rldesign.multd Multiple derivative compensators

Lec. rldesign.PD shows how to design a derivative compensator such that the compensated root locus of a control system can be made to include some test point $\psi \in \mathbb{C}$ where the designer would like a closed-loop pole (typically to satisfy transient response requirements). This derivative compensator has the form

$$C_D = K(s - z_c), \tag{1}$$

for gain $K \in \mathbb{R}$ and zero $z_c \in \mathbb{R}$. The crux of the design procedure is to compute via the root locus phase criterion[11] the required compensator phase contribution:

$$\theta_c = \pi - \angle GH(\psi) \tag{2}$$

for open-loop transfer function $GH(s)$. A trigonometric analysis shows that, for $\theta_c \in [-\pi, \pi]$, the compensator zero must be

$$z_c = \operatorname{Re}(\psi) - \operatorname{Im}(\psi)/\tan\theta_c. \tag{3}$$

The obvious limitation here is that if the required compensation θ_c is beyond $\pm\pi$, the derivative compensator of Eq. 1 cannot contribute sufficient phase. The strategy we adopt here is to augment the derivative compensator to include as many (equal) zeros as we need:

$$C_m = K(s - z_m)^m, \tag{4}$$

where z_m is a zero of multiplicity m. We call this a multiple derivative compensator or m-derivative compensator.

How do we select the compensator zero z_m and multiplicity m for a given θ_c? First, we determine m by determining how many π (or $-\pi$) contributions are required:[12,13]

$$m = \left\lceil \frac{|\theta_c|}{\pi} \right\rceil. \tag{5}$$

11. The phase criterion was defined in Lec. rlocus.def, Eq. 6.

Algorithm multd.1 the multiple derivative compensator algorithm.

```
function d_comp_m(ψ, GH(s))
    θ_c ← π − ∠GH(ψ)                  ▷ required phase comp
    m ← ceiling(θ_c/π)                ▷ zeros needed
    θ_m ← θ_c/m                       ▷ divide contributions
    z_m ← Re(ψ) − Im(ψ)/ tan θ_m      ▷ trig
    C'_m ← (s − z_m)^m                ▷ comp sans gain
    K_m ← |C'_m(ψ)GH(ψ)|^(−1)         ▷ angle criterion
    C_m ← K_m C'_m                    ▷ comp with gain
    return C_m
end function
```

12. The function $\lceil\cdot\rceil$ is called the ceiling function and rounds up to the nearest integer.

13. Note that if $\theta_c \in [-\pi, \pi]$, the multiplicity $m = 1$ and the compensator is a regular derivative compensator.

With this, we can divide-up the the required phase contribution θ_c among the m zeros:

$$\theta_m = \theta_c/m. \tag{6}$$

By construction, $\theta_m \in [-\pi, \pi]$, so the compensator zeros should be located at

$$z_m = \mathrm{Re}(\psi) - \mathrm{Im}(\psi)/\tan\theta_m. \tag{7}$$

This is summarized in Algorithm multd.1.

Causality

A complication can arise when derivative compensation yields a closed-loop transfer function with more zeros than poles—a type of system called non-causal (non-non-causal systems are called causal). Non-causal systems are those that depend on future states, something classically[14] impossible to instantiate in real-time, and therefore a controller that creates such a control system is of no practical use.[15] Adding multiple zeros to a controller can easily yield such undesirable systems.

To mitigate this, we can include ι pure integrators $1/s$ into the compensator. They will obviously affect the root locus, so their effects must be taken into account during the zero compensator calculations. This is done by treating the open-loop transfer function as if it already had the compensator integrators $1/s^\iota$. Algorithm multd.2 summarizes this approach.

14. It gets complicated when considering relativity and quantum mechanics, which we do not, here.

15. Non-causal system models are useful for digital signal post-processing, but these are always a posteriori—i.e. "future" time is known because it is in the analytic past. Controllers do not have this luxury.

Algorithm multd.2 the multiple derivative compensator algorithm with ι integrators.

```
function d_comp_mi(ψ, GH(s), ι)
    θ_c ← π − ∠GH(ψ)/s^ι              ▷ required phase comp
    m ← ceiling(θ_c/π)                 ▷ zeros needed
    θ_m ← θ_c/m                        ▷ divide contributions
    z_m ← Re(ψ) − Im(ψ)/tan θ_m        ▷ trig
    C'_m ← (s − z_m)^m/s^ι             ▷ comp sans gain
    K_m ← |C'_m(ψ)GH(ψ)|^−1            ▷ angle criterion
    C_m ← K_m C'_m                     ▷ comp with gain
    return C_m
end function
```

Example rldesign.multd−1

Design a controller to meet the