ME 4555 - Lecture 27 - More Root locus Design

We will investigate the use of a PID controller for two problems involving motors: speed control and position control. To keep things simple, we will use the same motor equations we've been using so far.

$$\frac{\Omega}{V} = G(s) = \frac{K}{(Js+b)(Ls+R)+K^2} \qquad \text{where: } \begin{cases} J = 2.5\times10^{-4} & R = 0.5 \\ b = 1.0\times10^{-4} & L = 1.5\times10^{-3} \\ K = 0.05 & \text{(all SI units)} \end{cases}$$

$\uparrow \frac{\text{rad/s}}{\text{volt}}$

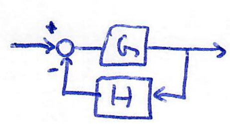
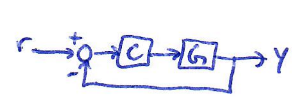
For speed, we will work in rpm. So let's find the transfer function from voltage to speed in rpm. Conversion is: $\frac{1\text{ rad}}{1\text{ sec}} = \frac{1/2\pi\text{ rev}}{1/60\text{ min}} = \frac{30}{\pi}\text{ rpm}$

So $\dfrac{\Omega\,[\text{rpm}]}{V\,[\text{volt}]} = \dfrac{30}{\pi} G(s) := G_s(s)$ ← subscript "s" is for "speed".

For position, we will work in degrees. The conversion from rad to deg is $\frac{180}{\pi}$.
But we also need to convert angular speed to angle, which means integrating.

So $\dfrac{\Theta\,[\text{deg}]}{V\,[\text{volt}]} = \dfrac{180}{\pi}\cdot\dfrac{1}{s}\cdot G(s) := G_p(s)$ ← subscript "p" is for "position".

New Matlab commands:

☆ feedback(G, H) returns tf of  so e.g.  is feedback(C*G, 1).

☆ pid(kp, ki, kd) returns the tf: $k_p + k_i/s + k_d s$. (pid controller).

More realistic PID control:

A pure differentiator is not realizable in practice. Instead, we must add a filter that mimics differentiation but rejects high-frequency noise. One example: a "low-pass filter":

Instead of $K_d \cdot s$, use: $\dfrac{K_d \cdot s}{\tau_d s + 1}$

- realizable since it has at least as many poles as zeros.
- typically we make τ_d as small as possible (pole far left).
- usually τ_d is fixed based on the hardware available, so we only have control over K_d.

PID controller (more realistic) looks like:

$$C(s) = K_p + K_i/s + \frac{K_d s}{\tau_d s + 1}$$

$$= \boxed{\frac{(K_p \tau_d + K_d) s^2 + (K_i \tau_d + K_p) s + K_i}{s(\tau_d s + 1)}}$$

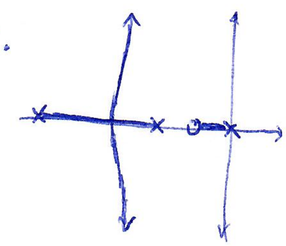
(two zeros and two poles)

Even if τ_d is fixed, changing K_p, K_d, K_i still gives us the freedom to make the numerator any quadratic polynomial (with positive coefficients). So we can place our two compensator zeros anywhere we like in the left-half plane.

<u>Speed control</u>: If our power source can only provide a max of 12 volts, and we want to achieve 0 to 1000 rpm with < 10% overshoot, how <u>fast</u> can we do it?

☆ investigate using rltool(Gs). {plot step r2y and r2u.}

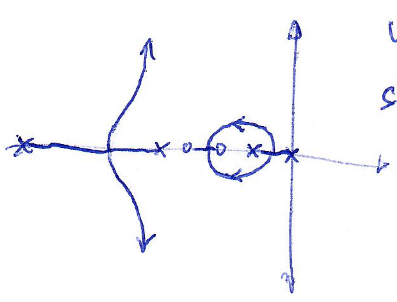
- P control is insufficient (steady-state error)
- PI control can work for suitably chosen gains. (can get $t_s \approx 0.07$ sec by using zero to cancel dom. pole!)
- If we remove voltage restriction, can get much faster response, now oscillation is the limiting factor.



☆ try PID controller

- with regular (ideal) PID, r2u doesn't make sense (tf is improper!). Appears we can make response arbitrarily fast...
- using pseudo-D ($\tau_d = 1ms$), again oscillation is the limiting factor, and large voltages are required

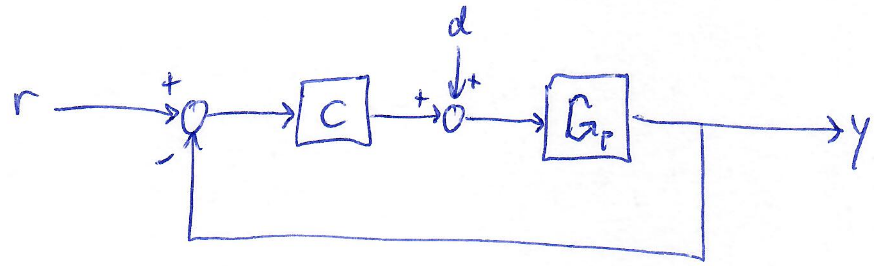
→ motor equations would probably break down. windings would explode... Not a realistic solution. (even if the voltage source were powerful enough).



☆ use "design requirements" feature!

Position control (if we have time)

Goal: design a reference position-tracking controller for the motor. We want zero error to a step disturbance input.



So y should track a step in r perfectly. Also, A step at d (disturbance) should result in $y \to 0$.

☆ since G_p already has an integrator, C doesn't need one to achieve zero error on reference step. Verify with P controller using rltool.

☆ verify "du2y" step response does not go to zero as required

$$du2y = \frac{G_p}{1 + CG_p} = \frac{G/s}{1 + CG/s} = \frac{G}{s + CG}.$$

if we want this to go to zero with a step input, we need by FVT: $\frac{G(0)}{0 + C(0)\,G(0)} = \frac{1}{C(0)} \to 0$ so we need $C(0) \to \infty$ i.e. C needs an integrator!

☆ experiment with PI and PID controllers.

☆ Note: pid(kp, Ki, Kd, τ) command in matlab

produces $k_p + K_i/s + \frac{K_d s}{\tau s + 1}$.

Extracting PID coefficients

When we use rltool, we will obtain a controller of the form:

$$\frac{\alpha s^2 + \beta s + \gamma}{s(s+\delta)}$$

This should be equal to

$$\frac{(k_p\tau + k_d)s^2 + (k_i\tau + k_p)s + k_i}{s(\tau s + 1)}.$$

How can we find k_p, k_i, k_d, τ ?

Match up coefficients:

$$\left.\begin{aligned} \frac{\alpha}{\delta} &= k_p\tau + k_d \\ \frac{\beta}{\delta} &= k_i\tau + k_p \\ \frac{\gamma}{\delta} &= k_i \\ \frac{1}{\delta} &= \tau \end{aligned}\right\} \overset{\text{solve}}{\Longrightarrow} \left\{\begin{aligned} k_p &= \frac{\beta\delta - \gamma}{\delta^2} \\ k_i &= \frac{\gamma}{\delta} \\ k_d &= \frac{\alpha\delta^2 - \beta\delta + \gamma}{\delta^3} \\ \tau &= \frac{1}{\delta} \end{aligned}\right.$$