A quick guide to PID Feedback Control

HUAN Q. BUI

July 2, 2021

1 An Example and Theory

Consider a 1D harmonic oscillator with natural resonance frequency ω_0 and damping parameter β, driven by some function $y(t)$. Its motion follows the following second-order ordinary differential equation:

$$\ddot{x}(t) + \beta\dot{x}(t) + \omega_0^2 x(t) = \omega_0^2 y(t).$$

There are many ways to solve this ODE, but in the context of control theory, we will proceed using the Fourier transform (or the Laplace transform). The point of the FT is twofold: (1) we convert a differential equation into an algebraic equation, and (2) we from the time domain to the frequency domain, where analysis becomes more "intuitive" (eventually, anyway).

In any case, define the FT of $x(t)$ as

$$x(\omega) = \int \frac{dt}{2\pi} e^{i\omega t} x(t).$$

We note that factors of 2π don't really matter here since we now take the FT of both sides of the ODE to find

$$-\omega^2 x(\omega) + i\beta\omega x(\omega) + \omega_0^2 x(\omega) = \omega_0^2 y(\omega). \tag{1}$$

Here, we have used the following identity:

$$\mathcal{F}\left[\frac{d^n}{dt^n} x(t)\right] = (i\omega)^n x(\omega).$$

From (1), we can define $G(\omega)$, the **impulse response function** (in frequency domain) as the ratio of output $x(\omega)$ over input $y(\omega)$:

$$G(\omega) \equiv \frac{x(\omega)}{y(\omega)} = \frac{1}{(\omega_0^2 - \omega^2) + i\beta\omega}$$

We can justify why G is called as such by considering the impulse input $y(t) = y_0\delta(t)$ and find that $G(\omega) = x(\omega)/y_0$. More commonly, $G(\omega)$ is also known as

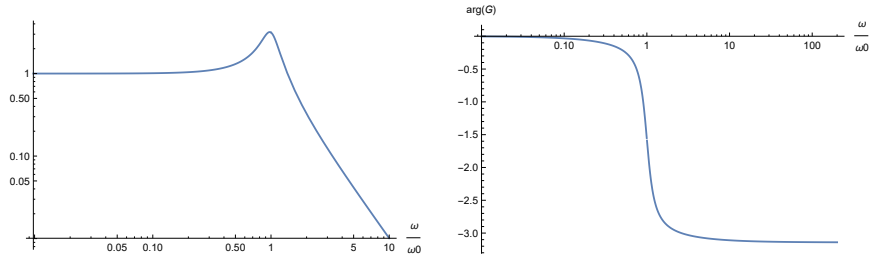

Figure 1: Bode plots for the transfer function G with $\beta = 2, \omega_0 = 2\pi$. The plot on the left is the amplitude Bode plot; the plot on the right shows the phase. Notice how $|G|$ exceeds **unity** when $\omega \approx \omega_0$ (resonance!). Notice also on the phase Bode plot how the phase lags behind as ω grows and exceeds ω_0.

the **transfer function** of the system.[1]

We have basically solved the problem: given some input $y(\omega)$, we can inverse-FT to find $x(t)$ from $x(\omega)$ via $G(\omega)$. Now, suppose we want to control the position of the oscillator so that $x(t)$ follows some desired position $r(t)$ as faithfully as possible. How do we do this?

A first guess is setting the driving function to be the desired position, i.e., $y(t) = r(t)$. In this case, we have

$$x(\omega) = G(\omega)r(\omega).$$

A convenient way to see how the system may respond to any driving function is to study the transfer function $G(\omega)$ using the **amplitude** and **phase Bode plots**. The former is almost always a log-log plot, while the latter can either be a linear-linear or log-linear plot. For our system, we can generate these plots (Figure 1) in Mathematica using the following Mathematica code.

```
\[Beta] = 2; \[Omega]0 = 2*Pi;

G[\[Omega]_] := \[Omega]0^2/((\[Omega]0^2 - \[Omega]^2) +
I*\[Beta]*\[Omega]);

LogLogPlot[Abs[G[a*\[Omega]0]], {a, 0, 10}, PlotRange -> Full,
AxesLabel -> {\[Omega]/"\[Omega]0", Abs[G]}]

LogLinearPlot[Arg[G[a*\[Omega]0]], {a, 0.01, 200},
AxesLabel -> {\[Omega]/"\[Omega]0", Arg[G]}]
```

To continue our analysis, we consider different regimes for ω:

- For low frequencies ($\omega \ll \omega_0$), the system follows the drive quite well: $|G| \approx 1$ and the phase lag $\arg(G) \approx 0$.

[1] There might be some confusion here regarding whether to use the Fourier or Laplace transform. Here's the resolution: If we're using the FT, then evaluate G at $\omega \in \mathbb{R}$. If we're using the LT, then evaluate G at $i\omega$.

- However, as we approach the resonance frequency ω_0 we become "too slow" because both $|G|$ and $\arg(G)$ deviate from "unity," and $x(\omega)$ no longer follows $r(\omega)$ well.

- On resonance ($\omega = \omega_0$), the amplitude becomes larger than that of the reference:

$$|x(\omega_0)| = |G(\omega_0)||r(\omega_0)| = \frac{\omega_0}{\beta}|r(\omega_0)| = Q|r(\omega_0)|,$$

  where Q is the **quality factor** of the oscillator. Meanwhile, the phase lags behind by

$$\arg(G(\omega_0)) = \arg\left(-i\frac{\omega_0}{\beta}\right) = \frac{\pi}{2}.$$

- For higher frequencies, we can take some limits and find that the phase lags by π and the amplitude decreases like

$$|x(\omega)| \to \frac{\omega_0^2}{\omega^2}|r(\omega)|.$$

  Because of this, we see a drop in frequency response in the amplitude Bode plot when $\omega \gg \omega_0$. The asymptote here is ω_0^2/ω^2, which appears on the log scale as a straight line.

So, we have a problem: using $y(t) = r(t)$ itself as a driving function is only good when ω is very small. But there's a way around this (otherwise this article wouldn't exist). For our system to follow $r(t)$ more faithfully, the driving function must also take into account the current position $x(t)$ **relative** to the reference $r(t)$. This is **feedback**: we control our system based on not just the reference but also the current state of the system. Feedback is nice because it detects if the system lags behind or if its amplitude is too large or too small and tries to compensate accordingly. Furthermore, it allows us to correct for **noise**, which appears in all real system. While we can simply let some function $\xi(t)$ represent noise, we won't worry about it for the forthcoming part of our analysis.

What we have to do now is instead of naively setting $y(t) = r(t)$, we will first define some **error signal** $\epsilon(t)$ and try to minimize it. The error signal is simply how far away our system is from the reference:

$$\epsilon(t) = r(t) - x(t).$$

With the error signal, we can now define the driving function $y(t)$ such that $y(t)$ changes the system to minimize $e(t)$. This is where the three letters **P-I-D** come in. Intuitively, we want $y(t)$ to do the following things:

- If $\epsilon(t)$ is large (large error), then $y(t)$ should also be large (large correction), and vice versa. So, we can set $y(t) = K_p\epsilon(t)$. This is **P**, which stands for **Proportional**. Here K_p is called the **proportional gain**.

- If $\epsilon(t)$ has been large *so far* (still far away from reference), then $y(t)$ must also be large (move faster towards reference), and vice versa. We can set $y(t) = K_i \int_0^t \epsilon(t')\,dt'$. This is **I**, which stands for **Integral**. Here, K_i is called the **integral gain**. [2]

- If $\epsilon(t)$ is changing fast, then we're still far away from the reference, so we make $y(t)$ large to approach the reference faster. So we set $y(t) = K_d\dot{\epsilon}(t)$. This is **D**, which stands for **Derivative**. Here, K_d is called the **derivative gain**.

Putting everything together, we have the **PID control function**

$$y(t) = K_p\epsilon(t) + K_i\int_0^t \epsilon(t')\,dt' + K_d\frac{d}{dt}\epsilon(t).$$

Now, this is just another ODE. And just we before, we can go to Fourier space to solve it. Here, our system is the PID controller, and the input is the driving function. Inspired by our previous approach, we can also define an impulse response function $K(\omega)$ for the PID controller:

$$y(\omega) = \left(K_p - i\frac{K_i}{\omega} + iK_d\omega\right)\epsilon(\omega) \equiv K(\omega)\epsilon(\omega),$$

where we have used the following identity of the FT:

$$\mathcal{F}\left[\int_{-\infty}^t \epsilon(t')\,dt'\right](\omega) = \frac{-i}{\omega}\epsilon(\omega).$$

So, we have $x(\omega) = G(\omega)y(\omega)$ and $y(\omega) = K(\omega)[r(\omega) - x(\omega)]$. Solving for $x(\omega)$, i.e., "closing the loop," we find that

$$x(\omega) = \frac{K(\omega)G(\omega)}{1 + K(\omega)G(\omega)}r(\omega) \equiv T(\omega)r(\omega),$$

which is now completely independent of $y(\omega)$ and $\epsilon(\omega)$ and is dependent only on the reference and ω. This is what we want. Note how we have chosen the error signal such that there is a $(-)$ sign in front of $x(t)$. This is so that the error signal goes in the direction which reduces the error.

Our system is now a combination of the "bare" system with the oscillator and the PID controller. Depending on K_i, K_p, K_d we may or may not have resonance. The goal of PID tuning, which we will do "experimentally" later, is to find these three parameters such that the system has good response in a large frequency range. Ideally, we want the gain to be **unity** and phase lag to be (near) zero for as large ω as possible.

[2] The lower bound of the integral can be 0 or $-\infty$ – this doesn't really matter too much here.

Before analyzing each of P-I-D separately, let us first consider a special set of K_p, K_i, K_d for the resonance in $G(\omega)$ is canceled out. Recall that the transfer function for the composite system is

$$T(\omega) = \frac{K(\omega)G(\omega)}{1 + K(\omega)G(\omega)}.$$

Let's find K_p, K_i, K_d such that $K(\omega)G(\omega)$ doesn't go bad:

$$K(\omega)G(\omega) = \frac{\omega_0^2(K_p - iK_i/\omega + iK_d\omega)}{\omega_0^2 - \omega^2 + i\beta\omega}.$$

It turns out that by fixing K_d and setting

$$K_p = \beta K_d \quad \text{and} \quad K_i = \omega_0^2 K_d,$$

we have

$$K(\omega) = \frac{-iK_i}{\omega}$$

which no longer have the ω_0 frequency. Also, we see that the open-loop (i.e., KG only) system now has a $1/\omega$ frequency response (1st-order) rather than a $\sim 1/\omega^2$ (2nd-order). Meanwhile, the closed-loop transfer function is

$$T(\omega) = \frac{K(\omega)G(\omega)}{1 + K(\omega)G(\omega)} = \frac{1}{1 + i\omega/K_i}.$$

This is nothing but a low-pass filter with time constant $\tau = 1/K_i$. To see why, we have to recall the ODE for the low-pass filter:

$$v_{\text{out}}(t) = v_{\text{in}}(t) - RC\frac{d}{dt}v_{\text{out}}(t)$$

Taking the Fourier transform and rearranging gives

$$T(\omega) \equiv \frac{v_{\text{out}}(\omega)}{v_{\text{in}}(\omega)} = \frac{1}{1 + iRC\omega}.$$

The Bode plots (Figure 2) associated with the open-loop system are generated using the following Mathematica commands.

```
Kd = 1; Kp = \[Beta]*Kd; Ki = \[Omega]0^2*Kd;

K[\[Omega]_] := Kp - I*Ki/\[Omega] + I*Kd*\[Omega];


LogLogPlot[Abs[K[a*\[Omega]0]*G[a*\[Omega]0]], {a, 0, 200},
AxesLabel -> {\[Omega]/"\[Omega]0", Abs[KG[\[Omega]]]}]

LogLinearPlot[Arg[K[a*\[Omega]0]*G[a*\[Omega]0]], {a, 0.01, 400},
PlotRange -> Full,
AxesLabel -> {\[Omega]/"\[Omega]0", Arg[KG[\[Omega]]]},
Exclusions -> None]
```

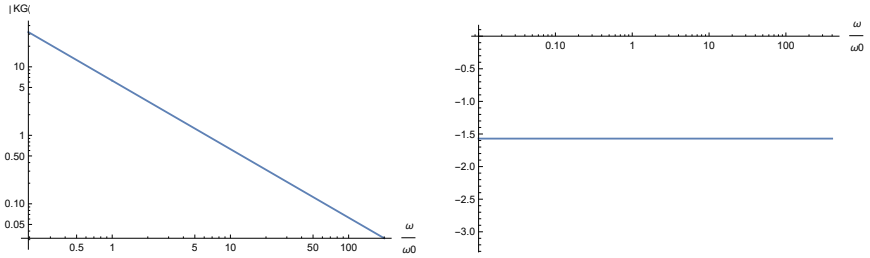

Figure 2: Open-loop gain KG for oscillator-PID system with natural resonance frequency ω_0 "tweaked away." The phase lag is constantly $\pi/2$ and the gain decreases like $1/\omega$ Again, $\beta = 2, \omega_0 = 2\pi$.

1.1 The P-gain

Suppose we set $K_i = K_d = 0$ and only have K_p. Is K_p enough to bring the system to $r(t)$? The answer is "Yes, but not very well." Here's why. In time domain, the ODE is

$$\ddot{x}(t) + \beta\dot{x} + \omega_0^2 x(t) = \omega_0^2 K_p(r(t) - x(t)).$$

Rearranging gives

$$\ddot{x}(t) + \beta\dot{x} + \omega_0^2(1 + K_p)x(t) = \omega_0^2 K_p P r(t).$$

This is basically a driven harmonic oscillator with a modified natural frequency of $\omega_P = \omega_0\sqrt{1 + K_p}$. When the proportional gain K_p is very high ($K_p \gg 1$), the modified natural frequency is much greater than the original natural frequency, $\omega_P \gg \omega$. As a result, the oscillator can follow the reference without significant phase lags even with ω's much larger than ω_0. Meanwhile, the gain can be calculated from

$$x(\omega \ll \omega_P) \approx \frac{K_p}{1 + K_p} r(\omega) \approx r(\omega),$$

which says that the gain is very close to unity ω within ω_P (again, assuming that $P \gg 1$). Both of these features can be seen in Figure 3, generated using the following Mathematica code.

```
Kd = 0; Kp = 100; Ki = 0;

K[\[Omega]_] := Kp - I*Ki/\[Omega] + I*Kd*\[Omega];
T[\[Omega]_] := G[\[Omega]]*K[\[Omega]]/(1 + G[\[Omega]]*K[\[Omega]]);

LogLogPlot[Abs[T[a*\[Omega]0]], {a, 0, 200},
AxesLabel -> {\[Omega]/"\[Omega]0", Abs[T]}]

LogLinearPlot[Arg[T[a*\[Omega]0]], {a, 0.01, 400}, PlotRange -> Full,
AxesLabel -> {\[Omega]/"\[Omega]0", Arg[T]}, Exclusions -> None]
```

So why not use only P rather than PID? Well, the catch is that we just pushed the resonance frequency of our system to a much higher value which could be comparable to variations due to noise. If noise is around the modified

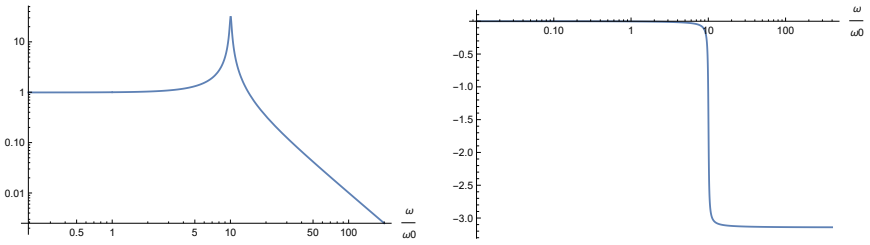

Figure 3: Bode plot for P-controller. Here, $\beta = 2, \omega_0 = 2\pi$, and $K_p = 100$.

resonance frequency ω_P then resonance will kick in. There are two types of noise: $d(t)$ due to disturbance to our system (more or less "mechanical") and $\xi(t)$ which is measurement noise (usually fast). The total dynamics will then be

$$x(\omega) = K(\omega)G(\omega)\epsilon(\omega) + d(\omega)$$

where

$$\epsilon(\omega) = r(\omega) - x(\omega) - \xi(\omega)$$

which gives

$$x(\omega) = \frac{K(\omega)G(\omega)}{1 + K(\omega)G(\omega)}[r(\omega) - \xi(\omega)] + \frac{1}{1 + K(\omega)G(\omega)}d(\omega).$$

From here, we see that for really high proportional gain (i.e., $|K| \gg 1$), the $d(\omega)$ noise will be suppressed up to ω_P, which is good. However, the measurement noise $\xi(t)$ gets mixed up with the referent signal $r(\omega)$ and amplified. One can define the tracking error, which is the difference between the reference r and the actual position x (all error free), in terms of the errors:

$$\epsilon_0 \equiv r - x = S(r - d) + T\xi$$

with the **sensitivity function** S and **complimentary sensitivity function** T. The goal is to make ϵ_0 as small as possible, but we're limited by the fact that $S + T = 1$ for all ω. For small S disturbances are rejected but then T is large and measurement noise is coupled in, and vice versa. For more information on this part, the reader may reference [1].

1.2 The I-gain

1.3 Instability

1.4 The D-gain

2 Experimentally tuning P-I-D with MATLAB

2.1 MATLAB code

```
% Author: Huan Q. Bui
% June 30, 2021
% Simple demo of PID control
% for a driven damped HO

% Inspired by: https://robotics.stackexchange.com...
% /questions/6859/how-to-implement-tracking-problem-with-pid-controller


clear all; close all; clc;

global error r dt time_array;

error = 0;
dt = 0.1;
stoptime = 100;

%%% create set(point)function
r = ones(stoptime/dt,1); % setpoint
%r(1:10) = 0;
r(floor((stoptime/dt)/4):(stoptime/dt)/2) = 0;
r(floor(3*(stoptime/dt)/(4)):end) = 0;
% define time array so that dimension matches setpoint r(t)
time_array = 0:dt:dt*(numel(r)-1);


x0 = [0; 0]; % initial condition, position & velocity = 0;
options = odeset('Reltol',dt,'Stats','off');
tspan = [time_array(1),time_array(end)];
[t_ode, x] = ode45(@ODESolver, tspan, x0, options);


% interpolate setpoint function
% so that we have r(t) with the same dimension as
% x(t) from ODE (which has finer resolution)
r_desired_ode = interp1(time_array,r,t_ode);
err = x(:,1) - r_desired_ode; % Error signal


%%% plotting %%%

figure(1)
plot(t_ode,x(:,1))
hold on
plot(t_ode,r_desired_ode)
xlabel('Time (sec)');
ylabel('Position', 'Interpreter','LaTex');
legend('Actual position', 'Desired position')
grid on
hold off


% figure(2)
% plot(t_ode, x(:,2), 'r', 'LineWidth', 2);
% title('Velocity','Interpreter','LaTex');
% xlabel('time (sec)');
% ylabel('$\dot{\theta}(t)$', 'Interpreter','LaTex');
% grid on

%%% %%%

function dxdt = ODESolver(t, x)

persistent r_old t_old;
global error r time_array;

if isempty(r_old)
```

```
r_old = 0;
t_old = 0;
end

% Parameters:
beta = 1;
omega0 = 2*pi;

% PID tuning
Kp = 100;
Ki = 20;
Kd = 7;

% dr/dt: find the derivative of r(t) for D-gain:
r_now = interp1(time_array,r,t);
if t == t_old
drdt = 0;
else
drdt  = (r_now - r_old)/(t-t_old);
end
r_old = r_now;

% u: control function
u = Kp*(r_now - x(1))... % P-gain
+ Kd*(drdt - x(2))... % D-gain
+ Ki*error; % I-gain;

% new error signal, obtained by accumulating errors (integral)
error = error + (r_now - x(1))*(t-t_old);

% 2x2 system of 1st-order ODEs
% initialize: matrix with position & velocity
dxdt = zeros(2,1);
dxdt(1) = x(2); % velocity
dxdt(2) = (omega0^2)*u - beta*x(2) - (omega0^2)*x(1); % acceleration

t_old = t;
end
```

References

[1] J. Bechhoefer, "Feedback for physicists: A tutorial essay on control," *Reviews of modern physics*, vol. 77, no. 3, p. 783, 2005.

[2] M. Zwierlein, "A short introduction to feedback control," 2008.