PD / PI / PID Controller Design

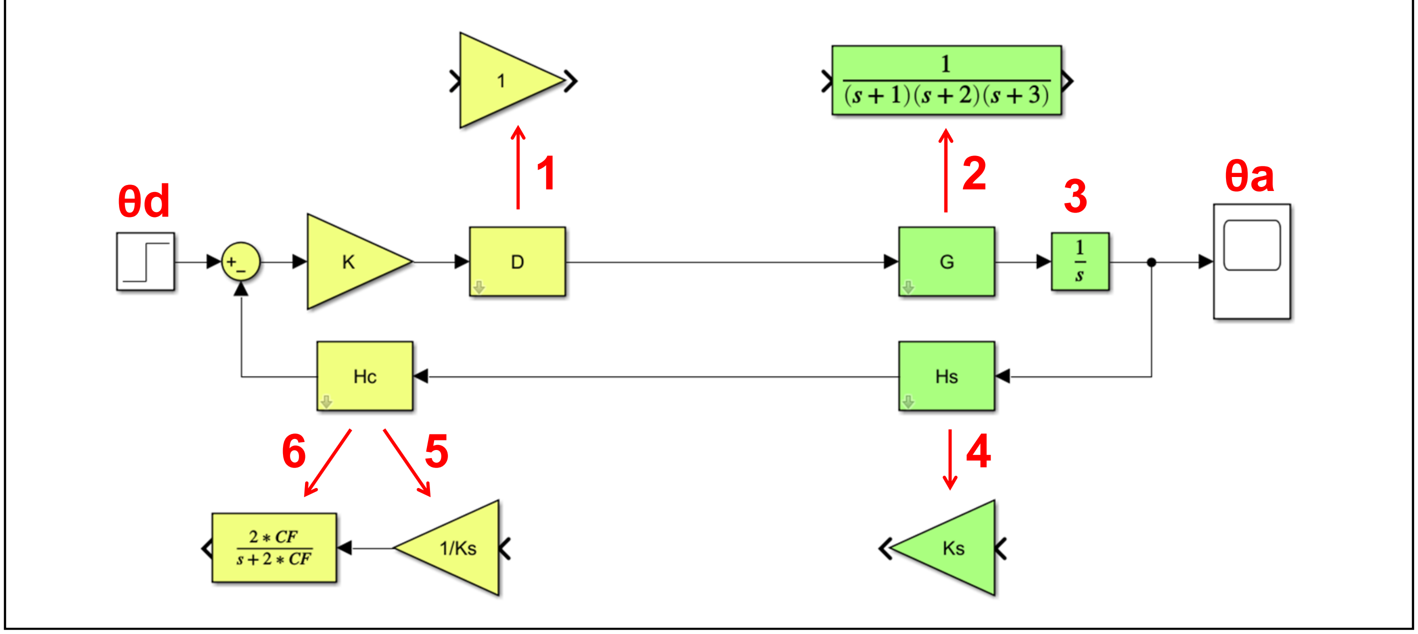
10-Step Process

(with PID Position-Control Example)

Copyright © 2022

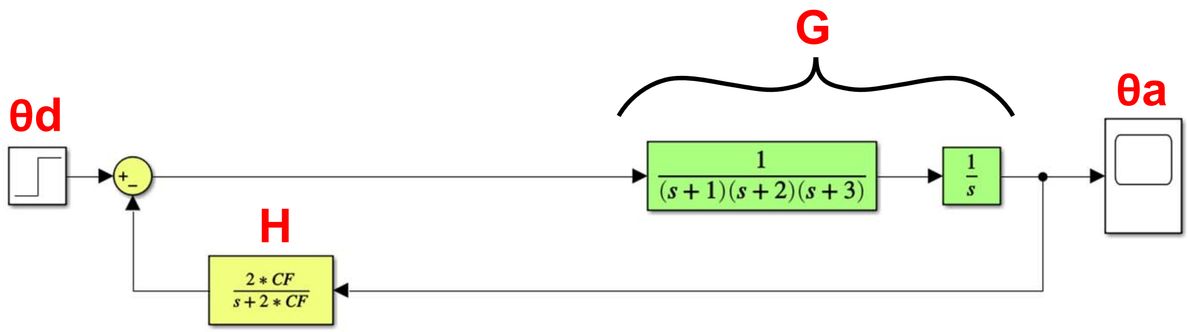
Prof. Leo Stocco, P.Eng
Electrical & Computer Engineering
University of British Columbia

Step 1a : System Identification



1) **Controller Dynamics**
   - No dynamics → Proportional Control
2) **System Model**
   - Linearize
   - 2nd Order Approximation
   - State-Space Model and/or Transfer Function
3) **Integrate**
   - Transform Speed → Position
4) **Sensor Model**
   - Often just a Sensor Gain
   - Include Sensor Dynamics if present
5) **Controller Feedback Gain**
   - Compensate for Sensor Gain
   - Entire Feedback Path must have Unity Gain
6) **Controller Feedback Dynamics**
   - Control Frequency from ISR Execution Time + Safety Factor
   - Implement PID Controller for Worst-Case Execution Time

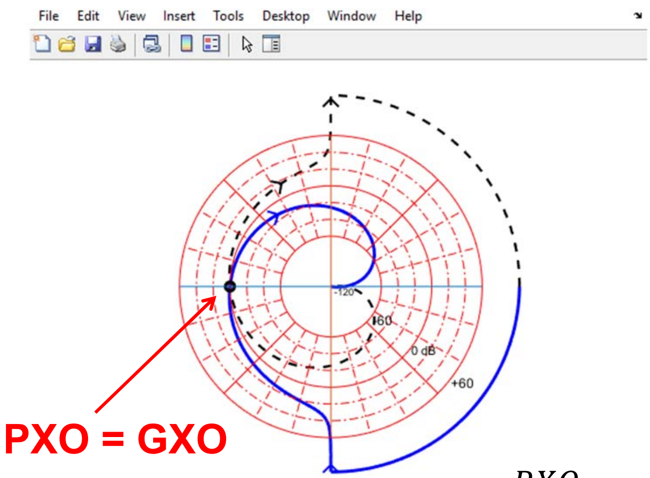
Step 1b : Mathematical Equivalent



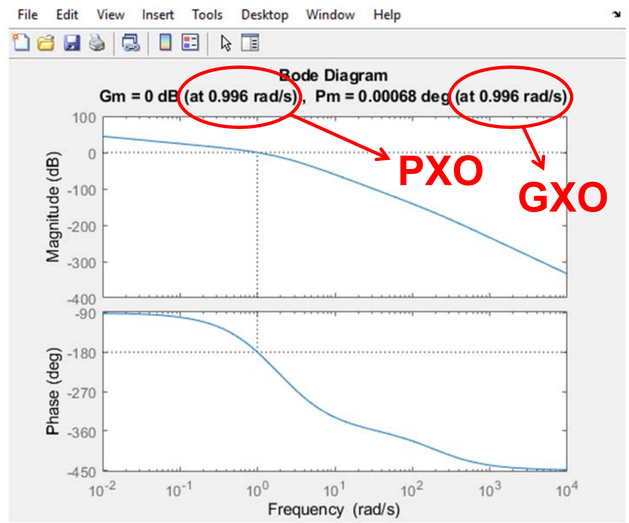
Initial System Model

- Many blocks cancel out
- Simple mathematical equivalent
- Use to CHECK model (GH)

Step 2 : Marginally Stable Reference



$$z = \frac{PXO}{10} = 0.1$$



Find Kappa

- Use ***margin(GH)***
- Kappa = Ultimate Gain using P-Control

Generate Nyquist Contour

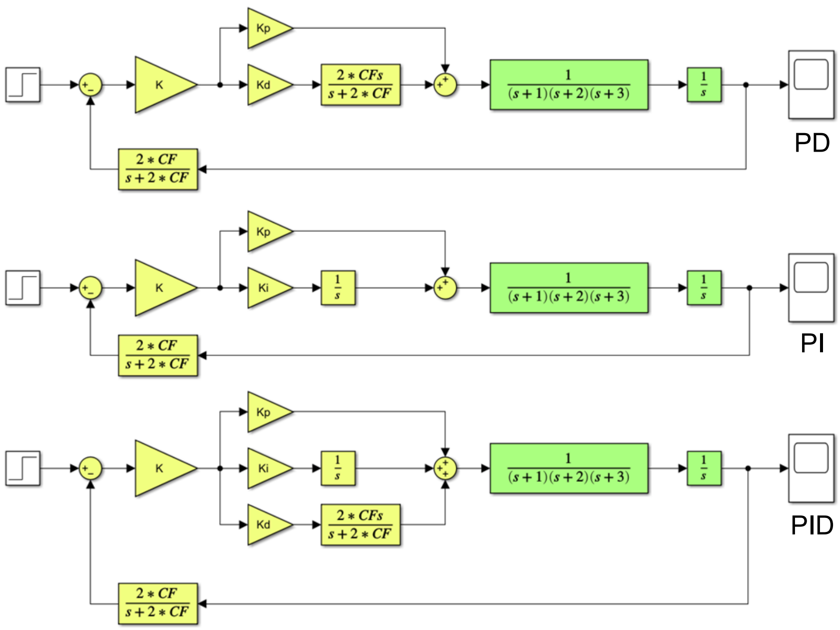
- Use ***nyqlog(*κ*GH)***
- Marginally Stable reference figure
- GXO = PXO

Find PXO

- Use ***margin(GH)*** *or* ***margin(*κ*GH)***
- GXO affected by Gain
- PXO NOT affected by Gain

Step 3 : Controller Dynamics

$$z = \frac{PXO}{10} = 0.1$$



$$D = K_p + K_d \frac{2CFs}{s + 2CF}$$

$$D = K_p + K_i \frac{1}{s}$$

$$D = K_p + K_i \frac{1}{s} + K_d \frac{2CFs}{s + 2CF}$$

Zero Location

- z = PXO/10

PD Controller (zero @ -z)

- Kp = 1
- Kd = 1/z – 1/p

PI Controller (zero @ -z)

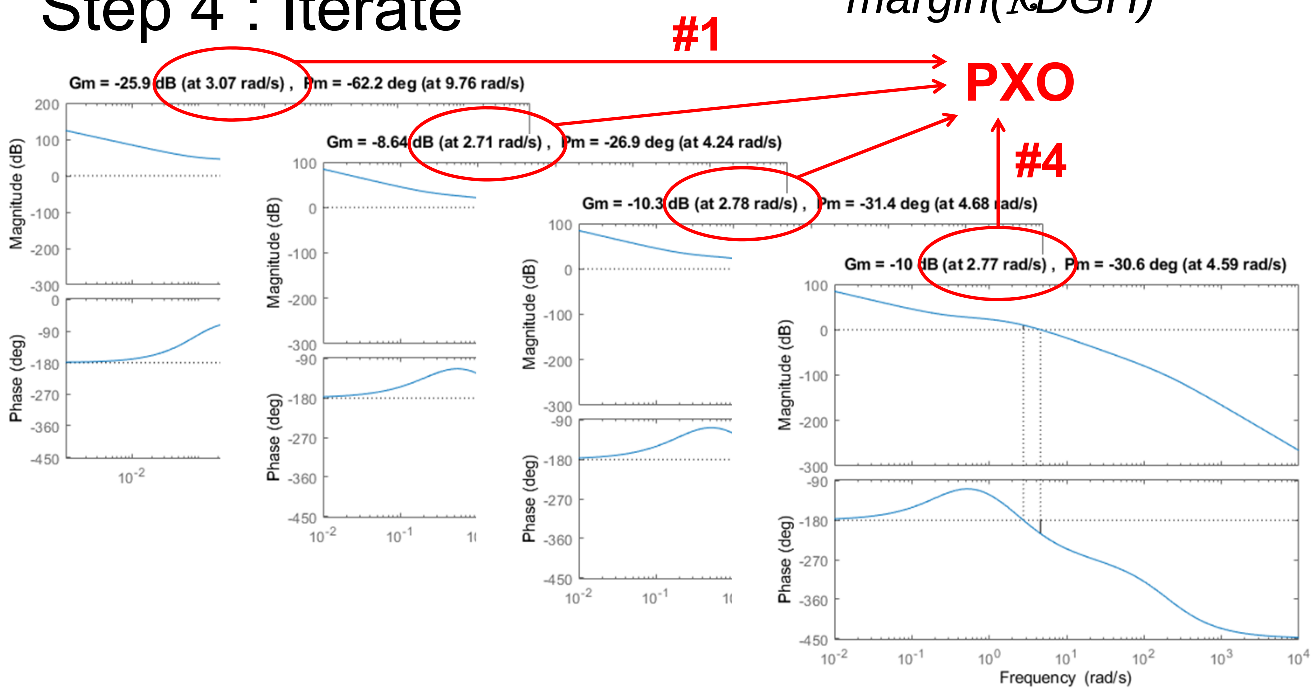
- Kp = 1/z
- Ki = 1

PID Controller (double-zero @ -z)

- Kp = 2/z – 1/p
- Ki = 1
- Kd = 1/z^2 – Kp/p

Step 4 : Iterate

margin(KDGH)



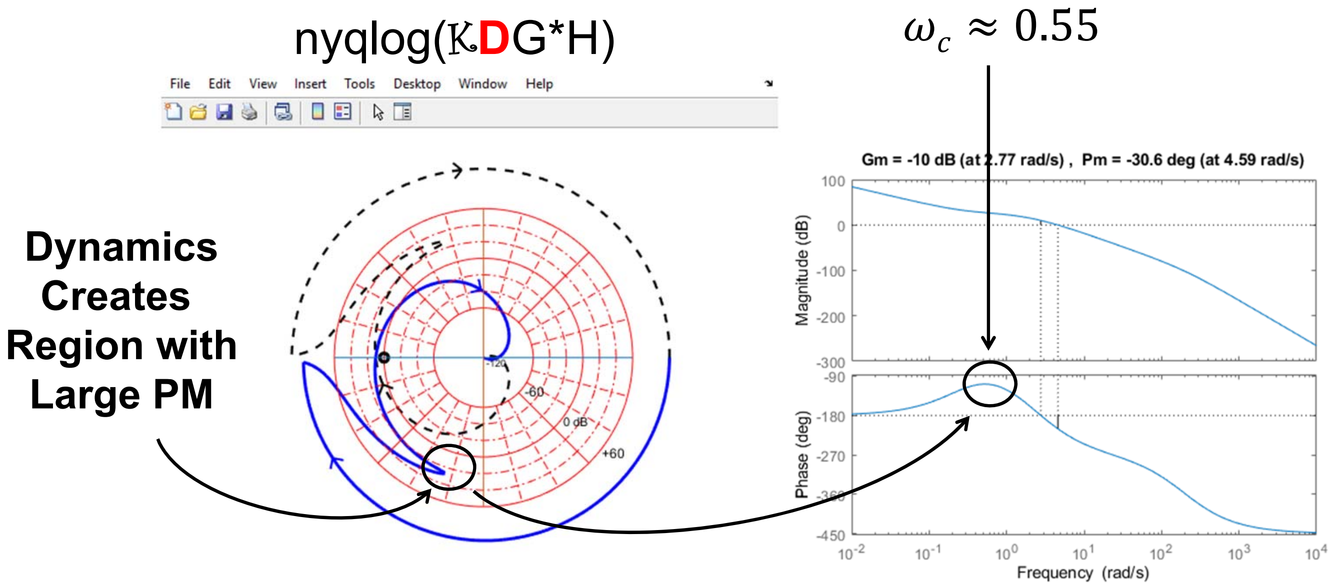
Controller Dynamics

- Use ***margin(KDGH)*** to find new PXO
- Use new PXO to adjust Zero(s)
- Re-compute Controller Dynamics
- Repeat until PXO stops changing

Multiple Solutions

- When phase = -180 at multiple frequencies you get multiple solutions
- Choose lowest frequency solution (closest to jω axis on pole-zero plot)

Step 5 : Identify Corner Frequency



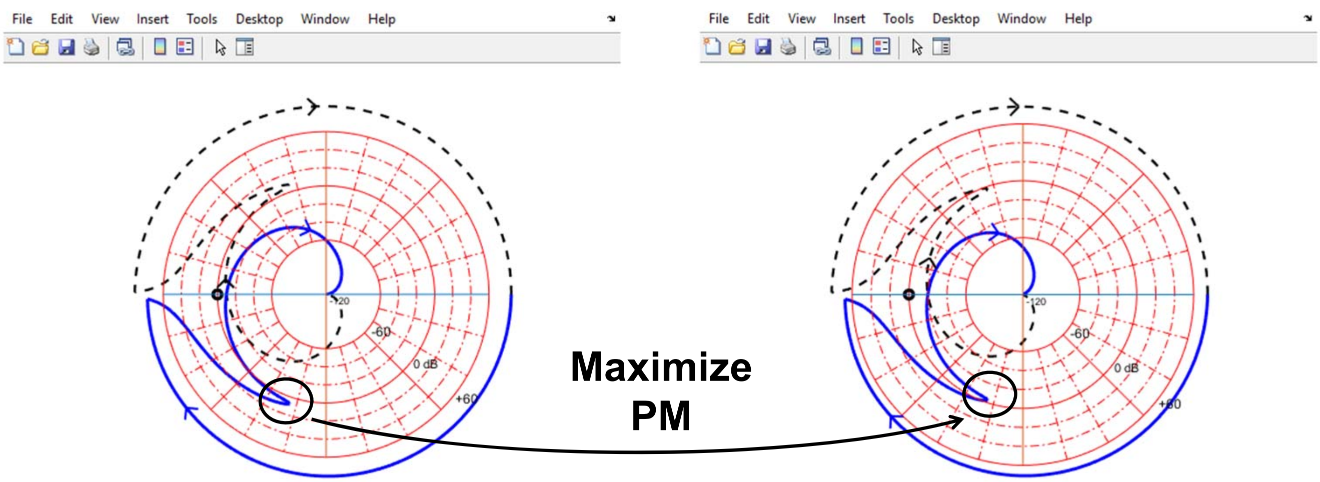
<u>Corner Frequency</u>

- Results from zero placement
- Large PM at that frequency
- Peak in Phase Bode plot
- Lookup frequency on Nyquist or Bode

Step 6 : Initial Gain

$$K_0 = \frac{1}{abs(freqresp(DGH, \omega_c))} \approx 0.44$$

nyqlog(**D**GH) nyqlog(**K_0D**GH)



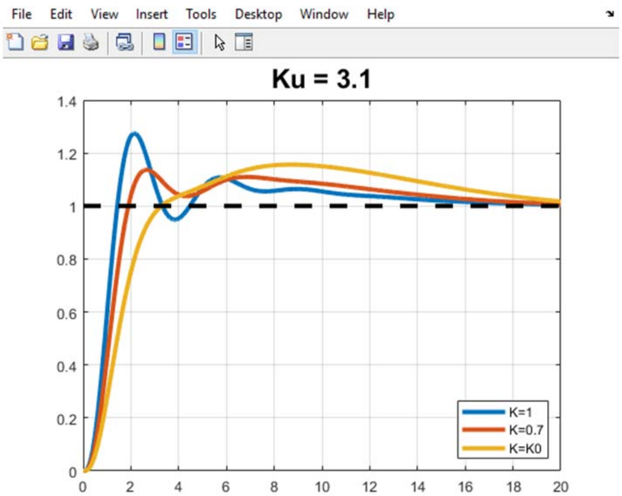
Corner Frequency

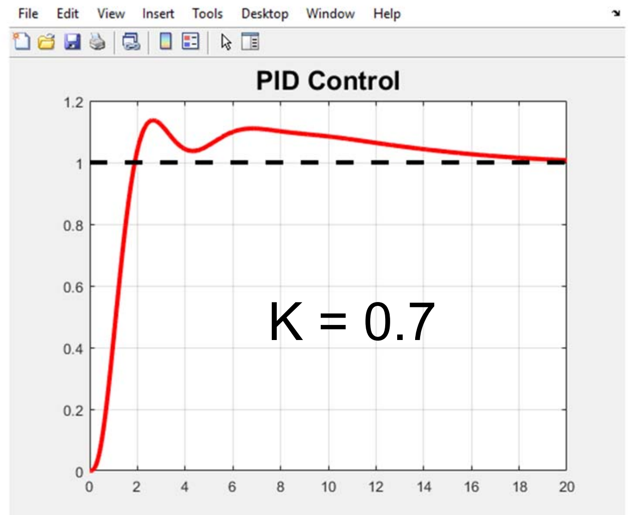
- Remove K to simplify math
- Use ***freqresp()***
- Find gain of **DGH** at wc

Applying Gain K0

- Corner intersects 0dB iso-line
- PM maximized

Step 7 : Tune Gain





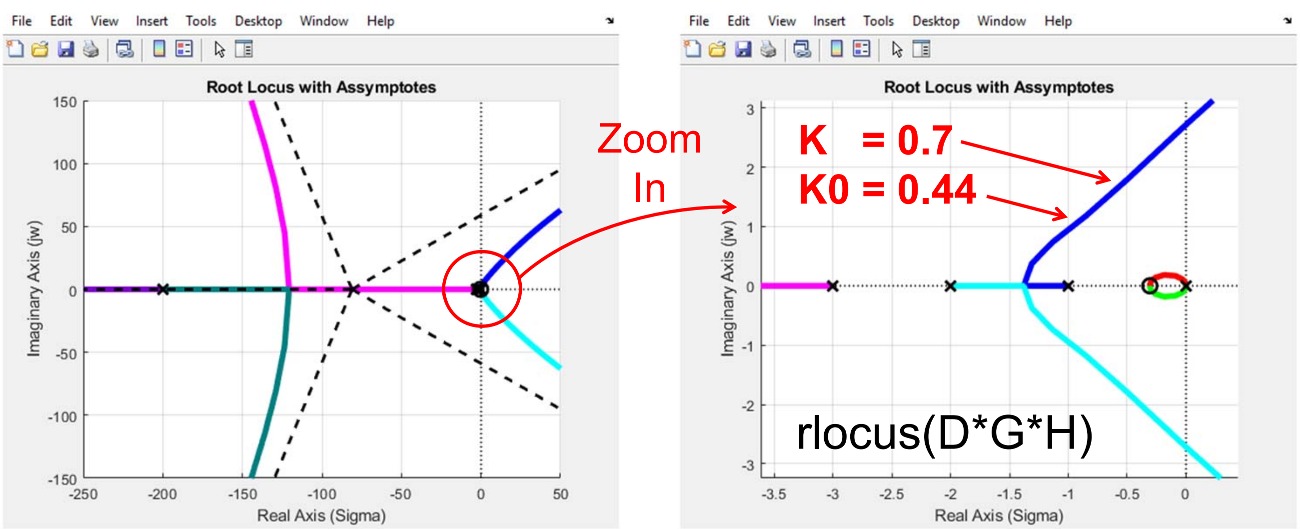
<u>Plot Step Response</u>

- Compute Closed-Loop transfer function
- Plot step response for range of K values

<u>Choose best compromise</u>

- Consult RCGs

Step 7b : Check Root Locus

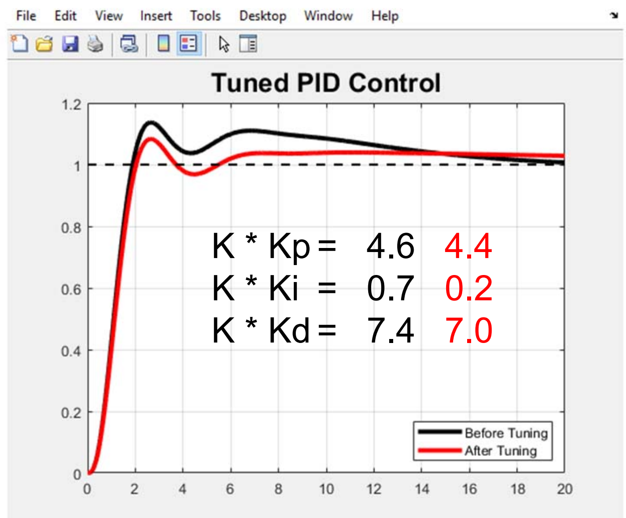


Root Locus (DGH)

- Never include gain K in RL plots
- Zoom in on Dominant Roots
- OL Poles & Zeros as expected?
- Least stable poles attracted to zeros?

Step 8a : Heuristic Tune

- Controller Gain K ↑
  - ↑ Kp, Ki, Kd Simultaneously
  - Poles follow Root Locus
- Proportional Gain Kp ↑
  - ↓ Rise Time & Steady-State Error
  - ↑ Overshoot
- Integral Gain Ki ↑
  - ↓ Steady-State Error
  - ↑ Overshoot, Settle Time
- Derivative Gain Kd ↑
  - ↓ Overshoot, Settle Time
    - Destabilizes when too large
    - Depends on filter pole



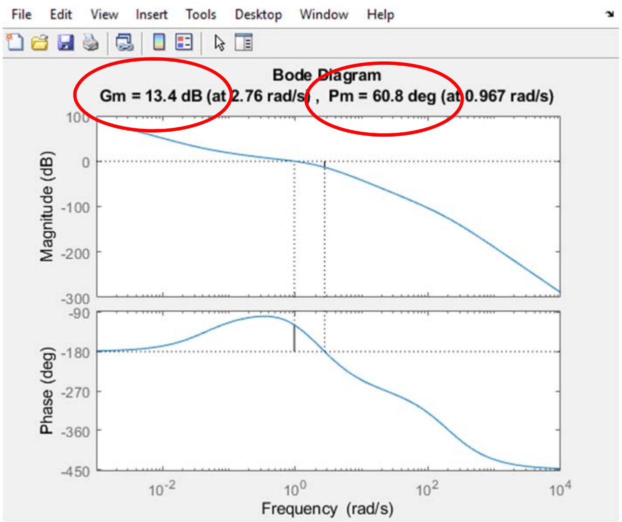
<u>Adjust Individual Gains</u>

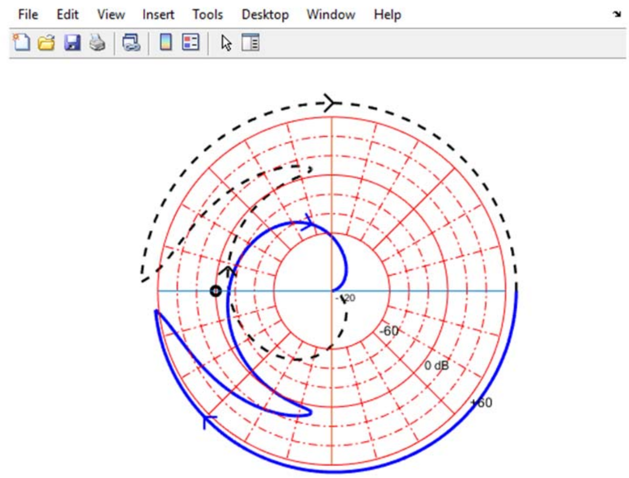
- #1 – Ki & K
  - Balance overshoot & steady-state error
- #2 – Kp & K
  - Balance rise time & stability
- #3 – Kd & K
  - Maximize stability

<u>Repeat until satisfied</u>

- Small increments
- Record good combinations

Step 8b : Evaluate Margins





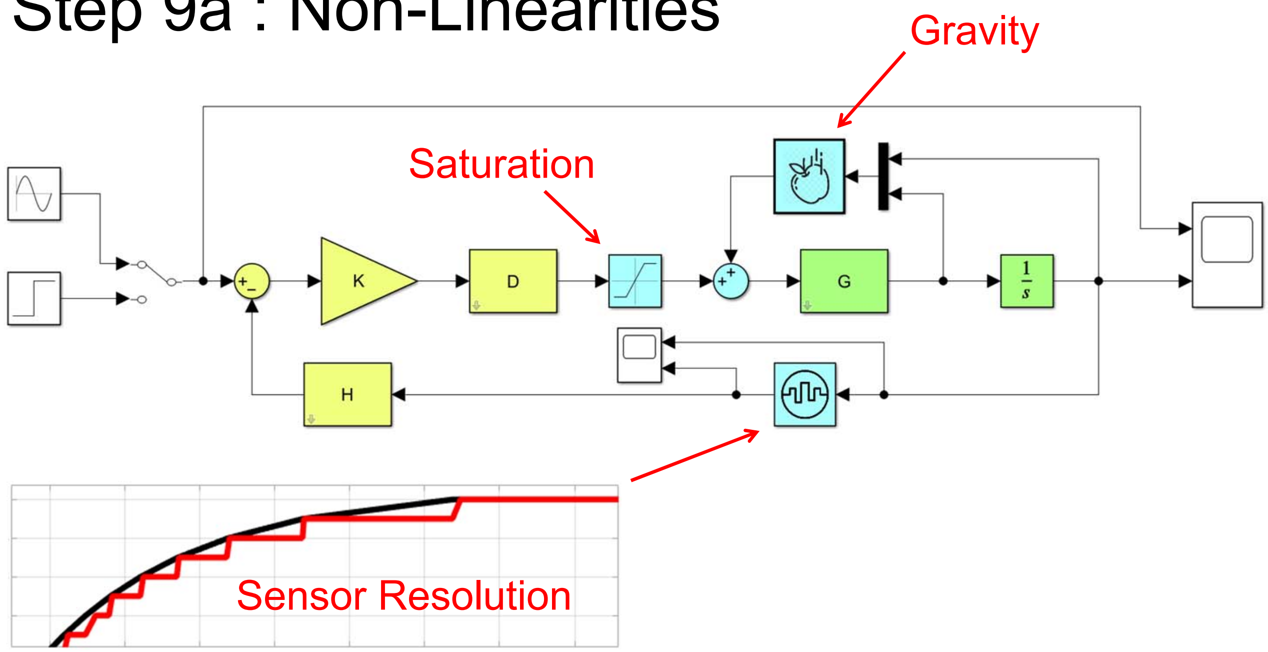
Generate Bode & Nyquist plots

- Evaluate GM
- Evaluate PM

Check

- Higher margins → Reduced sensitivity

Step 9a : Non-Linearities



Transfer to Simulink

- **Control System Toolbox / LTI System** block for transfer functions
- Non-linearities convenient to model in Simulink

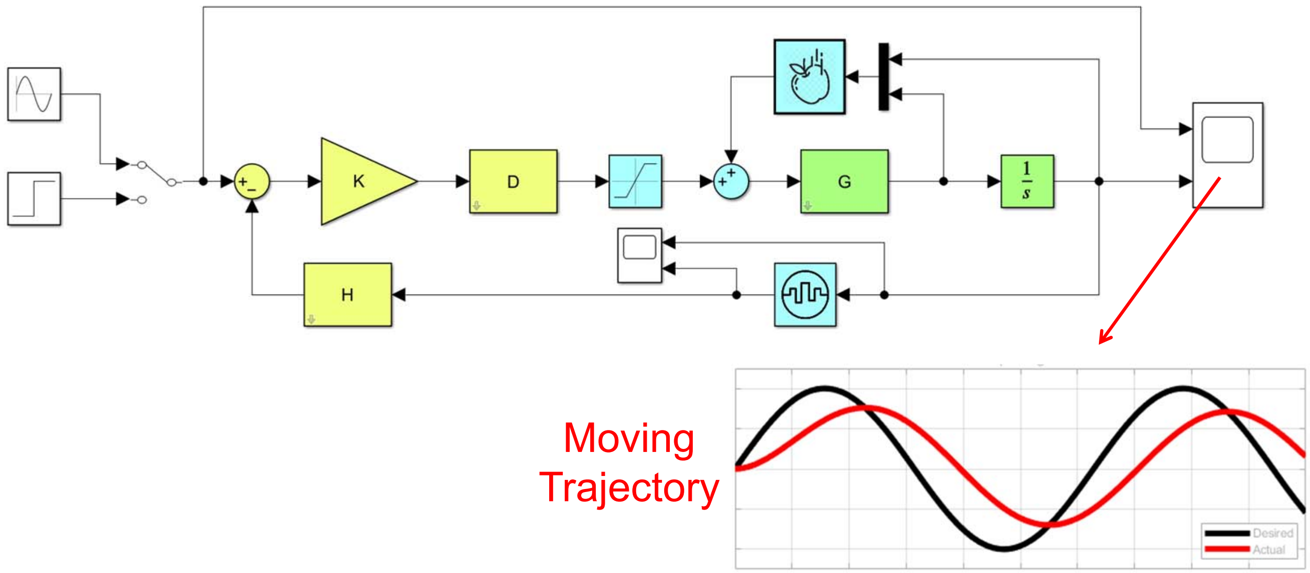
Add Non-Linearities

- **Discontinuities / Saturation** for Voltage / Current limits
- **Math Operations / Floor** for resolution
- **User Defined Functions / MATLAB function** for custom equations (Gravity / Friction)
- Explore all Simulink libraries for other features

Results Not Acceptable

- Adjust RCGs
- Go to Step 7

Step 9b : Application Requirements



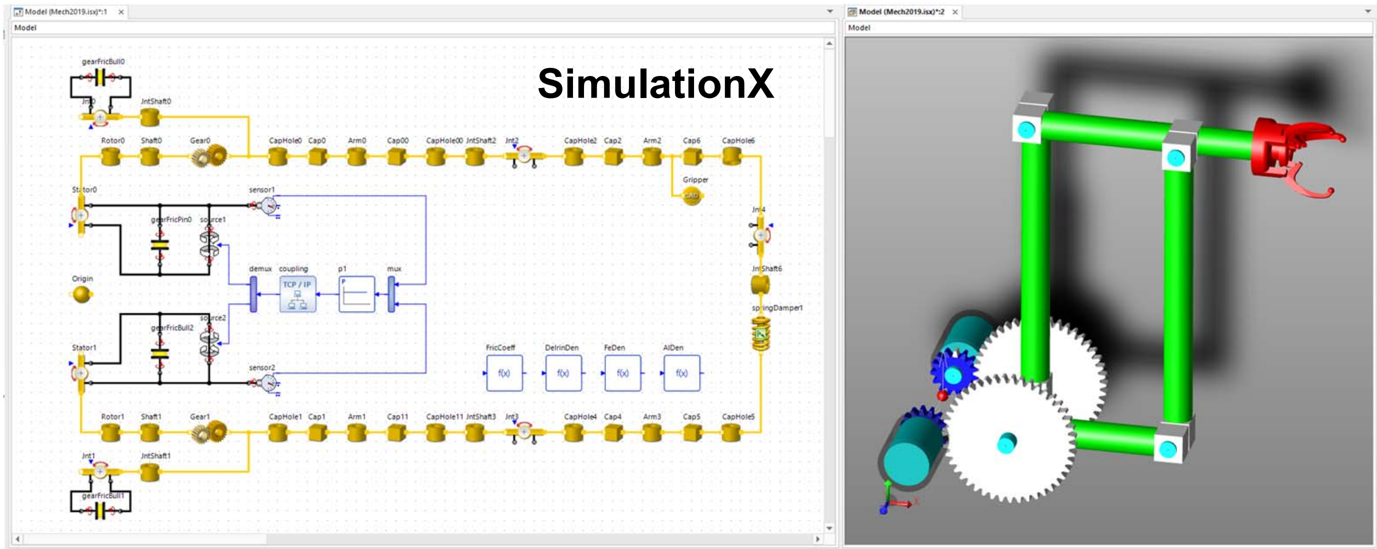
<u>Replace Step Input with Sine Input</u>

- Evaluate Delay
- Overshoot eliminated by moving target
- Better tracking when **STABILTY REDUCED**

<u>Results Not Acceptable</u>

- Adjust RCGs
- Go to Step 7

Step 10 : Practical Implementation



<u>Results similar?</u>

- Fix bugs
- Repeat process

<u>Results acceptable?</u>

- Heuristic Tune
- Use Intended System during tuning