16-299 Spring 2021: Lecture 3
Dynamical Systems Modeling

George Kantor

9 February 2021

1 Finishing Intuitive PID

So the overall behavior of the PID controller is complicated and the gains are usually selected by trial and error. But there is the some conventional wisdom about how the gains affect closed loop system behavior (note these are not always true!)

- P term increases speed of response and frequency of oscillation (increasing the P term increases the effective spring constant in a mass-spring-damper system)
- D term increases damping (increasing the D term increases the effective damping coefficient in a mass-spring-damper system.)
- I term eliminates steady state error.

2 PID Tuning Approaches

Here are a couple of methods you can follow to help guide the tuning of PID gains. Both of these methods assume that the open loop system is already stable and that the PID controller is just being used to improve its dynamic response.

2.1 Intuitive Method

This method come directly from the intuition for the P, I, and D terms described, here are the basic steps:

1. start with all gains equal to zero.
2. turn up k_p until rise time is about right.
3. turn up k_d until overshoot and/or settling time goal is reached.
4. iterate steps 2 and 3 until desired transient performance is achieved
5. *slowly* turn up k_i until steady state error goes to zero in the desired time.

2.2 Ziegler-Nichols Tuning Method

Here is a more formal approach, called the "ultimate gain" method by Ziegler and Nichols:

1. start will all gains equal to zero
2. turn up k_p until system becomes just barely unstable (i.e., it oscillates steadily)
   (a) call that gain k_u (the ultimate gain)
   (b) call the resulting oscillation period P_u
3. implement the PID controller using this equation (slightly different parameterization than we have used so far):

$$u(t) = k_p e(t) + k_p T_D \frac{de}{dt}(t) + \frac{k_p}{T_I} \int e(\sigma) d\sigma$$

   where the parameters k_p, T_I, and T_D are selected as follows:

controller type	k_p	T_I	T_D
P only	$0.5k_u$		
PI (no D)	$0.45k_u$	$P_u/1.2$	
PID	$0.6k_u$	$P_u/2$	$P_u/8$

3 PID Comments

PID controllers are the workhorse of the classical control world. They are usually hand tuned, which requires good intuition about what the various terms do. Here are some things to watch out for:

3.1 The D Term

As we've mentioned before, the derivative term is not causal, meaning that D control is technically impossible to implement. It is tempting to implement the D term with a computer approximation:

$$\frac{de}{dt} \approx \frac{e(t) - e(t-T)}{T}$$

but this is not a good idea because this approximation amplifies noise, especially at high frequencies. For this reason, D control is usually only used in cases where the derivative of the output signal can be measured directly.

3.2 The I Term

The integral term can also cause some problems and should be used carefully. The main problem is *wind-up*: if there is something preventing the error from going to zero (like a stuck gear or a blown circuit) then the integral term will keep growing without bound. Some things you can do about this:

- add a max value, when the integral exceeds this max value stop integrating.
- be very careful when turning controllers on and off (or switching between controllers).
- use pre-computed feed-forward instead (you lose some of the closed-loop benefits, but this approach is often less risky than integral control)

4 Modeling

Let's start by formalizing a model we already have:

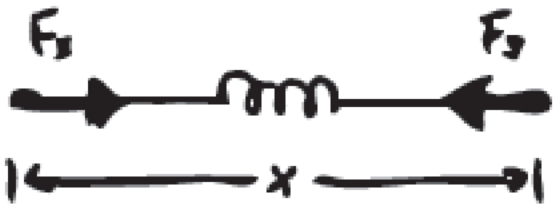
4.1 Mechanical Systems

Linear mechanical systems have two basic components, each with its own law that describes the relationship between force and motion.

linear spring:

$$F_s = kx$$

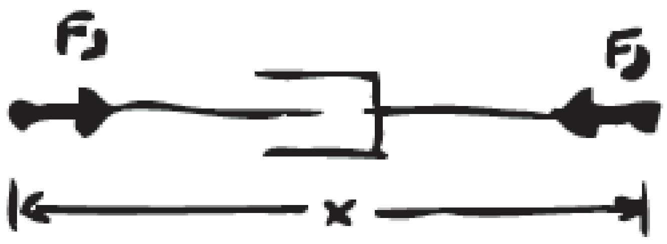
k is called the *spring constant*.



viscous damper:

$$F_d = \mu \dot{x}$$

μ is called the *damping coefficient*.

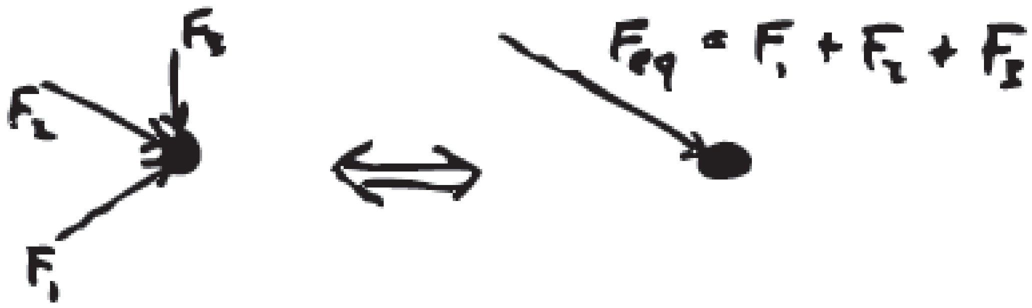


In addition to the component laws, there are physical laws that describe how forces act on point masses.

sum of forces: Multiple forces $F_1, F_2, \ldots, F_{n_f}$ applied to the same point are equivalent to a single force

$$F_{\text{eq}} = \sum_{i=1}^{n_f} F_i.$$

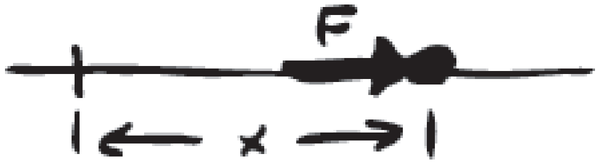
In other words, F_{eq} will produce motion identical to the motion caused by applying all of the F_is.



Newton's Second Law:

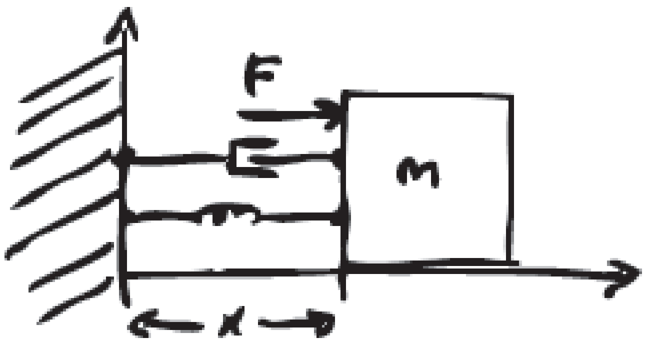
$$F = m\ddot{x}$$

m is called the *mass*.



We can create simple mechanical systems by connecting point masses together with springs and dampers.

Example: mass-spring-damper



$u = F$ is the input.

$y = x$ is the output.

The total force acting on the mass is

$$F_{\text{eq}} = F - F_s - F_d = u - kx - \mu\dot{x}$$

Applying Newton's law:

$$m\ddot{y} = F_{\text{eq}} = u - kx - \mu\dot{x}$$

so we get

$$\boxed{m\ddot{y} + \mu\dot{y} + ky = u} \tag{1}$$

This is a differential equation, specifically, it is a second order ordinary differential equation (ODE). We say it is the equation of motion for the mass spring damper system. What does that mean?

Solution: given initial conditions $y(0)$ and $doty(0)$ and an input $u(t)$ defined for $t \in [0, t_f]$, the solution is the function $y(t)$ also defined for $t \in [0, t_f]$ that satisfies the initial conditions and equation 1.

Some facts:

- for the class of systems we deal with, $y(t)$ always exists and is unique.
- in most cases, $y(t)$ is extremely hard to find
- but don't worry about that because:
    1. we can easily approximate it with a computer
    2. the whole point of control theory is to control y without actually solving for it
- equation 1 is called an *equation of motion* because $y(t)$ predicts how the output will move given the initial conditions and input

4.2 SISO LTI ODEs

So this gives us our first major class of model: ODE. We can write ODE's in many different forms. The mass spring damper as expressed above is a scalar, linear, second order ODE. We will often deal with the following generalization of this equation:

$$a_n \frac{d^n}{dt^n} y(t) + a_{n-1} \frac{d^{n-1}}{dt^{n-1}} y(t) + \cdots + a_1 \frac{d}{dt} y(t) + a_0 y(t) =$$
$$b_m \frac{d^m}{dt^m} u(t) + b_{m-1} \frac{d^{m-1}}{dt^{m-1}} u(t) + \cdots + b_1 \frac{d}{dt} u(t) + b_0 u(t)$$

Or:

$$\sum_{i=0}^{n} a_i \frac{d^i}{dt^i} y(t) = \sum_{i=0}^{m} b_i \frac{d^i}{dt^i} u(t). \tag{2}$$

Where:

- $u(t)$ is a scalar-valued input function
- $y(t)$ is a scalar-valued output function
- a_i, $i = 0, 1, 2, \ldots, n$ and b_i, $i = 0, 1, 2, \ldots, m$ are constant real numbers

Since there is one input and one output, and all of the coefficients are constant, this is called a single-input, single-output, linear time invariant ordinary differential equation (SISO LTI ODE).

4.3 Linear State Space

We can always convert an nth SISO LTI ODE into a system of n coupled first order ODEs. The resulting system is called a state space. Let's go back to the MSD model:

$$m\ddot{y} + \mu\dot{y} + ky = u$$

Define the state x to be a 2-vector

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Taking the derivative of x:

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}\left(u - \mu x_2 - kx_1\right) \end{bmatrix}$$

or more generally:

$$\boxed{\dot{x}(t) = f(x(t), u(t))}$$

This is known as a state space ODE.

Taking it a step further, since the original SISO LTI ODE was linear, the state space ODE is also linear:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & \frac{\mu}{m} \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

or more generally as:

$$\boxed{\dot{x}(t) = Ax(t) + Bu(t)} \qquad (3)$$

There are a few caveats that we will discuss later, but you can always go back and forth between equations 2 and 3.

4.4 Transfer Functions

The ODE models are in *time domain*, meaning that we consider everything as a function of time. For linear systems, we can also consider the signals and systems in the *frequency domain*, meaning everything is expressed as a function of frequency. This can be a difficult concept to grasp, and we will try to understand it better later, but for now here is a starting point (and a handful of little white lies).

The key link between the time domain and the frequency domain is the Laplace Transform, which eats time domain signals and spits out the corresponding signal in the frequency domain: The Laplace transform of a time-domain function $y(t)$ is defined to be:

$$Y(s) = \mathcal{L}[y(t)] = \int_{0_-}^{\infty} y(t)e^{-st}dt,$$

where 0_- represents $\lim_{t\to 0, t<0}$. A few key facts about $\mathcal{L}$:

- s is a complex variable, $s \in \mathbb{C}$.
- if the real part of s is zero, then the imaginary part is frequency in radians/sec, $s = i\omega$.
- $Y(i\omega)$ is complex:

   1. the magnitude of $Y(i\omega)$ is the magnitude of the sinusoid component of $y(t)$ at the frequency ω
   2. the angle of $Y(i\omega)$ is the phase shift of the sinusoid component of y at the frequency ω

- $\mathcal{L}$ has in inverse that converts $Y(s)$ back to $y(t)$
- $\mathcal{L}$ is linear: $\mathcal{L}[\alpha_1 y_1(t) + \alpha_2 y_2(t)] = \alpha_1 \mathcal{L}[y_1(t)] + \alpha_2 \mathcal{L}[y_2(t)]$
- differentiation in time domain is equivalent to multiplication by s in frequency domain: $\mathcal{L}\left[[\frac{dy}{dt}(t)\right] = sY(s)$

Recall the LTI ODE in input-output form:

$$\sum_{i=0}^{n} a_i \frac{d^i}{dt^i} y(t) = \sum_{i=0}^{m} b_i \frac{d^i}{dt^i} u(t).$$

Now let's take Laplace transform of both sides, assuming all of the initial contitions are zero ($0 = y(0_-) = \dot{y}(0_-) = \ddot{y}(0_-) = \cdots$ and $0 = u(0_-) = \dot{u}(0_-) = \ddot{u}(0_-) = \cdots$. I'll spell out the steps in excruciating detail:

$$\mathcal{L}\left[\sum_{i=0}^{n} a_i \frac{d^i}{dt^i} y(t)\right] = \mathcal{L}\left[\sum_{i=0}^{m} b_i \frac{d^i}{dt^i} u(t)\right]$$

Linearity of Laplace transform makes this

$$\sum_{i=0}^{n} a_i \mathcal{L}\left[\frac{d^i}{dt^i} y(t)\right] = \sum_{i=0}^{m} b_i \mathcal{L}\left[\frac{d^i}{dt^i} u(t)\right]$$

Then we use the formula for the Laplace transform of the derivative (with initial conditions zero) to get

$$\sum_{i=0}^{n} a_i s^i Y(s) = \sum_{i=0}^{m} b_i s^i U(s).$$

Rewrite in a more digestable form:

$$\underbrace{\left(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_2 s^2 + a_1 s + a_0\right)}_{\triangleq A(s)} Y(s) = \underbrace{\left(b_m s^m + b_{m-1} s^{m-1} + \cdots + b_2 s^2 + b_1 s + b_0\right)}_{\triangleq B(s)} U(s)$$

So we define the *transfer function* to be

$$H(s) = \frac{B(s)}{A(s)}$$

then we have

$$Y(s) = H(s)U(s).$$

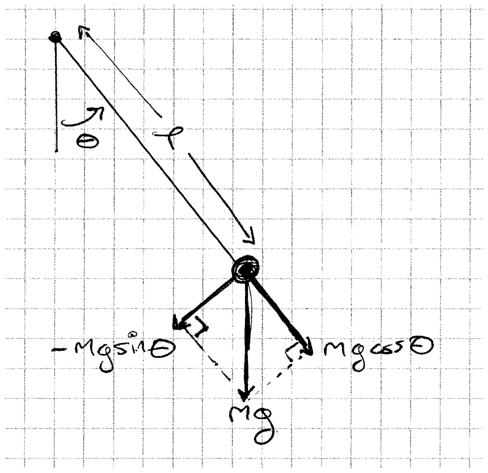
Some key points:

1. Solving a differential equation in time domain has become complex multiplication in frequency domain
2. $H(i\omega)$ gives the *frequency response* of the system.
3. If you feed the system a pure sinusoid $u(t) = A\sin(\omega t + \phi)$, the output will be:

$$y(t) = |H(i\omega)|\, A\sin(\omega t + \phi + \angle H(i\omega))$$

4.5 Nonlinear Systems

Let's look at another example, a pendulum:



Let's derive the equations of motion using what we know about physics. The pendulum bob will move on a circle around the pivot point. The velocity tangent to the circle is

$$v_t = \ell\dot{\theta}$$

Newton's law says

$$m\dot{v}_t = F_t$$

$$\Rightarrow \qquad m\ell\ddot{\theta} = -mg\sin\theta$$

$$\Rightarrow \qquad \ddot{\theta} = -\frac{g}{\ell}\sin\theta$$

We can put this into state space form as follows. Define

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

Then

$$\dot{x} = \begin{bmatrix} x_2 \\ -\frac{g}{\ell}\sin x_1 \end{bmatrix}.$$

If we extend that slightly to add a control torque $u = \tau$ and viscous friction $(-\gamma\dot{\theta})$ at the pivot, then the system becomes

$$\dot{x} = \begin{bmatrix} x_2 \\ -\frac{g}{\ell}\sin x_1 - \gamma x_2 + u \end{bmatrix}$$

This fits the general state space model $\dot{x} = f(x, u)$, but it does not fit any of the linear models! As a result, there is no such thing as frequency domain for nonlinear systems.