Introduction to Automatic Control

7 - PID Design

University of BRISTOL

Contents

Intro

- So far this term we have developed a mathematical toolbox
  - Week 2: Block Diagrams
  - Week 3: Stability Analysis
  - Week 4: Performance Analysis
  - Week 5: Root Locus (i.e. pole placement)
  - Week 6: Bode Plots (i.e. sculpting the frequency response)

- We will now see how these tools can be used to **design** controllers.

This Lecture

1. **Design requirements:** how to convert design requirements in to root locus/bode plot characteristics.

2. **PID Controller Design**
   - Root Locus
   - Bode Plot
   - Implementation Considerations

- In these slides/videos we will mainly focus on the **concepts**
  - The real process of controller design is hugely **problem specific** and **iterative**.
  - Ideas will be reinforced with problems on this week's problem sheet.

- Next Lecture: Lead/lag 'Compensator' Design

Contents

Design Requirements

- The two tools we have focussed on for system analysis are **root locus** and **bode plots**.

- We need to be able to recognise how to convert system **requirements** into desired characteristics of the root locus and bode plot.

Example: Design a controller so that the step response of the closed loop system has:

Steady-state error $\leq 5\%$.
Settling time $\leq 2s$.
All oscillations faster than 10 Hz.

- How do we interpret these requirements?

Steady-state Error

- We have previously covered how to analyse the steady-state error of a closed-loop control system (Week 4).

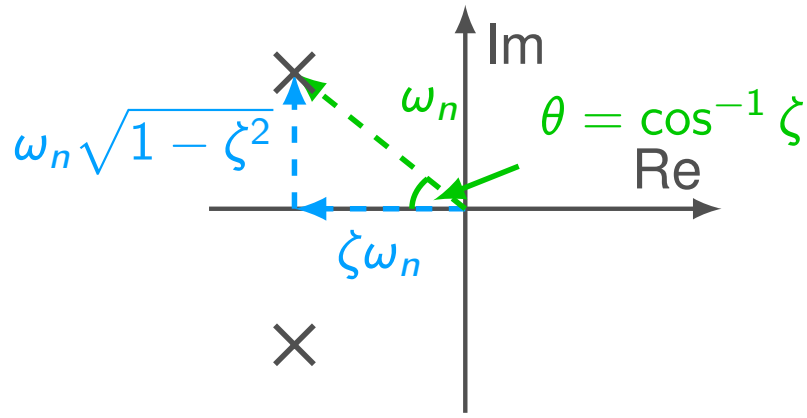
SS Error Analysis

1. Given system TF $G(s)$ and control TF $C(s)$
2. Choose input signal $u(t)$
3. Take Laplace transform for $U(s)$
4. Use final value theorem on $\frac{1}{1+C(s)G(s)}U(s)$

- We will not go over this again in detail, but this is a reminder as this approach will be needed in problem sheet.

Root Locus

- Recall the significance of pole locations we derived:



- Root locus is fundamentally a **time domain** design technique.
- Can be used to determine
  - Rise time.
  - Natural frequency.
  - Overshoot/damping ratio.
  - Settling time.
- How do requirements on these values translate into **feasible regions** of the s-plane?

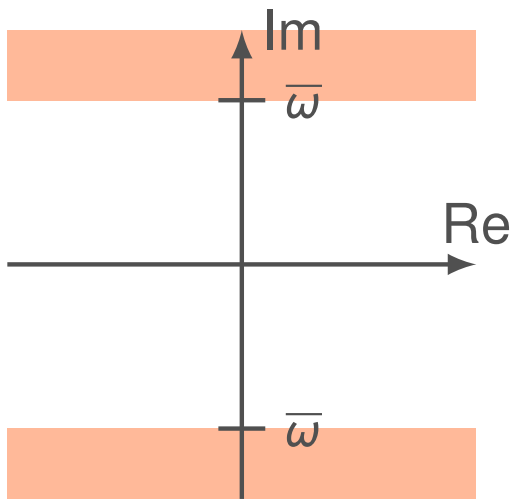
Natural Frequency

- We may have a requirement such as

Natural frequencies of the system must be within $\omega_d \leq \overline{\omega}$

  - ► This could be to avoid the system resonating with other systems.

- The **feasible region** in the s-plane is given by:



- This is **independent** of whether the poles are **dominant**.

Overshoot

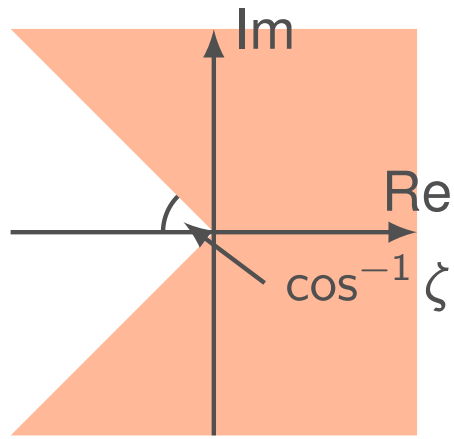
- We may have a requirement such as

Overshoot must be less than 25%

  - Overshoot could damage the system.

- Overshoot of $\leq 25\%$ corresponds to damping ratio of $\zeta \geq 0.4$ $\implies \theta \leq \cos^{-1}(0.4)$

- The **feasible region** in the s-plane is given by:



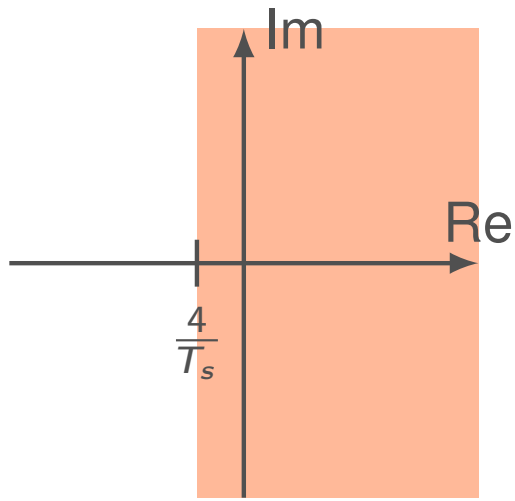
- This is **dependent** on whether the poles are **dominant**.

Settling Time

- We may have a requirement such as

> The step response transients must settle in $\leq$ 2s

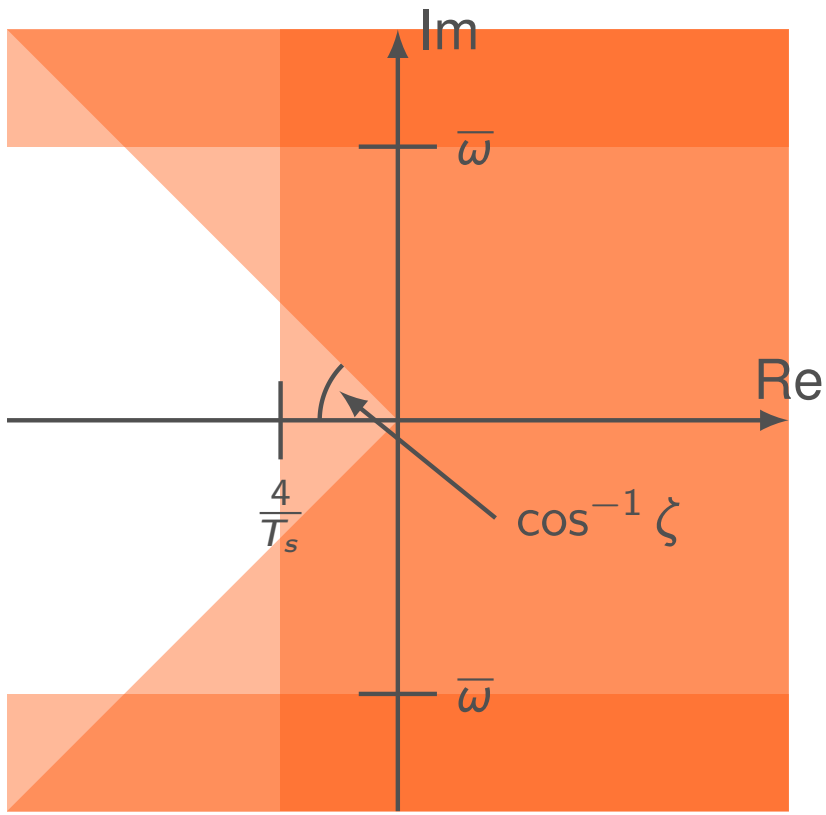
- The **feasible region** in the s-plane is given by:



- This is (somewhat) **dependent** on whether the poles are **dominant**.

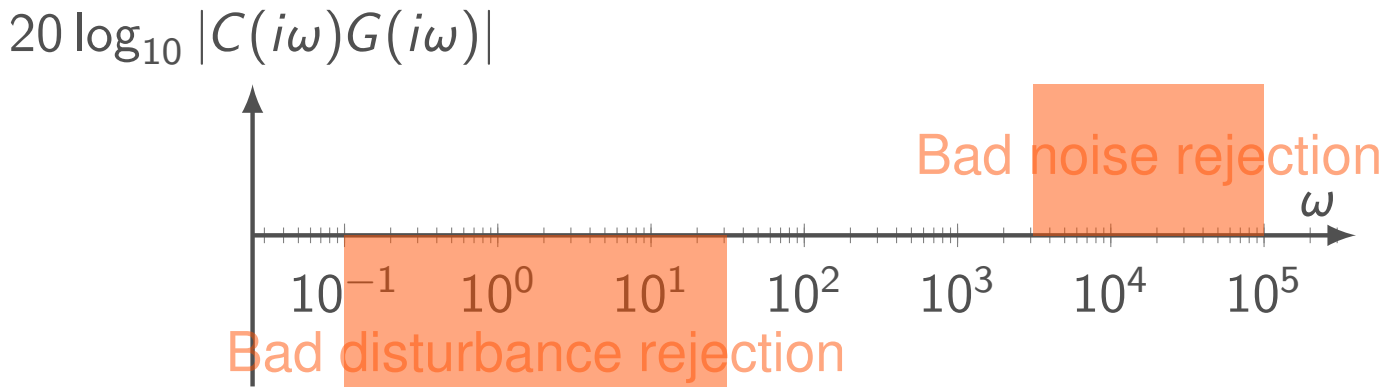
Settling Time

- These three requirement together result in the feasible region for the closed loop system poles:



Bode Plot

- We have already explored problematic areas of the **open-loop** bode plot:



- Bode plot also provides **Gain and Phase Margins**

- Many more possible requirements could be needed:
  - E.g. flat response for a range of frequencies
  - For the purposes of this course, we are only going to consider the aforementioned criteria.

Bode Plot vs Root Locus

- Bode plots and root locus have relative advantages for control system design.

- For example
  - Root locus can be used to infer transient response.
  - Bode plot can be applied directly to measured data.
  - Bode plot provides information on robustness.
  - Both can be used to infer closed-loop stability.

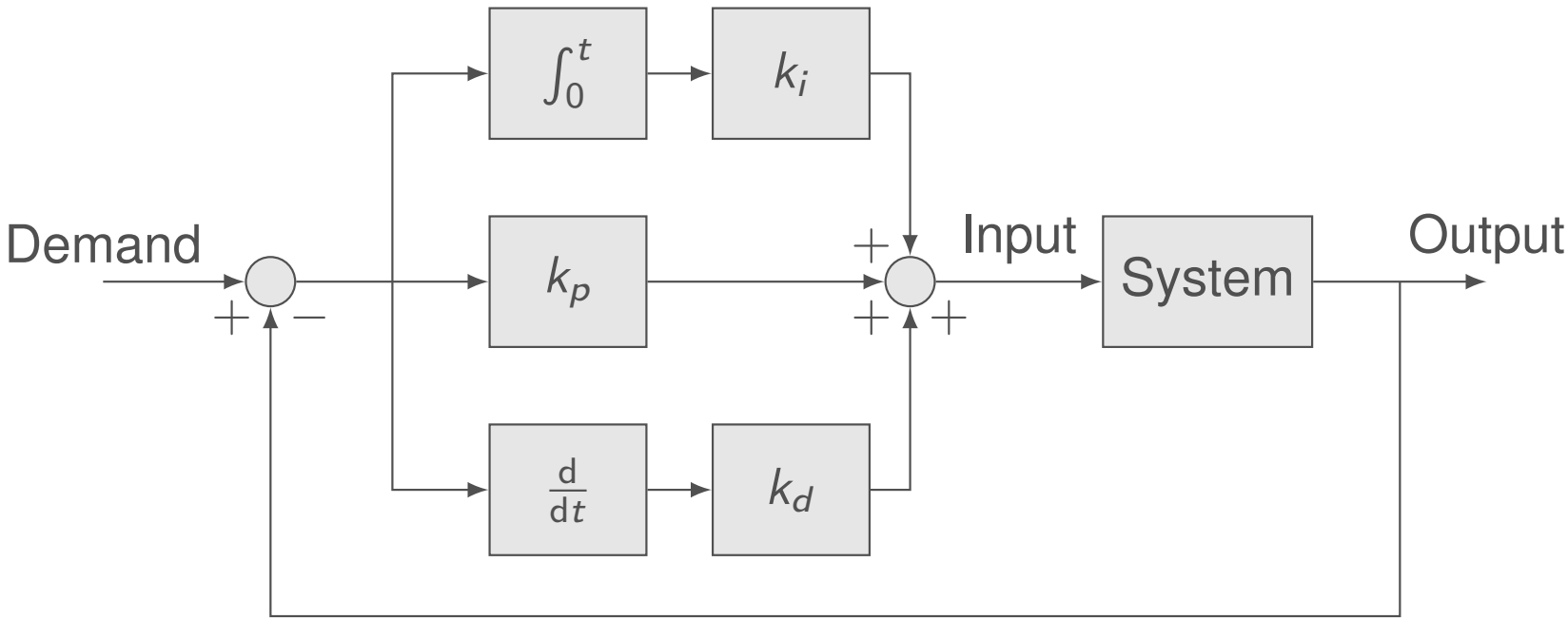
- Generally most powerful when used **together**.

- We will next look at the **intuition** behind PID design using root locus, and then the bode plot.

Contents

PID Control

- Recall the structure of a PID controller:



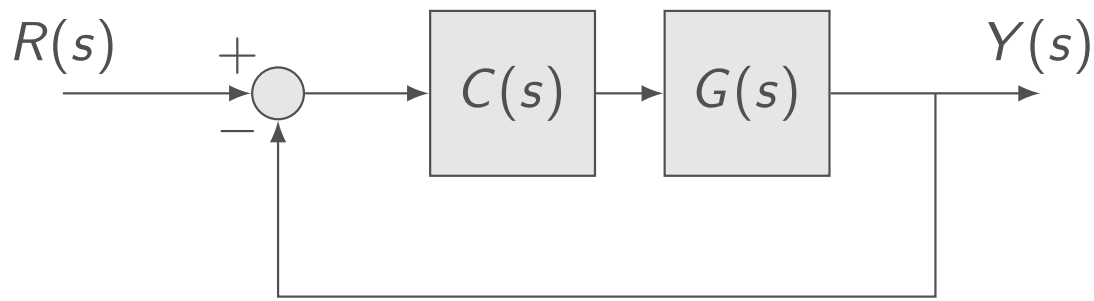
- Intuition developed in week 2:
  - k_p = ‘Stiffness’
  - k_d = ‘Damping’
  - k_i reduces (eliminates) steady-state error.

PID Control s-domain

- Also recall that PID controller is **linear** and **time invariant**, so can be represented using the Laplace transform as

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

in the equivalent representation[1]



[1] Note that the sensor dynamics are not explicitly included in either of these representation.

Root Locus Representation

- Fundamentally, the process of designing a PID controller is choosing the three gains.

- The **key** to interpreting a PID controller using root locus is to recognise that the PID controller is also equivalent to

$$C(s) = \frac{K(s-\lambda_1)(s-\lambda_2)}{s}$$

  where

$$k_d = K \quad k_p = -K(\lambda_1 + \lambda_2) \quad k_i = K\lambda_1\lambda_2$$

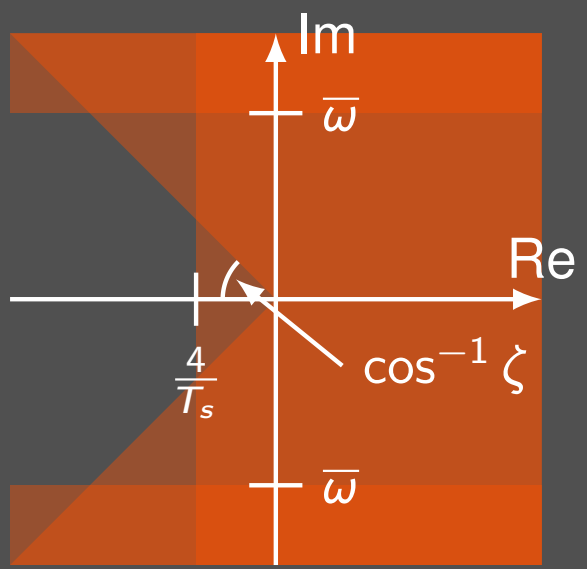
- Therefore, designing a PID controller with root locus is equivalent to
  1. Placing an open-loop pole at 0
  2. **Choosing** two open-loop zeros as λ_1 and λ_2
  3. Choosing gain K
  4. Determining PID gains from λ_1, λ_2, and K.

Illustrative Example

- We will demonstrate the intuition behind this idea using the second order system

$$G(s) = \frac{1}{0.02s^2 + 0.3s + 1}$$

and will place **closed-loop** poles in the feasible region:



Step 1: Place Open-loop Pole at 0

- Recall that the first step of designing the controller is placing an open-loop pole at zero:

$$C(s)G(s) = K\frac{(s-\lambda_1)(s-\lambda_2)}{s}\frac{1}{0.02s^2+0.3s+1}$$

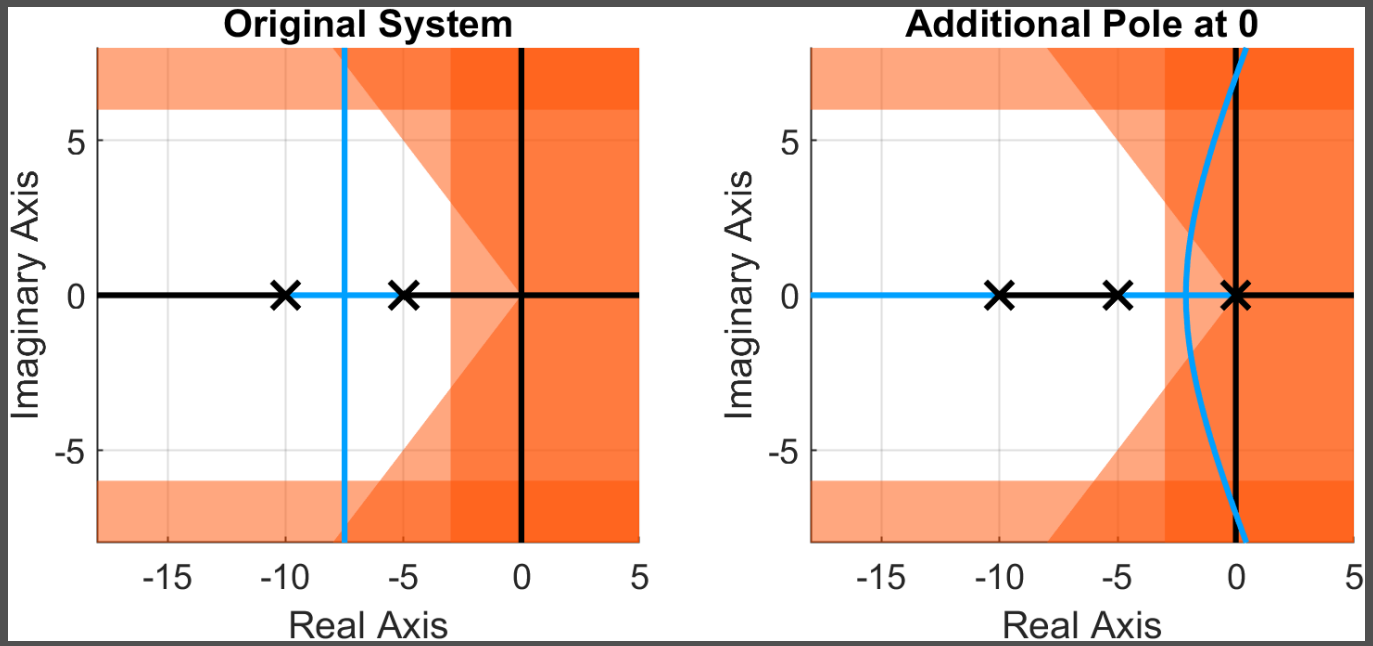
- Recall that root locus is plot of roots of

$$1 + K\hat{C}(s)G(s)$$

for $K \in (0, \infty)$

Step 1 Root Locus

- Root locus for original system and with additional pole at 0:



- After additional pole:
  - Three asymptotes
  - No gain K can achieve desired performance criteria.

Step 2. Choose λ_1 and λ_2

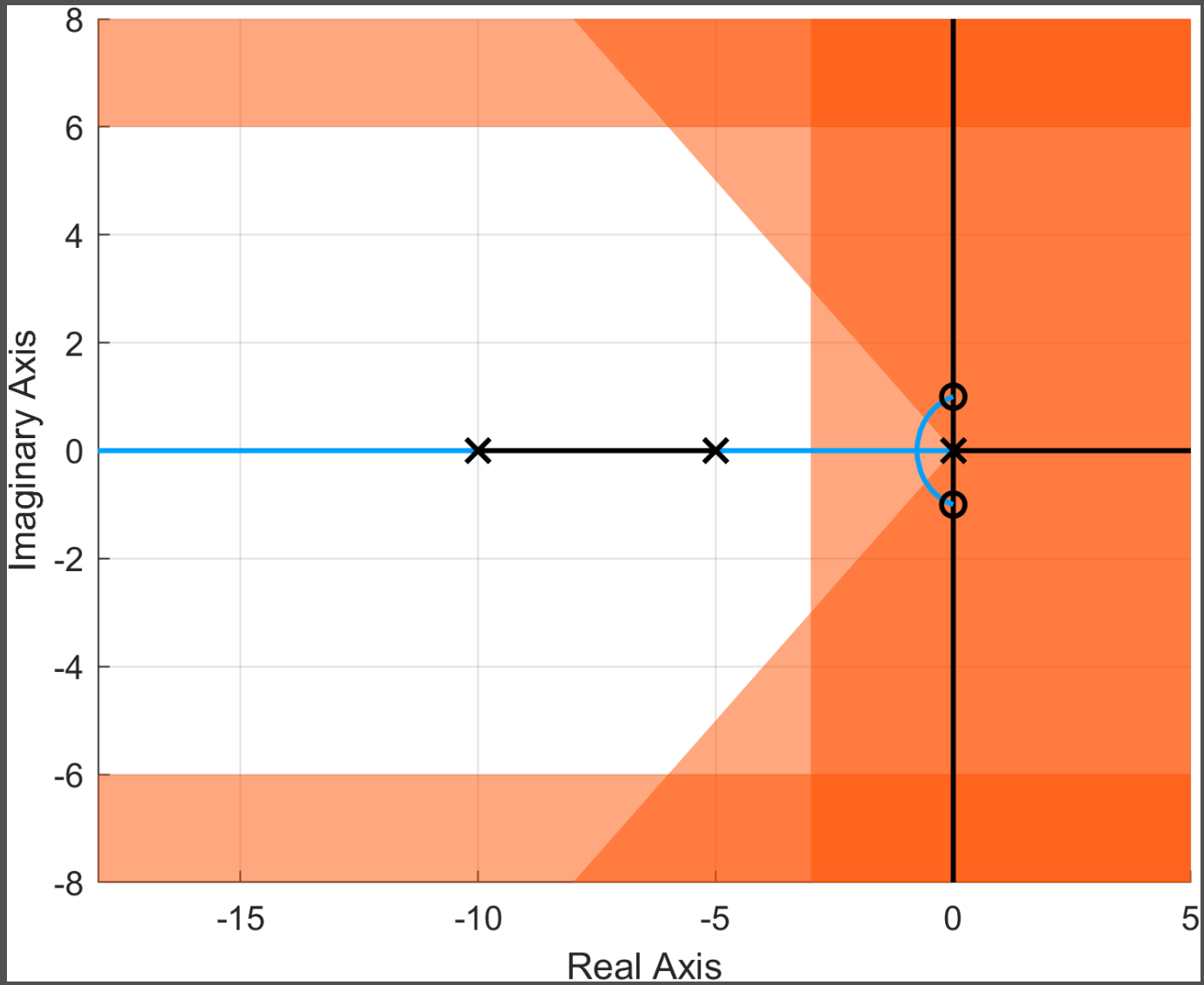
- We are now able to choose two open-loop zeros in λ_1 and λ_2

$$C(s)G(s) = K\frac{(s-\lambda_1)(s-\lambda_2)}{s}\frac{1}{0.02s^2+0.3s+1}$$

- Recall from root locus method:
  - Closed-loop poles move to open-loop zeros as $K \to \infty$
  - Three asymptotes reduced to one (at $-\infty$) with two additional zeros.

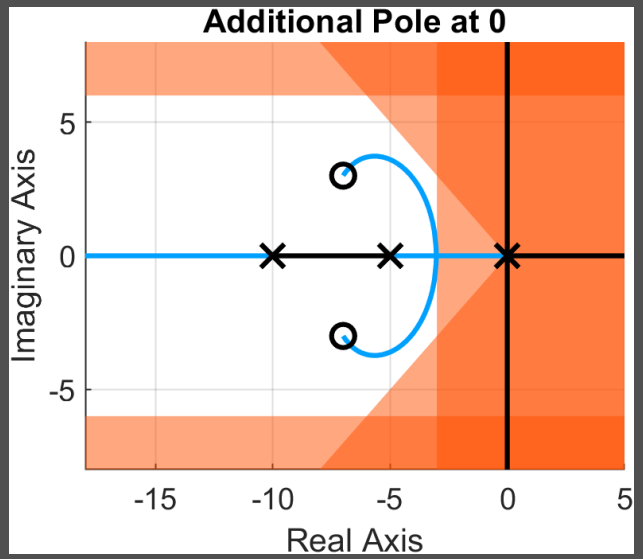
- Principle is that λ_1 and λ_2 **drag** root locus to desired region of complex plane.
  - No general rule
  - Trial-and-error required for most systems.

λ_1 and λ_2 Visualised



Steps 3 & 4. Choose Gain K and determine parameters

- Once λ_1 and λ_2 have been chosen so that root locus intersects with feasible region:



K can be determined to **place** close-loop poles using methods detailed in week 5.

- Original PID gains then obtained from

$$k_d = K \quad k_p = -K(\lambda_1 + \lambda_2) \quad k_i = K\lambda_1\lambda_2$$

Related Controllers

- Related controllers can be similarly interpreted:
- Proportional derivative (PD)

$$C(s) = k_p + k_d s = K(s - \lambda_2)$$

  - Choose zero location and gain K.

- Proportional integral (PI)

$$C(s) = \frac{k_i}{s} + k_p = \frac{K(s - \lambda_1)}{s}$$

  - Place open-loop pole at zero, then choose zero location and K.

Root Locus General Method

- There no **general algorithm** for designing a controller using the root locus.
  - The approach depends entirely on the locations of the open-loop system poles.
  - There are infinite possible variations.

- The primary intuition is that the additional zeros act as **attractors** for the root locus.

Conclusion

- We have demonstrated an analytical method for designing PID controllers that **can** be guaranteed to meet time-domain performance requirements.

- Method is subject to several limitations:
  - Performance is only guaranteed for **dominant** pole pair.
  - Can still require trial and error solution.
  - Can rapidly become more challenging for higher-order systems.
  - Requires a transfer function model.

- Next we will look at how the bode plot can inform PID design.

Contents

Bode Plot Design

- We previously demonstrated how PID design can be interpreted using root locus
- Several limitations
  - Particularly, performance is **only** guaranteed for dominant pole pair.
- We will now investigate how the bode plot can be used as a tool for PID design.

PID Decomposition

- Similarly to the previous lecture, we can show that the PID controller can be decomposed into

$$C(s) = K\frac{(s+\omega_1)(\frac{1}{\omega_2}s+1)}{s}$$

where

$$k_d = \frac{K}{\omega_2}, \quad k_p = K\left(1+\frac{\omega_1}{\omega_2}\right), \quad k_i = K\omega_1$$

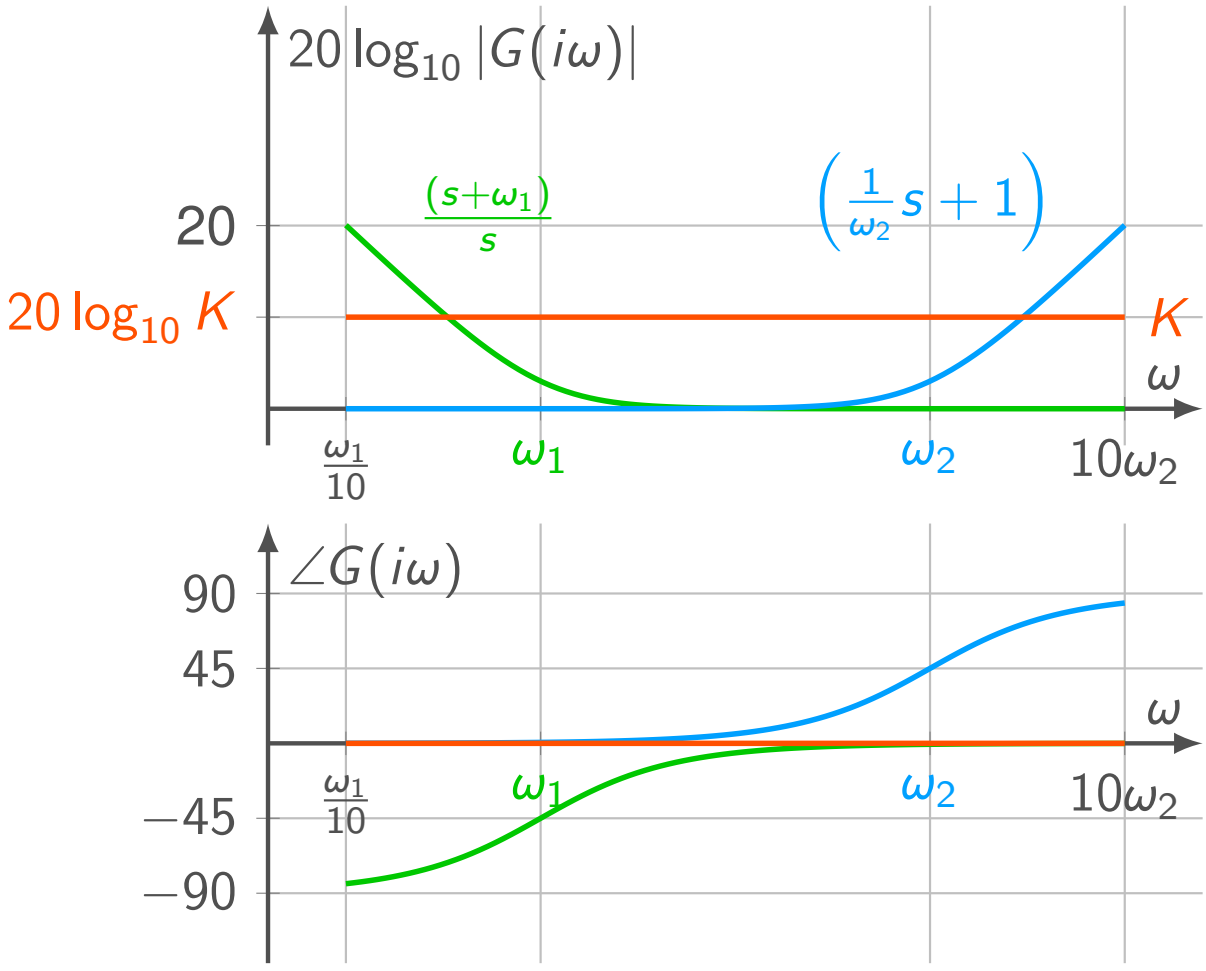
- We can therefore interpret this controller from the **frequency response** of the components:

$$C(s) = \frac{(s+\omega_1)}{s} \times \left(\frac{1}{\omega_2}s+1\right) \times K$$

- Therefore, instead of choosing the gains k_i, k_p, and k_d, we can choose the gain K and the **frequencies** ω_1 and ω_2

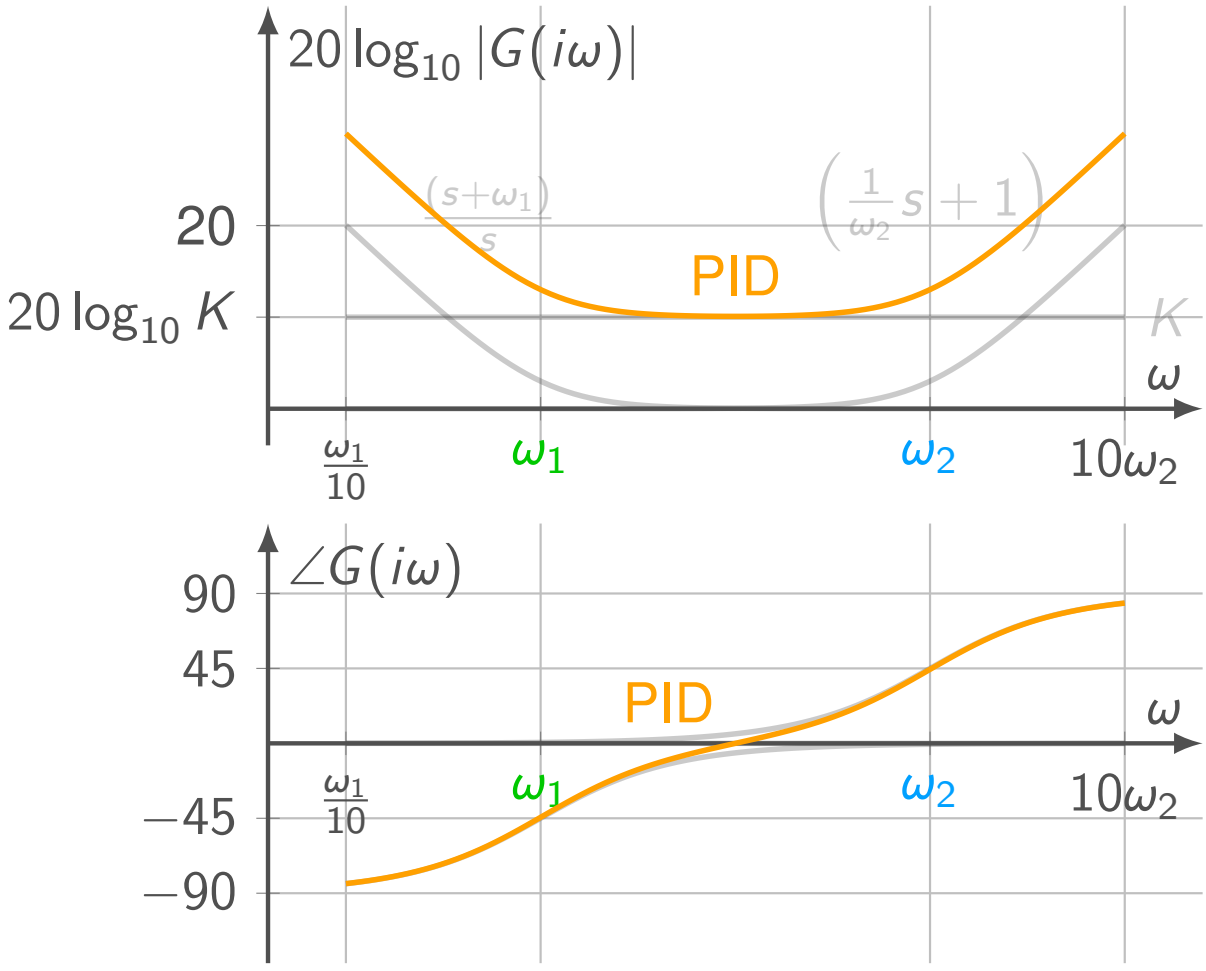
PID Frequency Components Visualised

- You should be familiar with these representations from last week.



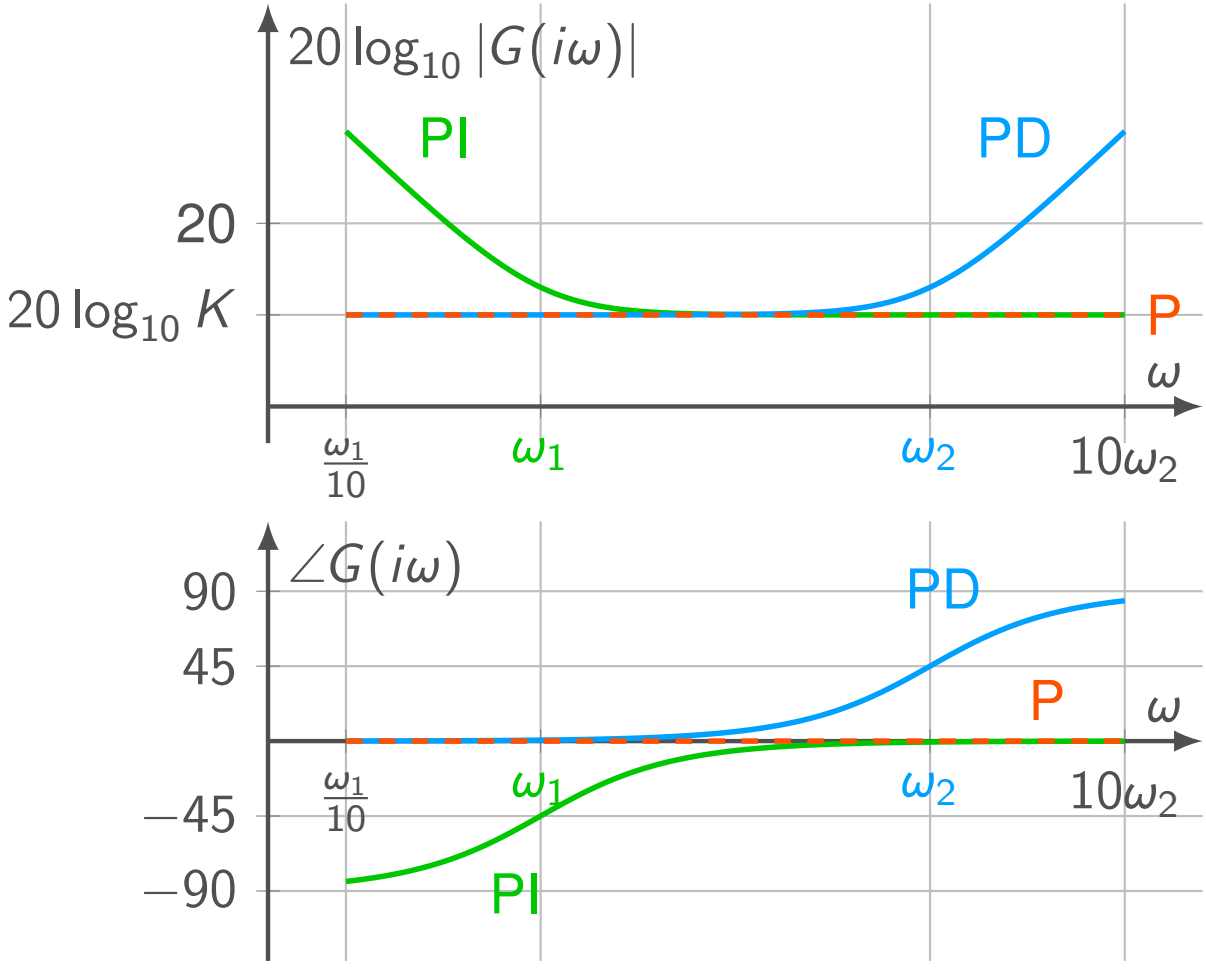
PID Frequency Response Visualised

- PID controller is therefore the sum of the three components.



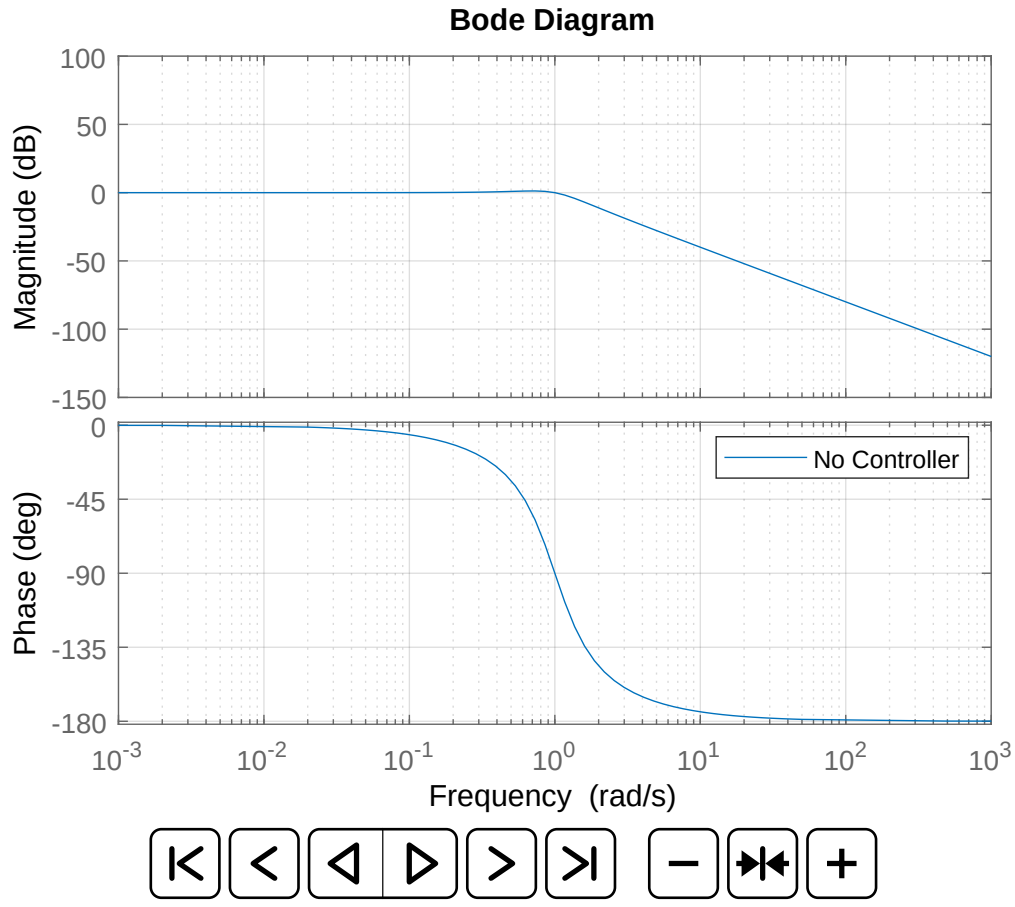
Related Controllers

- Related controllers can be interpreted similarly:



Bode Plot PID Intuition

- The **open-loop** transfer function can then be obtained by **adding the PID bode plot to the system bode plot**:



General Method

- Similar to Root Locus, there is no **general algorithm** for designing a controller using a bode plot.

- Sensible method similar to the heuristic case:
  - Maximize gain K subject to stability margins.
  - If needed, introduce low-frequency boost to reduce s.s. error (integral)
  - If needed, introduce high-frequency boost to increase damping (derivative).

- Generally, the bode-plot is a **useful tool** for inferring closed-loop performance, and making sensible design decisions.
  - Particularly useful **in conjunction** with other tools.

Contents

Implementation Considerations

- Analytical design methods sometimes ignore some of the **practicalities** of controller implementation.
- These are best learned through practice, as again, every hardware combination introduces different challenges.
- We will briefly discuss three **extremely common** considerations:

    1. Noise
    2. Integral windup
    3. Controller saturation

Noise and Derivatives

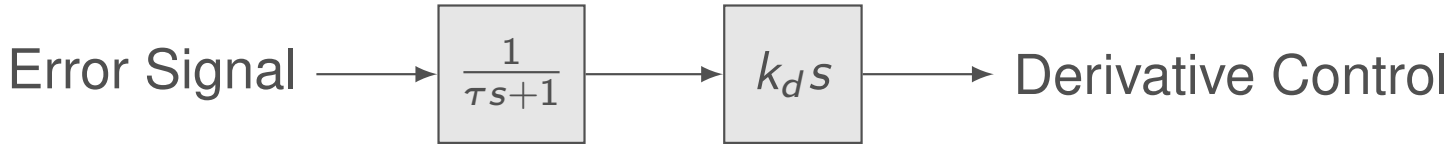
- Noise can have serious detrimental effects on controller performance, particularly derivative and integral terms.

- Recall that the derivative of a sinusoid given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\sin\omega t = \omega\cos\omega t$$

$\Longrightarrow$ Derivative action **amplifies** high frequency noise.

- Therefore, derivative gain generally implemented with **low-pass** filter beforehand:



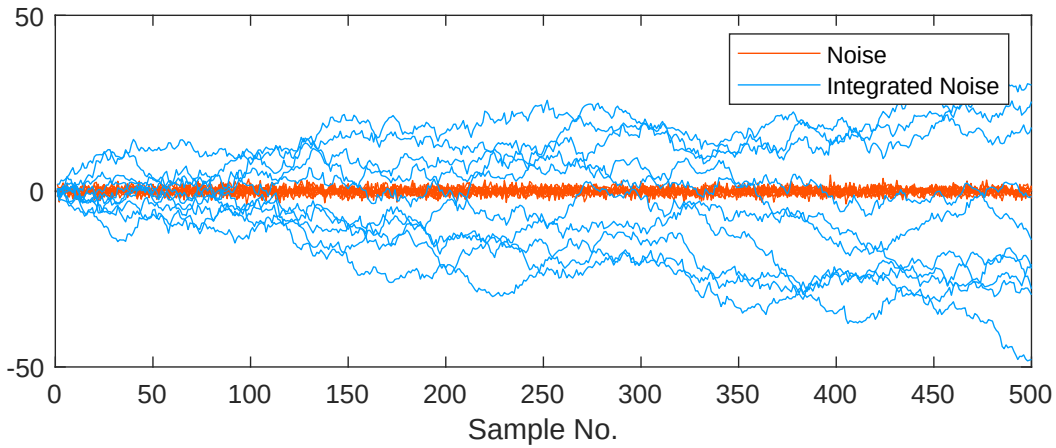
Noise and Integrals

- Noise can also be a problem for **integral** gains.

- For simplicity, consider the sum of N samples of zero-mean, unit variance noise:

$$\sum_{i=1}^{N} x_i, \quad x_i \sim \mathcal{N}(x_i|0,1)$$

- This sum **diverges** as $N \to \infty$ - known as a **random walk**.



- This is known as **drift** and also needs to be accounted for in signal processing.

Integral Windup

- Integrators are also susceptible to a phenomenon known as **windup**.
- To illustrate windup, consider a controlled system that is prevented from reaching error of zero.
  - Possibly caused by **controller saturation**
  - Integral-term continues to increase over time
  - Prevents system from reacting when demand is stepped down again
- Possible solution is to **saturate** integral action.

Controller Saturation

- Limits on realisable control inputs can also adversely affect control performance.

- From our knowledge of the root locus, it would appear to be desirable to put out closed-loop poles on the real axis and send them to $-\infty$
  - Infinitely fast, critically damped response.

- This **theoretical** performance cannot be achieved in practice.
  - Classical control design methods cannot readily deal with this issue, and can lead to instability.
  - **Model predictive control** is the most common way of addressing input and state saturation.

- Must test controller on saturated (**nonlinear**) system to validate performance.

Conclusion

- We have demonstrated how PID controller design can be interpreted using
  - Root Locus
  - Bode Plot

- These methods allow for a more robust approach to PID controller design.

- Also introduced some practical considerations for PID controller design.

- Next week: we will consider alternative controller structure - **compensators**.

You can now complete
this week's problem sheet