Velocity control
DC motor

Report

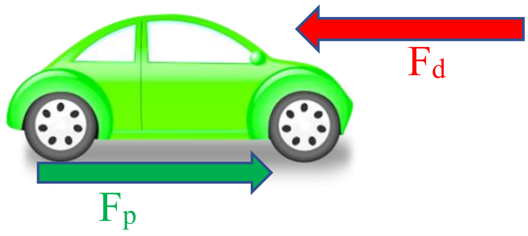
by

YA-CHE SHIH

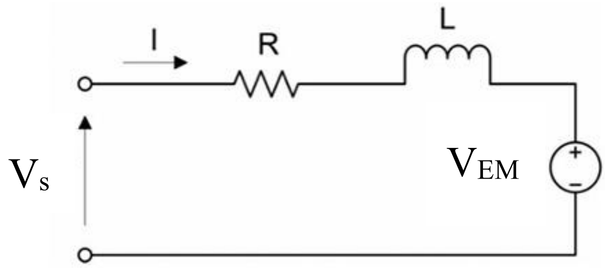
2022

Model

Toy car model:



DC motor:



Thus, we have equations:

$m(dV/dt) = F_p - F_d$

$F_d = 0.5 \cdot C_d \cdot \rho \cdot A \cdot V^2$

$V_s = I \cdot R + L(dL/dt) + V_{EMF}$

$V_{EMF} = K_e \cdot \omega$

$\tau_{DC} = K_t \cdot I$

$F_p \times r = \tau_p$

$V = \omega r / G_r$

$\tau_p = \tau_{DC} \cdot G_r$

m - toy car mass

V - car velocity

C_d - Drag Coefficient

ρ - air density

A - Front area of car

V_s, I, R, L - Armature Voltage, current, resistance and inductance of DC motor

V_{EMF} = Back EMF of DC motor

K_e, K_t = motor coefficients

τ_{DC}, τ_p = torque provided by DC motor, torque provided by wheels

r = radius of car wheels

G_r = Gear ratio

By combining all equations above, we have

$d\tau_{DC}/dt = (Kt/L)\cdot(V_s - R\tau_{DC}/K_t - K_e\omega)$

$d\omega/dt = (G_r/mr)\cdot(\tau_{DC}G_r/r - C_d\rho A\omega^2r^2/2G_r)$

Linearization

Linearize above system at $\omega = \omega_0$, and output Velocity, we'll have

$$\dot{x} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_eK_t}{L} \\ \frac{G_r{}^2}{mr^2} & -\frac{C_d\rho Ar\omega_0}{G_rm} \end{bmatrix}\cdot x + \begin{bmatrix} \frac{K_t}{L} \\ 0 \end{bmatrix}\cdot V_s$$

$$y = \begin{bmatrix} 0 & \frac{r}{G_r} \end{bmatrix}\cdot x + 0\cdot V_s$$

Notice that A_{44} is really small no matter what value of ω_0 is. Thus, we can write our linearized system in state space:

$A = \begin{bmatrix} -2 & 0 \\ 4069 & 0 \end{bmatrix}$ $\qquad$ $B = \begin{bmatrix} 0.046 \\ 0 \end{bmatrix}$

$C = \begin{bmatrix} 0 & 0.0128 \end{bmatrix}$ $\qquad$ $D = \quad 0$

Then compute system transfer function from Voltage to Velocity by linearized system above:

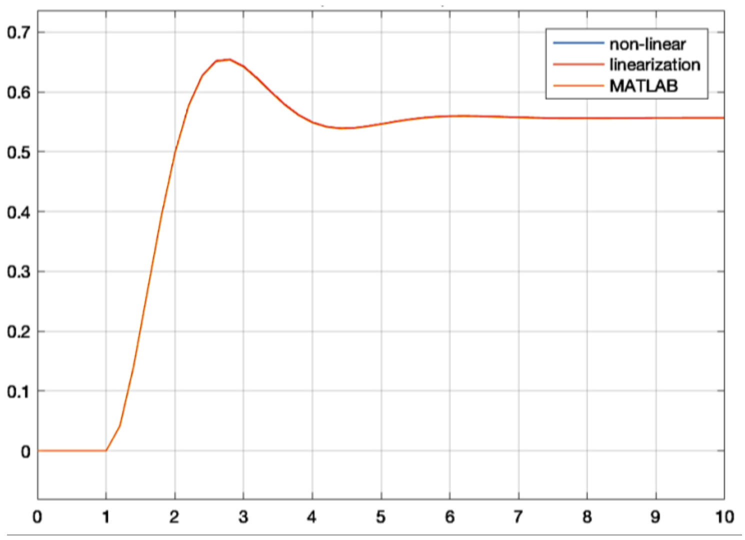
Transfer function by linearization= $\frac{2.398}{s^2+2.004s+4.314}$

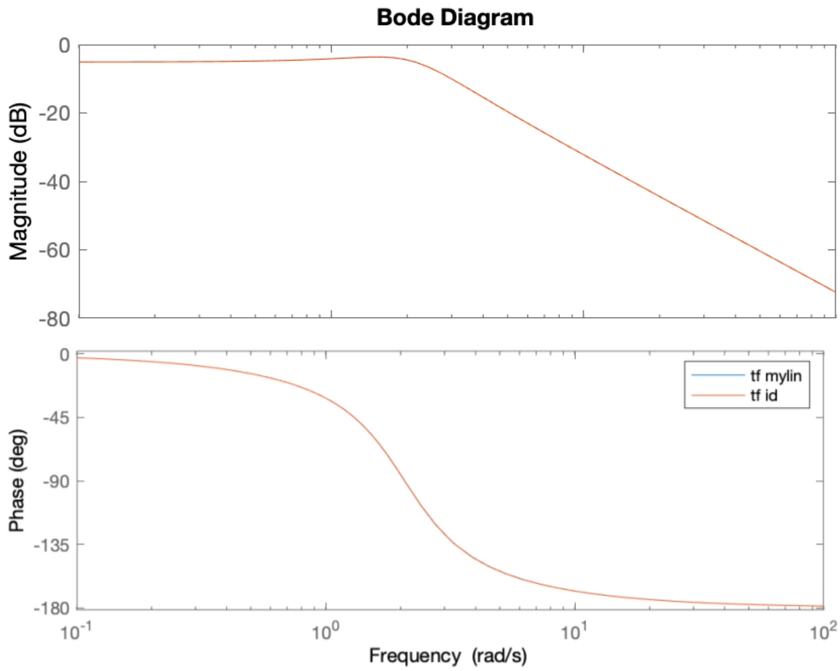
System Identification

By applying step input to our non-linear system, and then use System Identification in MATLAB app, we get a transfer function that is computed by MATLAB.

Transfer function by MATLAB = $\frac{2.396}{s^2+2s+4.305}$

Now we have three system, tf_MATLAB, tf_linearization and non-linear system. Notice transfer function by MATLAB and linearization looks alike. To check transfer function behaves like non-linear system





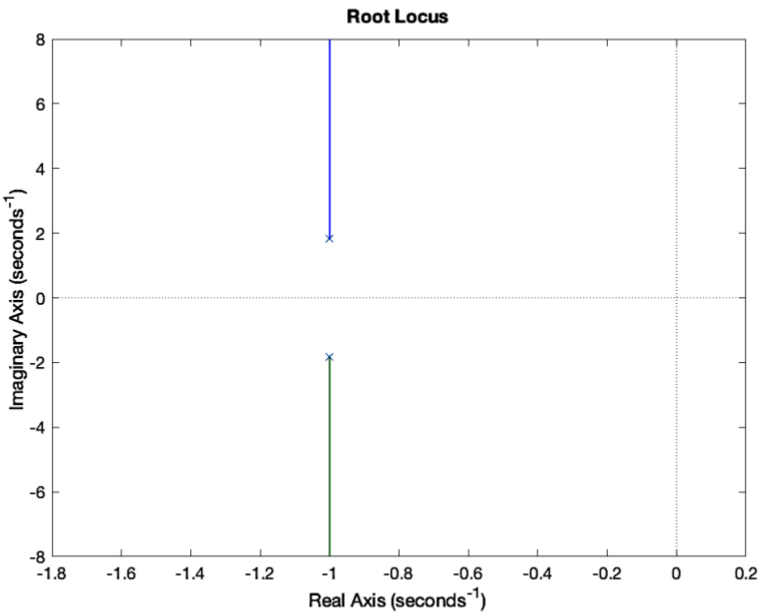
It's hard to tell there are more than one lines in the plots since they behave almost the same, which is a great news.

In the following, I'll use tf_sys representing system transfer function = $\frac{2.396}{s^2+2s+4.305}$

PID

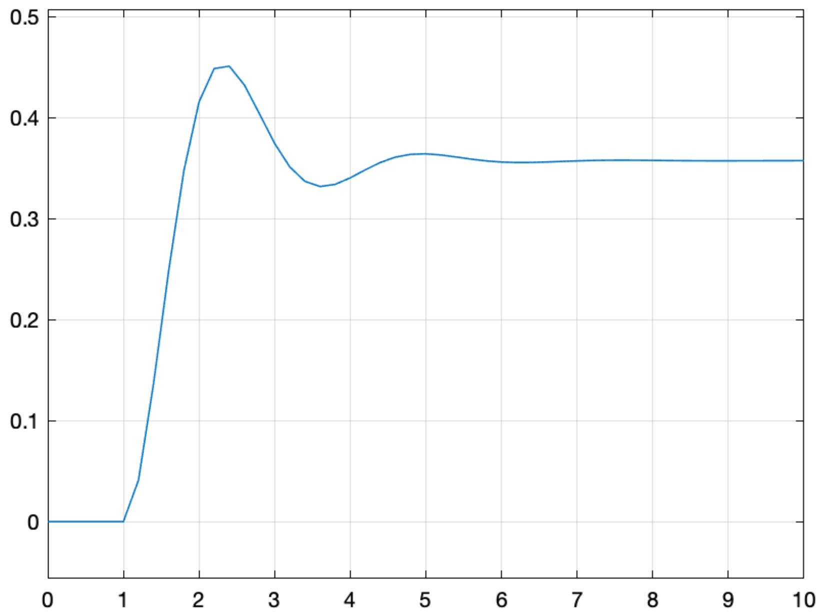
P controller

First, we can plot root locus to see how gain affects system's poles and zeros



From the plot above, notice that we can arbitrarily assign gain K_p since all poles will still remain in LHP.

Therefore, we let $K_p = 1$, and see how the system behaves after applying P controller.



After input a step input, notice output velocity doesn't go to 1, this is because our system with P controller has a final value according to K_p. We can find the property by looking at transfer function.

Transfer function

$$tf\_p = \frac{K_p{*}2.396}{K_p{*}2.396 + s^2{+}2{*}s + 4.305}$$

$$\lim_{s\to 0} tf\_p = \frac{K_p{*}2.396}{K_p{*}2.396 + s^2{+}2{*}s + 4.305} = \frac{K_p{*}2.396}{K_p{*}2.396 + 4.305}$$

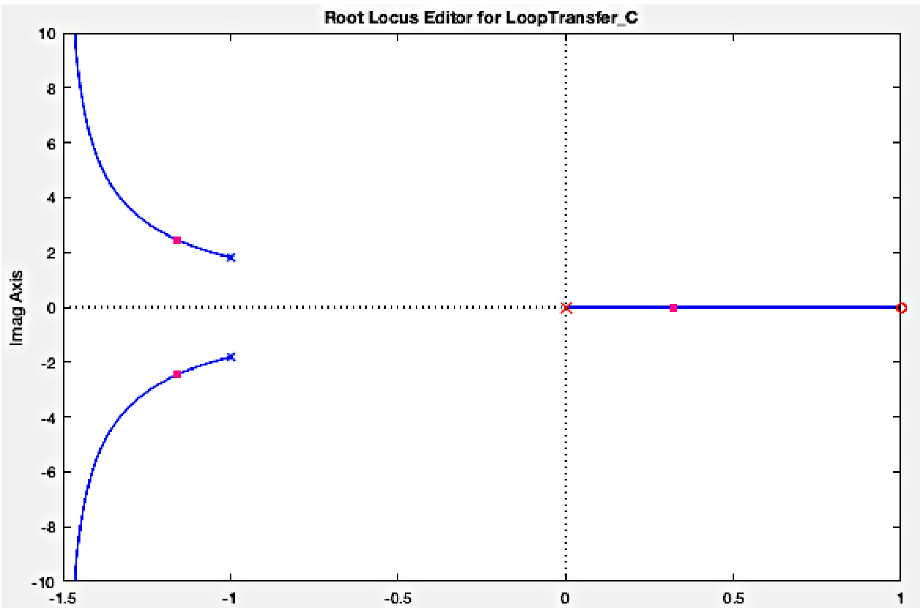
Notice that no matter what K_p we choose, steady state value can never go to 1. Following the value we choose above, $K_p = 1$, we have steady state value = 0.3576. Therefore, even out system is stable, its output doesn't match our requirements. To solve the problem, let's attempt PI controller.

PI controller

For PI controller, we can write its transfer function $K_p + \frac{K_i}{s}$. The transfer function can also be written by $\frac{K_p(s+a)}{s}$, where a is Ki/Kp.

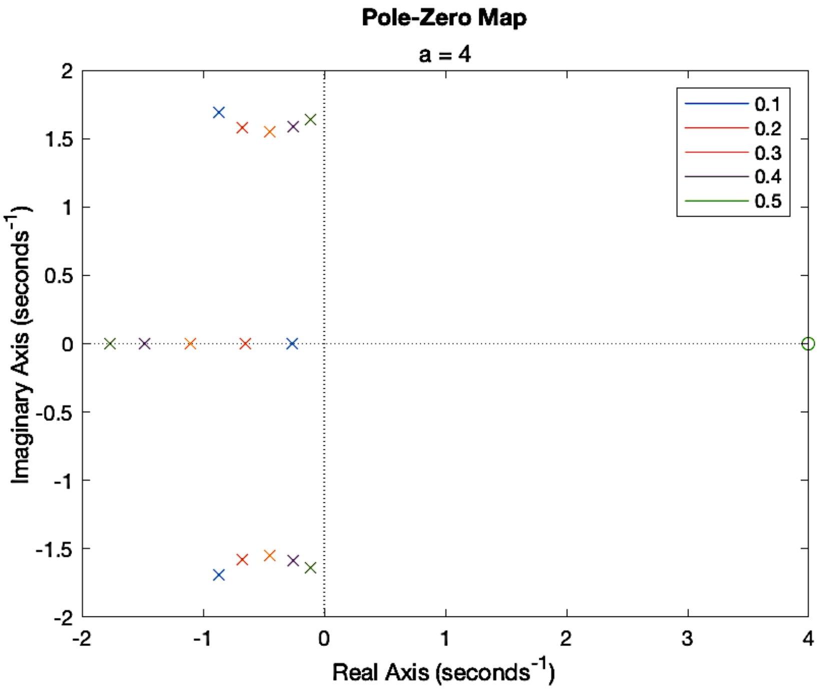
Assign a positive zero

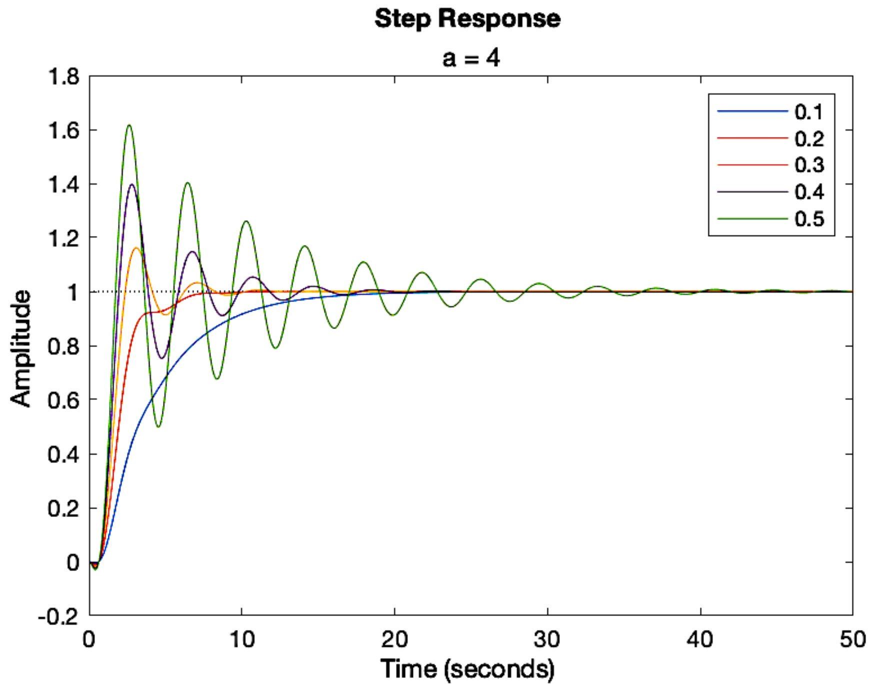
First, I simply assigned $K_p = 1$, a = 1 to see the performance of the system.



Notice that there is a zero on the right-hand side of pole at zero, which means no matter what value the K_p is, the system can never be stable. Therefore, I let K_p be negative. Now, the transfer function of PI controller is $\frac{K_p(s+a)}{s}$, with KP < 0. I plot several graphs to see the effects of K_p and a.

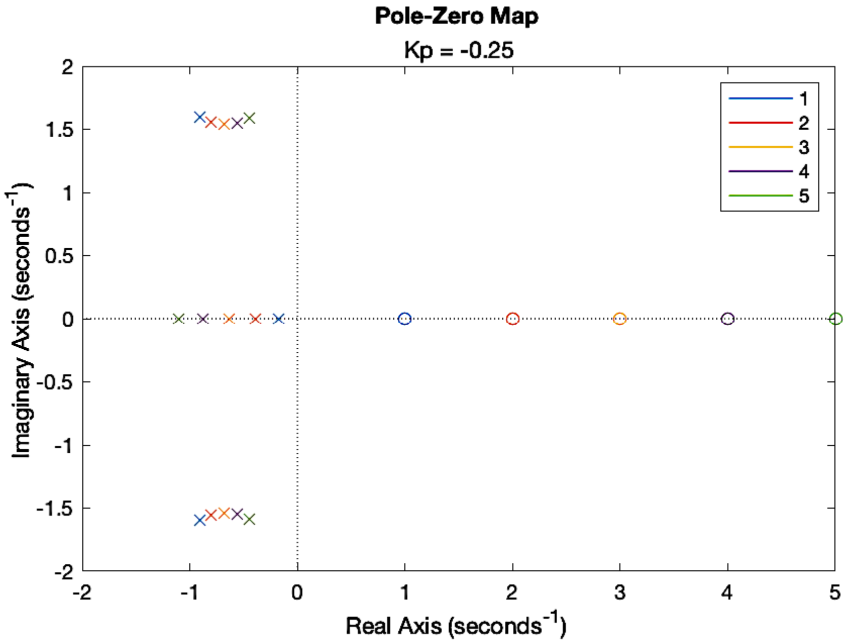
Adjust K_P

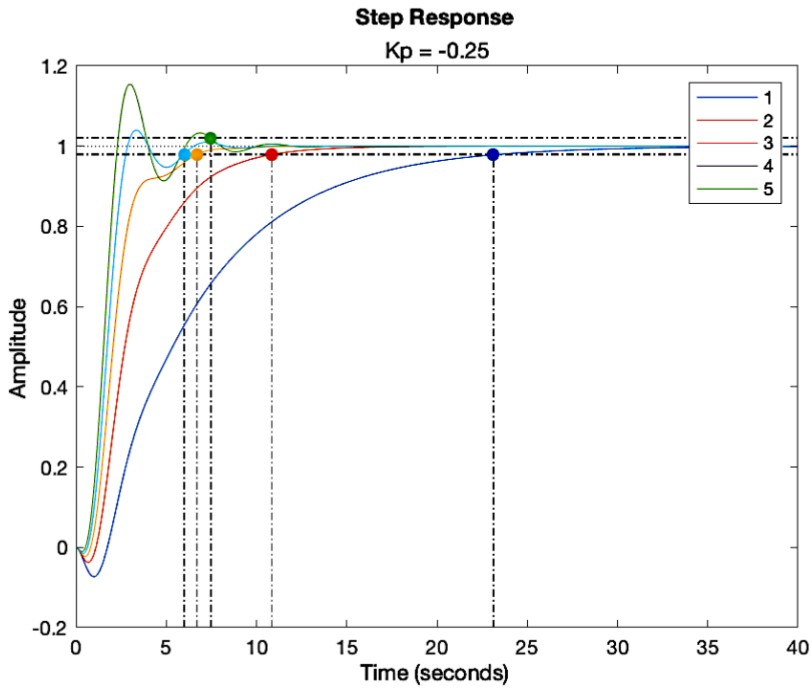




Notice as K_P goes from -0.1 to -0.5, the shorter the response time, but the larger the over shoot. Besides, there is a lag in the beginning of step response.

Adjust a





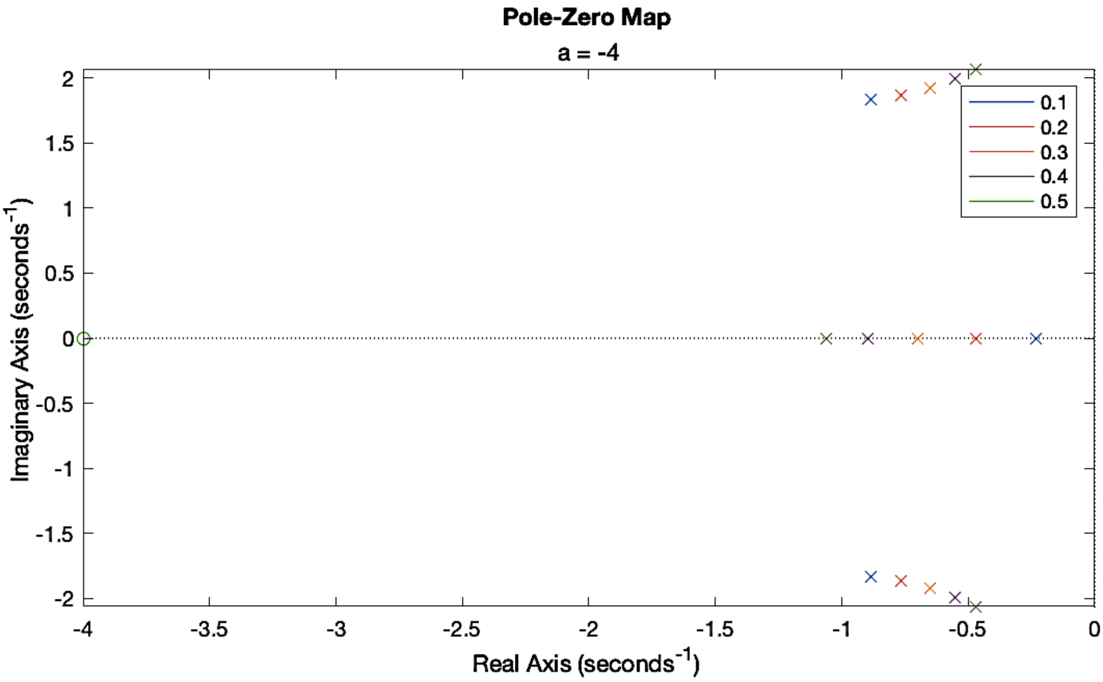
Notice the smaller the a is, the more obvious the negative response at the beginning. Besides, although larger a makes system response quicker, overshoot percentage is also larger. By all those plots, I choose a = 4, since it has the shortest settling time and the least overshoot.

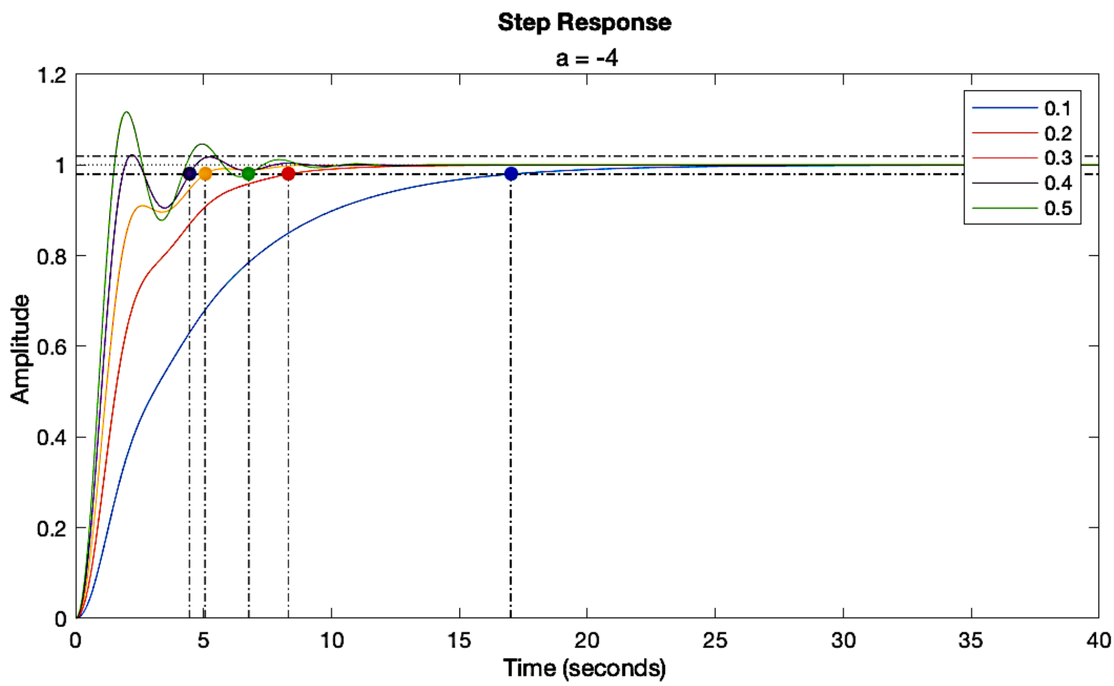
Transfer function

$$\frac{-0.25(s-4)}{s}$$

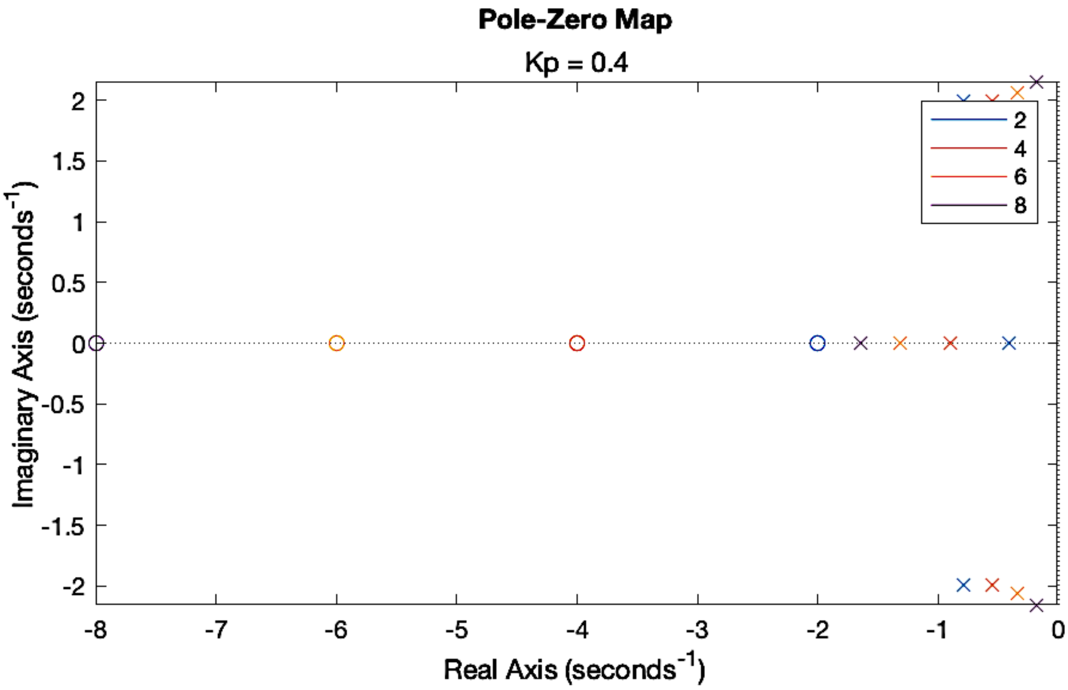
Assign a negative zero

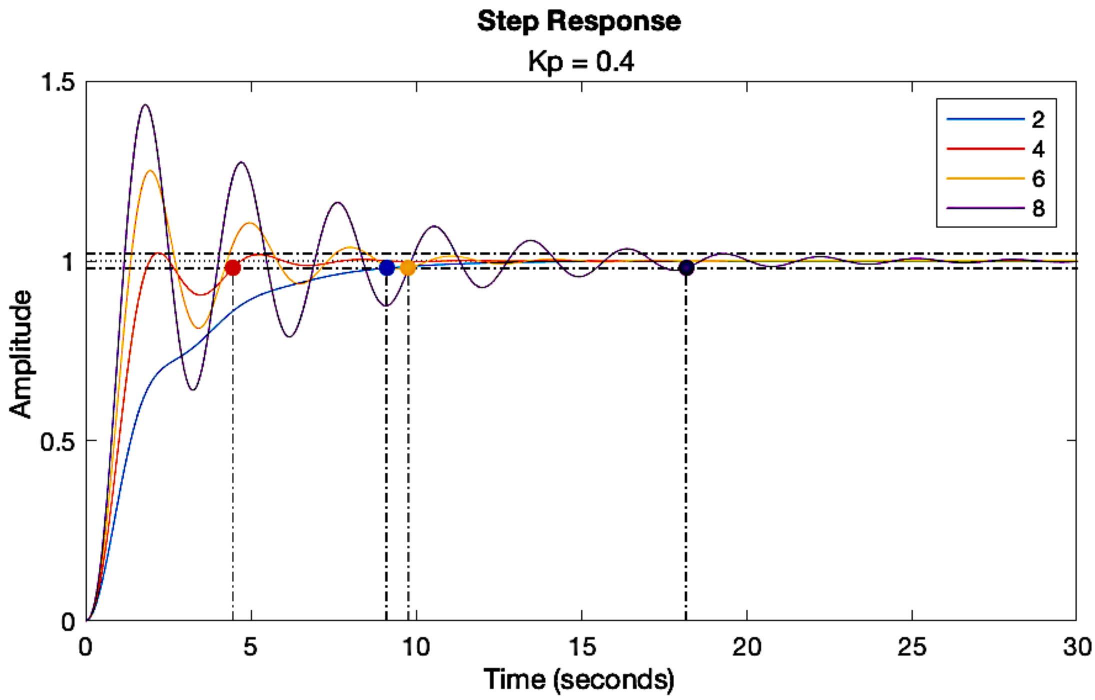
Adjust K_p





Adjust a





By assigning negative zero, there is no more lag in the beginning of step response, and settling time is a little bit shorter than positive zero. Therefore, I choose my PI controller to be this one.

Transfer function

$$\frac{0.4(s+4)}{s}$$

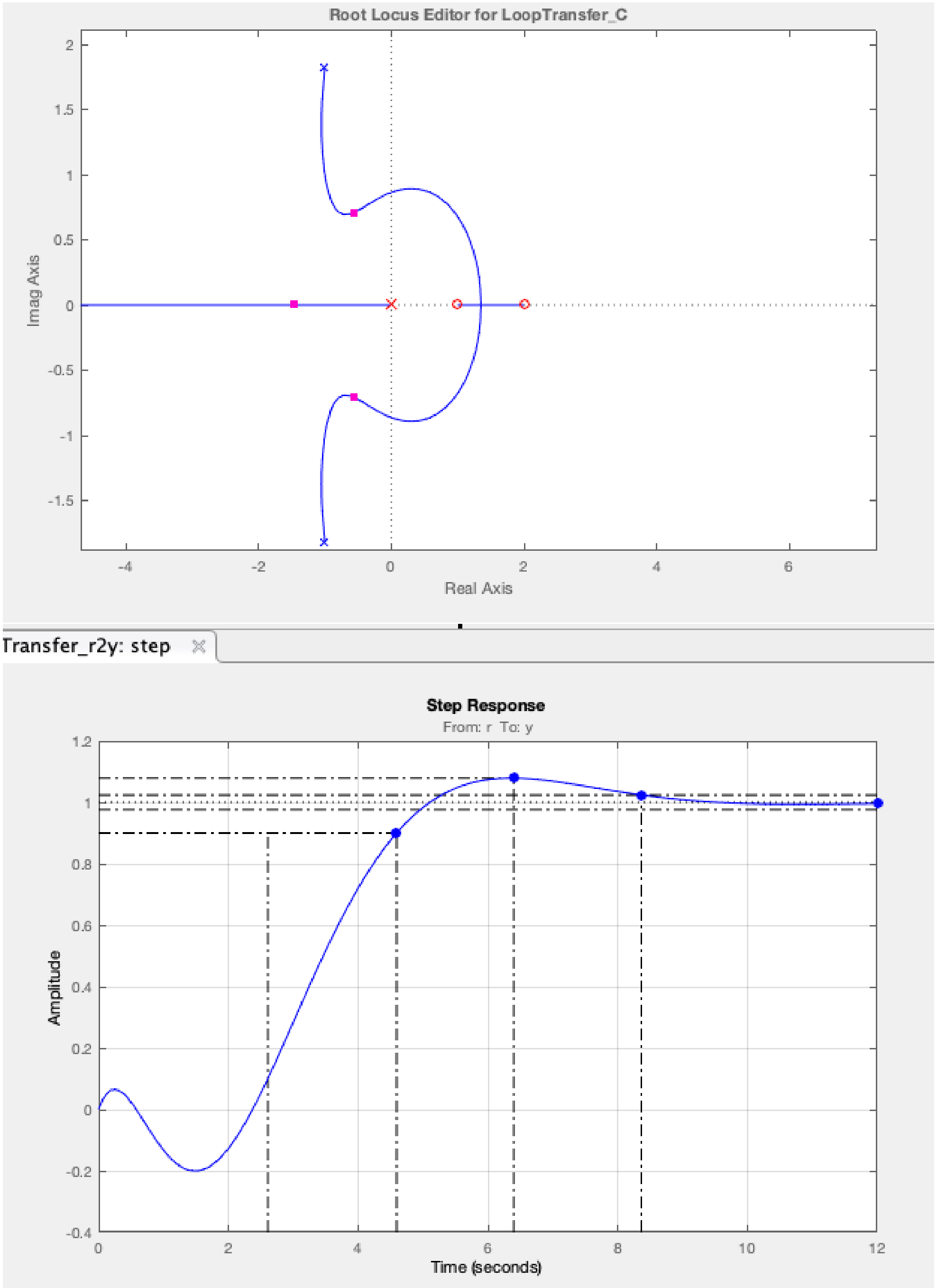
PID controller

Transfer function of PID controller is $K_P + \frac{K_i}{s} + K_D s$, it can also be represented in $\frac{K_D(s^2+as+b)}{s}$, where a = KP/KD, and b = K_i/K_D.

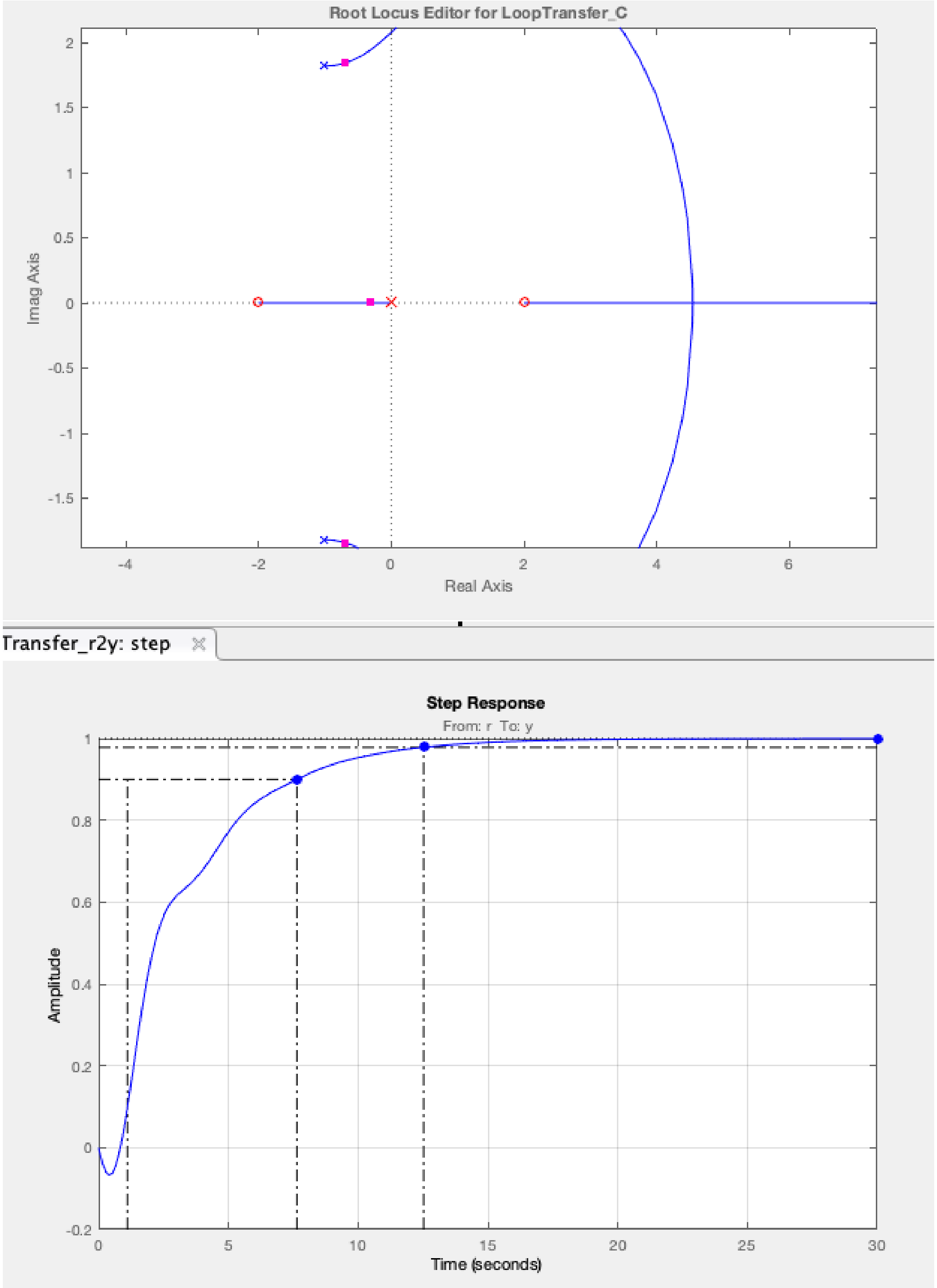
Positive zeros

No matter there is a positive zero or two positive zeros, lag in the beginning of step response is inevitable.

zero = 1, 2



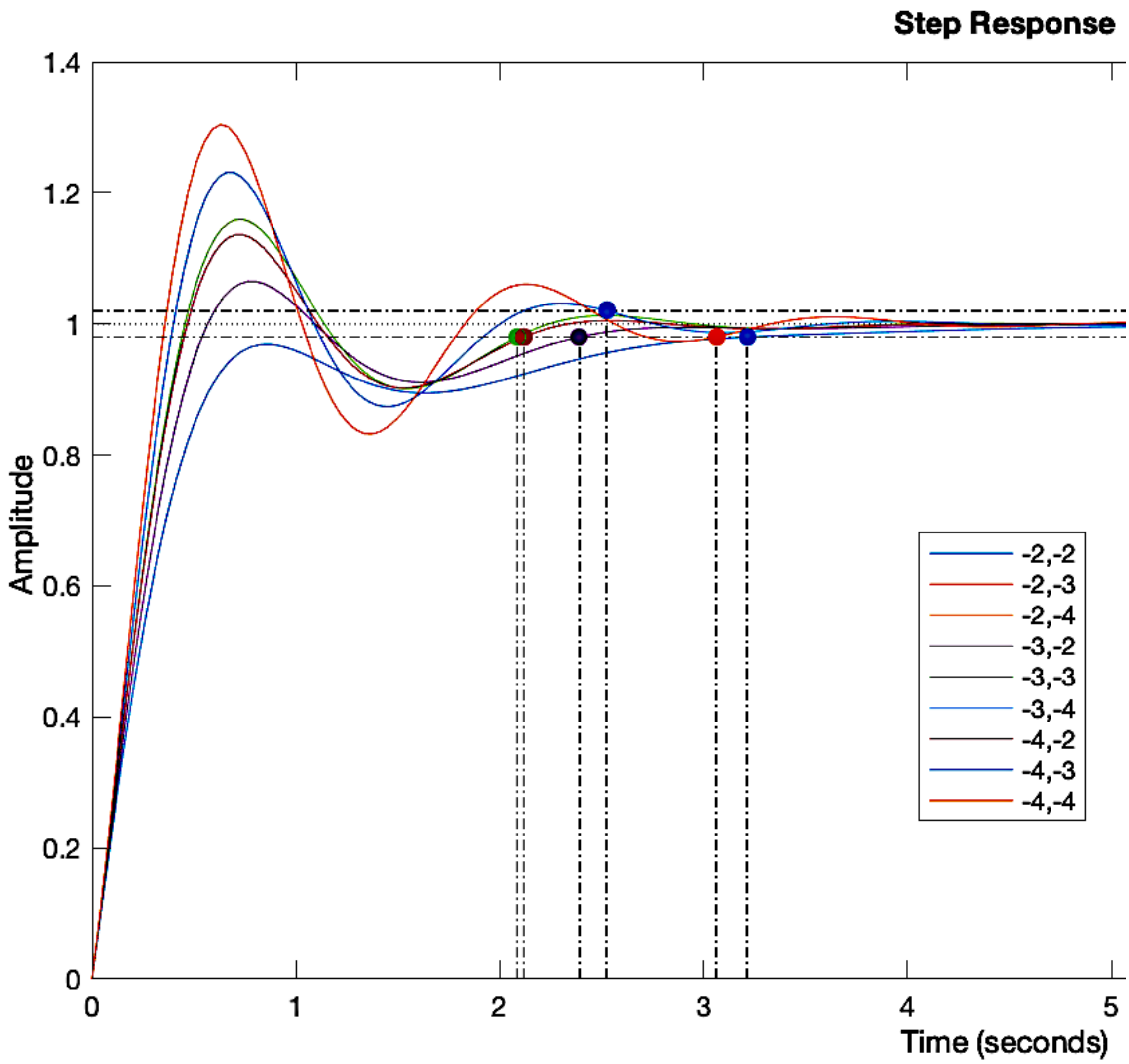
zero = 2, -2



To prevent from starting lag, I assign two zeros to be negative.

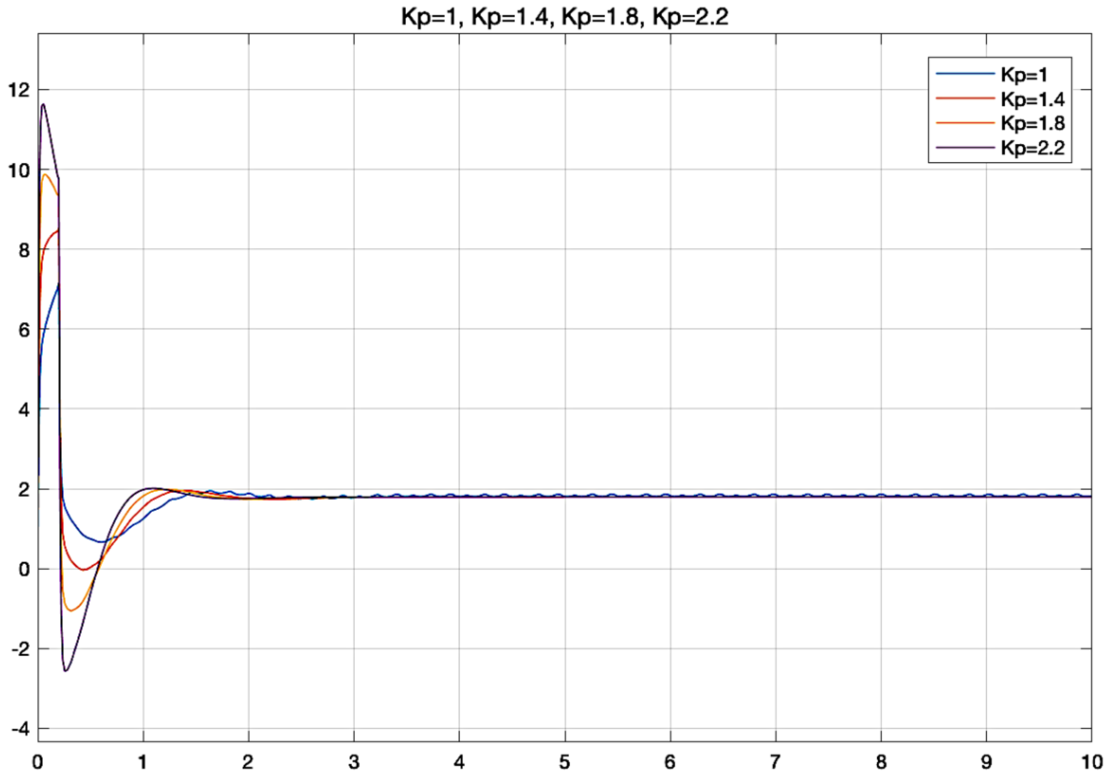
Negative zeros

Since we get to choose K_p, a and b, I tried a lot of values. The following plot is the values that close to our desired behavior.



I chose the values that have the shortest settling time before, but I noticed although (-4, -2) has the shortest settling time, its overshoot is much higher than (-3, -2). Besides, (-3, -2) settling is little bit longer than (-2, -4). Therefore, I chose (-3, -2) to be the zeros of PID controller.

Now we get to choose K_p, it's the value that relate to how much voltage will input to our DC motor. Let's assume we want our toy car at least has acceleration 5 m/s^2 within the limitation of battery. The battery's maximum output voltage is 12V.



Since $K_p = 2.2$ is the maximum gain that can meet our request, our PID controller is

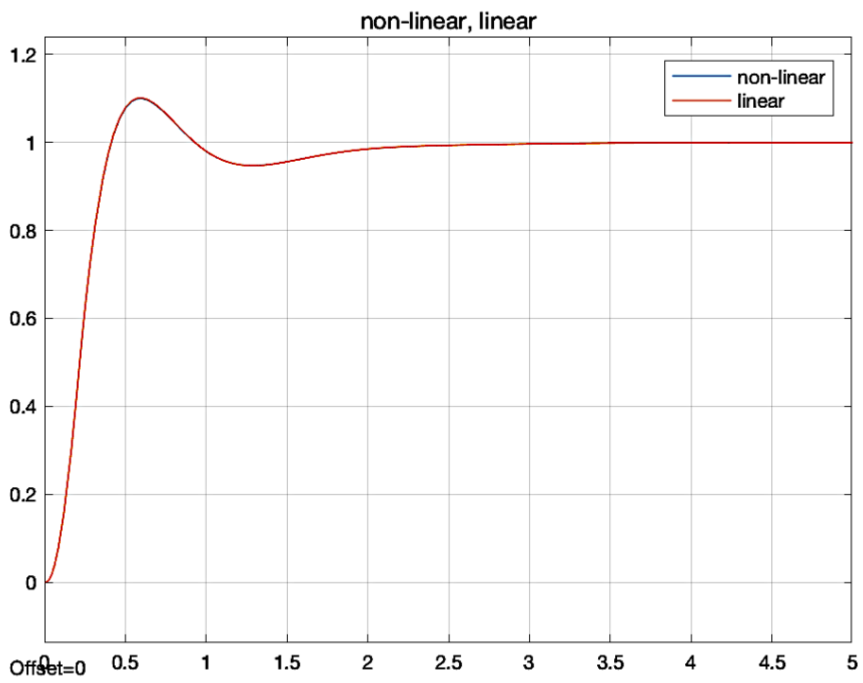
Transfer function

$$\frac{2.2(s+2)(s+3)}{s} = 11 + 13.2/s + 2.2s$$

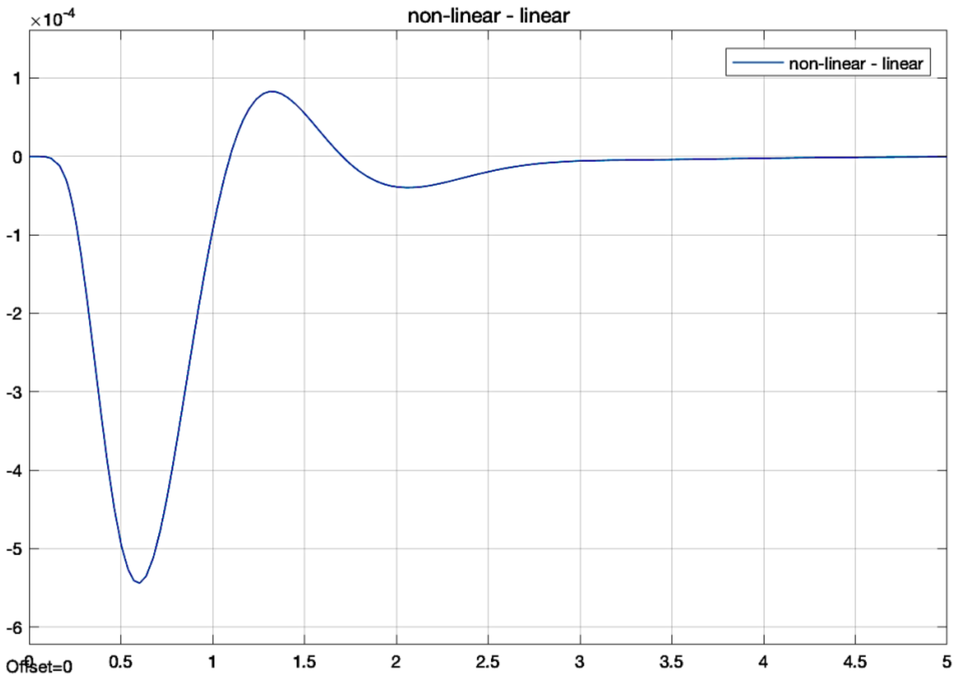
Apply PID controller into non-linear system

To make sure the PID controller we designed by linearized system can work as well in non-linear system, I plot the following figures.

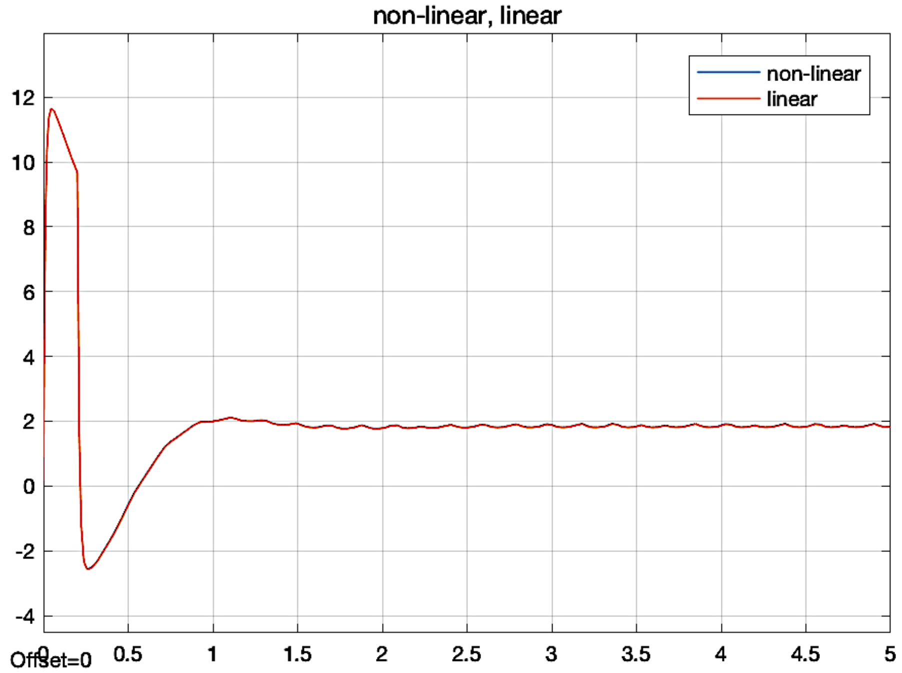
System performance



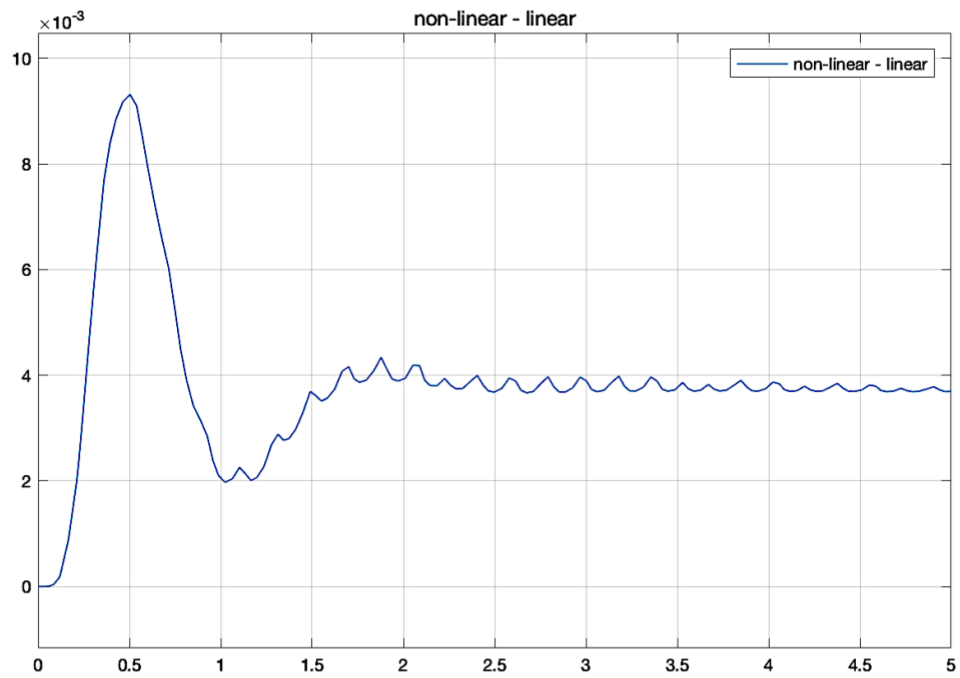
State error between nonlinear and linear



Voltage performance



Voltage error between nonlinear and linear



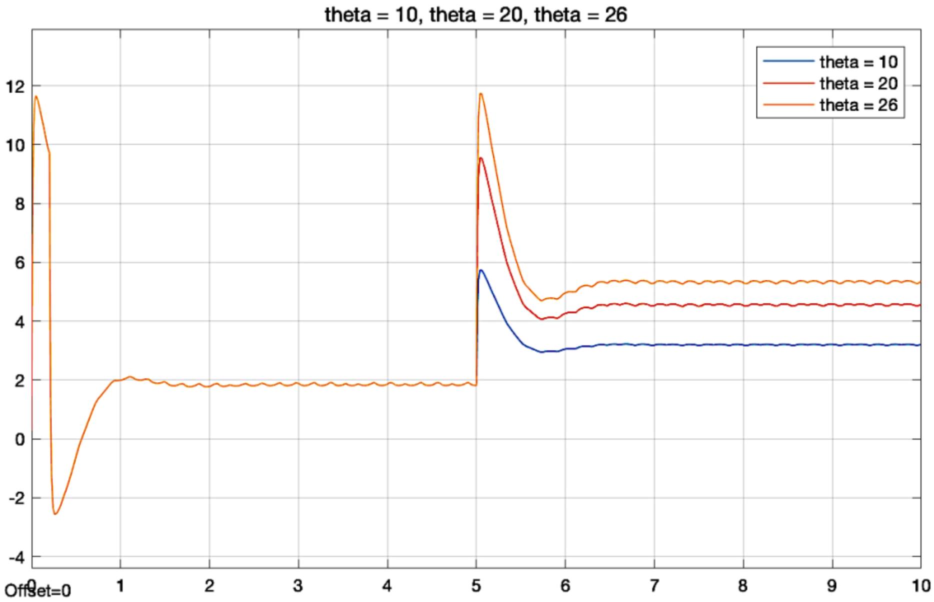
Input θ

After having our PID controller, I'd like to see if it can adjust DC motor when going to an uphill or a downhill. Besides, I'd like to see how much angle can our PID controller controls.

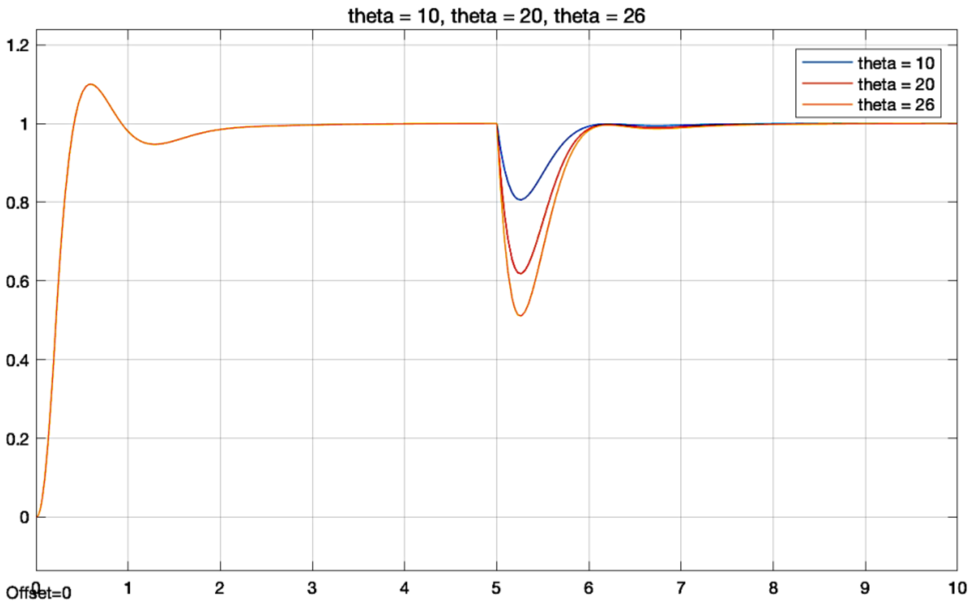
Uphill

Assume the car is in steady state (Velocity = 1m/s), and at t=5s, it meets an uphill, let's see how angle affects car's velocity and voltage input.

Voltage

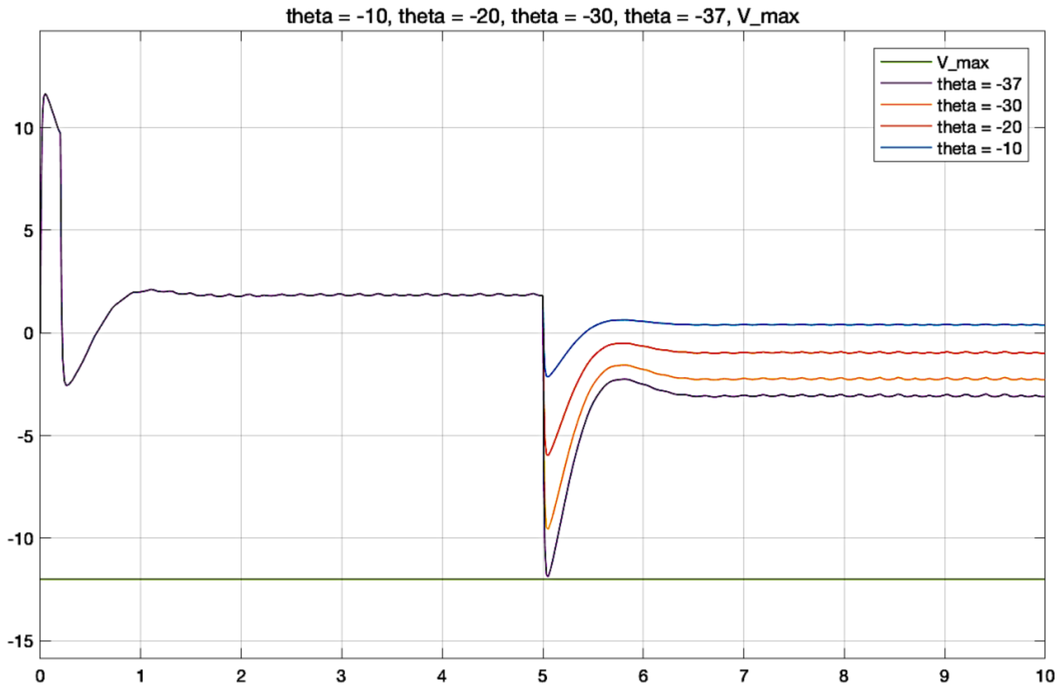


Velocity



Downhill

Voltage



Velocity