase loss at ω_c is $6°$($a = 0.1\omega_c$ rule of thumb from lecture 11).

The parameter a in the lag link corresponds to $\frac{1}{T_i}$ in the PID controller.

$$a = 0.1\omega_c \Rightarrow T_i = \frac{1}{a} = \frac{1}{0.1\omega_c} = \frac{1}{0.1 \cdot 2} \approx 5.0$$

Final bode plot, G_0

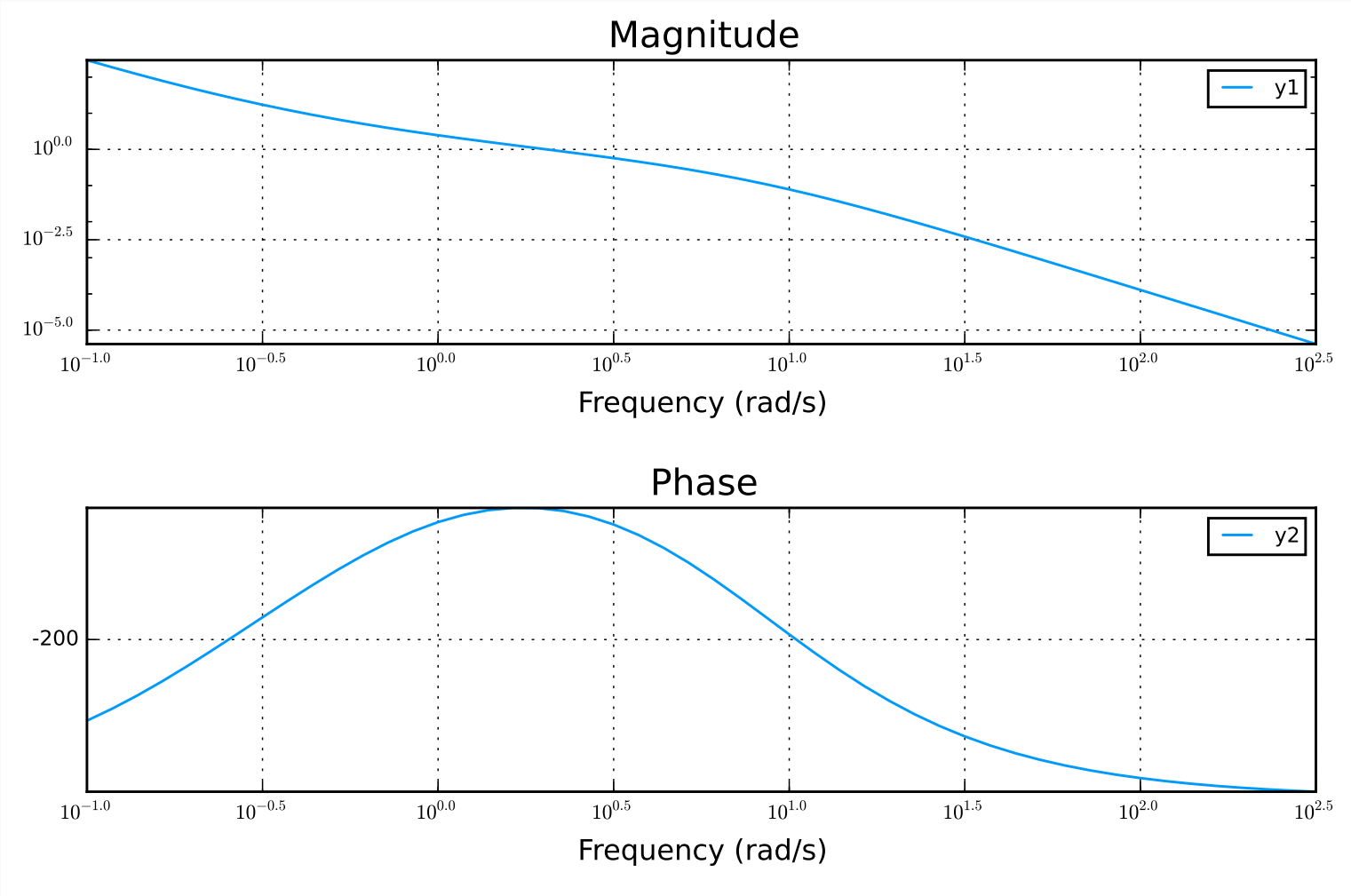


Figure 5: Bode plot of the open loop system

Final bode plot, G_0

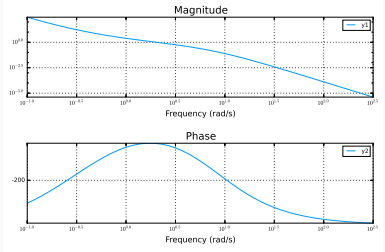


Figure 5: Bode plot of the open loop system

Here we clearly see that the PID controller has lifted the phase with the peak exactly where we have the crossover frequency.

Other approaches

- Could do pole placement with feedback from estimated states $\rightarrow$ simple design procedure.
- Could do P-control of angle and pole placement for the position.
- Many other approaches possible.