ENEL441
Control systems I
Winter 2020
Unit 1 Laplace analysis of LTI systems

Objective of Unit 1

Before we can get into control systems we need to have a solid background in linear systems analysis. In ENEL441 we will use both time domain and frequency domain analysis. In a typical design you will apply both time and frequency domain analysis simultaneously going back and forth between the domains.

We will approximate components and systems as being **Linear and Time Invariant (LTI)**. This approximation allows us to go between time and frequency domain using very efficient Laplace transform math.

First a brief review of the Laplace transform and LTI systems as covered in ENEL327 (primarily) and ENEL 343 (secondary) will be done. Then Matlab control system toolbox utilities will be introduced that you will find very helpful in the lab work and working through control problems. Then analysis of more complex systems will be tackled based on state space formulation. We will conclude with an introduction to feedback analysis based on signal graphs.

There are a number of problems included on D2L that you should go through. These will help in preparation for the first quiz.

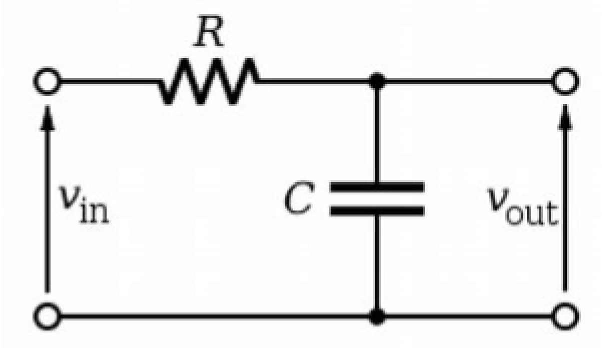
Review of LTI transfer functions and Laplace transformations

You should have a good background in Fourier and Laplace transform analysis, transfer functions, LTI , impulse responses and application of this to basic linear circuits. We will review the concepts that are essential for this course.

LTI – linear time invariant system

Everything in ENEL441 is based on the LTI system

Consider an example of a simple circuit consisting of a resistor R, and a capacitor C, as shown in the diagram below.



Say the input is the applied voltage and the output is the voltage across the capacitor. The LTI system consists of the circuit of R and C. Assuming that R and C are ideal components then their values are independent of the voltage across them or the current through them. As such the circuit is **linear**. Furthermore, we assume that the values of R and C are constant with time. That is the circuit is **time invariant**. If for example, R was temperature dependent such that the resistance decreased with temperature then the circuit would no longer be time invariant if the circuit temperature were to change slowly with time. Also, the circuit would no longer be linear in that the power dissipation in R, which is dependent on the signal applied to the circuit, would increase the temperature and change R as a function of the input signal resulting in nonlinear behavior.

LTI is therefore an idealization that is used as we need and LTI system for Laplace analysis. However, the LTI approximation is generally sufficiently accurate. In fact it is generally good enough that we often make the mistake of assuming the LTI approximation is ok when it is not.

Unless the system is LTI, then we cannot use Laplace analysis. Laplace analysis is so fundamental to control systems that we live with the fact that our analysis of systems is only approximate. Generally, the LTI approximation is good enough that we can get a very good understanding of how the system works. For real systems that are not LTI, we generally start with the analysis assuming that it is LTI and then use forms of quasi-static approximations or perturbations for a more accurate analysis.

An example is an airplane which is a system that slowly changes dynamic characteristics as fuel is being burned and the airplane becomes lighter with time. Hence, the airplane is not a time invariant system. Furthermore, the interaction of the control surfaces of the airplane with the surrounding air is nonlinear. Hence, the airplane is not a linear system either. Yet, the go-to initial analysis is heavily dependent on Laplace which approximates the airplane as an LTI system.

You will experience through the examples and lab assignments given in this course, that analysis and design of rather complex systems is possible, based on Laplace analysis.

Back to our example of the RC circuit. The input is $v_{in}(t)$ and the output is $v_{out}(t)$.

One way of systematically generating the DEQ is by starting with the element that is involved with energy storage which in this case is the capacitor. We have

$$i = C\frac{dv_{out}}{dt}$$

where i is the current. As there is only one energy storage device we know that the DEQ that represents the overall transfer function of the RC network can be written as a first order equation. We also look for the variable that is a time derivative of other variables which in this case is v_{out}. This becomes our 'state variable'. More on state variables later and as the course evolves you will get a much better idea of what a state variable really is. For now just stick to the process of looking for the variable associated with an energy storage component (such as the capacitor in this case) that is expressible as a first order derivative of this variable in terms of other variables. Hence we have

$$\frac{dv_{out}}{dt} = \frac{1}{C}i$$

But now i can be written as

$$i = \frac{1}{R}(v_{in} - v_{out})$$

Then we combine the DEQ as

$$\frac{dv_{out}}{dt} = \frac{1}{C}\frac{1}{R}(v_{in} - v_{out})$$

which is the desired DEQ written as a function of one state variable in terms of the independent input voltage source. Typically we would write the DEQ in the format

$$\frac{dv_{out}}{dt} = -\frac{1}{RC}v_{out} + \frac{1}{RC}v_{in}$$

That is we have the equation as:

{time derivative of state variable}

= {linear weighting of state variables} + coefficient {independent input source}

From this form we can directly map this into the Laplace domain as

$$sV_{out}(s) = -\frac{1}{RC}V_{out}(s) + \frac{1}{RC}V_{in}(s)$$

where

$$V_{out}(s) \Leftrightarrow v_{out}(t)$$

$$V_{in}(s) \Leftrightarrow v_{in}(t)$$

The variable ‘t’ denotes time in appropriate units which will always be seconds in this course. The variable ‘s’ is the complex frequency of

$$s = j\omega$$

where $j = \sqrt{-1}$ and ω is frequency in units of radians per second. Time domain signals are (generally) written with small letters and frequency domain with capital letters. The symbol of $\Leftrightarrow$ will refer to a Laplace transform pair.

You will likely recall that time derivatives can be replaced with s. That is

$$\mathcal{L}\left(\frac{dx(t)}{dt}\right) = sX(s)$$

$$\mathcal{L}\left(\frac{d^2x(t)}{dt^2}\right) = s^2X(s)$$

and so forth. Here we have tacitly assumed that all the initial conditions are zero. More on this later.

Going back to the DEQ we can extract the transfer function as

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{\frac{1}{RC}}{s + \frac{1}{RC}}$$

The transfer function links the input V(s) to the output Y(s) as

$$V_{out}\left(s\right) = H\left(s\right)V_{in}\left(s\right)$$

We can also express this in time domain as

$$h\left(t\right) \Leftrightarrow H\left(s\right)$$
$$v_{out}\left(t\right) = h\left(t\right) * v_{in}\left(t\right)$$

where

$$*$$

denotes convolution operation such that

$$v_{out}(t) = h(t) * v_{in}(t) = \int_{-\infty}^{\infty} h(\tau) v_{in}(t-\tau) d\tau$$

I hope that you are bored at this point as you have seen all of this before! If not, then you have a bit of review work to do.

The Dirac Delta Function

The first function to review is the dirac delta function which is a bit of a weird mathematical construct. In the time domain it is generally denoted as

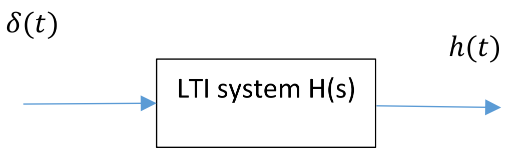
$\delta(t)$

it is defined by two properties:

Property 1: $\delta(t) = 0$ for all t except at t=0 where $\delta(0) = \infty$

Property 2: $\int_{-\infty}^{\infty} \delta(t) dt = 1$. That is the area of the delta function is one.

Obviously, we cannot generate such a function as the amplitude at one instance in time is infinity and therefore also if we consider the energy of the delta function it is infinite. Despite this weirdness it is a very useful function for LTI systems analysis. For instance assume an input of $\delta(t)$ into an LTI system with a transfer function of H(s). The output time domain response is then h(t).



Convolution vs Laplace transform

The time domain convolution is an integral that is generally tedious to do. This is where the Laplace processing is often beneficial.   Consider two functions

$$h(t) \Leftrightarrow H(s)$$
$$g(t) \Leftrightarrow G(s)$$

and we want $e(t) = h(t) * g(t)$.

This represents an integral of

$$e(t) = \int_{\tau} h(t-\tau)\, g(\tau) d\tau$$

We can also express this as

$e(t) = \mathcal{L}^{-1}(H(s)G(s))$

if H(s) and G(s) are known and that the product of H(s)G(s) can be put into a standard form for a table conversion of the inverse laplace transform then this is much more convenient as it avoids the integration. In this course, the transfer functions and excitation functions will generally consist of rational polynomials of s or exponentials and then then the product of H(s)G(s) can be put into a form where the inverse laplace can be determined with identity tables.

Another advantage of the Laplace formulation is that algebraic simplifications can be done in the ‘s’ or Laplace domain before conversion back to time domain. There will be many examples of this.

We can map time domain DEQs into Laplace domain expressions and extract out the transfer function. Consider an example:

An LTI system is described by the following DEQ

$$4\frac{d^2y}{dt^2} = y - x$$

where x(t) is the input excitation and y(t) is the output response. Assume the initial conditions are

$$y\left(0^-\right) = 0 \qquad \frac{dy\left(0^-\right)}{dt} = 0$$

Determine the transfer function

This is easy as we just replace the derivatives with s as follows:

$$4s^2Y(s) = Y(s) - X(s)$$

then group Y(s) and X(s) and express as a ratio:

$$(4s^2 - 1)Y(s) = -X(s)$$

and

$$\frac{Y(s)}{X(s)} = -\frac{1}{4s^2 - 1}$$

Then we can say that the transfer function is

$$H(s) = \frac{Y(s)}{X(s)} = -\frac{1}{4s^2 - 1}$$

Input Excitation Functions

In control systems we are generally interested in two things:

1. Time domain transient response to a given input

2. Eventual steady state time domain response after the transient has died off

Hence we have some specific input functions that we use repeatedly. The first is the impulse function $\delta(t)$. Applying $\delta(t)$ to the system with a transfer function of H(s) gives us the transient response which is the 'system impulse response' of h(t).

Next we have the step excitation function denoted in this course as u(t).

$$u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

The Laplace transform of u(t) is U(s) given as

$$U(s) = 1/s$$

We will show this later on. Hence if we have a system of H(s), then the step response is H(s)U(s)=H(s)/s. The time domain transient response of $\mathcal{L}^{-1}(H(s)/s)$ is very much of interest to us as we will see. Also the steady state response after the transient given as $\lim_{t\to\infty} \quad \mathcal{L}^{-1}(H(s)/s)$ is of interest.

Next we have the ramp excitation function

$$tu(t) = \begin{cases} t & t \geq 0 \\ 0 & t < 0 \end{cases}$$

The Laplace transform of tu(t) is given as 1/s^2. We will also show this later on.

We can also have the parabolic input excitation function as

$$t^2u(t) = \begin{cases} t^2 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

with a Laplace transform as of tu(t) is given as 2/s^3.

Relevant Matlab tools for Laplace Analysis

In the following we will give some examples of some Laplace transform calculations using some Matlab routines. Start with the time domain function of

$$f(t) = u(t-5)e^{-4t}$$

that we want to determine the Laplace transform of. Use the symbolic toolbox and first introduced the variables s as

```
>> syms f F s t
```

Then input the function f(t) using the symbolic notation

```
>> f=heaviside(t-5)*exp(-4*t)
```

The function Heaviside is the unit step function, u(t). We compute the Laplace as

```
>> F = laplace(f,t,s)
```

and get the result of

```
F =(exp(-5*s)*exp(-20))/(s + 4)
```

To make it more readable use:

```
>> pretty(F)
```

which returns a slightly more readable output as

```
exp(-5 s) exp(-20)
------------------
     s + 4
```

Now we want to take the inverse Laplace of F to get back f(t). This is done by

```
>> f = ilaplace(F)
```

which gives

```
f =heaviside(t - 5)*exp(-20)*exp(20 - 4*t)
```

Note that heaviside(t - 5) is the same as u(t - 5), the delayed unit step.

Another example of an inverse Laplace of

$$F(s)=\frac{1}{s(s+1)}$$

```
>> syms F s t
```

```
>> ilaplace(1/(s*(s+1)),s,t)
```

which gives a solution for f(t) as

```
1 - exp(-t)
```

Note that the heaviside(t) is implied. Matlab only includes it in the solution for a delayed step function.

Note that Matlab defaults s and t such that we don't have to explicitly input these. We can write

```
>> ilaplace(1/(s*(s+1)))
```

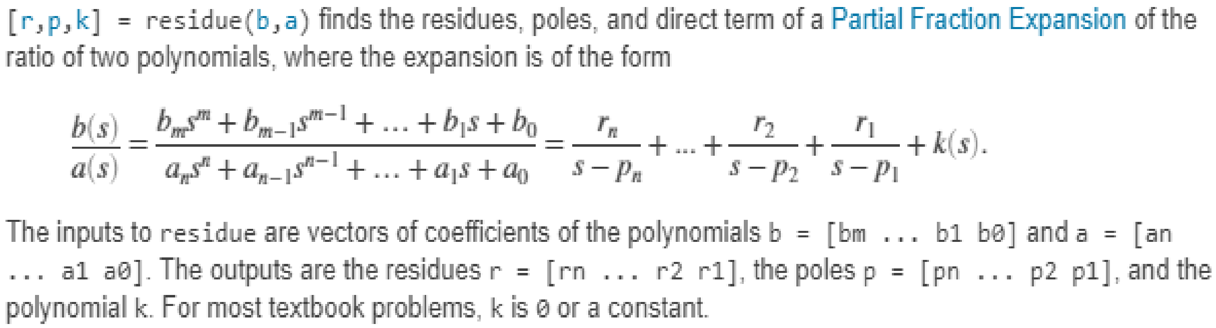
and get the same result of

```
1 - exp(-t)
```

Partial Fraction Expansion

Partial fraction expansions are not fun to do so fortunately Matlab has several useful routines to do this.

The first routine is residue which converts between a rational polynomial of the transfer function and the partial fraction expansion. Let r denote the residue at each pole p and k is the coefficient of the improper fraction part. That is k is the coefficient vector of the quotient of the division of the numerator and the denominator and (r,p) describes the proper fractional part.



`[r,p,k] = residue(b,a)` finds the residues, poles, and direct term of a Partial Fraction Expansion of the ratio of two polynomials, where the expansion is of the form

$$\frac{b(s)}{a(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+\dots+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+\dots+a_1 s+a_0}=\frac{r_n}{s-p_n}+\dots+\frac{r_2}{s-p_2}+\frac{r_1}{s-p_1}+k(s).$$

The inputs to `residue` are vectors of coefficients of the polynomials `b = [bm ... b1 b0]` and `a = [an ... a1 a0]`. The outputs are the residues `r = [rn ... r2 r1]`, the poles `p = [pn ... p2 p1]`, and the polynomial `k`. For most textbook problems, `k` is `0` or a constant.

Start with an example that we want to convert to a partial fraction expansion

$$F(s)=\frac{1}{s(s+1)}$$

We first write this as

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

and then put this into Matlab as polynomial coefficients as

```
>> num = 1;
>> den = [1 1 0];
>> [r,p,k]=residue(num,den)
```

Note that the polynomials are entered from the highest to lowest power.

This gives the output vector for the residues of the poles:

```
r = [-1  1]
```

The pole values as another vector

```
p = [ -1  0]
```

The last vector k is the ‘whole part’ of the polynomial fraction expansion. It is empty if the input rational polynomial fraction is proper meaning the order in the numerator is smaller than the order of the denominator.

```
k =    []
```

Putting this together for this example we have

$$F(s) = \frac{-1}{s+1} + \frac{1}{s} + []$$

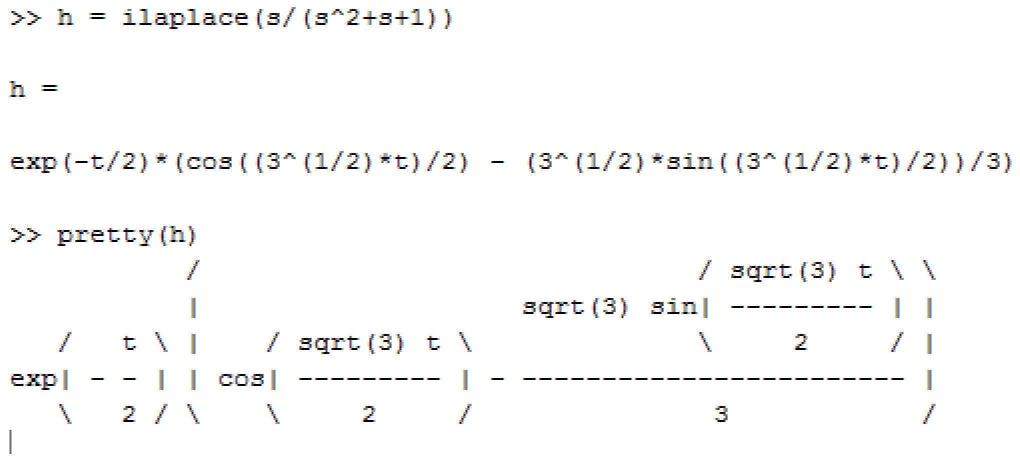
which can then easily be converted by the inverse laplace transform as

$$F(s) = \frac{-1}{s+1} + \frac{1}{s} + [] \Rightarrow -e^{-t}u(t) + u(t) = u(t)\left(1 - e^{-t}\right)$$

Another example is

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

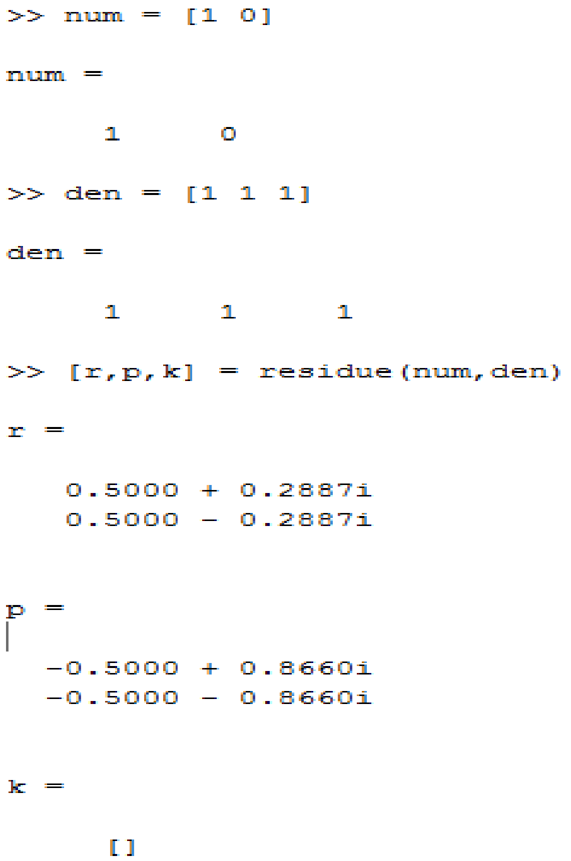
which we will do first with the symbolic solver and then with the numerical residue function.



```
>> h = ilaplace(s/(s^2+s+1))

h =

exp(-t/2)*(cos((3^(1/2)*t)/2) - (3^(1/2)*sin((3^(1/2)*t)/2))/3)

>> pretty(h)
           /                               / sqrt(3) t \ \
           |                    sqrt(3) sin| --------- | |
   /   t \ |    / sqrt(3) t \              \     2     / |
exp| - - | | cos| --------- | - ------------------------ |
   \   2 / \    \     2     /               3            /
```

Note that pretty() does a reasonable job of making the output readable.

Now solve the same problem using the residue() routine.



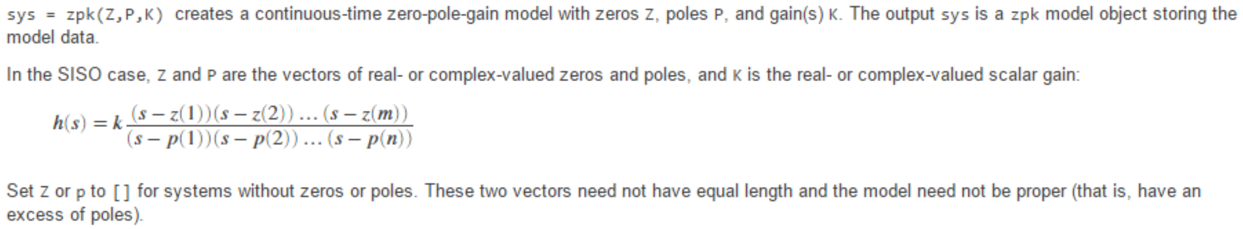
```
>> num = [1 0]

num =

     1     0

>> den = [1 1 1]

den =

     1     1     1

>> [r,p,k] = residue(num,den)

r =

   0.5000 + 0.2887i
   0.5000 - 0.2887i


p =

  -0.5000 + 0.8660i
  -0.5000 - 0.8660i


k =

     []
```

We have complex residues and poles that can be combined and shown to be the same as the symbolic result. However we are more interested in presenting the partial fraction expansion here.

Now consider a problem with repeated roots which gets tedious to do by hand.

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

It is awkward to have to multiply out the denominator before applying the residue(). Hence we convert the transfer function into a system first using zpk().



`sys = zpk(Z,P,K)` creates a continuous-time zero-pole-gain model with zeros `Z`, poles `P`, and gain(s) `K`. The output `sys` is a `zpk` model object storing the model data.

In the SISO case, `Z` and `P` are the vectors of real- or complex-valued zeros and poles, and `K` is the real- or complex-valued scalar gain:

$$h(s)=k\frac{(s-z(1))(s-z(2))\ldots(s-z(m))}{(s-p(1))(s-p(2))\ldots(s-p(n))}$$

Set `Z` or `p` to `[]` for systems without zeros or poles. These two vectors need not have equal length and the model need not be proper (that is, have an excess of poles).

Note that F has no zeros and three poles at -1, -2 and -2. Hence for the zeros we enter an empty set and for the poles an array of the poles. The scaling factor of k is 2 in this case.

```
>> F = zpk([],[-1,-2,-2],2)
```

This responds back with the transfer function object as

```
F =

     2
-------------
  (s+1) (s+2)^2
```

To use the partial fraction expansion of residue() we need to extract the numerator and denominator polynomials from F. This is done be tfdata(). This is the routine that gives access to the members of the transfer function object. the ‘v’ is an option for the output that gives the coefficients as a numerical array.

```
>> [num,den] = tfdata(F,'v')

num =    0    0    0    2

den =    1    5    8    4
```

Now we can put this into residue.

```
>> [r,p,k] = residue(num,den)


r =

  -2.0000
  -2.0000

   2.0000

p =

  -2.0000

  -2.0000

  -1.0000

k =[]
```

Note we could have got the same result by multiplying out the polynomial of the denominator. Note that the pole at -2 is repeated and that there are two residue values for this pair of repeated poles. You write the partial fractions of the denominator as increasing denominator order as below. Having this expansion we can convert this into the time domain response as

$$F(s) = \frac{-2}{s+2} + \frac{-2}{\left(s+2\right)^2} + \frac{2}{s+1} + [\,] \Rightarrow -2e^{-2t}u(t) - 2e^{-2t}tu(t) + 2e^{-t}u(t)$$
$$f(t) = -2e^{-2t}(1+t)u(t) + 2e^{-t}u(t)$$

To verify we can use the symbolic solver as

```
>> f = ilaplace(2/((s+1)*(s+2)^2))

f =2*exp(-t) - 2*exp(-2*t) - 2*t*exp(-2*t)
```

Impulse and step response of a transfer function

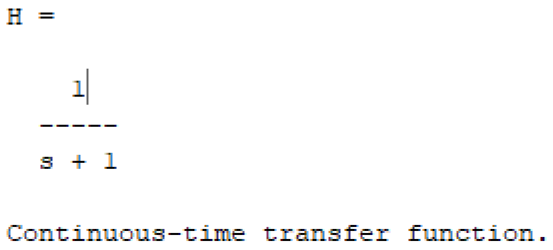
The transfer function is directly put into tf() from which the poles, zeros, impulse response step response and so forth can be calculated. Best to show this with an example. Consider the RC circuit with R=1 and C=1 such that the transfer function is

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

We can enter this as

```
H=tf(1,[1,1])
```

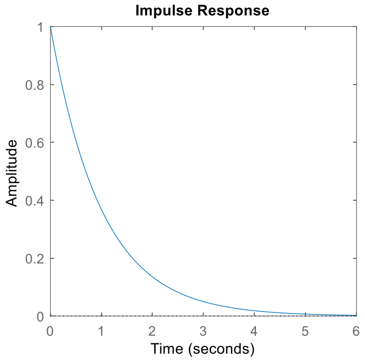
which results in a system object of the system H as



```
H =

     1
   -----
   s + 1

Continuous-time transfer function.
```

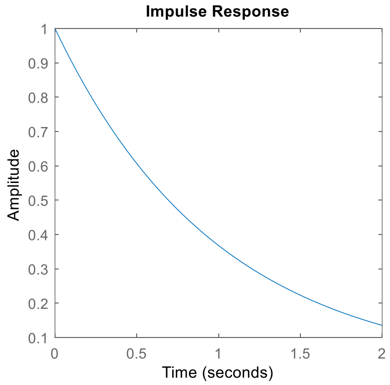
Plotting the impulse response

>> impulse(H)

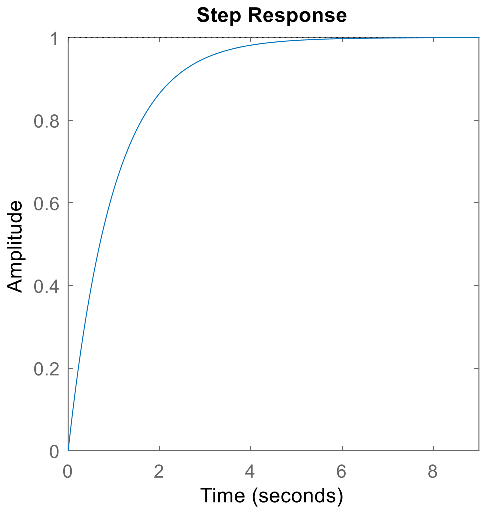


Note you can also input the time sample vector instead of letting Matlab choose the default.

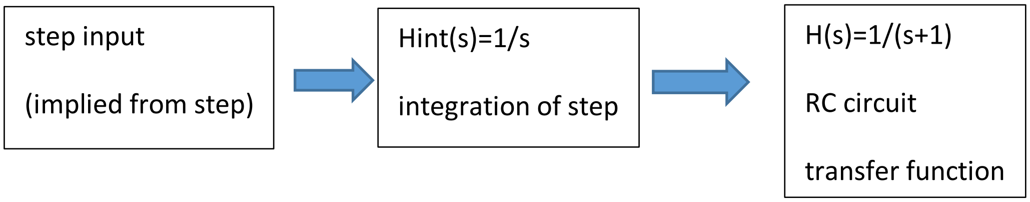
```
t = linspace(0,2,500)';
impulse(H,t)
```



For a step response



For the ramp we set up Hint as an integrator as the step input integrated becomes a ramp.



step input

(implied from step)

Hint(s)=1/s

integration of step

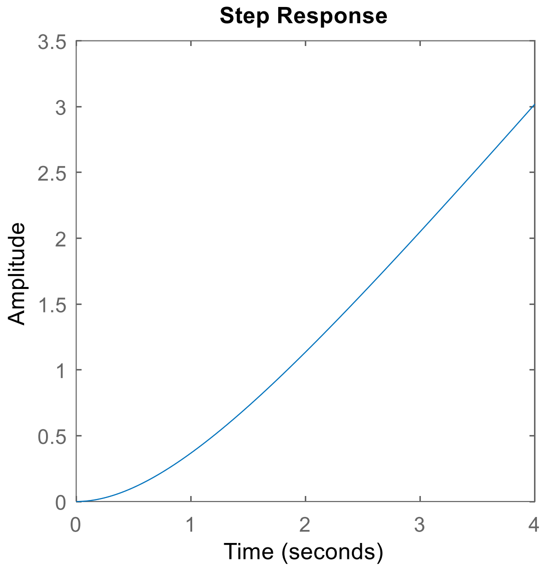
H(s)=1/(s+1)

RC circuit

transfer function

The matlab code to do this:

```
Hint=tf(1,[1,0])
t = linspace(0,4,500)';
step(H*Hint,t)
```

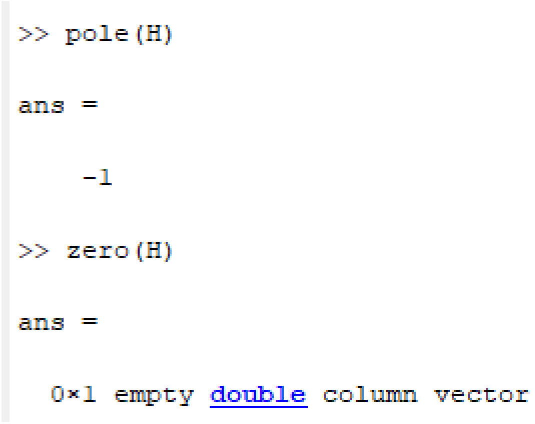


Another useful shorthand here in the Matlab notation is that H and Hint are transfer function objects. The series connection is simply expressed as H multiplied by Hint. Matlab uses overloaded operators to do this.

Extracting set of poles and zeros from the transfer function

Use pole() to get the set of poles of a transfer function

Use zero() to get the set of zeros of a transfer function



```
>> pole(H)

ans =

    -1

>> zero(H)

ans =

  0×1 empty double column vector
```

This is useful for more complex transfer functions.

Bode Plot of frequency response of a transfer function

Matlab can draw a bode plot of a transfer function or system space object using bode() to get the frequency response as a bode plot with:

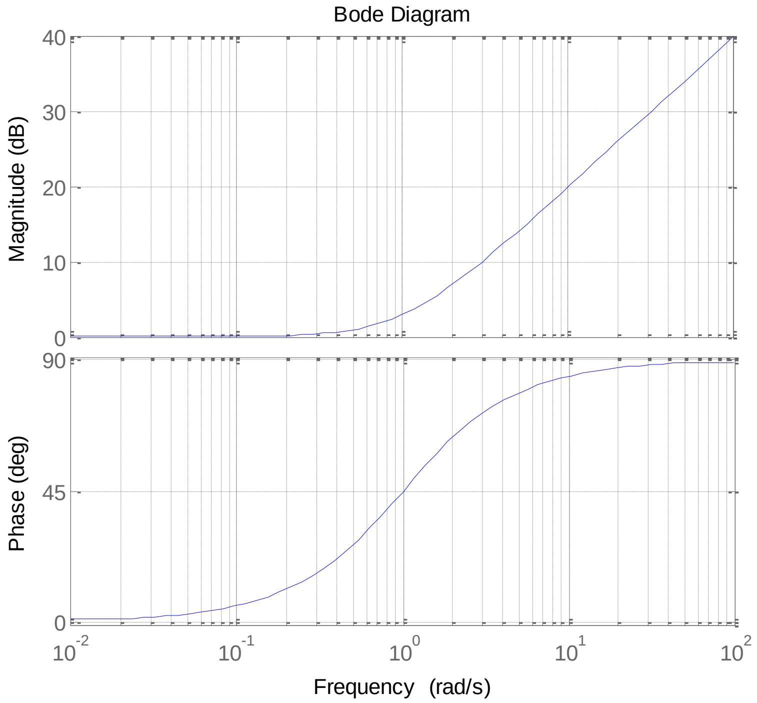
```
bode(H)
```

As an example consider a single zero at s=-1

$$H(s) = s + 1$$

```
den = 1;
num = [1 1];
H = tf(num,den)
bode(H)
grid on
```

Transition point (3 dB point) should occur at |s|=|jw|=1

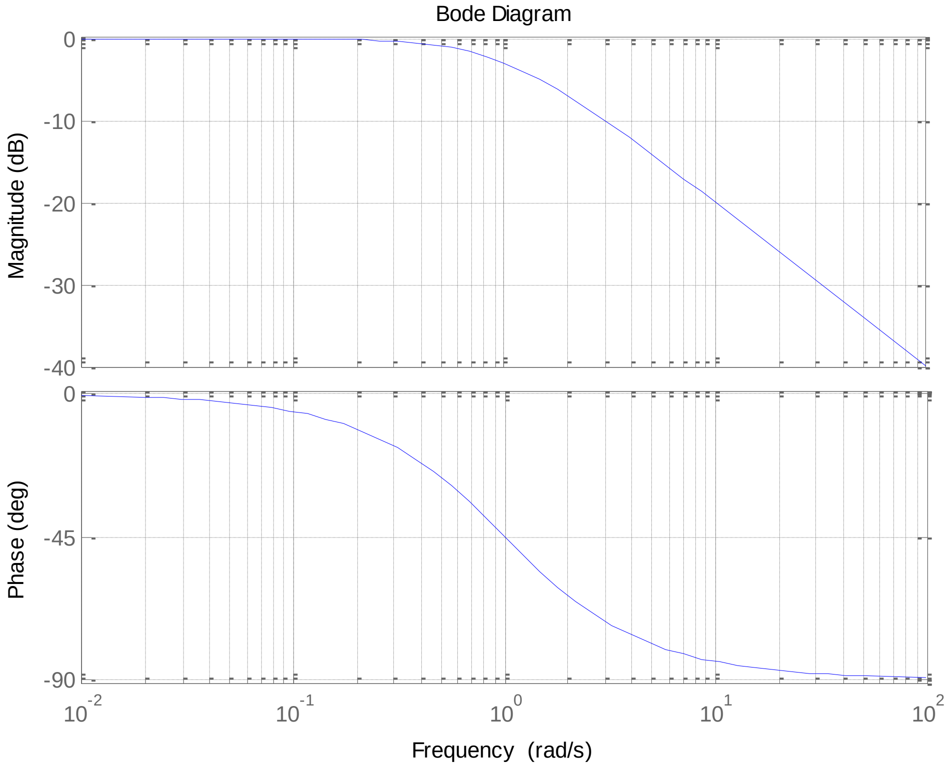


Note break point at w=1, |H| goes through 3 dB and phase through -45 deg.

Now show this as a pole at s=-1

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

```
num = 1;
den = [1 1];
H = tf(num,den)
bode(H)
```



More control over bode()

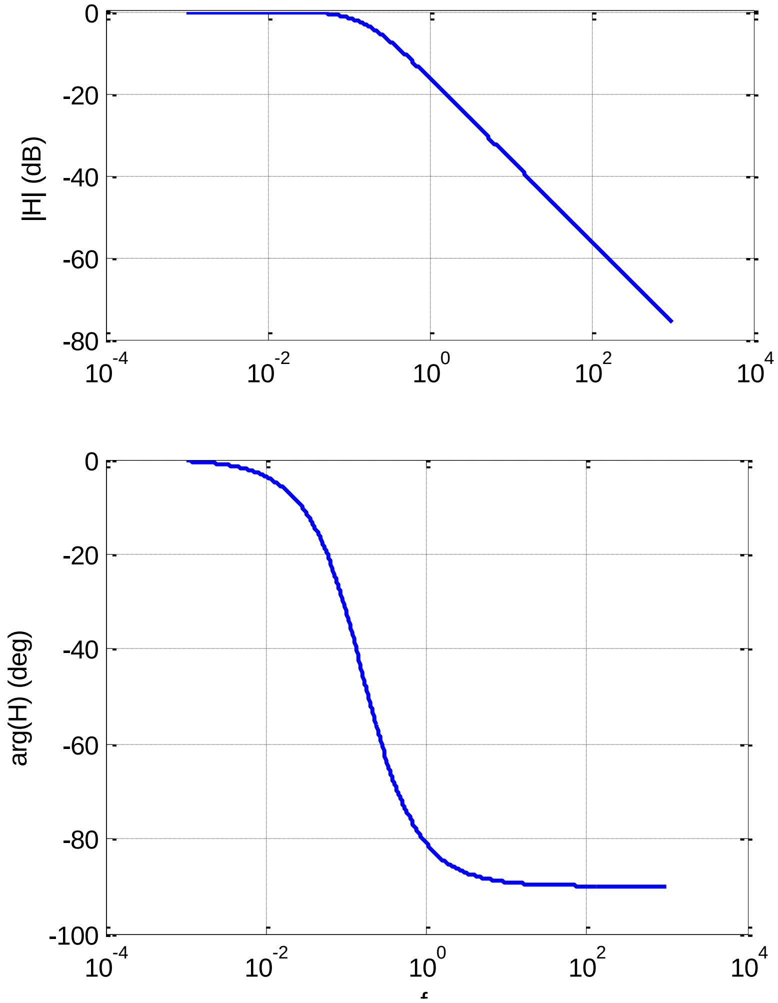
Put in your own frequency vector to control the points at which the frequency response of the transfer function is determined.

```
f = logspace(-3,3,500)';
[Hm,Ha] = bode(H,2*pi*f);
figure(1);semilogx(f,squeeze(20*log10(Hm)),'linewidth',2);
xlabel('f');ylabel('|H| (dB)');grid on;
figure(2);semilogx(f,squeeze(Ha),'linewidth',2);
xlabel('f');ylabel('|H| (dB)');grid on;
```

bode outputs the magnitude of the response, Hm and the phase angle of the response in degrees. You then have to use 'squeeze' to remove the redundant dimension of the data matrix to reduce it to a one dimensional array of values.

>> Hm(1:2)

```
ans(:,:,1) =    1.0000
ans(:,:,2) =    1.0000
```

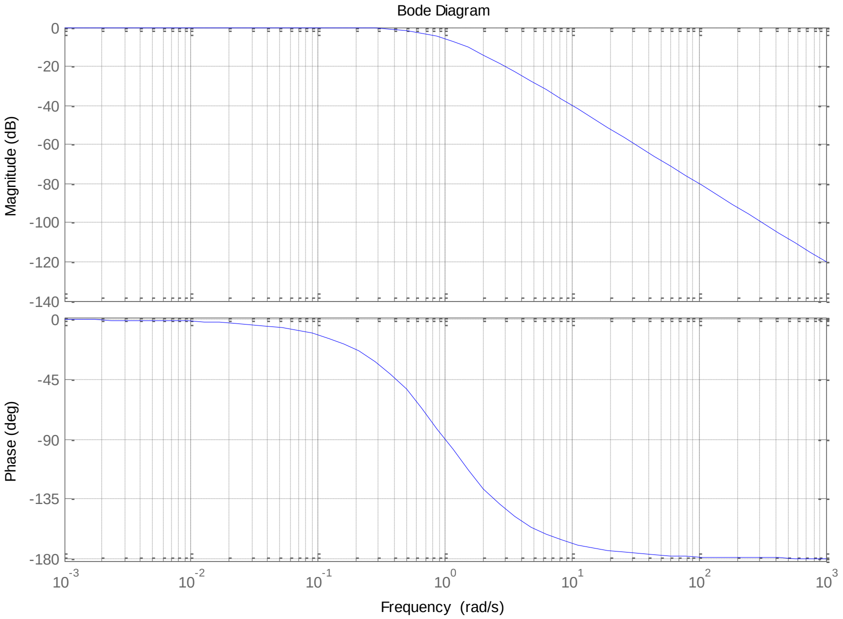


Remember: bode() outputs in degrees and not radians!

Some further examples with bode() - H(s) with a double pole

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

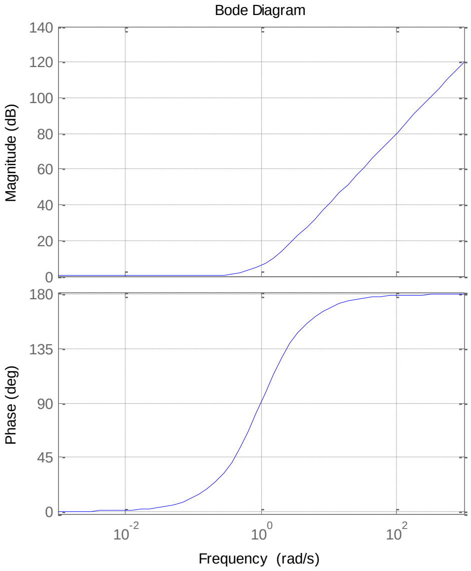
```
H = zpk([],[-1 -1],1);
bode(H,w);grid on;
```



Example with a double zero

$$H(s) = (s+1)^2$$

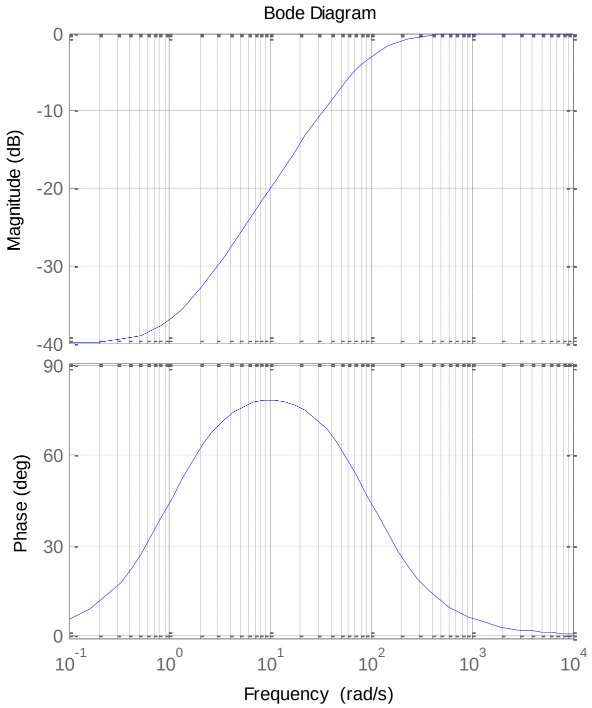
```
H = zpk([-1 -1],[],1);
bode(H,w);grid on;
```



Example with a zero and pole

$$H(s)=\frac{s+1}{s+100}$$

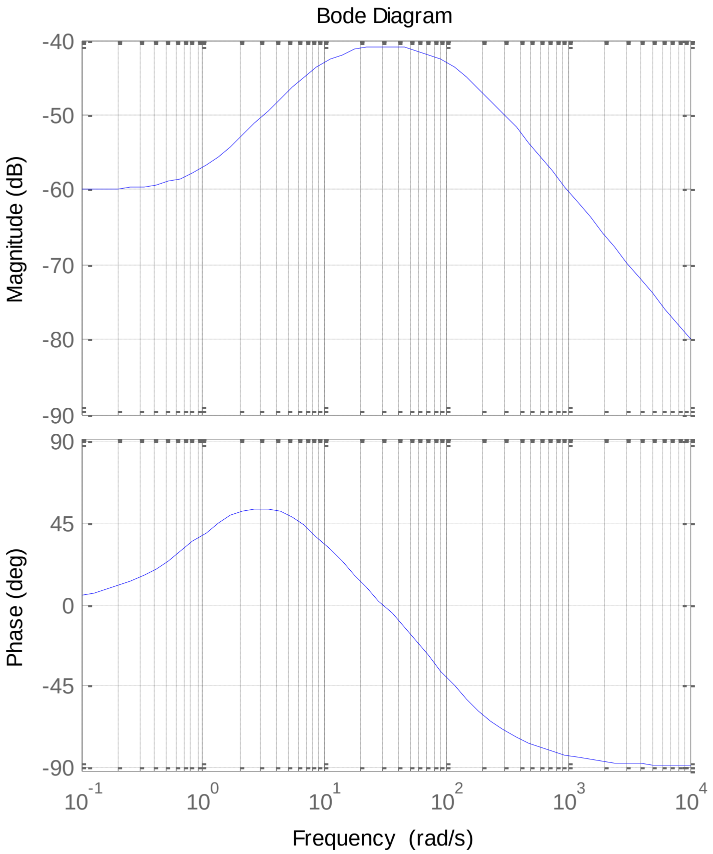
```
H = zpk(-1,-100,1);
w = logspace(-1,4);
bode(H,w);grid on;
```



Example with a zero and two poles

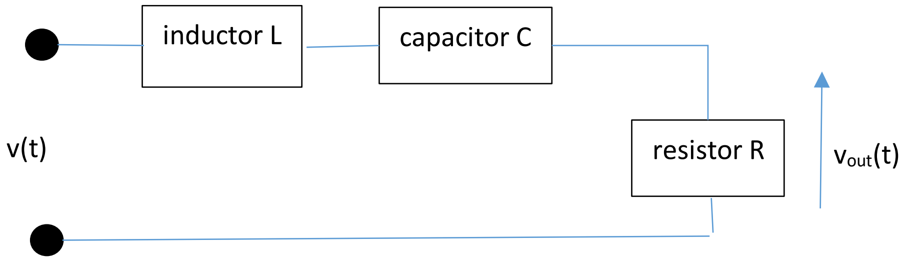
$$H(s)=\frac{s+1}{(s+100)(s+10)}$$

```
H = zpk(-1,[-10,-100],1);
w = logspace(-1,4);
bode(H,w);grid on;
```



Band pass filter example

Consider a circuit as in the figure below:

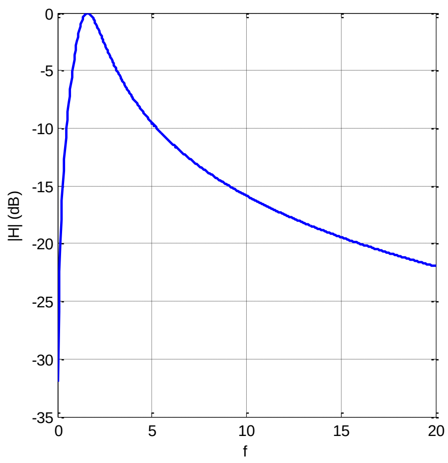


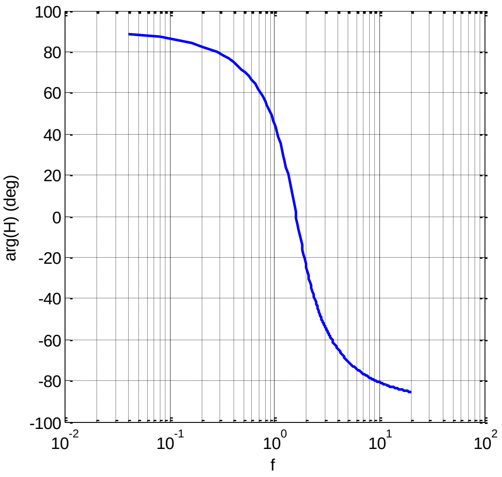
Here the input is the applied voltage and the output is the voltage across the resistor. Therefore the transfer function is

$$H(s) = \frac{\frac{R}{L}s}{s^2 + \frac{R}{L}s + \frac{1}{LC}}$$

```
R = 1;
C = .1;
L = .1;
num = [R/L,0];
den = [1,R/L,1/(C*L)];
f = linspace(0,20,500)';
[Hm,Ha] = bode(tf(num,den),2*pi*f);


figure(1);plot(f,squeeze(20*log10(Hm)),'linewidth',2);
xlabel('f');ylabel('|H| (dB)');grid on;
figure(2);semilogx(f,squeeze(Ha),'linewidth',2);
xlabel('f');ylabel('arg(H) (deg)');grid on;
```

The magnitude plot is given below:

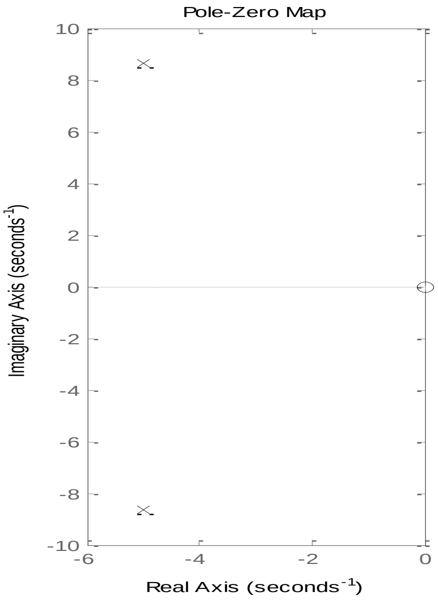




Note that the series RLC circuit with the output across the resistor forms a bandpass filter.

Pole zero map

Another useful utility is the plotting of the pole-zero map using `pzmap(tf(num,den),'k')`



Poles are indicated by 'x' and the zeros by 'o'.

Poles and Zeros of a system

In this section we take a slightly deeper look at poles and zeros. Consider the general system of

$x(t) \Leftrightarrow X(s)$

as the input excitation

$h(t) \Leftrightarrow H(s)$

as the transfer function of the system plant and

$y(t) \Leftrightarrow Y(s)$

as the output signal.

The excitation signal can be arbitrary except for the condition that it has a Laplace transform as

$$X(s) = \int_0^\infty x(t)e^{-st}dt$$

that needs to converge. That is x(t) cannot be an increasing exponential function as

$x(t) = u(t)\exp(t)$

as this does not have a Laplace transform.

In this course the transfer functions will be assumed to be LTI. Hence we have the output given by the convolution of the input excitation and the system impulse response and the Laplace relation

$$y(t) = h(t) * x(t)$$
$$Y(s) = H(s)X(s)$$

Furthermore, it is of significance that since $h(t) \Leftrightarrow H(s)$ and that the system is an LTI system that H(s) can only be expressed as a rational function of polynomials of s and delay terms. That is

$$H(s) = Ae^{-sT}\frac{\prod_{m=0}^{M}(s - z_m)}{\prod_{n=0}^{N}(s - p_n)}$$

where:
A is an amplitude coefficient.
M is the number of zeros
z_m is the set of zeros
N is the number of poles
p_m is the set of poles
T is the delay

In general we can have a superposition of the of the terms with poles, zeros and the delay. For example we can have

$$H(s) = \frac{e^{-s2}}{s+1} + \frac{e^{-3s}}{s+10}$$

With the restriction of an LTI system we can see that the poles, zeros and delay take on a more special significance. They are not just an arbitrary convenient mathematical form but rather they have physical significance. We can trace this back to the underlying differential equations that describe the system which themselves must be linear and time invariant.

Next consider the input excitation x(t) which can be any arbitrary function. However we can choose to restrict x(t) to being of the polynomial form of

$$x(t) = (a + bt + ct^2 + \cdots)u(t)$$

where {a,b,c…} are coefficients. The Laplace transform of X(s) is then a rational polynomial of s describable by poles and zeros. We can also add delay terms such as

$$x(t) = tu(t-1) + t^2u(t-2)$$

With this restriction we can have X(s) in the same form as H(s) consisting of exponential terms, poles and zeros. Hence Y(s)=H(s)X(s) will also consist only of exponential terms, poles and zeros.

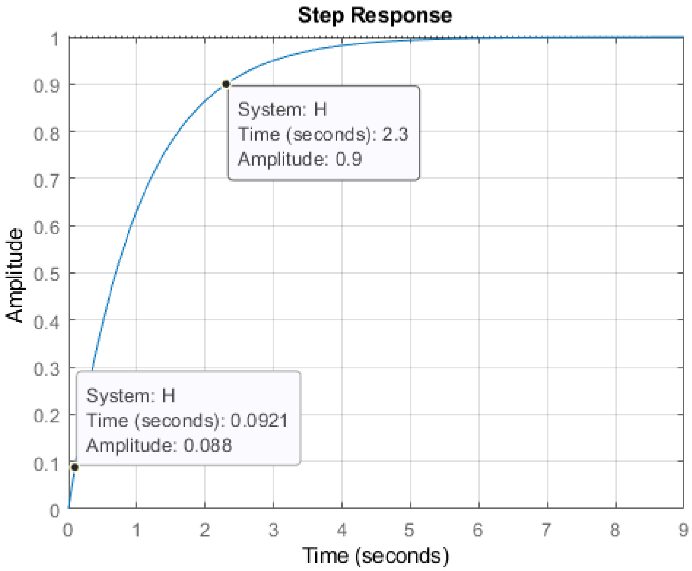
A significant point is that in control system design and analysis we are generally primarily interested in the transient response. Hence in terms of the excitation of x(t) we can approximate this with a truncated Taylor series and then have X(s) expressible in terms of poles, zeros and exponential terms.

Overall, this is why poles and zeros are so fundamental to control systems. We will next look at a the characteristic of a pole.

Consider the usual example of $H(s) = \frac{1}{s+1}$ which has a single pole at s=-1. The step response is

$$\frac{H(s)}{s} = \frac{1}{s(s+1)} \Leftrightarrow (1 - e^{-t})u(t)$$

We can calculate the rise time which is defined as the time taken for the step response to go from 10% of the final value to 90% of the final value. In the plot below we see that this is about 2.2 seconds.



Can we get a closed form expression of the rise time? Consider the more general transfer function of

$$H(s) = \frac{1}{s+p} \Leftrightarrow u(t)\frac{1}{p}(1 - e^{-pt})$$

where we have a single pole at s=-p. We have an equation for the 10% value as

$$0.1 = 1 - e^{-pt_1}$$

and an equation for the 90% value as

$$0.9 = 1 - e^{-pt_2}$$

The rise time is then given as

$$\tau_r = t_2 - t_1$$

Manipulating these two equations results in

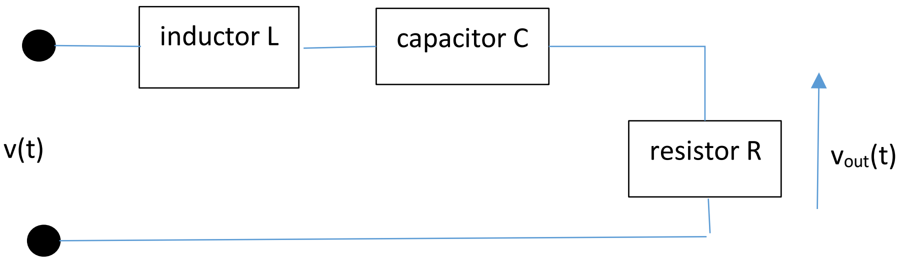
$$\tau_r = \frac{1}{p} ln(9) = \frac{2.2}{p}$$

Note that is consistent with the graph given above.

Hence we can say that a transfer function with a single pole at s=-1 results in the transfer function having a step response as in the above figure with a rise time or response time of about 2.2/p seconds.

When the transfer function has multiple poles then the response time is more complex to calculate. However, we can approximate a given individual pole in a transfer function as having a step response time of about 2.2/p seconds.

Next go back to the second order system with two poles. Reconsider the example of the series LRC



Here the transfer function was written as:

$$H(s) = \frac{\frac{R}{L}s}{s^2 + \frac{R}{L}s + \frac{1}{LC}}$$

The conventional way of notating second order systems is to introduce two variables:

ω - natural resonance frequency

D – damping coefficient

The denominator of the second order system is notated as

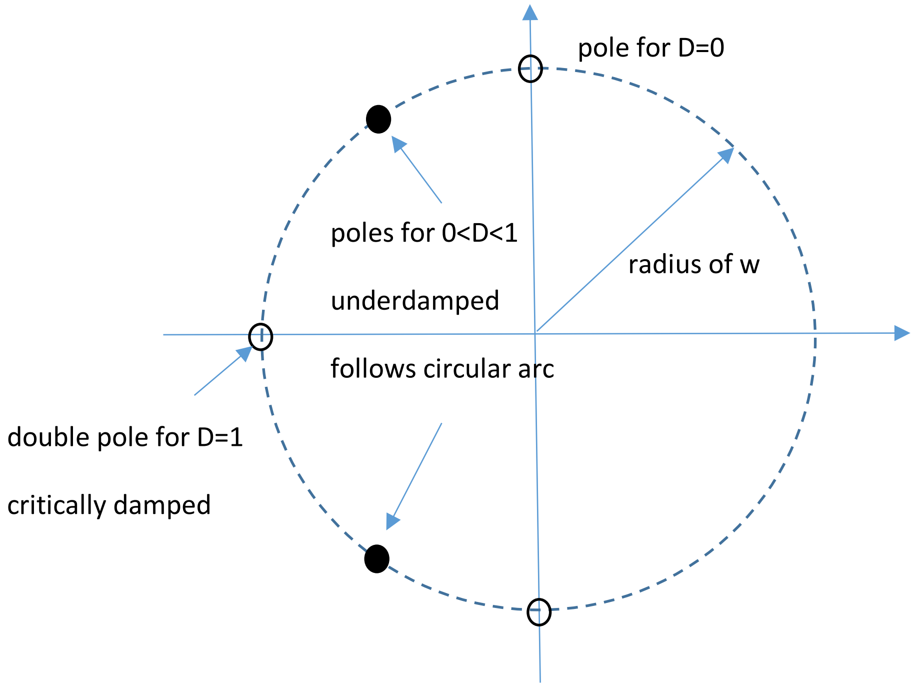
$$s^2 + 2D\omega s + \omega^2$$

The roots are

$$s = \frac{-2D\omega \pm j\sqrt{\omega^2 - D^2\omega^2}}{2} = -D\omega \pm j\omega\sqrt{1 - D^2}$$

Hence the roots have an imaginary part of $\omega\sqrt{1 - D^2}$ and a real part of $-D\omega$.

Note that the smaller D is the closer the pole is to the jw axis. If D=0 then there is zero damping and the poles will be on the jw axis. This is a marginally stable case. If D=1 then the poles are on the real axis and we have a double pole at s=-w.



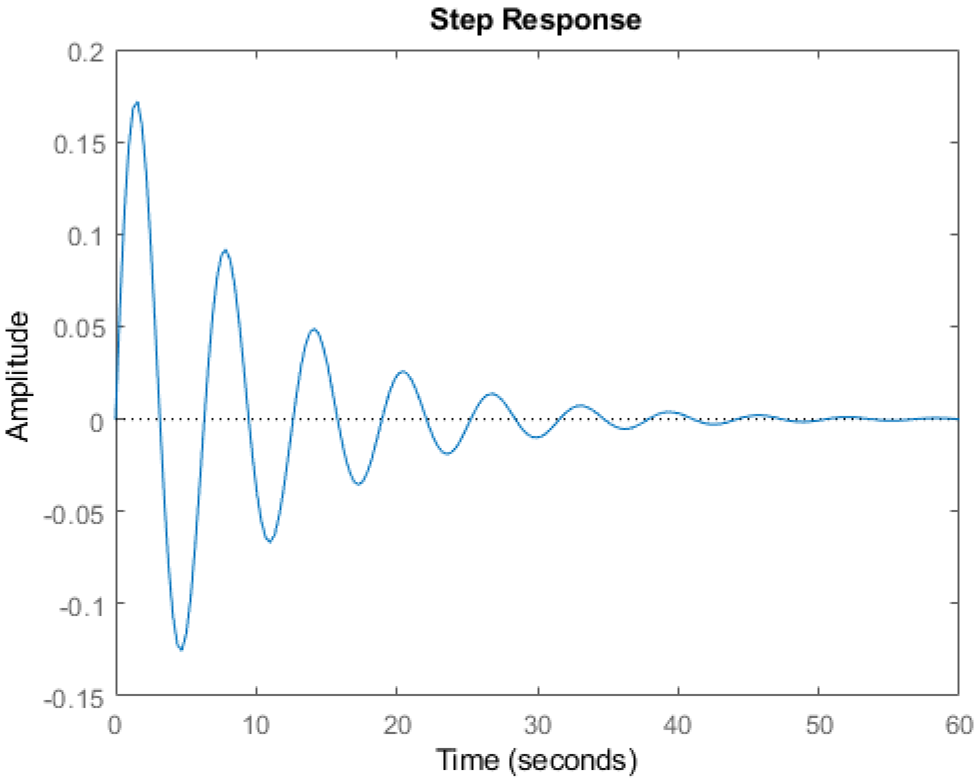
Plot of the location of the poles for D=0, 0<D<1, and D=1. For D>1 the system is overdamped and the poles are on the real axis. For 0<D<1 the poles follow a circular arc of radius of w.

An important observation is that the poles are in conjugate pairs. In general, for realizable transfer functions that have real valued coefficients, the poles are always distributed symmetrically about the real axis in conjugate pairs.

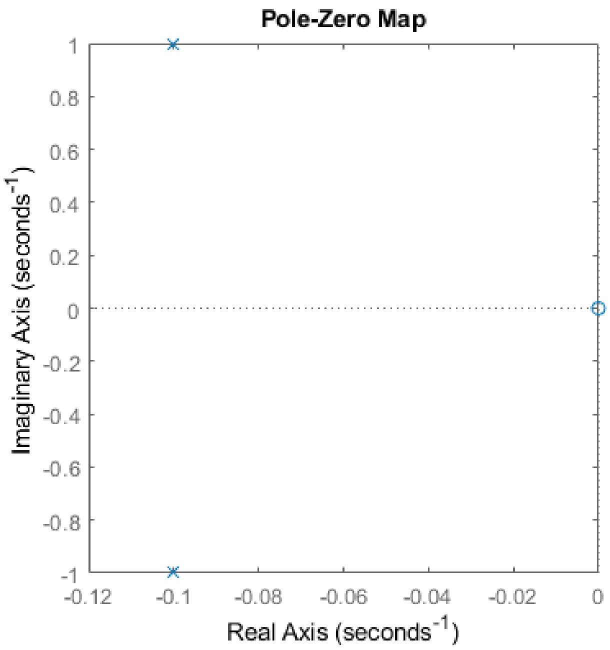
Another observation is that in the second order transfer function, the poles start at ω for D=0 and as D is increased towards 1, the roots follow a circular trajectory of constant radius ω until D=1 when the poles merge and sit on the real axis at s=-w. If D is increased beyond this point then the poles track along the real axis.

The example of the step function for D=0.1 and w=1 is given below:

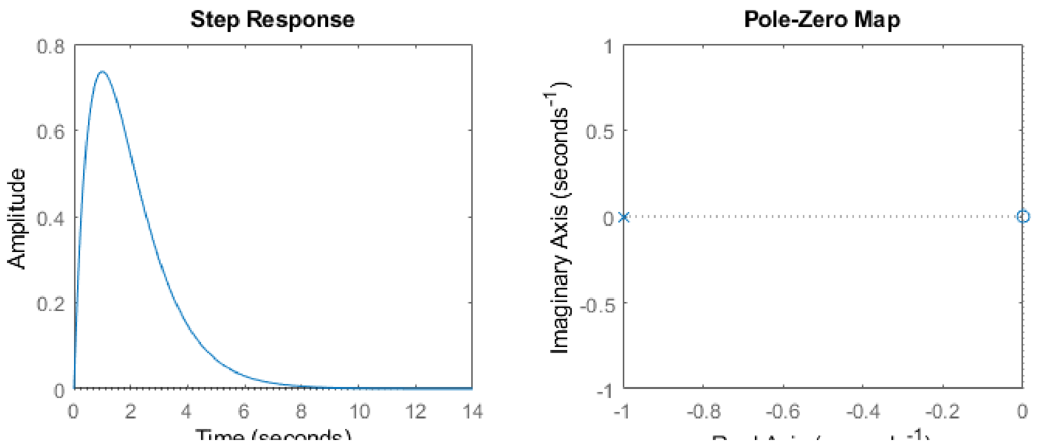
```
D = .1;
w = 1;
H = tf([2*D*w,0],[1,2*D*w,w^2]);
step(H)
```



The conjugate pair of poles and single zero is plotted below:



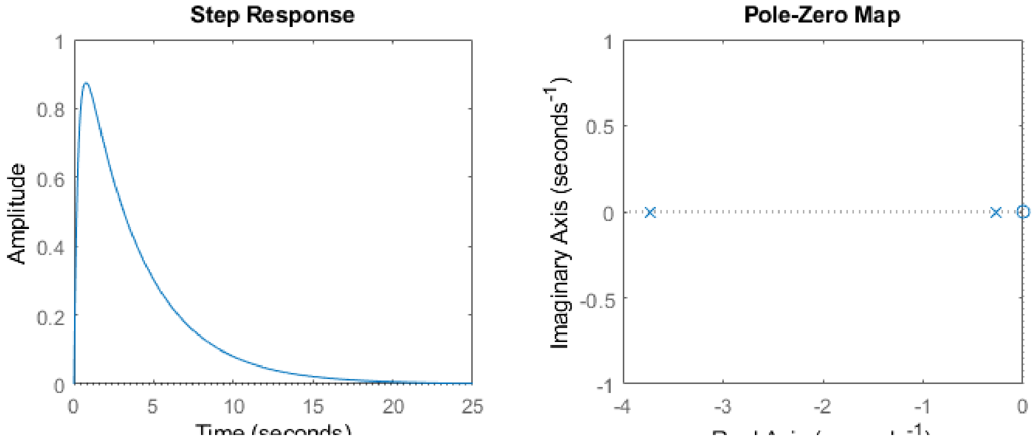
Next the sample for D = 1;



```
D = 1;w = 1;
H = tf([2*D*w,0],[1,2*D*w,w^2]);
figure(1);subplot(121);step(H)
subplot(122);pzmap(H)
```

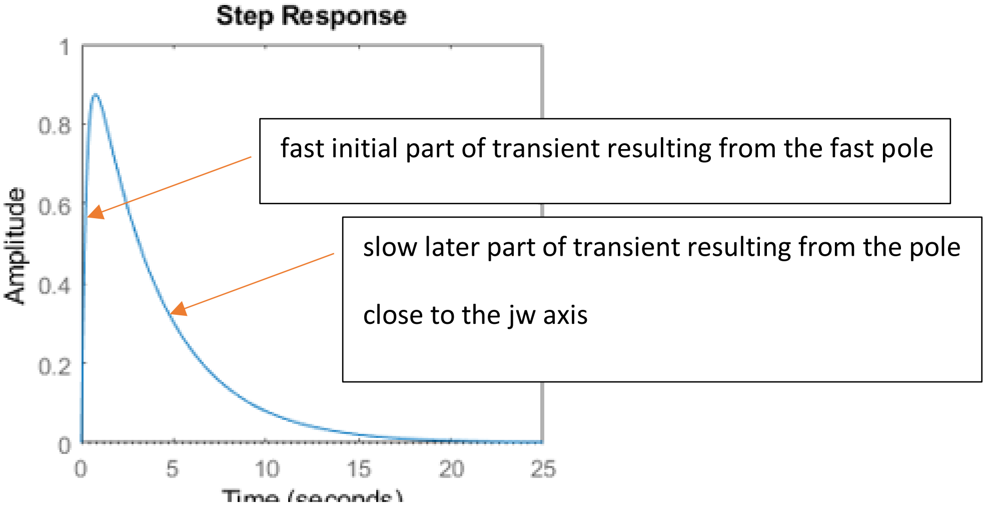
Note the critically damped step response is not oscillatory (no ringing in the transient). The poles are in the real axis at s=-1.

One further example with D=2 and w=1



Here in the overdamped case the poles are on the real axis with one pole close to the jw axis which makes it have a long time constant. The other pole on the real axis near s=-3.7 is much faster and only affects the initial part of the transient.

The observation in the previous example is typical of the transient in control problems. That is there are often a mix of fast and slow poles. The fast pole results in an initial fast transient immediately after t=0. Then there is a slow component of the transient that follows.



fast initial part of transient resulting from the fast pole

slow later part of transient resulting from the pole

close to the jw axis

So that introduces what poles do. What about zeros?

A way of describing what zeros do is as follows. Poles are set by the natural resonances of the system and have specific component responses as discussed. These are the eigenmode responses or the natural resonance responses of the system. Zeros essentially modify the frequency response of the system by superposition and weighting of these eigenmode responses.

This is best described by an example.

$$H(s) = \frac{c}{s+C} + \frac{d}{s+D}$$

H(s) is a weighted superposition of two natural modes. If we combine we get

$$H(s) = \frac{(c+d)s + cD + Cd}{(s+C)(s+D)}$$

In this form there are two poles at s=-C and s=-D and a zero at

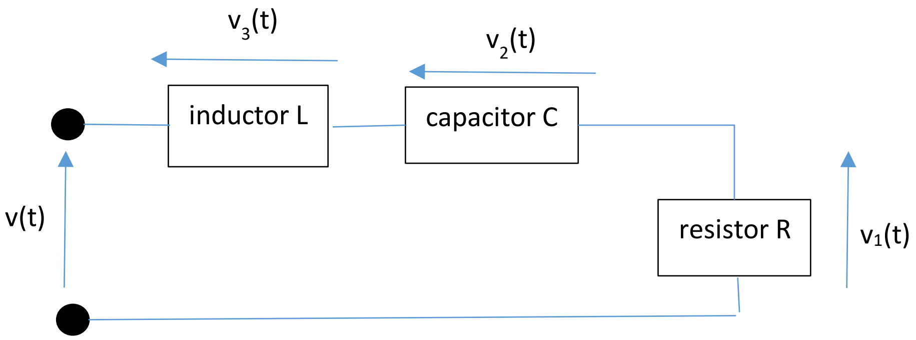
$$z = \frac{cD + Cd}{c+d}$$

Note that the role of the zero at s=z is that it sets the relative weighting of the natural modes of

exp(-Ct)u(t) and exp(-Dt)u(t).

Another example of the role of zeros

Go back to the example of the series LRC and determine the transfer functions from the input applied voltage of v(t) and the voltages across the three components which is taken as $v_1(t)$ across the resistor, $v_2(t)$ across the capacitor and $v_3(t)$ across the inductor.



The transfer functions are given as:

$$H_1(s) = \frac{\frac{R}{L}s}{s^2 + \frac{R}{L}s + 1/LC}$$

$$H_2(s) = \frac{\frac{1}{LC}}{s^2 + \frac{R}{L}s + \frac{1}{LC}}$$

$$H_3(s) = \frac{s^2}{s^2 + \frac{R}{L}s + 1/LC}$$

The second order LRC circuit is interesting as the voltage taken as the output variable for the three different components makes the difference between a low pass, band pass and a high pass filter circuit. H1(s) is a bandpass circuit with one zero at the origin. H2(s) is a low pass with no zeros at the origin and H3(s) is a high pass with two zeros at the origin.

Note in these transfer functions we have the natural resonance as

$$w = \frac{1}{\sqrt{LC}}$$

and the damping given by

$$2Dw = \frac{R}{L}$$

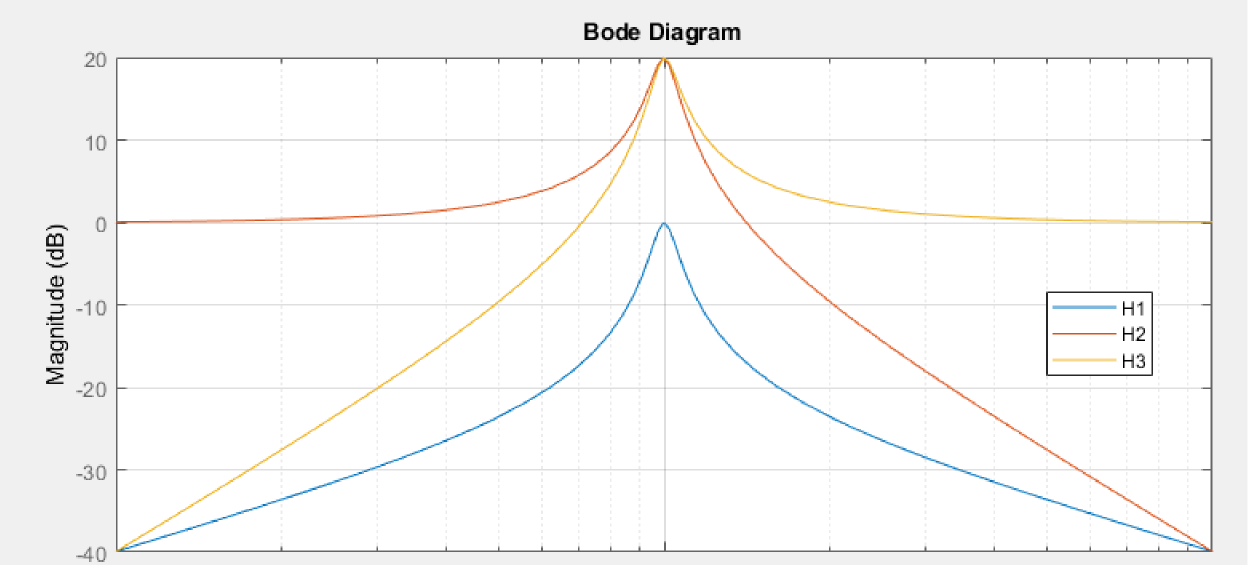
such that

$$D = \frac{R}{2wL} = \sqrt{\frac{C}{L}\frac{R}{2}}$$

Example with L=C=1 and R = 0.1

With these values the resonance frequency is 1 and the dampling coefficient is 0.05.

```
D = .05;
w = 1;
H1 = tf([2*D*w,0],[1,2*D*w,w^2]);
H2 = tf(w^2,[1,2*D*w,w^2]);
H3 = tf([1,0,0],[1,2*D*w,w^2]);
bode(H1,H2,H3);
grid on;
legend('H1','H2','H3');
```



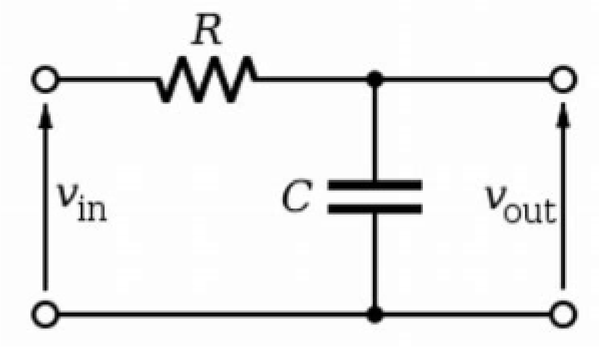
H1(w) in blue is a bandpass response with one zero at the origin

H2(w) in red is a lowpass response with no zeros at the origin

H3(w) in yellow is a highpass response with two zeros at the origin

Introduction to state space formulation of systems

Initially we started with the example of the first order transfer function of the RC network shown in the diagram below.



As discussed, we look for elements of energy storage and associate state variables with the variable associated with the energy value.   We have

$$i = C\frac{dv_{out}}{dt}$$

where i is the current. As there is only one energy storage device we know that the DEQ that represents the overall transfer function can be written as a first order equation. We also look for the variable that is a time derivative of other variables which in this case is v_{out}. This becomes our 'state variable'. We also have

$$i = \frac{1}{R}(v_{in} - v_{out})$$

Combining we write the DEQ in the format

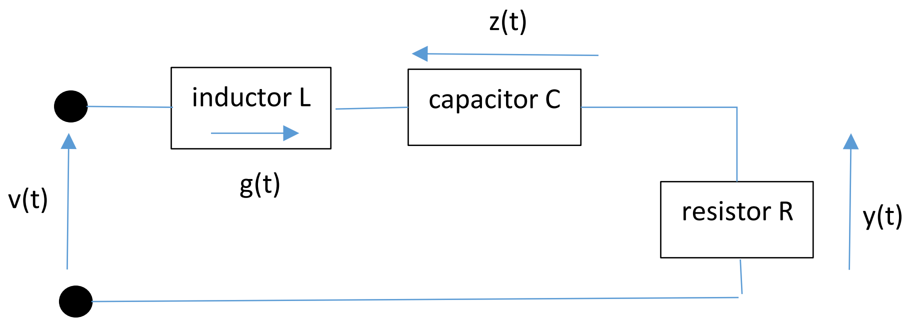
$$\frac{dv_{out}}{dt} = -\frac{1}{RC}v_{out} + \frac{1}{RC}v_{in}$$

That is we have the equation as:

{time derivative of state variable}

= {linear weighting of state variables} + coefficient {independent input source}

Next we will consider a second order system and in this way also introduce the concept of state space.



Here we have v(t) as the independent input and y(t) as the output. We first start by looking for appropriate state variables which are:

z(t) - The voltage across the capacitor

g(t) – the current through the inductor

Now we have

$$\frac{dz}{dt} = \frac{1}{C}g$$

where g is the current through the capacitor which is fortunately in this case also a state variable. For the inductor we have

$$\frac{dg}{dt} = \frac{1}{L}(v - z - y)$$

Looing on the RHS y is an output but not a state variable and we need to express this in terms of state variables {g,z}. Hence y=Rg. Therefore write

$$\frac{dg}{dt} = \frac{1}{L}(v - z - Rg)$$

Now we have two state equations of the derivatives of the state variables on the LHS as a function of the input and the state variables on the RHS. We can chose to combine these into a single differential equation relating the input independent source variable of v(t) to the output dependent variable y(t). However, this is unnecessary and is in general tedious and error prone to do. Instead we solve this in a systematic way by using state space formulation and analysis.

First having identified the state variables we put them into a vector format that few denote x, the state variable vector.

$$x = \begin{bmatrix} z \\ g \end{bmatrix}$$

Then we write the two state equations in a matrix form as

$$\frac{d}{dt}\begin{bmatrix} z \\ g \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} z \\ g \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} v$$

This is the main part of the state space formulation. The matrix of $A = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix}$ is the system matrix and the matrix $B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}$ is the input matrix. Hence we have

$$\frac{d}{dt}x = Ax + Bv$$

To complete the state space formulation express the output variable y in terms of the state variables and the input as

$$y = \begin{bmatrix} 0 & R \end{bmatrix}\begin{bmatrix} z \\ g \end{bmatrix} + [0]v$$

or

$$y = Cx + Dv$$

where $C = [0 \quad R]$ and D=0.

The matrices of {A,B,C,D} are the set of state space matrices representing the system and x is the state variable vector.

The matrix formulation for the second order LRC system given above is somewhat overkill. However, it should be clear that for a system of higher order and hence complexity a systematic approach to generating the set of DEQs that represent the system is necessary. The state space approach aptly fills that role.

Furthermore, there are many advantages of the matrix formulation. The poles of the system are determined as the eigenvalues of the system matrix A. This is a huge insight into how the system will behave. We will pick up on this later. Also the transfer function from the input to the output can be determined from the {A,B,C,D} matrices. We will derive this later when it is needed.

State space to transfer function

Often it is necessary to convert the state space {A,B,C,D} to a transfer function. Matlab does this as

[num,den] = ss2tf(A,B,C,D}

where num is the numerator polynomial coefficients and den is the denominator coefficients.

What it does is compute the transfer function as

$$H(s) = C(Is - A)^{-1}B + D$$

This is derived as follows:

First we get the transfer function of the state variables as a function of the input based o

$$\frac{d}{dt}x = Ax + Bv$$

This is transformed into Laplace where initial conditions are taken to be zero. Then we get

$$sX = AX + BV$$

where V is the laplace transform of the input excitation signal v(t). Then we can write

$$(sI - A)X = BV$$

where I is the identity matrix of dimension which is the same as the number of state variables. Assuming that the matrix of sI-A is invertable then

$$X = (sI - A)^{-1}BV$$

Now we use

$$y = Cx + Dv$$

which then gives

$$Y(s) = (C(Is - A)^{-1}B + D)U(s)$$

which gives the transfer function as above.

Dominant and Secondary poles

Generally in control systems there will be a number of poles in the transfer function to consider. However, typically only one to two of these are close to the jw axis and the rest are further into the LHP. Poles that are close to the jw axis have long transients in time and poles further away from the jw have relatively short transients. Poles that are close to the jw axis are the dominant poles as they control how fast the transfer function settles given an input excitation of an impulse function or a step function. Poles further away from the jw axis are the secondary poles as they have little influence of the settling time of the transfer function.

As an example consider

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

where the pole is on the real axis at s=-p. The impulse function is

$$h(t) = u(t)e^{-pt}$$

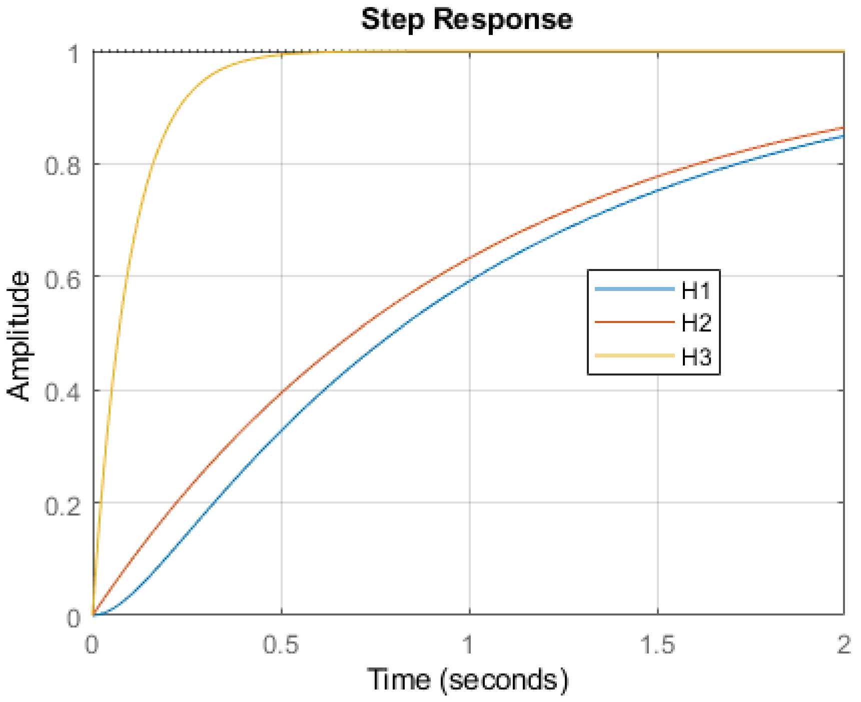
and therefore the settling time is on the order of 1/p.

Now consider the transfer function of

$$H(s) = \frac{10}{(s + 1)(s + 10)}$$

Here we have two poles, one at s=-1 and one at s=-10. The pole at s=-1 is relatively close to the jw axis and is the dominant pole. The pole at s=-10 is the secondary pole.

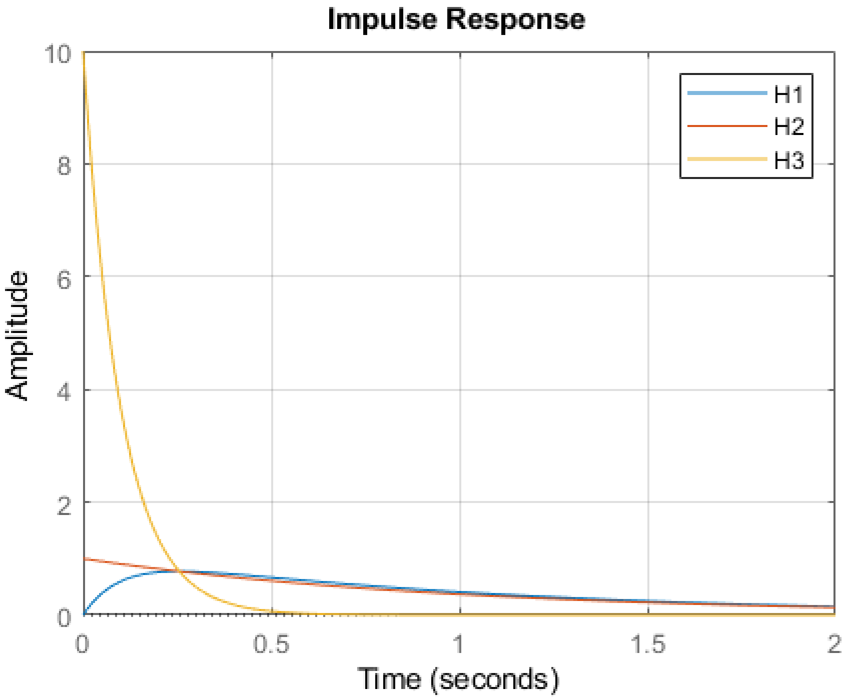
```
H1 = tf(10,[1 11 10]);
H2 = tf(1,[1 1]);
H3 = tf(10,[1,10]);
step(H1,H2,H3,2);
grid on;
legend('H1','H2','H3');
```



Note that the impulse response of $H_1(s) = \frac{10}{(s+1)(s+10)}$ is almost the same as the impulse response of

$$H_2(s) = \frac{1}{s+1}$$

which only includes the dominant pole at s=-1. It is also interesting to look at the impulse response which is given in the figure below:



Note that the impulse response of H1 converges to the impulse function of H2 at longer times. However, the impulse response of H2 is not a good approximation for the impulse response of H1 for short time lags between t=0 and t=0.25. This is because the faster secondary pole is not included. Hence if the transient response immediately after the impulse excitation is important then the secondary poles must be included. Otherwise if we are just interested in the transient response after a short interval after the impulse response then the primary or dominant poles are sufficient. The concept of dominant poles is therefore a convenient way of getting an approximation of the transient provided we are not interested in the response immediately after the impulse excitation.

Example of a pair of dominant poles

Suppose we have a system of

$$H_1(s) = \frac{0.2s}{s^2 + 0.2s + 1}$$

in cascade with a system of

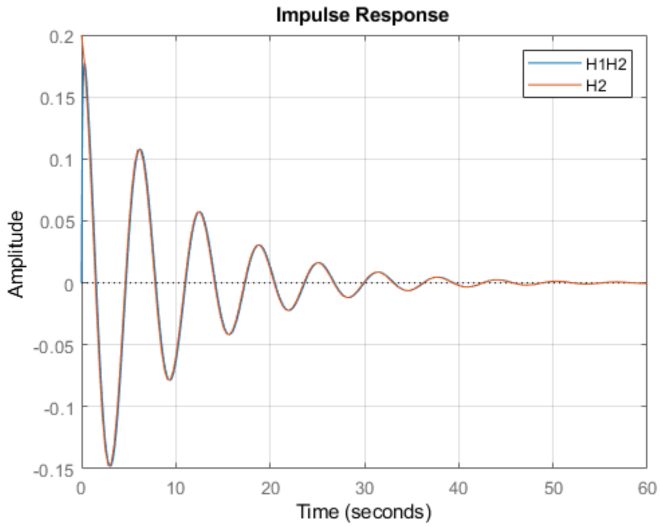
$$H_2(s) = \frac{10}{s + 10}$$

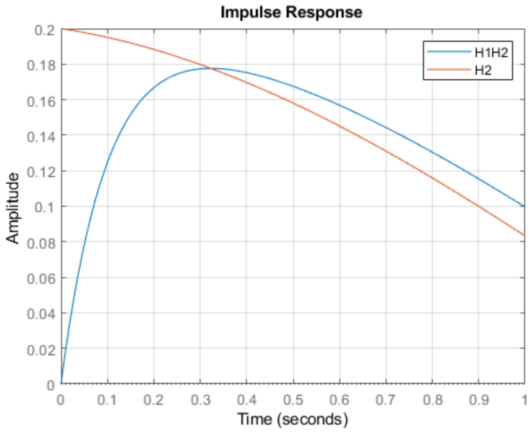
such that the system considered is $H = H_1H_2$. Plot the impulse response of H and H1 and compare.

```
H1 = tf([0.2,0],[1,0.2,1]);
```

```
H2 = tf(10,[1,10]);
impulse(H1*H2,H1)
grid on;
legend('H1H2','H2');
```

The impulse response is shown in the figure below with a zoomed in portion around t=0

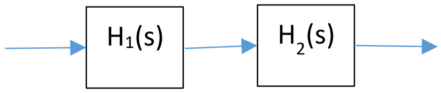




The poles of H_1(s) are $s = -0.1 \pm j\sqrt{0.99}$ and the pole of H2 is at s=-10. Hence the two conjugate poles of H1(s) are the primary or dominant poles and the pole of H2 is the secondary. Note that in plot of the impulse responses that the H2(s) does not contribute except for the portion of the impulse response right after t=0.

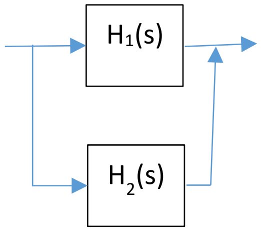
Series, Parallel and feedback connections of transfer functions

So far we have considered several examples of series connections of transfer functions as shown in the figure below:

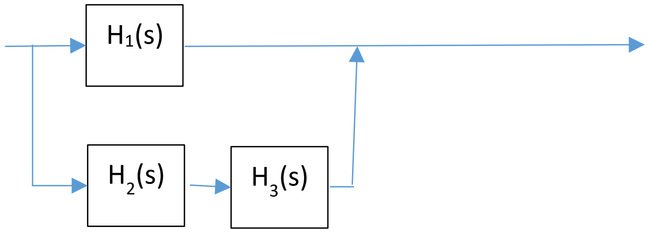


The overall transfer function of the series connection can be written as $H=H_1H_2$

A parallel connection of H1(s) and H2(s) as shown in the figure below can be expressed as H1+H2.

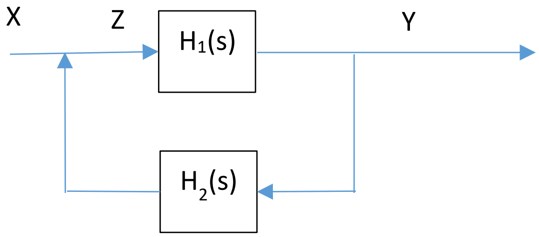


Note we do not explicitly present the sum block as this is implied with the arrow connection. We can of course generalize this as combinations of series and parallel connections is below:



The overall transfer function here can be expressed as $H_1+H_2H_3$.

Next consider feedback



The relation between the input and output is

Y=H1 Z = H1 (X+H2 Y)

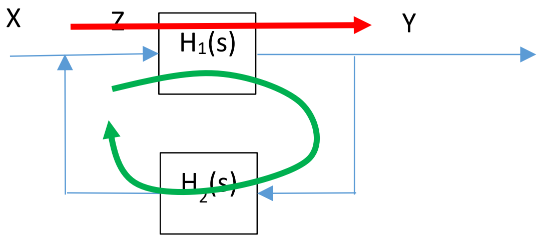
such that

Y(1-H1H2) = H1 X

and therefore the transfer function of the feedback system is

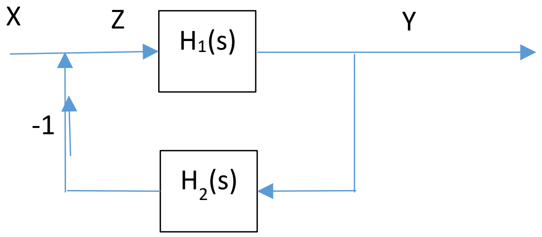
$$H = \frac{H_1}{1 - H_1 H_2}$$

We will be using this feedback connection extensively in ENEL441 so memorize this equation so you don't have to rederive it every time. A better way of remembering it that can be applied in general is that H is equal to the 'through connection' over 'one minus the loop gain'. This is illustrated in the following diagram:



Definition of the through connection as the red arrow connecting the input to the output being the product of all of the components encountered in the through path. The loop gain is the product of all of the components encountered in the feedback loop.

This feedback loop is a positive feedback as the feedback signal is not multiplied by -1 prior to the summing with the input signal. In ENEL 441 we will generally encounter negative feedback loops wherein the feedback is multiplied by -1 prior to summing as indicated in the diagram below.



The multiplication by -1 is indicated by the '-1' and arrow. However, often we will ignore the arrow as the direction of signal flow is obvious. In this case the transfer function of the negative feedback loop is given as

$$H = \frac{H_1}{1 + H_1 H_2}$$

As a first example consider that we have a transfer function of a plant given as

$$H_1 = \frac{1}{s+1}$$

and that a negative feedback loop is applied around the plant with

$$H_2 = G$$

The transfer function of the overall negative feedback system is

$$H = \frac{\frac{1}{s+1}}{1 + G\frac{1}{s+1}}$$

which simplifies to

$$H = \frac{1}{s+1+G}$$

It is interesting to see what the feedback accomplishes here. Note that H_1 has a single pole at s=-1. When the negative feedback is added then the pole is moved to s=-1-G. Hence the feedback can be used to move the pole of the transfer function. Suppose that we considered the plant with an initial pole of s=-1, resulting in a step function rise time of 2.2 seconds is too slow. We would like to move this pole to say s=-4 so that the rise time is a factor of four faster. Then we can use feedback consisting of a gain block of gain G=3.

This control over the pole position is fundamental to classical feedback control systems in that the feedback can be arranged around a plant transfer function to move the poles to a *more desired location*. This *more desired location* is such that the impulse or step response of the plant with feedback is more desirable than the response of the plant without the feedback.

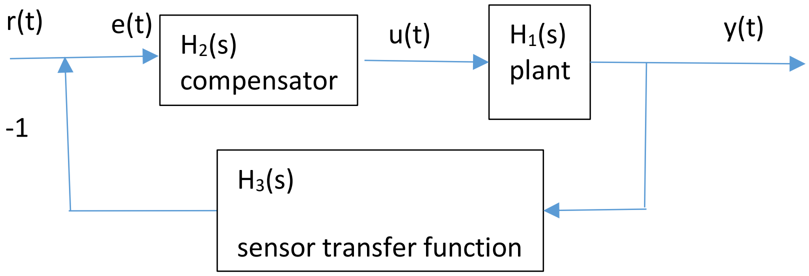
We need a couple of definitions at this point. In the above example the **open loop transfer function** is the transfer function from X to Y without the feedback. This is given as $H_1(s)$. Generally, the notation that will be used is

$$H_{ol} = H_1$$

If the feedback path is closed around the plant then the overall transfer function is the **closed loop transfer function** which for this example is

$$H_{cl} = H = \frac{1}{s+1+G}$$

The configuration for the general negative feedback control problem is shown in the figure below.



Here r(t) is the input reference signal which is typically the desired output of the plant which is denoted by y(t). The sensor transfer function is a measure of the output y(t) but the measure is not perfect and therefore a transfer function of H_3 is considered here. H_3 may model the sensor measurement delay or frequency response. Regardless, the difference between the desired reference input, r(t) and the sensor output from H_3 is the error signal e(t). This error signal passes through the compensator with a transfer function of H_2 with an output of u(t) that is the drive signal into the plant with a transfer function of H_1. The overall closed loop transfer function of

$$H_{cl} = \frac{H_1 H_2}{1 + H_1 H_2 H_3}$$

has poles and zeros. The compensator of H2 is designed to optimally place these closed loop poles and zeros. 'Optimum' is based on criteria such as transient response of the closed loop system, stability and cost of operation etc. In a subsequent section there will be detailed discussion of design optimality, and implementation.