Interconnections of Systems (§ 2.3 of Ogata)

One advantage of Laplace Transforms / Transfer functions is that ODEs for interconnections of systems can be easily derived.

Three common system interconnections:

A) Cascade (Serial) Interconnection

\[ u \rightarrow G_1 \rightarrow X \rightarrow G_2 \rightarrow Y \]

\[ u \rightarrow H \rightarrow Y \]

In the Laplace domain:
\[ Y(s) = G_2(s) X(s) \]
\[ X(s) = G_1(s) U(s) \Rightarrow Y(s) = [G_2(s) G_1(s)] U(s) \]

An example:

The transfer function for a cascade interconnection of \( G_2(s) \) and \( G_1(s) \) is given by multiplying the transfer functions, i.e. \( H(s) = G_2(s) \cdot G_1(s) \).

**Ex. 1**

System 1: \( x + 4x = 3u \rightarrow G_1(s) = \frac{3}{s+4} \)

System 2: \( 3y + 6y = 5x \rightarrow G_2(s) = \frac{5}{3s+6} \)

The transfer function from \( U(s) \) to \( Y(s) \) is

\[ H(s) = G_2(s) G_1(s) = \left[ \frac{5}{3s+6} \right] \left[ \frac{3}{s+4} \right] \]

\[ \Rightarrow H(s) = \frac{15}{8s^2 + 18s + 24} \]
We could also compute the model for the serial interconnection directly from the ODEs:

\[ \text{Sys 1} \quad x' + 4x = 3u \]
\[ \text{Sys 2} \quad 3y' + 6y = 4x \]

The easiest path is to solve for \( x \) and \( x' \) in terms of \( y \) and then plug into Sys 1. From Sys 2:

\[ 3y' + 6y = 4x \]
\[ \Rightarrow \begin{cases} x = \frac{3}{5} y' + \frac{6}{5} y \\ x' = \frac{3}{5} y' + \frac{6}{5} y \end{cases} \]

Plug these equations for \( x \) and \( x' \) back into Sys 1:

\[ \left[ \frac{3}{5} y + \frac{6}{5} y \right] + 4 \left[ \frac{3}{5} y' + \frac{6}{5} y \right] = 3u \]

Multiply both sides by 5 and group terms:

\[ 3y + 18y + 24y = 15u \]

The transfer function from \( u \) to \( y \) is

\[ H(s) = \frac{15}{3s^2 + 18s + 24} \]

This is the same result as derived before by multiplying \( G_1(s) \times G_2(s) \). The use of transfer functions is quite a bit simpler.
B) Parallel Interconnection

In the Laplace domain:

\[ Y(s) = G_1(s) U(s) + G_2(s) U(s) \]

\[ \Rightarrow Y(s) = [G_1(s) + G_2(s)] U(s) \]

The transfer function for a parallel interconnection of \( G_1(s) \) and \( G_2(s) \) is given by adding the transfer functions:

\[ H(s) = G_1(s) + G_2(s) \]

C) Feedback Interconnection

In the Laplace domain:

\[ Y(s) = G_1(s) U(s) \]

\[ U(s) = R(s) - G_2(s) Y(s) \]

\[ \Rightarrow Y(s) = G_1(s) \left[ R(s) - G_2(s) Y(s) \right] \]

\[ \Rightarrow \left[ 1 + G_1(s) G_2(s) \right] Y(s) = G_1(s) R(s) \]

\[ \Rightarrow Y(s) = \frac{G_1(s)}{1 + G_1(s) G_2(s)} R(s) \]

There are many variations on this basic feedback interconnection. Note that the diagram is drawn with negative feedback.
The transfer function for the feedback interconnection
with \( G_1(s) \) in the forward path and \( G_2(s) \) in the
feedback path is given by: [Assuming negative feedback]

\[
H(s) = \frac{G_1(s)}{1 + G_1(s) G_2(s)}
\]

In MATLAB, transfer functions are objects. These objects
have overloaded functionality, e.g.:

```matlab
>> G1 = tf([1 2], 1);
>> G2 = tf([3 4 5 6], 1);
```

Parallel
```
>> G1 + G2
```

Serial
```
>> G2 * G1
```

Negative Feedback
```
>> feedback(G1, G2)
```
where: \[ K: \quad u = 4e + 5 \int_0^t e(t) \, dt \quad \rightarrow \quad \text{PI control with} \quad K_p = 4 \quad \text{and} \quad K_i = 5 \]

\[ P: \quad \dot{x} + 2x = 3u \quad \rightarrow \quad 1^{st} \text{ order system} \]

What is the transfer function from \( r \rightarrow x \)?

We encountered this question earlier in the course.

Let's first derive the transfer function using the "old" way.

Differentiate the model for \( K \) and \( P \):

\[ \begin{align*}
\dot{u} &= 4 \dot{e} + 5e = 4(r - \dot{x}) + 5(r - x) \\
\ddot{x} + 2\dot{x} &= 3\ddot{u}
\end{align*} \]

Plug \( \dddot{u} \) into the second equation to get

\[ \ddot{x} + 2\dot{x} = 3 \left[ 4(r - \dot{x}) + 5(r - x) \right] \]

\[ \Rightarrow \quad \dddot{x} + 14\dot{x} + 15x = 12r + 15r \]

From this ODE we can derive the closed-loop transfer function:

\[ \boxed{H(s) = \frac{12s + 15}{s^2 + 14s + 15}} \]
Let's derive the same result using transfer functions:

\[
K \quad u = 4e + 5 \int_0^t e \, dt \, dz
\]

\[
P \quad \dot{x} + 2x = 3u
\]

The transfer functions for \( K \) and \( P \) are:

\[
K(s) = \frac{u(s)}{e(s)} = 4 + \frac{5}{s} = \frac{4s + 5}{s}
\]

\[
P(s) = \frac{x(s)}{u(s)} = \frac{3}{s + 2}
\]

Thus the transfer function in the forward path is:

\[
G_1(s) = P(s) \cdot K(s) = \left( \frac{3}{s + 2} \right) \left( \frac{4s + 5}{s} \right) = \frac{12s + 15}{s^2 + 2s}
\]

The transfer function in the feedback path is simply:

\[
G_2(s) = 1
g\text{ because there is no dynamics in the feedback.}
\]

This is called unity feedback.

Using our previous derivation, the transfer function for the closed-loop system is:

\[
H(s) = \frac{G_1(s)}{1 + G_1(s) \cdot G_2(s)} = \frac{\frac{12s + 15}{s^2 + 2s}}{1 + \left( \frac{12s + 15}{s^2 + 2s} \right)}
\]

To simplify, multiply the numerator and denominator by \( s^2 + 2s \):

\[
H(s) = \frac{12s + 15}{s^2 + 14s + 15}
\]

This is the same result as we derived the "old" way using ODEs. The use of transfer functions simplifies the algebra when the dynamics get more complicated.
Up to this point we've considered only fairly simple control strategies. Going forward we'll be designing control laws that are themselves governed by ODEs. We'll use transfer functions to design and analyze these more general control laws. To put this into context, the transfer functions of the control laws we've considered are:

a) Proportional Control

\[ u = K_p e \quad \Rightarrow \quad K(s) = \frac{U(s)}{E(s)} = K_p \]

b) PI Control

\[ u = K_p e + K_i \int e(t) \, dt \quad \Rightarrow \quad K(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} \]

\[ \Rightarrow \quad K(s) = \frac{K_p s + K_i}{s} \]

Note that a PI controller has a pole at \( s = 0 \) and zero at \( z = -K_i/K_p \).

c) PD Control

\[ u(t) = K_p e + K_d \dot{e} \quad \Rightarrow \quad K(s) = \frac{U(s)}{E(s)} = K_p s + K_d \]

Note that the PD controller is a polynomial rather than a ratio of polynomials, i.e., there is no denominator polynomial. This is a "non-proper" transfer function.

**Def:** A transfer function \( G(s) = \frac{N(s)}{D(s)} \) is **proper** if the degree of the denominator polynomial is \( \geq \) the degree of the numerator. It is **strictly proper** if \( \deg \text{ of } D(s) > \deg \text{ of } N(s) \). It is **non-proper** if \( \deg \text{ of } D(s) < \deg \text{ of } N(s) \).
Pole-Zero Cancellations

Consider the following serial interconnection:

\[
\begin{array}{c}
\text{u} \\
\downarrow \\
\times \\
\downarrow \\
\text{y}
\end{array}
\]

\[G_1(s) = \frac{1}{s+1} \quad \text{and} \quad G_2(s) = \frac{s+1}{s+2}\]

The transfer function from \(u(s)\) to \(y(s)\) is:

\[H(s) = G_2(s) \cdot G_1(s) = \left(\frac{s+1}{s+2}\right) \left(\frac{1}{s+1}\right) = \frac{1}{s+2}\]

Notice that the pole at \(p=-1\) in \(G_1\) perfectly cancels the zero at \(z=-1\) in \(G_2(s)\). The resulting model from \(u \rightarrow y\) appears to be a first order system \(\frac{1}{s+2}\) even though the combined dynamics of \(G_1 + G_2\) are second-order. One of the states is "unobservable" if we only look at the input-output behavior from \(u \rightarrow y\). However, these hidden, unobservable dynamics caused by pole-zero cancellations can have, in some cases, a very important role on the feedback system.

\[\text{Note: If you try to construct } G_1 \text{ and } G_2 \text{ in Matlab and then compute } H = G_2 \times G_1 \text{, then Matlab will return } H(s) = \frac{s+1}{s^2+3s+2}. \text{ This } H(s) \text{ has a zero at } z=-1 \text{ and poles at } p=-1,-2. \text{ If you want to force Matlab to perform the pole/zero cancellation then use the MINREAL function: } H_2 = \text{minreal}(G_2 \times G_1) \text{ returns } \frac{1}{s+2}. \]
Consider another example with a RHP pole-zero cancellation:

\[ G_1(s) = \frac{1}{s-1} \quad \text{and} \quad G_2(s) = \frac{s-1}{s+2} \]

The transfer function from \( u(s) \) to \( y(s) \) is:

\[ H(s) = G_2(s) G_1(s) = \left( \frac{s-1}{s+2} \right) \left( \frac{1}{s-1} \right) = \frac{1}{s+2} \]

Thus if we put a unit step \( u(t) \) into the dashed box and measure \( y(t) \) we see a nice, stable response:

But the internal signal is shooting off to \( \infty \). In other words, \( x(t) \) looks like:

Thus even though the cancellation looks ok from \( u \) to \( y \), there is something going terribly wrong (unstable) with one of the hidden internal signals.