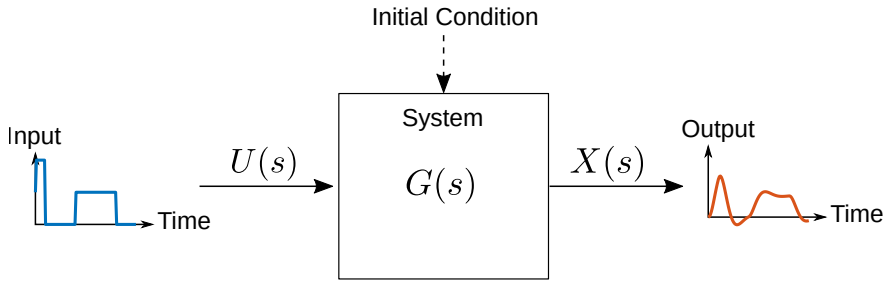
Lecture 12: Transfer Functions

The diagram below illustrates a system as a block that receives an input $u(t)$ (or $U(s)$ in the Laplace domain) on the left-hand side and produces an output $x(t)$ (or $X(s)$) on the right-hand side.



We can represent the sytem dynamics by the *transfer function*

$$G(s) = \frac{X(s)}{U(s)}$$

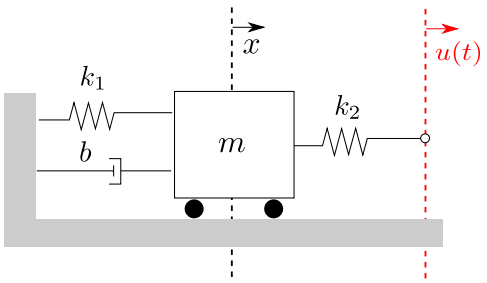
that is the ratio of the output $X(s)$ (e.g., the state of our system) to the input $U(s)$ (e.g., an applied force or disturbance). Finding the transfer function of a system involves the same steps as we've studied in previous lectures; however, we assume zero initial conditions and keep the forcing function $u(t)$ as a variable (rather than specifiying a specific input such as a step, ramp, etc.). Then, by taking the Laplace transform, we can rearrange to find $G(s)$ as a ratio of two polynomials in s.

Why is this new rearranged form of our system useful? Since the transfer function is derived without assuming a specific form of $U(s)$ we can easily recover the actual response to a specific input. Given $G(s)$ and some input $u(t)$ (with Laplace transform $U(s)$) we find the response in the time domain as by multiplying $G(s)$ with $U(s)$ and taking the inverse Laplace transform:

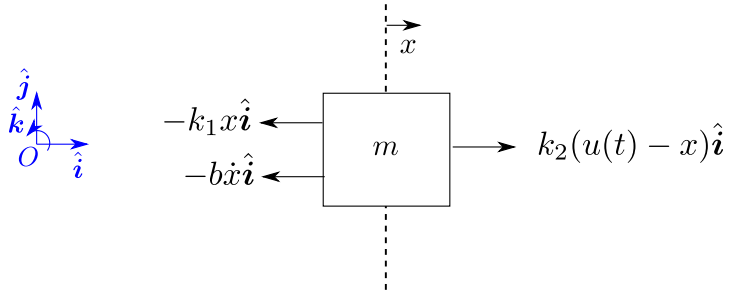
$$x_{zs}(t) = \mathcal{L}^{-1}[G(s)U(s)] = \mathcal{L}^{-1}[X(s)]\,. \tag{1}$$

The "zs" subscript is a reminder that we assumed zero initial conditions (zero state) when finding the transfer function $G(s)$.

Example. Consider the following system that resembles a mass-spring-damper system with a second connected spring. The end of the spring is displaced relative to its nominal (unstretched spring) position according to $u(t)$. The constant b is a damping coefficient and k_1, k_2 are spring constants. The displacement of the block from its nominal (unstretched spring) position is $x(t)$.



Define a reference frame $\mathcal{I} = \{O, \hat{\boldsymbol{i}}, \hat{\boldsymbol{j}}, \hat{\boldsymbol{k}}\}$ and draw the free body diagram (ignoring gravity and normal forces). The spring on the right exerts a force that depends on both x and u. For example, if $u = x$ the spring length remains unchanged even if the block moves. To determine the direction/sign of the force, consider a thought experiment: if the applied displacement u is greater than the displacement x, then in what direction will the force act? Intuitively, if $u > x$ the spring will be stretched (rather than compressed) and the force will pull the block to the right.



Let P be a point fixed on the center of the block. The acceleration of point P with respect to O is $\boldsymbol{a}_{P/O} = \ddot{x}\hat{\boldsymbol{i}}$. Then, summing the forces and applying Newton's 2nd Law:

$$\sum \boldsymbol{F} = (k_2 u(t) - b\dot{x} - k_1 x - k_2 x)\hat{\boldsymbol{i}} = m\boldsymbol{a}_{P/O} \tag{2}$$

$$= (m\ddot{x})\hat{\boldsymbol{i}} \tag{3}$$

which can be rearranged to give the scalar equation (in the $\hat{\boldsymbol{i}}$ direction)

$$m\ddot{x} + b\dot{x} + (k_1 + k_2)x = k_2 u(t)\ . \tag{4}$$

Taking the Laplace transform

$$m(s^2 X(s)) + bsX(s) + (k_1 + k_2)X(s) = k_2 U(s) \tag{5}$$

$$X(s)(ms^2 + bs + (k_1 + k_2)) = k_2 U(s) \tag{6}$$

we rearrange to obtain the transfer function

$$G(s) = \frac{X(s)}{U(s)} = \frac{k_2}{ms^2 + bs + (k_1 + k_2)} \tag{7}$$

Simulating transfer functions in MATLAB. A transfer function can be defined in MATLAB using the `tf` command with the syntax

```
G = tf(num,den)
```

where `num` is a vector containing the coefficients of a transfer function numerator (assumed to be in decreasing order of s) and `den` is a vector containing the coefficients of the transfer function denominator. To simulate the system we may use the `lsim` command with the syntax

```
x = lsim(G,u,t)
```

where `G` is the transfer function defined with the `tf` command and `u` and `t` are two vectors of equal length that contain the inputs and corresponding times for the simulation.

Example. Consider the transfer function corresponding to the previous example

$$G(s) = \frac{X(s)}{U(s)} = \frac{k_2}{ms^2 + bs + (k_1 + k_2)} \tag{8}$$

with $k_1 = 10$ N/m, $k_2 = 30$ N/m, $b = 0.1$N /(m/s), $m = 1$ kg. We wish to simulate the system subject to a two square wave pulses, one with amplitude one from $t = 0$ to $t = 1$ and another with amplitude 0.5 from $t = 4$ to $t = 8$. The input function using Heaviside step notation is

$$u(t) = H(t) - H(t-1) + 0.5H(t-4) - 0.5H(t-8)$$

(Note: In MATLAB, the Heaviside function is implemented with `heaviside`.).

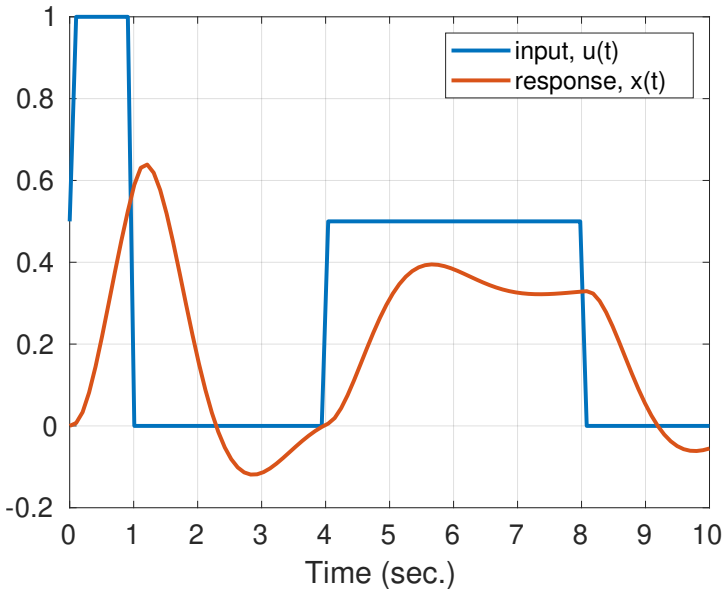
```
clear; close all; clc;

% define constants
m = 10; b = 20; k1 = 15; k2 =30;

% define transfer function
num = [k2]; den = [m b (k1+k2)];
G = tf(num,den)

% define input signal
t = linspace(0,10);
u = heaviside(t) - heaviside(t-1) + 0.5*heaviside(t-4) - 0.5*heaviside(t-8);

% use lsim to simulate linear system
x = lsim(G,u,t);

% plot
figure;
plot(t,u,'linewidth',2);  hold on;
plot(t,x,'linewidth',2);
grid on;
legend('input, u(t)','response, x(t)')
xlabel('Time (sec.)')
set(gca,'FontSize',14)
```

The variable G displays the numerical transfer function:

```
G =

          30
  ------------------
```

```
  10 s^2 + 20 s + 45

Continuous-time transfer function.
```

The resulting plot is



Impulse and Step Response. Suppose that the input is an impulse $f(t) = \delta(t)$. The impulse is defined as

$$\delta(t) = \begin{cases} \infty & \text{at } t = 0 \\ 0 & \text{at } t \neq 0 \end{cases} \tag{9}$$

It has the special property that the integral $\int_{-\infty}^{\infty} \delta(t)dt = 1$. An impulse can be physically approximated by a very large input over a short period of time compared to the dynamics. Since the Laplace transform of an impulse is $F(s) = \mathcal{L}^{-1}[\delta(t)] = 1$, the impulse response of the system $x(t)$ is found from above as

$$x_{\text{impulse}}(t) = \mathcal{L}^{-1}[G(s)] \tag{10}$$

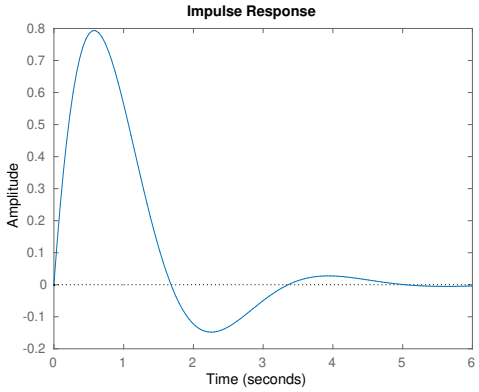
Given a transfer function in MATLAB the `impulse(G)` command can be used to observe the response (this command generates a plot by default). The `step` function is similar and shows the response to a step input.

Example: Using the same system as in the previous example:

```
clear; close all; clc;

% define constants
m = 10; b = 20; k1 = 15; k2 =30;

% define transfer function
```

```
num = [k2]; den = [m b (k1+k2)];
G = tf(num,den)

% generate impulse response
impulse(G)
```

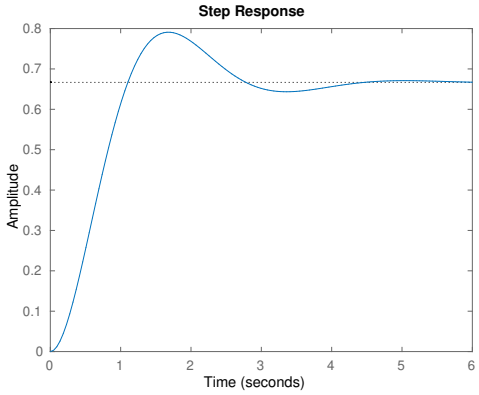
The resulting plot is



or using

```
step(G)
```

The resulting plot is



Response to Non Zero Initial Conditions and an Input. Since a transfer function is derived without assuming particular initial conditions, how can we use it to solve more general ODEs with non-zero initial conditions? The key idea is to leverage the superposition principle for linear systems. The complete response can be split into a sum of two parts:

$$x(t) = x_{\text{zs}}(t) + x_{\text{zi}}(t)$$

where

- $x_{\text{zs}}(t)$ is the response to the applied input assuming zero initial conditions (zero-state response) found from (1).

- $x_{\text{zi}}(t)$ is the response to the initial conditions assuming zero input (zero-input response).

Unfortunately, to find $x_{\text{zi}}(t)$ we cannot use the transfer function directly (recall the transfer function was derived assuming zero initial conditions). Instead, we return to our original ODE and follow the usual steps for solving an IVP with *no input* ($u(t) = 0$). That is, we take the Laplace *including* the initial conditions, then rearrange for $X(s)$ and apply the inverse Laplace transform for $x_{\text{zi}}(t)$ (or use any other suitable method to find $x_{zi}(t)$. By summing the zero-state response (forced response) and the initial condition response (free response) we obtain the actual response $x(t)$ of the system.