16-642 Fall 2017: Reference Notes for Classical Controls

George Kantor

6-13 November 2017

1 Transfer Functions

Transfer functions are one of the core concepts of classical controls. Transfer functions are always a ratio of two polynomials of a complex variable s, for example:

$$H(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} = \frac{B(s)}{A(s)}$$

If $m <= n$, the transfer function is *causal*. If $m < n$, the transfer function is strictly proper. Almost all of the transfer functions you will ever see are causal, and most will be strictly causal.

Note that we are in new territory, so m does not have the same meaning as it did before. n does, but that is kind of a coincidence.

A transfer function is often called the "frequency domain" representation of a SISO system. It is just another way of representing a linear dynamical system. There is a deep story that we will not get too far into, but the important thing to know is that in frequency domain, the system output can be found by simply multiplying the transfer function with the input:

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

Here $Y(s)$ and $U(s)$ are frequency domain representations of the output $y(t)$ and the input $u(t)$, respectively. You can go back and forth between time domain and frequency domain representations using the Laplace transform $\mathcal{L}$, e.g.:

$$Y(s) = \mathcal{L}\left[y(t)\right] \quad \text{and} \quad y(t) = \mathcal{L}^{-1}\left[Y(s)\right]$$

We won't say much about Laplace transforms, though there is much to say! We will settle for this one useful (approximately true) fact:

$$\mathcal{L}\left[\dot{y}(t)\right] = sY(s).$$

Some facts:

- a transfer function is a frequency domain representation of an ODE that looks like this:

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

  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

- we can think of $H(s)$ as an *operator* that maps inputs to outputs: either $H : U(s) \to Y(s)$, or $H : u(t) \to y(t)$.
- H is a *linear* operator:

$$H(\alpha_1 u_1 + \alpha_2 u_2) = \alpha_1 H(u_1) + \alpha_2 H(u_2).$$

- H is a *time invariant* operator: if $H(u(t)) = y(t)$, then $H(u(t - \tau)) = y(t - \tau)$.

The fact that H is linear and time invariant (LTI) leads to the **fundamental theorem of LTI systems**:

Theorem 1 *Let $H : u \mapsto y$ be a linear time invariant operator, and let $u(t)$ be a pure sinusoid with amplitude A_{in} and phase ϕ_{in}, i.e.,*

$$u(t) = A_{in} \sin(\omega t + \phi_{in}).$$

Then the corresponding output $y = T(u)$ is a pure sinusoid of the same frequency, though the magnitude may be scaled and the phase may be shifted:

$$y(t) = A_\omega A_{in} \sin(\omega t + \phi_{in} + \phi_\omega).$$

Further, the gain A_ω *and* phase shift ϕ_ω *depend only on the frequency ω.*

Now we can get one of the key insights into what a transfer function does. Let $s = i\omega$, where $\omega \in \mathbb{R}$ is a frequency expressed radians per second (i.e., $\omega = 2\pi f$, where f is the frequency in Hertz). Then $H(i\omega)$ is a complex number, which can be expressed in polar coordinates as the magnitude $r = \|H(i\omega)\|$ and angle $\phi = \angle H(i\omega)$. Two key facts:

1. $\|H(i\omega)\|$ is the gain of the system for the frequency ω.
2. $\angle H(i\omega)$ is the phase shift of the system for the frequency ω.

In other words, if the input is

$$u(t) = A_{in} \sin(\omega t + \phi_{in}),$$

then the output is

$$y(t) = \|H(i\omega)\| A_{in} \sin(\omega t + \phi_{in} + \angle H(i\omega)).$$

Also, because of linearity, if the input is a sum of sinusoids at different frequencies

$$u(t) = \sum_k A_k \sin(\omega_k t + \phi_k),$$

Then the output will also be a sum of sinusoids, where each one is waited and shifted according to the gain and pahse shift at that frequency:

$$y(t) = \|H(i\omega_k)\| A_k \sin(\omega_k t + \phi_k + \angle H(i\omega_k)).$$

This is why the classical controls approach is often called frequency domain: because the math is easy if we represent signals weighted and shifted sums of sinusoids. The Fourier transform (and the closely related Fourier series and transform) effectively transforms systems from functions of time into functions of frequency.

1.1 Transfer Functions and State Space

You can (pretty much) go back and forth between transfer function and state space represenations of a system. Here we look more closely at how to get a transfer function from a state space representation, i.e., how to get $H(s)$ from (A, B, C). Given (A, B, C), there is a unique transfer function that will provide the same relationship between input and output, the key to finding it is to take the Laplace transform of the state equation and do a bit of algebra. Start with the normal linear state space equations:

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

$$y(t) = Cx(t),$$

then take the Laplace transform of both sides of both equations to get

$$sX(s) = AX(s) + BU(s)$$

$$Y(s) = CX(s).$$

Now solve the top equation for $X(s)$

$$(sI - A)X(s) = BU(s)$$

$$X(s) = (sI - A)^{-1}BU(s),$$

and finally subsititue this expression for $X(s)$ into the output equation to get:

$$Y(s) = \underbrace{C(sI - A)^{-1}B}_{H(s)}U(s).$$

So we have clear formula to go from (A, B, C) to $H(s)$:

$$H(s) = C(sI - A)^{-1}B.$$

You can compute it by yourself, or you can use the `ss2tf` function in MATLAB.

The question of going from a transfer function $H(s)$ to a state-space representation is a bit more difficult. A triple of state space matrices (A, B, C) is said to be a *realization* of a transfer function $H(s)$ if will provide the same relationship between input and output. A realization (A, B, C) is a *minimal realization* if (A, B) is controllable and (A, C) is observable.

In order for a realization to exist, $n > m$, i.e., the transfer function must be *strictly causal*. If it is just causal ($n = m$), then the state space realization takes a slightly different form:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Any strictly causal transfer function has an infinite number of minimal realizations. For example, consider the realization

$$\dot{x} = Ax + Bu$$

$$y = Cx.$$

Now define a change of coordinates on x to generate a new state variable z as follows:

$$z = Tx,$$

where $T \in \mathbb{R}^{n\times n}$ is a constant invertible matrix. Now look what happens:

$$\dot{z} = \frac{d}{dt}(Tx) = T\dot{x}$$
$$= T(Ax + Bu) = TAx + TBu$$
$$\boxed{\dot{z} = TAT^{-1}z + TBu}$$

And looking at y:

$$y = Cx = CT^{-1}z$$

So the triple (TAT^{-1}, TB, CT^{-1}) is also a minimal realization.

The bottom line is that, for a given $H(s)$, there are an infinite number of realizations. One common realization to use is called Controllable Cannonical Form:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} b_0 & b_1 & \cdots & b_m & 0 & \cdots & 0 \end{bmatrix} x,$$

where the a_is and b_is are the coefficients in $H(s)$

1.2 Poles and Zeros

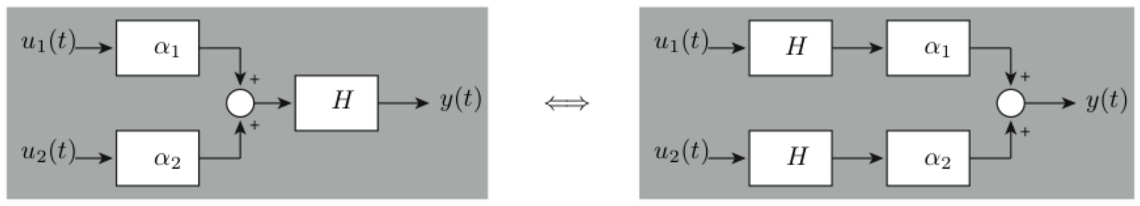
The roots of the numerator of the transfer function are called its *zeros*. The roots of the denominator of the transfer function are called its *poles*. Here is an important fact: let (A, B, C) be a realization of $H(s)$, then the poles of $H(s)$ are identical to the eigenvalues of A.

2 Block Diagrams

Now let's take a step back and think about systems again. He have shown that we can represent one block with a transfer function. So what happens when we put multiple blocks together?

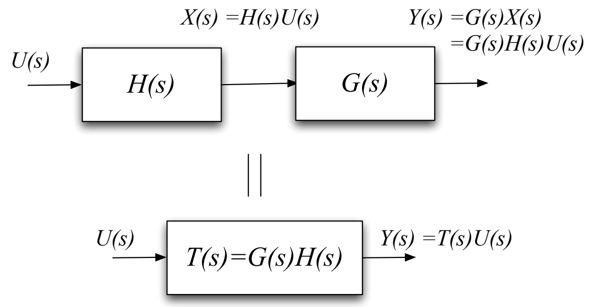
2.1 Linearity in Block Diagrams

Linearity means that the two block diagrams below are equivalent:

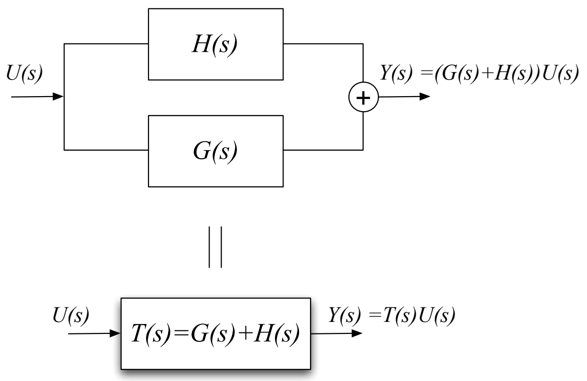


The pictures show it in time domain, but it is also true in frequency domain.

2.2 Blocks in Series

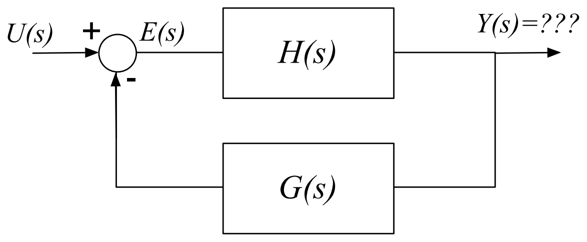


2.3 Blocks in Parallel



2.4 Feedback

We'll start with the simplest feedback system. Once we understand it, we will have all of the tools we need to analyze the more complex systems found in the book.



For the system above, what is the relationship between U and Y? If plant and controller are both linear, then we expect the system to be linear, so we should be able to find a transfer function $T(s)$. Lets try:

$$Y(s) = H(s)E(s),$$

$$E(s) = U(s) - G(s)Y(s).$$

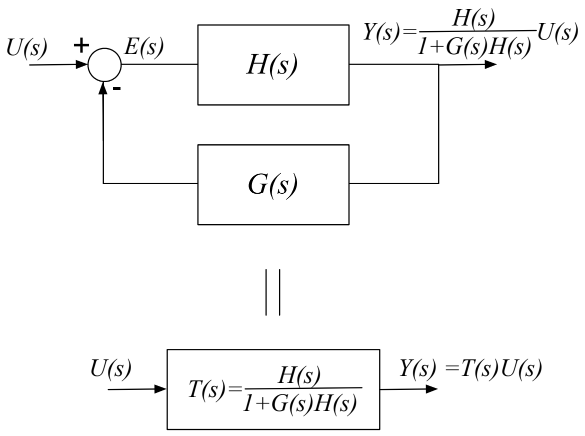
For convenience, we'll quit writing the s's (we know they're still there, though). Combining the above eqs:

$$Y = H(U - GY)$$

$$Y + GHY = HU$$
$$Y = \frac{H}{1 + GH}U$$

So the closed loop transfer function is

$$T(s) = \frac{H(s)}{1 + G(s)H(s)}$$



Note that when $G(s) = 1$, this configuration is called "unity negative feedback", which you might find useful in doing problems 3 and 4 in the second problem set.

3 Analysis Preview

So we know what a tranfer function is, and we know that serial, parallel, and feedback combinations of LTI systems can be boiled down to a single transfer function. Next, we start to understand how properties of a tranfer function correspond to system behavior. But first we have to learn how to describe system behavior.

4 Stability of Transfer Functions

We can't directly apply stability results from state space because those notions of stability were defined with respect to the state. In classical controls there is not explicit state, so we need to look at things a bit differently.

Think back to the idea of the impulse response. You have a system at rest (*i.e.*, all initial conditions equal to zero) and you "kick it" by applying an impulse at $t = 0$. Then you stand back and watch to see what the output does. It could do a lot of different things, but every possible response falls into one of the following two categories:

- $h(t) \to 0$ as $t \to \infty$ (the system is *asymptotically stable*).
- $h(t) \nrightarrow 0$ as $t \to \infty$ (the system is *unstable*).

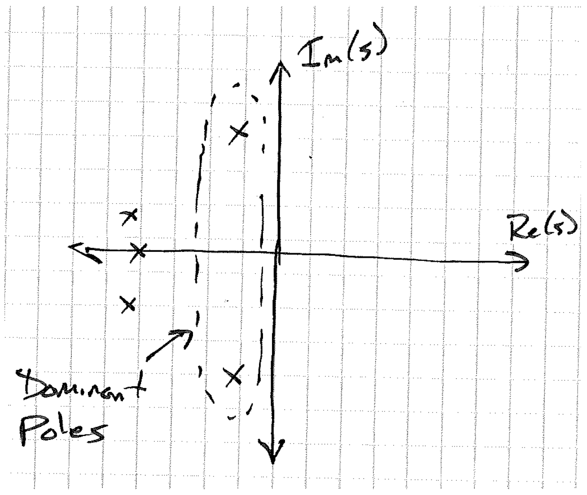
We will use this notion as our description of stability for transfer functions. It would be nice to determine the stability of a system by just looking at the transfer function $H(s)$. And given that the poles of H are equivalent to the eigenvalues of A, you probably have a pretty good notion already of how to to that. But let's make it official:

Poles and Stability: A system defined by a transfer function $H(s)$ is stable if and only if all of the poles of $H(s)$ have negative real part. Such systems are said to be *Hurwitz*.

Note that by these definition, systems that have poles with zero real part are lumped into the unstable category. This is not consistent with state space definitions, but we have to live with it.

4.1 Reminder of Dominant Poles

It is not unusual for one or two poles to be dominant, allowing a higher order system to be approximated by a first or second order system. This is part of the reason why it is important to thoroughly understand second orders systems: many higher order systems act like second order systems!



5 Step Response

The impulse response is one way of characterizing the transient properties of an LTI system, however it is not a very practical way because the impulse is not a physically real signal. In other words, it is difficult to experimentally determine the impulse response of a system because it is difficult to generate something approximating an impulse input (infinitely high, infinitessimally narrow).

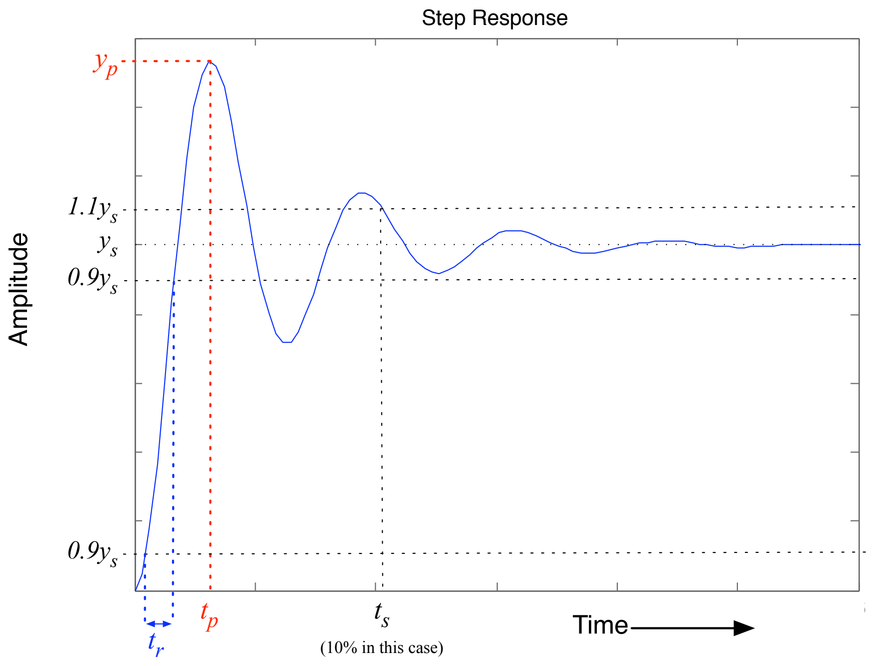
A more commonly used method of characterizing an LTI system is to look at its response to a unit step input. This is appropriately called the *step response*. Here is specifically what that means:

- assume all initial conditions are zero
- apply a unit step input:

$$u_s(t) = \begin{cases} 0 & \text{if} \quad t < 0 \\ 1 & \text{if} \quad t \geq 0 \end{cases}$$

- see what happens

A typical step response looks something like this:



We can use this figure to define a number of parameters that can be used to quantify the time-domain performance of the system:

1. steady state value y_s: the value that $y(t)$ goes to as $t \to \infty$.
2. 10%-90% rise time: the amount of time it takes to go from $0.1y_s$ to $0.9y_s$.
3. maximum percent overshoot:

$$M_p = 100\frac{|y_p - y_s|}{|y_s|}$$

   where $y_p = y(t_p)$, and t_p is the time at which $|y(t)|$ reaches its maximum value.

4. $n\%$ settling time: the smallest time t_s so that for all $t > t_s$

$$|y(t) - y_s| < \frac{n|y_s|}{100}$$

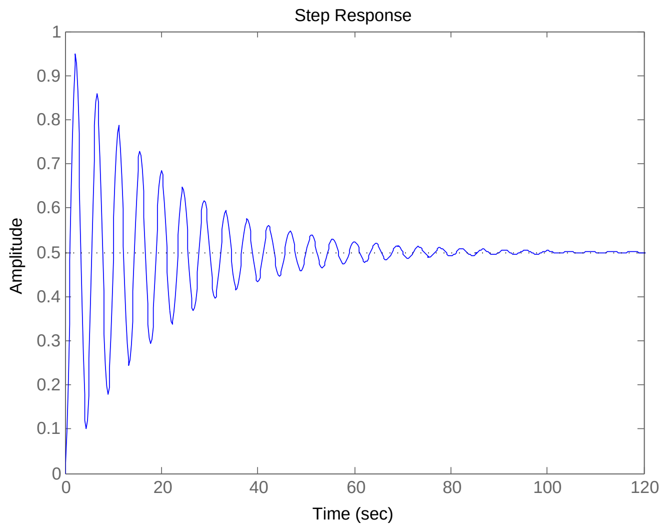
Design goals for control systems are often stated in terms of these parameters. But we will do the design in frequency domain, not time domain. We will learn how to relate these time-domain characteristics directly to the transfer function without having to do a Laplace transform or solve a differential equation.

6 Step Response for Second Order Systems

The mass-spring-damper system is a second order system with no zeros. For $m = 1$, $\mu = 0.1$, and $k = 2$ the transfer function is

$$H(s) = \frac{1}{s^2 + 0.1s + 2}$$

The MATLAB `step` function makes it easy to generate a step response from a transfer function, here is what the step response for the system above looks like (see `simpleStep.m`):



From this step response we can directly get the time domain performance parameters of y_s, M_p, t_r and t_s. But we would like to be able to know what these parameters are directly from the transfer function, without having to actually look at the time domain response.

We will start to answer this question by looking at the step response of second order systems with no zeros (like the mass-spring-damper system). This is partially to keep things simple while we build up intuition, but actually many systems act approximately like second order systems due to the presence of dominant poles. So the results we get from studying this class of systems are much more general than they seem at first.

Consider a second order system in a slightly different form:

$$H(s) = K\frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

Noncoincidentally, this is a form that often shows up in Laplace transform tables. So lets use Laplace transforms to solve a differential equation. In this case its really easy. Since this is a step response, the input is a step input, which in frequency domain becomes $U(s) = \frac{1}{s}$. This means that

$$Y(s) = H(s)U(s) = K\frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}\frac{1}{s}$$

The equivalent time domain case has three basic cases depending on the value of ξ. We'll look at the most interesting case first:

6.1 Underdamped Response

When $-1 < \xi < 1$ the system is said to be *underdamped*[1]. You could loop up the inverse Laplace transform of this second order step response to get:

$$\mathcal{L}^{-1}\left[\frac{\omega_n^2}{s(s^2+2\xi\omega_n s+\omega_n^2)}\right] = 1 + \frac{1}{\sqrt{1-\xi^2}}e^{-\xi\omega_n t}\sin\left(\omega_n\sqrt{1-\xi^2}t+\phi\right),$$

where ϕ (which wasn't listed in the table) is:

$$\phi = \tan^{-1}\frac{\sqrt{1-\xi^2}}{\xi}+\pi.$$

so this means that

$$y(t) = K\left(1-\frac{1}{\sqrt{1-\xi^2}}e^{-\xi\omega_n t}\sin\left(\omega_n\sqrt{1-\xi^2}t+\phi\right)\right).$$

This look a lot like the impulse response we found yesterday, and we can understand it in a similar way by writing things in terms of the poles

$$p_{1,2} = -\xi\omega_n \pm i\omega_n\sqrt{1-\xi^2}$$

to get

$$y(t) = K\left(1-\frac{1}{\sqrt{1-\xi^2}}e^{Re(p_1)t}\sin\left(Im(p_1)t+\phi\right)\right).$$

Some observations:

1. the basic form of the solution is a constant plus an exponential times a sinuosoid.
2. if the $Re(p_1) < 0$, then the exponential will go to zero and the solution will look like a decaying sinusoid that converges to the constant.
3. if the $Re(p_1) > 0$, then the exponential will go to infinity and the solution will look like a sinusoid whose amplitude grows exponentially.
4. if the real part is zero, then the solution will have a pure sinusoid component that neither shrinks nor grows with time.
5. the frequency of the oscillation will be $Im(p_1)$ radians per second.

Look at MATLAB examples for different poles (see `poleStep.m`).

6.2 Critically Damped Repsonse

When $\xi = 1$, then the imaginary part of the poles is zero and there are two repeated poles, $p_{1,2} = -\omega_n$. The response can be thought of as the limit of the underdamped case as ξ goes to one: the oscillatory component completely goes away. The time domain step response in this case works out to be

$$y(t) = 1-(1+\omega_n t)e^{\omega_n t}.$$

[1]some sources consider only stable systems to be underdamped, so they state the limits as $0 < \xi < 1$. But the math works out when ξ is negative, so I lump negative damping ratios into the underdamped case.

6.3 Overdamped Response

When $|\xi| > 1$, the poles of the system are real and the system is said to be overdamped[2]. The $1 - \xi^2$ under the square root sign is negative, so the poles in this case are

$$p_{1,2} = -\xi\omega_n \mp \omega_n\sqrt{-\left(1 - \xi^2\right)}.$$

The time domain step response in this case will have the form

$$y(t) = 1 - C_1 e^{p_1 t} - C_2 e^{p_2 t}$$

where C_1 and C_2 are real constants that result from the partial fraction expansion.

7 Getting to Some Numbers

The intuition of how the system responds to where the poles are placed is very useful, but it is also useful to have good hard formulas to relate time domain performance to poles. Here are some rule of thumb assuming there are one or two dominant poles:

settling time: generally, the $n\%$ settling time is a function of the real part of the dominant poles. Clearly, when the amplitude of the sinusoid shrinks to $n\%$ of the final value, then the solution will never go outside of the $n\%$ envelope again. So

$$\frac{e^{\mathrm{Re}(p_1)t_s}}{\sqrt{1-\xi^2}} = \frac{n}{100}$$

or

$$e^{\mathrm{Re}(p_1)t_s} = \frac{n}{100}\sqrt{1-\xi^2}$$

then take the natural log of both sides to get and divide by $\mathrm{Re}(p_1)$ to get t_s:

$$t_s = \frac{1}{\mathrm{Re}(p_1)} \ln\left(\frac{n}{100}\sqrt{1-\xi^2}\right)$$

rise time: the rise time is a function of the magnitude of the domninant poles; the smaller the magnitude the greater the rise time. The 10%-90% rise time of t_r obeys the following approximate equality:

$$t_r \approx \frac{1.8}{|p|}$$

percent overshoot: percent overshoot is a function of the angle between the positive dominant pole and the imaginary axis. This angle is

$$\theta = \mathrm{asin}(\xi) = \mathrm{atan}\left(\frac{-\mathrm{Re}(p_1)}{\mathrm{Im}(p_1)}\right).$$

The relationship between percent overshoot and this angle is nonlinear and hard to write as an explicit equation, but here are a few useful data points:

$$\theta > 45^o \quad \Rightarrow M_p < 5\%$$

[2]again, some sources consider only stable systems to be overdamped, so they state the limits as $\xi > 1$. But as in the case of underdamped systems, the math works out when ξ is negative, so I still consider the case where $\xi < -1$ to be "overdamped" even though the resulting system is unstable

$$\theta > 30^o \quad \Rightarrow M_p < 25\%$$
$$\theta > 17.5^o \quad \Rightarrow M_p < 35\%$$

steady state value: The above numbers describe the transient response of the system. To get the steady state response y_s we need the following famous result:

8 Final Value Theorem

The parameters in the previous section describe the transient response. To get the answer for the steady state value we can use the *final value theorem*:

If $Y(s)$ is the output of a stable LTI system, then

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s)$$

This is a general result that applies to all stable LTI systems (not just second order ones with no poles!).

To apply this, note that the step response of a system with transfer function $H(s)$ is

$$Y(s) = \frac{H(s)}{s}$$

so

$$y_s = \lim_{t\to\infty} y(t) = \lim_{s\to 0} H(s).$$

9 Adding Poles and Zeros

For details, see pages 121-127 of Franklin, Powell, and Emami-Naeini.

Adding zeros (numerators roots) or poles can change the responses described above in strange ways, but they can still be used as rules of thumb. Here is a quick overview of what can happen:

1. If zero is added in the left half plane and far away from the real part of the poles, it will not have much effect. (“far away” is defined as greater than 4 times the real part of the poles).
2. If a zero is added in the left half plane near real part of the poles, then percent overshoot will be increased.
3. If a zero is added in the right half plane, then the percent overshoot may be decreased and the step response might start off in the “wrong” direction. Systems with zeros in the right half plane are said to be *non-minimum phase*.
4. Added left half plane poles that have real part far away and to the left of the real part of the original poles do not have much effect (agrees with idea of “dominant poles”.
5. Added left half plane poles that have real parts near the original poles can significantly increase rise time.

See `addZeroStep.m` and `addPolesStep.m` for MATLAB examples.

10 Designing Controllers in the Complex Plane

The inequalities that relate the step response parameters to the dominant poles turn out to give us a useful approach in the design controllers. Design goals are often stated in terms of t_r, t_s, and M_p, we can turn around the equalities above to use the goals to identify "good" regions in the complex plane:

settling time: To have an actual settling time that is less than or equal to the design specification given by t_s, we must have

$$t_s > \frac{1}{\mathrm{Re}(p_1)} \ln\left(\frac{n}{100}\sqrt{1-\xi^2}\right)$$

$$= \frac{\ln\left(\frac{n}{100}\right)}{\mathrm{Re}(p_1)} + \frac{\ln\left(\sqrt{1-\xi^2}\right)}{\mathrm{Re}(p_1)}$$

Since $\sqrt{1-\xi^2} < 1$ and $\mathrm{Re}(p_1) < 0$, the second term on the right hand side is positive, so the equality still holds if remove it

$$t_s > \frac{1}{\mathrm{Re}(p_1)} \ln\left(\frac{n}{100}\right)$$

Doing a little more math, we can turn this inequality around to express a bound on $\mathrm{Re}(p_1)$:

$$\mathrm{Re}(p_1) < \frac{1}{t_s}\ln\left(\frac{n}{100}\right).$$

For the 1% settling time, this becomes approximately

$$\mathrm{Re}(p_1) < -\frac{4.6}{t_s}.$$

So given a desired t_s, we can draw a vertical line in the complex plane that intersects the the real axis at the value given by the right hand side of this inequality. As long as the dominant poles fall to the left of this line, the actual settling time will be less than t_s.

rise time: To have an actual rise time that is less than or equal to the design specification given by t_r, we must have

$$t_r > \frac{1.8}{|p_1|}$$

which can be turned around to yeild

$$|p| > \frac{1.8}{t_r}$$

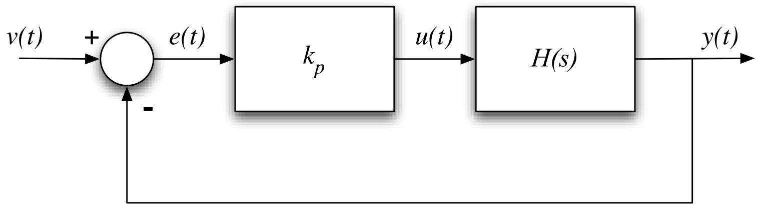
So given a desired t_r, we can draw a circle in the complex plane centered at the origin with radius $1.8/t_r$. As long as the dominant poles are outside of this circle (and stable!) the actual time will be less than t_r.

percent overshoot: Similarly, the inequalities above that relate settling time to the angle θ define wedge-shaped regions in the left half plane that are "good".

multiple design goals: There are usually multiple design goals which must be met simultaneously. In this case, we can draw the good regions for the individual goals above and then take their intersection. If the system poles lie within the intersection of the individual good regions, then the resulting response will satisfy all of the goals.

11 Proportional Control

Finally, we can introduce our first feedback controller: proportional feedback. Here is the block diagram:



Here the output $y(t)$ is fed back, subtracted from a reference input $v(t)$ to create the error signal $e(t)$. The error signal is multiplied by a constant k_p and then fed into the input of the plant. When $k_p = 1$, this is called unity negative feedback. The closed loop transfer function is

$$T_{cl} = \frac{k_p H}{1 + k_p H}$$

If $H = \frac{B}{A}$, then we can write

$$T_{cl} = \frac{k_p \frac{B}{A}}{1 + k_p \frac{B}{A}} = \frac{k_p B}{A + k_p B}$$

Note that if we can design a controller like this so that so that $e(t) \to 0$, then $y(t) \to v(t)$ and the output will track the reference input $v(t)$.

12 Root Locus

The root locus is a plot that contains the locations of all of the poles of the closed loop system as k_p ranges from 0 to ∞. The poles are the solutions of the equation

$$A + k_p B = 0$$

We think of the root locus as "starting" at $k_p = 0$ and "ending" as $k_p \to \infty$.

- the root locus starts with poles at roots of A (i.e., the poles of the open loop system)
- the root locus ends with some of the poles at the roots of B (i.e., the zeros of the open loop system) and the rest going off to infinity.

Root Locus Examples:
play with MATLAB, examples on the blackboard. (`go1, go2, go2b, go2c, go3, go4, go5, go6, go7`).

Summary: For second order systems there are 3 basic cases:

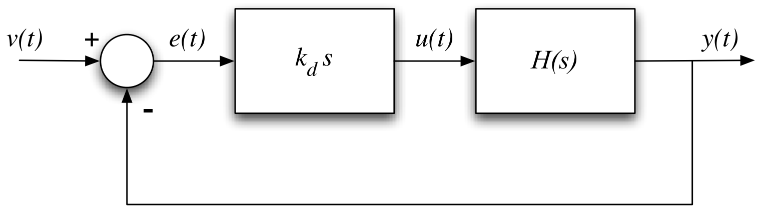
1. no zeros ⇒ poles go vertical. Or if they're real they come together then go vertical.
2. one zeros ⇒
    - poles circle around zero

- top pole goes counter clockwise
- poles meet to left of zero
- one pole goes to $-\infty$, one goes to the zero
- other cases turn out the same

3. two zeros $\Rightarrow$ poles go to zeros

13 Derivative Control

Derivative control, also known as D control, is similar to proportional control but the error signal is differentiated in addition to being multiplied by a constant k_d:



Note that the controller is non-causal. Oh well. We could use a causal approximation, but usually thats not a good idea because it amplifies noise. D control is usually only employed when the derivative (i.e. velocity) can be measured directly.

Anyway, the closed loop transfer function is

$$T_{cl} = \frac{sk_d\frac{B}{A}}{1 + sk_d\frac{B}{A}} = \frac{sk_dB}{A + sk_dB}$$

So the D root locus will act a lot like the P root locus with an extra zero at the origin.

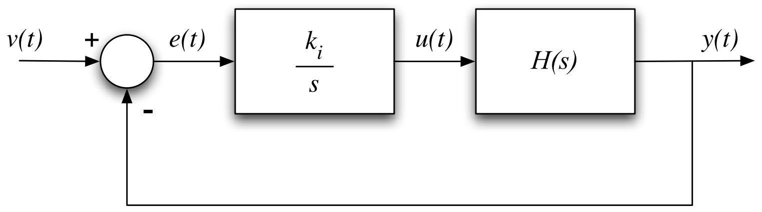
Play with MATLAB (`d_go1, d_go2, d_go3`, which can be found on the blackboard with the notes from Lecture 11).

Three cases:

1. no zeros $\Rightarrow$ acts like P with a zero at origin
2. one zeros $\Rightarrow$ acts like P with two zeros (one at origin)
3. two zeros $\Rightarrow$ acts like P with two zeros, plus another pole comes in from $-\infty$.

14 Integral Control

Another simple controller is integral control, or I control.



The closed loop transfer function is

$$T_cl(s) = \frac{k_i \frac{1}{s} \frac{N(s)}{D(s)}}{1 + k_i \frac{1}{s} \frac{N(s)}{D(s)}} = \frac{k_i N(s)}{sD(s) + k_i N(s)}$$

Effectively adds another pole at zero.

play with MATLAB (`i_go1, i_go2, i_go3`).

15 PID Control

Normally, P, I, and D control are combined into a single controller, called a PID controller. The PID controller can be written as a transfer function:

$$G(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k_d s^2 + k_p s + k_i}{s}$$

One way to get some intuition for how PID control works is to examine what it does to our favorite example of the mass-spring-damper system:

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

which we can put into transfer function form

$$Y(s) = \frac{1}{ms^2 + \mu s + k_s} U(s) = H(s)U(s)$$

If we define B_H, A_H, B_G and A_G to be the numerator and denominator polynomials of H and G, then we can write the closed loop transfer function:

$$\begin{aligned} T_{cl} &= \frac{B_H B_G}{A_H A_G + B_H B_G} \\ &= \frac{k_d s^2 + k_p s + k_i}{s(ms^2 + \mu s + k_s) + k_d s^2 + k_p s + k_i} \\ &= \frac{k_d s^2 + k_p s + k_i}{ms^3 + (\mu + k_d)s^2 + (k_p + k_s)s + k_i} \end{aligned}$$

Now, the intution. First, lets only worry about PD control, i.e., assume $k_i = 0$. Then we can write:

$$T_{cl} = \frac{k_d s + k_p}{ms^2 + (\mu + k_d)s + (k_p + k_s)}$$

The denominator of this looks just like the original mass spring system, only the spring constant and damping coefficients have changed:

$$\mu' = \mu + k_d$$

$$k_s' = k_s + k_p$$

So the gains k_p and k_d allow us to manipulate the effective spring constant and damping coefficient of the closed loop system.

But what about when you add the I term? It eliminates steady state error. There are two ways to understand how the I term eliminates steady state error. The first is intuitive. If there is a steady state error, the integral of that error will grow with time. Eventually it will get big enough to push the system toward the goal.

A more formal argument can be had by using the final value theorem on the closed loop transfer functions, which will show that the steady state error under PID control is zero as long as $k_i \neq 0$.

16 PID Control Recap

See `go_PID.m` for the effect of P, PD, and PID control on a mass-spring-damper system.

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.

17 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.

17.1 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.

17.2 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 transfer function (slightly different parameterization than we have used so far):

$$H(s) = k_p \left(1 + \frac{1}{T_I s} + T_D s\right)$$

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	$0.6k_u$	$P_u/2$	$P_u/8$

18 PID Comments

So we reviewed PID control, to remind you:

$$G(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k_d s^2 + k_p s + k_i}{s}$$

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:

18.1 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.

18.2 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)