ECE 3413 Lab 05
Time Response of First- and Second-Order Systems

Leomar Durán

10^{th} March 2023

1 Introduction

The purpose of this lab is to analyze a transfer function and a set of modifications to that transfer function, whether that be scaling the components of the poles, the natural frequency or adding poles.

This allows us to form relations between these parameters and the characteristics that we expect from a transfer function such as the transient response.

Not only can we estimate the effects that a modification will have on a transfer function, but we can use these relations to find new transfer functions just as one would pick components to have specific effects on a circuit.

2 Procedure

2.1 Part 01 – Changing poles

(a) Given the transfer function

$$G_2(s) = \frac{b}{s^2 + as + b}, \tag{1}$$

let the damping ratio and natural frequency, respectively $\zeta, \omega_n \in \mathbb{R}$ s.t. $a = 2\zeta\omega_n$ and $b = \omega_n^2$. So

$$\begin{cases} \omega_n = \sqrt{b}, \\ \zeta = \dfrac{a}{2\sqrt{b}}. \end{cases} \qquad (b > 0) \tag{2}$$

Furthermore

$$\zeta\omega_n = \frac{a}{2}, \tag{3}$$

and

$$\sqrt{1-\zeta^2} = \sqrt{1 - \left({}^{a}/_{\left(2\sqrt{b}\right)}\right)^2} = \frac{1}{2\sqrt{b}}\sqrt{4b - a^2}. \quad \left(|a| \leq 2\sqrt{b}\right) \tag{4}$$

We find that the peak time

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} = \frac{\pi}{\left(\sqrt{b}\right)\left(\frac{1}{2\sqrt{b}}\sqrt{4b-a^2}\right)} = \frac{2\pi}{\sqrt{4b-a^2}}. \tag{5}$$

We find that the overshoot rate

$$\begin{aligned} \%OS &:= \exp\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)100\,\% \\ &= \exp\left(-\zeta\omega_n\left(\frac{\pi}{\omega_n\sqrt{1-\zeta^2}}\right)\right)100\,\% \\ &= \exp\big(-(\zeta\omega_n)T_p\big)100\,\% \\ &= \exp\left(-\left(\tfrac{a}{2}\right)T_p\right)100\,\% \\ &= \exp\left(\frac{-aT_p}{2}\right)100\,\%. \end{aligned} \tag{6}$$

We find that the settling time

$$T_s \approx \frac{4}{\zeta\omega_n} = \frac{4}{(a/2)} = \frac{8}{a}. \tag{7}$$

To find rise time, we must first find the step response in the time domain with parameters substituted.

2.1.1 Step response

We have

$$G_2(s) = \frac{b}{s^2+as+b},$$

Well

$$C(s) = R(s)G_2(s) = \frac{1}{s}\frac{b}{s^2+as+b}. \tag{8}$$

So the second order factor of the denominator

$$\begin{aligned} s^2 + as + b &= \left(s^2 + 2\left(\frac{a}{2}\right)s + \left(\frac{a}{2}\right)^2\right) + \left(b - \left(\frac{a}{2}\right)^2\right) \\ &= \left(s + \frac{a}{2}\right)^2 + \left(\sqrt{b - \left(\frac{a}{2}\right)^2}\right)^2 \quad \left(|a| \leq 2\sqrt{b}\right) \\ &= (s + \hat{a})^2 + \omega_d^2. \end{aligned} \tag{9}$$

Thus, for $G_2(s)$, we have exponential and the damped frequencies

$$\begin{cases} \hat{a} = \dfrac{a}{2}, \\ \omega_d = \sqrt{b - \hat{a}^2}. \end{cases} \tag{10}$$

Note that the damped frequency and peak time respectively

$$\begin{aligned} \omega_d = \frac{\pi}{2\pi/\sqrt{4b - a^2}} = \frac{\pi}{T_p}. \\ \hline\hline \\ T_p = \frac{\pi}{\omega_d}. \end{aligned} \tag{11}$$

Thus, we find coefficients A, B, C s.t.

$$b = A(s^2 + as + b) + B(s^2 + \hat{a}s) + C\omega_d s. \tag{12}$$

Assume $s = 0$. Then

$$s = 0 : b = A(\cancel{(0)^2} + \cancel{a(0)} + b) + B\left(\cancel{(0)^2} + \cancel{\hat{a}(0)}\right) + \cancel{C\omega_d(0)}. \tag{13}$$

So $b = Ab$. Thus $A = 1$. Thus

$$b = (s^2 + as + b) + B(s^2 + \hat{a}s) + C\omega_d s. \tag{14}$$

For quadratic terms, we have $0s^2 = s^2 + Bs^2$. So $B = -1$. Thus

$$b = (s^2 + as + b) - (s^2 + \hat{a}s) + C\omega_d s. \tag{15}$$

Finally, for linear terms, we have $0s = as - \hat{a}s + C\omega_d s$. Thus

$$C = \frac{\hat{a} - a}{\omega}_d = \frac{\left(\frac{a}{2}\right) - a}{\omega}_d = \frac{-a}{2\omega_d},\ \omega_d = \sqrt{b - \left(\frac{a}{2}\right)^2} \neq 0. \tag{16}$$

Then by the inverse Laplace transform, in the time domain, we have step response

$$c(t) = \left(1 + \left(-\cos(\omega_d t) - \frac{a}{2\omega_d}\sin(\omega_d t)\right)e^{-(a/2)t}\right)u(t), \tag{17}$$

given $\omega_d = \sqrt{b - \left(\frac{a}{2}\right)^2}$.

After this step, we cannot solve the general case for t. So we must first evaluate $c(t)$ at some values for a, ω_d.

2.1.2 Evaluations

Let's evaluate at $a = 4, b = 25$. Well

$$\begin{cases} b = (25) > 0, \\ |a| = |4| = 4 < 10 = 2\sqrt{(25)} = 2\sqrt{b}, \\ \omega_d = \sqrt{(25) - \left(\frac{(4)}{2}\right)^2} = \sqrt{21} \neq 0 \end{cases} \tag{18}$$

The peak time

$$T_p = \frac{\pi}{\sqrt{21}} = 0.6\overline{9}56\,\text{s}. \tag{19}$$

The overshoot rate

$$\%OS = \exp\left(\frac{-(4)(0.6856\,\text{s})}{2}\right)100\,\% = \left(0.2\overline{5}3\,802\right)100\,\% = 2\overline{5}.3802\,\%. \tag{20}$$

The setting time

$$T_s \approx \frac{8}{4} = 2\,\text{s}. \tag{21}$$

For the rise time, in the time domain, the step response

$$c(t) = \left(1 + \left(-\cos(\omega_d t) - \frac{a}{2\omega_d}\sin(\omega_d t)\right)e^{-(a/2)t}\right)u(t). \tag{22}$$

Assuming that $a > 0$, then the final value is s.t.

$$c(t) \xrightarrow{t\to+\infty} \left(1 + \left(-\cos(\omega_d t) - \frac{a}{2\omega_d}\sin(\omega_d t)\right)\cancel{\left(e^{-(a/2)t}\right)^0}\right)\cancel{u(t)}^1 = 1 = c_f. \tag{23}$$

Thus, the 90 % and 10 % values are $.9c_f = 0.9$ and $.1c_f = 0.1$ respectively.

Well $a = 4,\ \omega_d = \sqrt{21} \neq 0$. Using these known values, we can use the script in Appendix subsection A.1, we find the times corresponding to these values

$$\begin{bmatrix} t_{.9f} \\ t_{.1f} \end{bmatrix} = \begin{bmatrix} 0.388\,761\,\mathrm{s} \\ 0.096\,063\,\mathrm{s} \end{bmatrix}. \tag{24}$$

Thus, rise time $T_r = t_{.9f} - t_{.1f} = 0.388\,761\,\mathrm{s} - 0.096\,063\,\mathrm{s} = 0.292\,698\,\mathrm{s}$.

2.1.3 Plotting the poles and zeroes

In (9), we found that the denominator of transfer function $G_2(s)$ is $s^2 + as + b = (s + \hat{a})^2 + \omega_d^2$.

Thus the poles s s.t.

$$\begin{aligned} 0 &= (s + \hat{a})^2 + \omega_d^2 \\ &= (s + \hat{a})^2 - (j\omega_d)^2 \\ &= (s + (\hat{a} - j\omega_d))(s + (\hat{a} + j\omega_d)), \end{aligned} \tag{25}$$

that is

$$s_{1,2} = -\hat{a} \pm j\omega_d = -\frac{a}{2} \pm j\omega_d. \tag{26}$$

Substituting in the parameters, the poles

$$s_{1,2} = -\frac{4}{2} \pm j\sqrt{21} = -2 \pm j\sqrt{21}. \tag{27}$$

Now, although the poles and zeros may be plotted with the built-in Matlab function `pzmap`, which accepts a transfer function or system and plots its zeroes and poles, I have written the Matlab script in Appendix subsection A.2 showing how it works with results in Fig. 1.

(b) **2.1.4 Effect of the pole's real and imaginary parts**

On the parameters From (26) and since $|a| \leq 2\sqrt{b}$, $\omega_d \in \mathbb{R}$. So let $s_0 \in s_{1,2}$, then $s_{1,2} = \mathfrak{Re}(s_0) \pm j\mathfrak{Im}(s_0)$ and the real part of s_0,

$$\mathfrak{Re}(s_0) := \frac{s_1 + s_2}{2} = -\frac{a}{2}, \tag{28}$$

and the imaginary part

$$\mathfrak{Im}(s_0) := \left|\frac{s_1 - s_2}{2j}\right| = \omega_d = \sqrt{b - \hat{a}^2} = \sqrt{b - \left(\frac{a}{2}\right)^2}. \tag{29}$$

Thus, we have parameters a, b s.t. $a = -2\mathfrak{Re}(s_0)$, and

$$\begin{aligned} b &= \mathfrak{Im}^2(s_0) + \hat{a}^2 \\ &= \mathfrak{Im}^2(s_0) + (-\mathfrak{Re}(s_0))^2 \\ &= \left(\mathfrak{Re}^2 + \mathfrak{Im}^2\right)(s_0). \end{aligned} \tag{30}$$

Let $m, n \in \mathbb{R}$, and let $s_0^{'} \in s_{1,2}^{'} = m\mathfrak{Re}(s_0) \pm jn\mathfrak{Im}(s_0)$ s.t.

$$\begin{cases} \mathfrak{Re}(s_0^{'}) = m\mathfrak{Re}(s_0), \\ \mathfrak{Im}(s_0^{'}) = n\mathfrak{Im}(s_0). \end{cases} \tag{31}$$

Thus the new parameters,

$$\begin{cases} a^{'} = -2\mathfrak{Re}\left(s_0^{'}\right) = -2m\mathfrak{Re}(s_0), \\ b^{'} = \left(\mathfrak{Re}^2 + \mathfrak{Im}^2\right)(s_0^{'}) = \left((\tilde{m}\mathfrak{Re})^2 + \tilde{n}^2\mathfrak{Im}^2\right)(s_0). \end{cases} \tag{32}$$

Further, in comparison with the original parameters

$$\begin{cases} a^{'} = -2m\Big(-\dfrac{a}{2}\Big) = ma, \\ b^{'} = \left(\Big((m)\Big(-\dfrac{a}{2}\Big)\Big)^2 + n^2\left(\sqrt{b - \Big(\dfrac{a}{2}\Big)^2}\right)^2\right) = \big(m^2 - n^2\big)\Big(\dfrac{a}{2}\Big)^2 + n^2 b. \end{cases} \tag{33}$$

On step response characteristics From (11), we have peak time

$$\begin{aligned} T_p^{'} &= \frac{\pi}{\mathfrak{Im}(s_0^{'})} \\ &= \frac{\pi}{n\mathfrak{Im}(s_0)} \\ &= \frac{1}{n}\left(\frac{\pi}{\mathfrak{Im}(s_0)}\right) \\ &= \frac{T_p}{n} \end{aligned} \tag{34}$$

From (6), we have the rate of overshoot

$$\begin{aligned} \%OS^{'} &= \exp\left(\frac{-(a^{'})T_p}{2}\right)100\,\% \\ &= \exp\left(\frac{-(ma)T_p}{2}\right)100\,\% \\ &= \exp\left(\frac{-maT_p}{2}\right)100\,\%. \\ &= \left(\frac{\exp\left(\dfrac{-aT_p}{2}\right)100\,\%}{100\,\%}\right)^m 100\,\%. \\ &= \left(\frac{\%OS}{100\,\%}\right)^m 100\,\%. \end{aligned} \tag{35}$$

From (7), we have the settling time

$$T_s^{\prime} \approx \frac{8}{a^{\prime}} = \frac{8}{ma} = \frac{1}{m}\left(\frac{8}{a}\right) = \frac{T_s}{m}. \tag{36}$$

As stated before, the relationship between parameters a, b and therefore the poles of the transfer function and the rise time are too complicated to calculate in a general form.

Evaluations on the real part Suppose that we modified the poles, so that the imaginary part stays the same, and the real part is $2\times$. So $m = 2,\ n = 1$ and the roots $s_{1,2}^{\prime} = 2(-2) \pm j\sqrt{21} = -4 \pm j\sqrt{21}$, plotted in Fig. 1. Then we have the parameters

$$\begin{cases} a^{\prime} = ma = (2)(4) = 8, \\ b^{\prime} = \left(m^2 - n^2\right)\left(\dfrac{a}{2}\right)^2 + n^2 b = \left((2)^2 - (1)^2\right)\left(\dfrac{(4)}{2}\right)^2 + (1)^2(25) = 37. \end{cases} \tag{37}$$

Thus the new transfer function

$$G_2^{\prime}(s) := \frac{37}{s^2 + 8s + 37}, \tag{38}$$

which is used to plot the pole-zero map in Fig. 1, generated by the Matlab statement `pzmap(tf(29, [1 8 29]))`

We saw that peak time is independent of m. Thus $T_p^{\prime} = T_p = 0.6\overline{9}56\,\mathrm{s}$. Now the rate of overshoot

$$\begin{aligned} \%OS^{\prime} &= \left(\frac{\left(\left(0.2\overline{5}3\,802\right)\cancel{100\,\%}^{1}\right)}{\cancel{100\,\%}^{1}}\right)^{(2)} 100\,\% \\ &= \left(0.06\overline{4}\,415\right)100\,\% \\ &= 6.\overline{4}415\,\%. \end{aligned} \tag{39}$$

and the settling time

$$T_s^{\prime} \approx \frac{(2\,\mathrm{s})}{(2)} = 1\,\mathrm{s}. \tag{40}$$

Now for rise time, we have $a' = 8$ and $\omega_d' = \mathfrak{Im}(s_0') = \sqrt{21}$. Thus, the Matlab script in Appendix subsection A.1 produces the limits

$$\begin{bmatrix} t_{.9f}' \\ t_{.1f}' \end{bmatrix} = \begin{bmatrix} 0.411\,704\,\mathrm{s} \\ 0.082\,401\,\mathrm{s} \end{bmatrix}, \tag{41}$$

and rise time $T_r' = 0.411\,704\,\mathrm{s} - 0.082\,401\,\mathrm{s} = 0.329\,303\,\mathrm{s}$.

(c) 2.1.5 Effect of the pole's imaginary part

This time, let's modified the poles, so that the real part stays the same, and the imaginary part is $\frac{1}{2}\times$. So $m = 1,\ n = {}^1\!/_2$ and the roots $s_{1,2}'' = -2 \pm j({}^1\!/_2)\sqrt{21}$, plotted in Fig. 1. The parameters

$$\begin{cases} a'' = ma = (1)(4) = 4, \\ b'' = \left(m^2 - n^2\right)\left(\dfrac{a}{2}\right)^2 + n^2 b = \left((1)^2 - ({}^1\!/_2)^2\right)\left(\dfrac{(4)}{2}\right)^2 + ({}^1\!/_2)^2(25) = {}^{37}\!/_4. \end{cases} \tag{42}$$

So the transfer function is

$$G_2''(s) = \frac{{}^{37}\!/_4}{s^2 + 4s + \frac{37}{4}}. \tag{43}$$

From (34), the peak time

$$T_p'' = \frac{0.6\overline{9}56\,\mathrm{s}}{({}^1\!/_2)} = 1.\overline{4}712\,\mathrm{s}. \tag{44}$$

In (35), (36), we see that $\%OS$, T_s are independent of n. Thus $\%OS'' = \%OS = 2\overline{5}.3802\,\%$ and $T_s'' \approx T_s \approx 2\,\mathrm{s}$.

Again for rise time, we have $a'' = 4$ and $\omega_d'' = \mathfrak{Im}(s_0'') = ({}^1\!/_2)\sqrt{21}$. Thus, the Matlab script in Appendix subsection A.1 produces the limits

$$\begin{bmatrix} t_{.9f}'' \\ t_{.1f}'' \end{bmatrix} = \begin{bmatrix} 0.823\,409\,\mathrm{s} \\ 0.164\,802\,\mathrm{s} \end{bmatrix}, \tag{45}$$

and rise time $T_r'' = 0.823\,409\,\mathrm{s} - 0.164\,802\,\mathrm{s} = 0.658\,607\,\mathrm{s}$.

(d) **2.1.6 Effect of the natural frequency**

On the parameters As we defined earlier, $a = 2\zeta\omega_n$ and $b = \omega_n^2$. Let $\nu \in \mathbb{R}$ and let $\omega_n''' = \nu\omega_n$. Thus

$$\begin{cases} a''' = 2\zeta\omega_n''' = 2\zeta\nu\omega_n = \nu(2\zeta\omega_n) = \nu a, \\ b''' = (\omega_n''')^2 = (\nu\omega_n)^2 = \nu^2 b. \end{cases} \tag{46}$$

On step response characteristics From (5), (11), the damped frequency for the inverse Laplace transform,

$$\omega_d = \omega_n\sqrt{1-\zeta^2}. \tag{47}$$

So the new damped frequency

$$\omega_d''' = \omega_n'''\sqrt{1-\zeta^2} = (\nu\omega_n)\sqrt{1-\zeta^2} = \nu\omega_d. \tag{48}$$

From (5), we have peak time

$$T_p''' = \frac{\pi}{\omega_d'''} = \frac{\pi}{\nu\omega_d} = \frac{1}{\nu}\left(\frac{\pi}{\omega_d}\right) = \frac{T_p}{\nu}. \tag{49}$$

From (6), we have rate of overshoot

$$\%OS''' = \exp\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)100\,\% = \%OS. \tag{50}$$

Thus, we see that the natural frequency has no effect on the overshoot. However, from (7), we have settling time

$$T_s''' \approx \frac{4}{\zeta\omega_n'''} = \frac{4}{\zeta(\nu\omega_n)} = \frac{1}{\nu}\left(\frac{4}{\zeta\nu\omega_n}\right) = \frac{T_s}{\nu}. \tag{51}$$

Evaluations Suppose that we modify the natural frequency so that it is $4\times$. So $\nu = 4$. Thus we have parameters

$$\begin{cases} a''' = (4)(4) = 16, \\ b''' = (4)^2(25) = 400. \end{cases} \tag{52}$$

Thus the transfer function

$$G_2'''(s) = \frac{400}{s^2 + 16 + 400}. \tag{53}$$

So the peak time

$$T_p''' = \frac{(0.6\overline{9}56\,\mathrm{s})}{4} = 0.1\overline{7}14\,\mathrm{s}, \tag{54}$$

the rate of overshoot, which is independent of the natural frequency, $\%OS''' = \%OS = 2\overline{5}.3802\,\%$ and the settling time

$$T_s''' \approx \frac{2\,\mathrm{s}}{4} = 0.5\,\mathrm{s}. \tag{55}$$

From $a''' = 16$, $\omega_d''' = (4)\left(\sqrt{21}\right) = 4\sqrt{21}$, using the script in Appendix subsection A.1, we find the limits of the rise time

$$\begin{bmatrix} t_{.9f}''' \\ t_{.1f}''' \end{bmatrix} = \begin{bmatrix} 0.097\,190\,\mathrm{s} \\ 0.024\,016\,\mathrm{s} \end{bmatrix}. \tag{56}$$

Thus the rise time $T_r''' = 0.097\,190\,\mathrm{s} - 0.024\,016\,\mathrm{s} = 0.073\,175\,\mathrm{s}$.

As for the poles, from (3), (28),

$$\mathfrak{Re}(s_0) = -\zeta\omega_n, \tag{57}$$

and with (29), the poles

$$s_{1,2} = -\zeta\omega_n \pm j\omega_d. \tag{58}$$

So the new poles

$$s_{1,2}''' = -\zeta\omega_n''' \pm j\omega_d''' = -\zeta(\nu\omega_n) \pm j(\nu\omega_d) = \nu(-\zeta\omega_n \pm j\omega_d) = \nu s_{1,2}. \tag{59}$$

Thus $\mathfrak{Re}(s_0''') = \nu\mathfrak{Re}(s_0)$ and $\mathfrak{Im}(s_0''') = \nu\mathfrak{Im}(s_0)$.

2.2 Part 02 – Adding poles

As Nise (2015) warns, the step response formulas in 5, 6, 7 are designed only for second order transfer functions with no zeros. When a transfer function falls outside of those parameters, there are other techniques to analyze it.

(a) **2.2.1 Non-positive real pole with great magnitude**

If the extra pole is a real, non-positive $-\alpha_R \in \mathbb{R}_{\leq 0}$ and has a magnitude much greater than that of the real part of the second order poles, then the step response of the transfer function can be approximated by ignoring this extra real pole.

This is the case with adding the pole $-\alpha_R = -200$, since $+\alpha_R = 200 \gg 2 = |\mathfrak{Re}(s_0)|$ of the pole that we found in sub-subsection 2.1.3.

(b) **2.2.2 Positive real pole**

Now, if the pole is a real and positive pole $+\alpha_R \in \mathbb{R}_{>0}$, then this pole adds an addend $\frac{D}{s-\alpha_R}$ to the partial fraction decomposition as in

$$C(s) = \frac{A}{s} + \frac{B(s+\hat{a}) + C\omega_d}{(s+\hat{a})^2 + \omega_d^2} + \frac{D}{s-\alpha_R}. \tag{60}$$

In the time domain, this addend produces a growing exponential

$$c(t) = \Big(A + \big(B\cos(\omega_d t) + C\sin(\omega_d t)\big)e^{-\hat{a}t} + De^{\alpha_R t}\Big)u(t) \tag{61}$$

Now we know that

$$\Big(A + \big(B\cos(\omega_d t) + C\sin(\omega_d t)\big)\cancel{\big(e^{-\hat{a}t}\big)^0}\Big)\cancel{u(t)^1} \xrightarrow{t\to+\infty} A. \tag{62}$$

However

$$\big(De^{\alpha_R t}\big) \xrightarrow{t\to+\infty} +\infty, \tag{63}$$

that is, the growing exponential component goes to infinity. Thus so does $c(t)$ since $\lim_{h\to+\infty} A + h = \lim_{h\to+\infty} h$. Because of that, the new transfer function is unstable.

3 Results

3.1 Part 01 – Changing poles

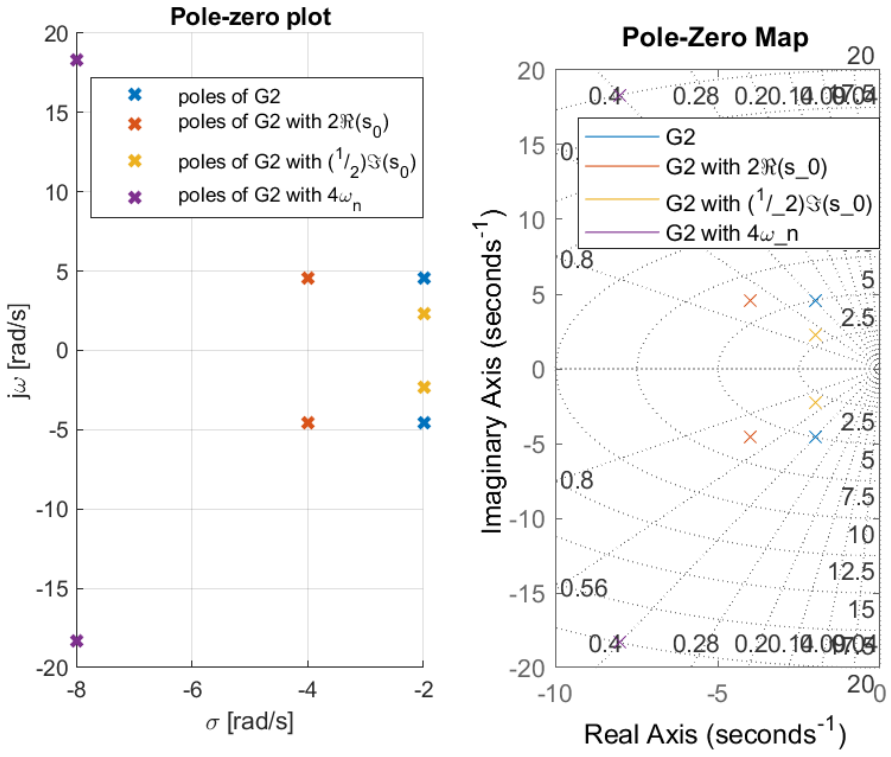

Figure 1: Comparison pole-zero plot of the transfer functions in Part 01 (subsection 2.1) using a custom plot function that I made on the left available in Appendix subsection A.2, and using Matlab's built-in `pzmap` on the right.

We can see in Fig. 1 that none of the variations of G_2 will have zeroes as G_2 did not either.

The poles with $\mathfrak{Re}(s_0') = 2\mathfrak{Re}(s_0)$ show that manipulating the real part of the poles scales the x-component of the pole.

The poles with $\mathfrak{Im}(s_0'') = (1/2)\mathfrak{Im}(s_0)$ show that manipulating the imaginary part of the poles scales the y-component of the pole.

In the Pole-Zero Map on the right in Fig. 1, we see that the poles with $\omega_n''' = 4\omega_n$ show that manipulating the natural frequency of the transfer function scales the magnitude of the pole.

Table 1: Proportional relationships of peak time, overshoot and settling time.

parameter	T_p'/T_p	$\frac{\log\left(\frac{\%OS'}{100\,\%}\right)}{\log\left(\frac{\%OS}{100\,\%}\right)}$	T_s'/T_s
$\mathfrak{Re}(s_0') = 2\mathfrak{Re}(s_0)$	1	2	$1/2$
$\mathfrak{Im}(s_0') = (1/2)\mathfrak{Im}(s_0)$	2	1	1
$\omega_n' = 4\omega_n$	$1/4$	1	$1/4$

Table 1 shows the parameter to modify if we want to change any characteristic of the step response. Particularly, we have shown that

(b) multiplying the real part exponentiates the overshoot and divides the settling time by the same amount, but does not effect peak time,

(c) dividing the imaginary part multiplies the peak time by the same amount, but does not effect overshoot or settling time, and

(d) multiplying the natural frequency divides peak time and settling time by the same amount, but does not effect overshoot.

Table 2: Numerical comparison of rise times from changing poles.

parameter	$t_{.9f}$ [s]	$t_{.1f}$ [s]	T_r [s]
reference	0.388 761	0.096 063	0.292 698
$\mathfrak{Re}(s_0') = 2\mathfrak{Re}(s_0)$	0.411 704	0.082 401	0.329 303
%error	5.9016 %	−14.2219 %	12.5061 %
$\mathfrak{Im}(s_0') = (1/2)\mathfrak{Im}(s_0)$	0.823 409	0.164 802	0.658 607
%error	111.8034 %	71.5562 %	125.0125 %
$\omega_n' = 4\omega_n$	0.097 190	0.024 016	0.073 175
%error	−75.0001 %	−74.9997 %	−74.9998 %

In Table 2, we see that for changing the real part, the upper and lower bounds of the rise time chance in opposite directions (positive and negative respectively). This suggests that manipulating the real part causes a widening of both margins for rise time. For changing the imaginary part, both bounds move up by varying amounts, with the rise time itself growing.

Finally, for the settling frequency, we see both bounds and the rise time itself all move by about the same amount (−75.0002 % by harmonic mean). This seems to be a uniform movement.

3.2 Part 02 – Adding poles

3.2.1 Non-positive real pole with great magnitude

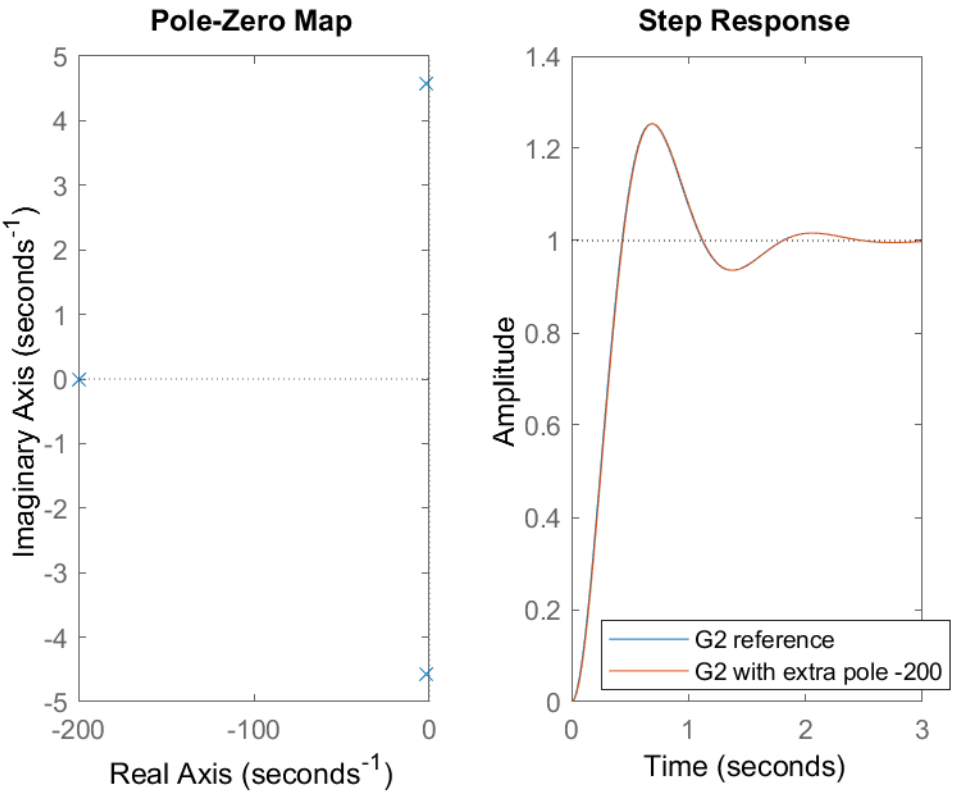

Figure 2: The pole-zero map of the new transfer function on the left, and the comparison of the step response of the original reference transfer function G_2 and the resulting transfer function of adding non-positive real pole $-\alpha_R = -200$ with great magnitude.

As predicted, we see in Fig. 2 that the resulting transfer function is very close to the original reference transfer function.

Table 3: Numerical comparison of reference function and the result of adding the pole $-\alpha_R$.

name	T_p [s]	$\%OS$	T_s [s]	T_r [s]
reference	0.6856	25.3802 %	2	0.292 698
$-\alpha_R$	0.6908	25.3746 %	1.6870	0.2934
%error	−0.7528 %	0.0221 %	18.5536 %	−0.2393 %

To find the transient response of the 3^o transfer function, we will use the Matlab function `stepinfo` because we do not have formulas to find the exact values as we did for 2^o underdamped systems.

Furthermore, Table 3 shows satisfactory errors for peak time, rate of overshoot and rise time. However, the difference for the settling time is quite high at 18.5 %. Since Fig. 2 is such a close fit, I believe that this may be due to differing definitions of the settling time. Both Nise (2015) and Matlab (2022) use a 2 % settling time. However, the difference may come down to the formula from Nise (2015) assuming a settling time from above and Matlab (2022) assuming a settling time from below.

3.2.2 Positive real pole

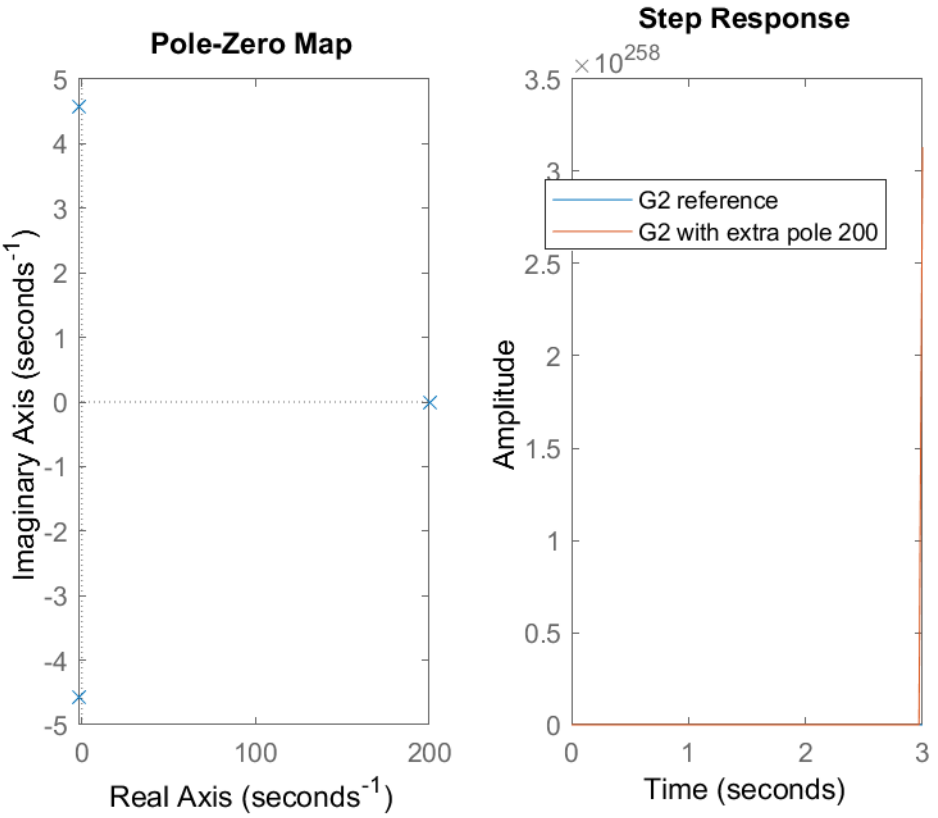

Figure 3: The pole-zero map of the new transfer function on the left, and the comparison of the step response of the original reference transfer function G_2 and the resulting transfer function of adding positive real pole $\alpha_R = 200$.

In this case, we saw the transfer function grow to 3.5×10^{258}, which would trend to positive infinity. Just as predicted, this shows that the new transfer function is unstable.

Further inspection of Table 4 shows that the step response has infinite peak time, indeterminate rate of overshoot, settling time and rise time. This is expected because the peak is never reached. As such the peak time would be infinite. The overshoot, settling time and rise time are questions that do

not make sense without a possible final value.

Table 4: Numerical comparison of reference function and the result of adding the pole $+\alpha_R$.

name	T_p [s]	$\%OS$	T_s [s]	T_r [s]
reference	0.6856	25.3802 %	2	0.292 698
$+\alpha_R$	$+\infty$	—	—	—
%error	$\frac{\infty}{\infty}$	—	—	—

4 Discussion

In this lab, we learned that manipulating the real and imaginary parts of the poles will effect their x- and y-components respectively, and that manipulating the natural frequency will effect the poles' magnitudes.

Regarding this, I learned that graphing in the polar coördinate system is not as well supported by Matlab as the Cartesian coördinate system. There is a `polarplot` function. However it does not support multiple plots and cannot be combined with a plot in Cartesian. Because of this, going forward, I will use `pzmap` instead; and if I need to use the more primitive facilities of polar coördinate system, then I will have to prefer to find an alternative system such as Python.

This lab also gave us the opportunity to test the hypothesis that a positive real pole will always be result in an unstable transfer function, whereas a transfer function where all poles only have non-positive real part will be stable. Additionally, we have learned that the poles with real part closest to 0 dominate. It is also a good time to remember that a pole that has $5\times$ the real part $-\zeta\omega_n$ of another pole will have very little significant effect in comparison. This former pole is 5τ away from the latter pole.

Now we can see that the rise time has no direct relationship with the roots or the natural frequency. Because of this, we had to use the Matlab equation solver to find the appropriate times at 10 % and 90 % final output, and from this calculate the rise times.

From Table 2, we can confirm no obvious relation for the components of the pole. However, there seems to be a scaling effect to the natural frequency. Let's investigate further.

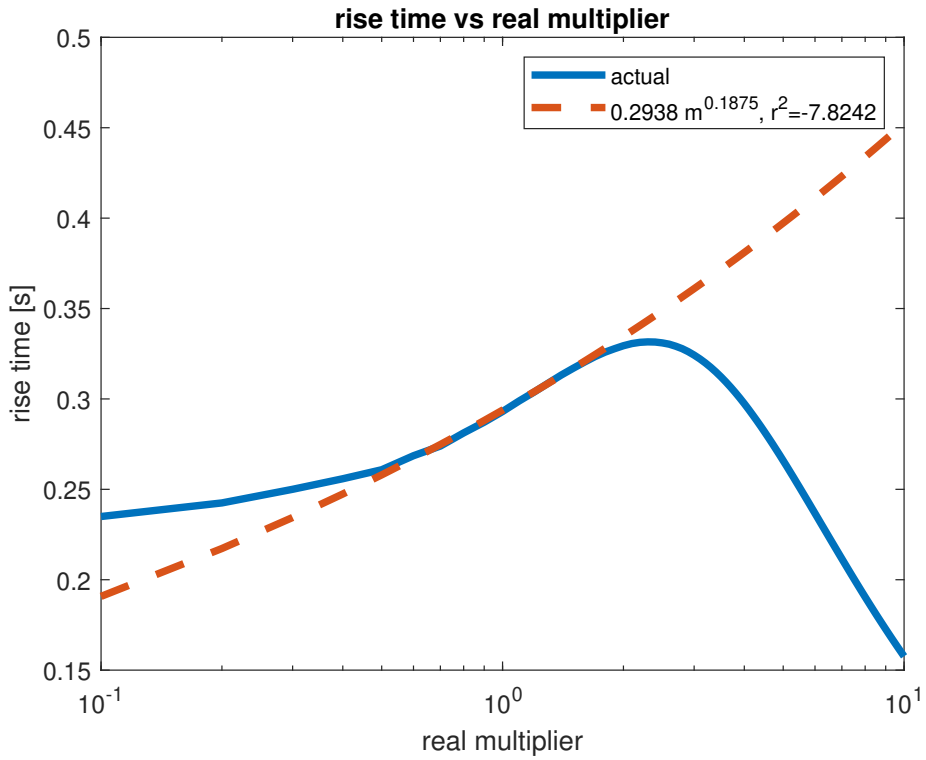

Figure 4: Plot in a semi-log scale on the x-axis showing the effect of multiplying the real component of the poles on the rise time and log-log linear regression.

The script in Appendix subsection A.4 plots the rise time vs the parameters that we have investigated in a semi-log scale on the x-axis from the reciprocal

of the modification to the modification. (For the imaginary part, this is the reverse because the modification was scaling by 1/2.) A log-log linear regression is taken from these, and then the entire plot is scaled by 5 on each side to allow for scaling.

This produces Fig. 4, 5 and 6.

Note that since the Matlab equation solver is time expensive and each plot will use 100 samples, we use the Matlab `stepinfo` function to find the rise time, albeit it does not the bounds of the rise time.

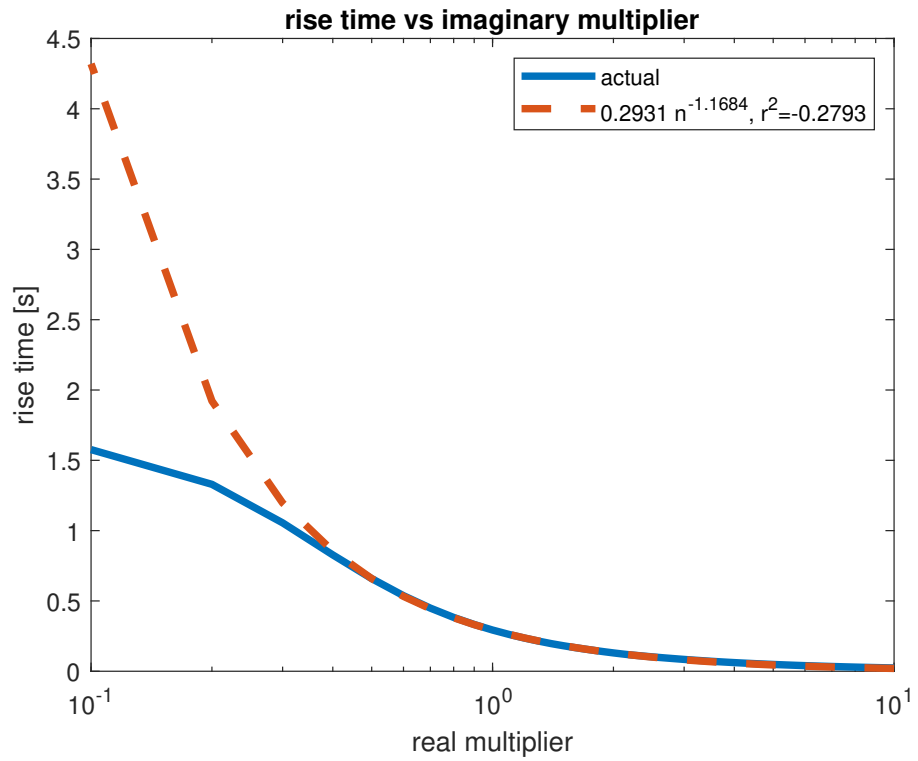

Figure 5: Plot in a semi-log scale on the x-axis showing the effect of multiplying the imaginary component of the poles on the rise time and log-log linear regression.

Now, the coefficients in [0.2938 0.2931 0.2930] represent the rise time of the original transfer function, which we found to be 0.292 698 sin sub-subsection 2.1.2. These values represent that rise time with deviation 0.000 856 51.

For modifying the real component, the linear regression (in Fig. 4) was only successful locally at $[0.3, 2.3]$ with $r^2 = 0.956\,348$. Thus, there does not seem to be a relation between the real component and rise time.

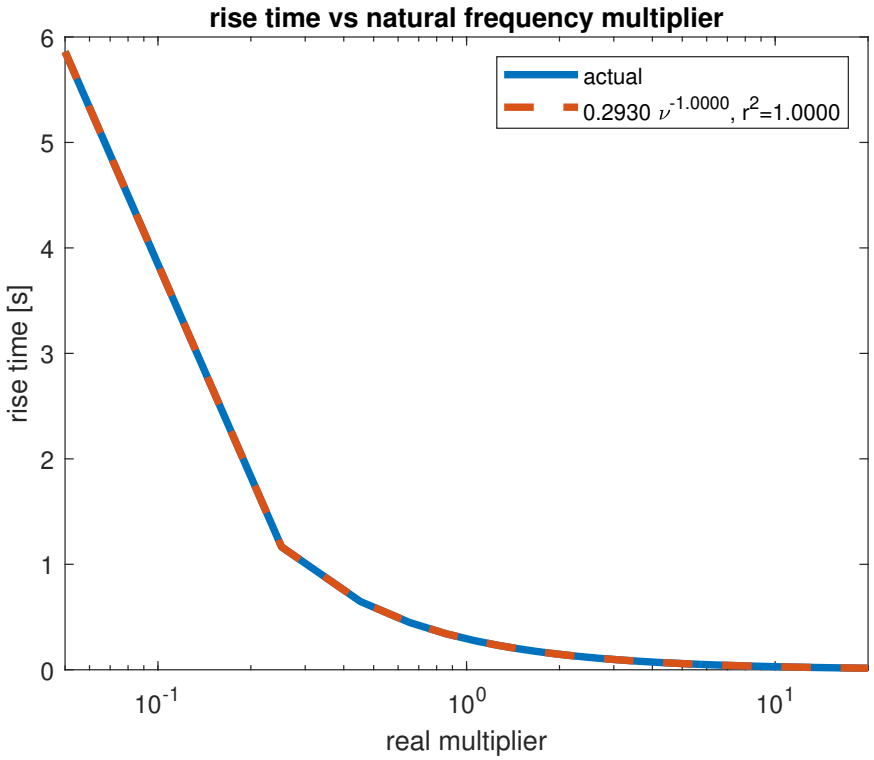

Figure 6: Plot in a semi-log scale on the x-axis showing the effect of multiplying the natural frequency on the rise time and log-log linear regression.

For modifying the imaginary component, the linear regression (in Fig. 5) is much better, having $r^2 = 0.991558$ in $[0.3, 9.8]$. I predict that after $n = 0.3$, the effect of multiplying the imaginary part by imaginary part of the pole by n will result in a division of the rise time by $n^{1.1684}$. In fact, taking the

original rise time 0.292 698 sand dividing by $(1/2)^1$.1684 gives 0.657 874 s. The calculated value for the rise time halving the imaginary part of the pole was 0.658 607 s. That's a rate of error of −0.111 295 %. Interestingly since the modification was to half the imaginary part, this prediction may not be so useful.

Earlier, we speculated that modifying the natural frequency uniformly moves the range of the rise time. In Fig. 6, we find an $r^2 = 1.0000$ from $(0.05, 19.3955)$. This appears to be an exact match! We can predict that multiplying the natural frequency by ν will result in dividing the rise time by ν. Now

$$\frac{T_r}{\nu} = \frac{0.292\,698\,\mathrm{s}}{4} = 0.073\,174\,5\,\mathrm{s} \approx 0.073\,175\,\mathrm{s} = T_r''' \tag{64}$$

with a negligible $-6.832\,93 \times 10^{-6}\,\mathrm{s}^0$ rate of error.

Moreover, it the natural frequency scales the time domain of a transfer function. Earlier we saw that changing the natural frequency $\omega_n' = \nu\omega_n$ results in new peak time $T_p' = {}^{T_p}/_{\nu}$ and new settling time $T_s' = {}^{T_s}/_{\nu}$, and now we have the relation $T_r' = {}^{T_r}/_{\nu}$ as well as the bounds $t_{.9f}' = {}^{t_{.9f}}/_{\nu}$, $t_{.1f}' = {}^{t_{.1f}}/_{\nu}$ from Table 2. In the case of overshoot, it makes sense that this would not be affected as overshoot is a characteristic of the range of step response, not the domain.

Thus, we see that generally it is the case that if c is a transfer function with natural frequency ω_n and c' is a transfer function that is almost equivalent, but with natural frequency $\nu\omega_n$, then

$$c'\left(\frac{t}{\nu}\right) = c(t). \tag{65}$$

References

Matlab. (2022). *Rise time, settling time, and other step-response characteristics: Matlab stepinfo*. The MathWorks, Inc.

Nise, N. S. (2015). *Control systems engineering* (Ed. VII). John Wiley & Sons, Inc.

A Appendix

A.1 Part 1 – Rise time, Matlab script

```
%% part01a_rise_time.m

% Calculates the rise times Tr given the following transfer function
% denominator linear coefficients, a, and damped frequencies, wd.
% By       : Leomar Duran
% Version  : v1.4.2
% For      : ECE 3413 Classical Control Systems
%
% CHANGELOG:
%       v1.4.2 - 2023-03-09t18:46
%           labeled damped frequency correctly
%
%       v1.4.1 - 2023-03-09t13:57
%           use zpk for system, rather than tf
%           added G2 to table
%
%       v1.4.0 - 2023-03-09t13:31
%           split from convpow
%
%       v1.3.0 - 2023-03-09t05:21
%           halfing Im part, t >= -1e-5 so it works
%
%       v1.2.0 - 2023-03-09t02:48
%           loop and table for multiple transfer functions
%
%       v1.1.0 - 2023-03-08t13:49
%           modeling G2
%
%       v1.0.0 - 2023-03-07t23:21
%           rise time script
%

clear
```

```
% Given the transfer function
%       G2(s) = ((a/2)^2 + wd^2)/((s + a/2)^2 + wd^2)
% s.t.

% linear coefficient of transfer function denominator
aVec = [4 8 4 16]';
% damped frequency
wdVec = [sqrt(21) ; sqrt(21) ; sqrt(21)/2 ; 4*sqrt(21)];

% set up time domain
syms t          % symbol for time [s]
% conditions for t: t must be non-negative
% We use -1e-5, rather than 0 because (4, sqrt(21)/2)[a,wd] does not
% find a t[.1f] when t >= 0. However, the t[.1f] found satisfies
% t >= 0.
assume(t >= -1e-5)

% number of transfer functions
nTfs = numel(aVec);
% allocate rise time limit matrix
TrLims = zeros(nTfs, 2);

% set up frequency domain
syms s
% allocate transfer function numerator and denominator
syms G2numerator G2denominator [3 1]

for k=1:numel(aVec)
    % the transfer function
    a = aVec(k)
    wd = wdVec(k)
    % calculate the roots
    s12 = (-a/2 + [1, -1]*j*wd)
    % calculate the transfer function
    G2 = zpk([], s12, prod(s12))
    % store numerator, denominator
    G2tf = tf(G2);
    G2numerator(k) = poly2sym(G2tf.Numerator{1}, s);
```

```
    G2denominator(k) = poly2sym(G2tf.Denominator{1}, s);

    % handle to function c(t)
    xC_t = @(t) 1 + (-cos(wd*t) - a/(2*wd) * sin(wd*t))*exp(-a/2 * t);
    % handle to inverse function c<-(t)
    xT_c = @(c) solve(c == xC_t(t), t);

    % find the final value of the step response
    cf = limit(xC_t, t, inf)

    % apply c<-(t) to cf*[.1, .9]
    TrLims(k,:) = arrayfun(xT_c, cf*[.1 .9]);
end % next k

% find rise time Tr = -t(.1)c + t(.9)c
Tr = TrLims*[-1 1]';

% create a table from the results
Tr_table = table(G2numerator, G2denominator, aVec, wdVec, TrLims, Tr)

disp("Done.")

%% end
```

A.2 Part 1 – Pole-zero plot, Matlab Live Script

```
%% Plotting the poles and zeroes.

% part01_pzplot_mlx.m
% Plots the poles and zeroes of the given transfer function
% using both a custom plot (pzplot) and the builtin pzmap.
% By      : Leomar Duran
% When    : 2023-03-09t20:11
% For     : ECE 3413 Classical Control Systems
%

clear
```

```
%%
% The transfer functions
G2 = [
    tf(25, [1 4 25]) ;
    tf(37, [1 8 37]) ;
    tf(37/4, [1 4 37/4]) ;
    tf(400, [1 16 400])
]
%%
% name the transfer functions
G2_name = [
    "G2" ;
    "G2 with 2\Re(s_0)" ;
    "G2 with (^1/_2)\Im(s_0)" ;
    "G2 with 4\omega_n" ;
]
%%
% get the number of transfer functions
nG2 = numel(G2)
%%
% convert to zero-pole-gain form
G2_zpk = zpk(G2)

%%
% Set up the figure.
figure
%%
% use pzplot for subplot #1
subplot 121

for k=1:nG2
    %%
    % We find the zeroes
    G2_zero = G2_zpk.Z{k}
    %%
    % and poles
    G2_pole = G2_zpk.P{k}
```

```
    %%
    % Next we plot these as points with the real part as the x-component
    % and the imaginary part as the y-component.

    % get (x, y) from zeroes
    G2_zero_x = real(G2_zero);
    G2_zero_y = imag(G2_zero);
    % get (x, y) from poles
    G2_pole_x = real(G2_pole);
    G2_pole_y = imag(G2_pole);

    % generate the legends
    G2pzLegends = ["zeroes of G2", "poles of G2"];
    % ignore either if we might not find any zeroes or poles
    G2pzLegendIdx = (cellfun(@numel, {G2_zero_x, G2_pole_x}) ~= 0);
    G2pzLegends = G2pzLegends(G2pzLegendIdx);

    % plot the zeroes, then poles
    hold on
    plot(G2_zero_x, G2_zero_y, 'o', ...
         G2_pole_x, G2_pole_y, 'x', ...
            'LineWidth', 2, ...
            'DisplayName', strcat("zeroes of ", G2_name(k)), ...
            'DisplayName', strcat("poles of ", G2_name(k)) ...
    )
    hold off
end % next k

% label the plot and grid
grid
title('Pole-zero plot')
xlabel('\sigma [rad/s]')
ylabel('j\omega [rad/s]')
legend(gca,'show')

%%
% use pzmap in subplot #2
subplot 122
```

```
for k=1:nG2
    hold on
    pzmap(G2(k))
    hold off
end % next k
grid
legend(G2_name(:))
```

A.3 Part 2 – Comparing results of adding poles, Matlab Script

```
%% part02_adding_poles.m
% Compares the resulting transfer functions of adding a nonpositive
% real root and a positive real root.
% By      : Leomar Duran
% When    : 2023-03-09t22:53
% For     : ECE 3413 Classical Control Systems
%

clear

%% The reference transfer function
G2_ref = tf(25, [1 4 25])

%% The new poles.
a_R = 200
% We also add an equivalent gain so that the new transfer function is
% within the original magnitude.
newPoles = [
    zpk([], [-a_R], a_R) ;
    zpk([], [+a_R], a_R)
]
% the number of poles
nPoles = numel(newPoles);

for k=1:nPoles
    % new reference figure
```

```
    figure(k)
    % create third order function
    G2_3o = (G2_ref * newPoles(k))
    % transient response
    stepinfo(G2_3o)
    % the poles and zeroes
    subplot 121
    pzmap(G2_3o)
    % step response of reference
    subplot 122
    step(G2_ref)
    % add the new pole to step response
    hold on
    step(G2_3o)
    hold off
    % label it
    legend("G2 reference", ...
        strcat("G2 with extra pole ", string(newPoles.P{k})) ...
    )
end % next k
```

A.4 Part 1 Discussion – Plotting rise time vs parameters, Matlab Script

```
%% part01_rise_time_vs_parameters_plot.m
% Plots the rise time against each parameter changed in part 01.
% By      : Leomar Duran
% When    : 2023-03-10t16:21
% For     : ECE 3413 Classical Control Systems
%

clear;

% reference parameters
a = 4
b = 25
```

```
% powers of parameter in approximation equation
pVec = [0.1875, -1.1684, -1]
% the coefficient for each approximation
cVec = [0.293804, 0.293134, 0.293041]

% parameters to analyze
paramDescs = ["real", "imaginary", "natural frequency"]
paramNames = ["m", "n", "\nu"]
% which parameter to update for each figure
paramIdxs = [1 2 3]
% the upper limit of each parameter (lower limit is reciprocal)
scale = 5
paramLims = [2*scale, (1/2)/scale, 4*scale]
% count of the parameters (for each figure)
figCount = numel(paramNames);

% allocate settling time
TrCount = 100
Tr = zeros(TrCount, figCount);
% multiplier of real part
for figIdx=1:figCount
    figure(figIdx)
    paramLim = paramLims(figIdx)
    % set up the input space
    ipSpace = linspace(1/paramLim, paramLim, TrCount);
    for TrIdx=1:TrCount
        % clear the parameters
        params = ones(1, max(paramIdxs));
        % use paramIdx to pick the next parameter
        params(paramIdxs(figIdx)) = ipSpace(TrIdx);
        m = params(1)
        n = params(2)
        nu = params(3)
        % modify polynomial by real and imaginary parts
        a2 = m*a;
        b2 = (m^2 - n^2)*(a/2)^2 + n^2*b;
        % modify polynomial by natural frequency
        a3 = nu*a2;
```

```
        b3 = nu^2*b2;
        % update the polynomial
        a2 = a3;
        b2 = b3;
        % find the rise time
        Tr(TrIdx,figIdx) = stepinfo(tf(b2, [1 a2 b2])).RiseTime;
    end % next TrIdx
    % plot the result
    semilogx((ipSpace), (Tr(:,figIdx)), 'LineWidth', 3)
    hold on
    p = pVec(figIdx)
    c = cVec(figIdx)
    TrHat = ipSpace.^p * c;
    semilogx((ipSpace), (TrHat), '--', 'LineWidth', 3)
    hold off
    TrMean = mean(Tr(:,figIdx))
    SSres = sum((Tr(:,figIdx) - TrHat').^2)
    SStot = sum((Tr(:,figIdx) - TrMean).^2)
    rsq = (1 - SSres/SStot)
    % label the plot
    title(strcat('rise time vs', " ", paramDescs(figIdx), ' multiplier'))
    xlabel('real multiplier')
    ylabel('rise time [s]')
    legend(["actual", ...
        sprintf('%.4f %s^{%.4f}, r^2=%.4f', ...
                c, paramNames(figIdx), p, rsq ...
        )] ...
    )
end % next figIdx
```