070928/ Thomas Munther
Halmstad University
School of Information Technology, Computer Science and Electrical Engineering

Exercises in Matlab/Simulink III

The goal with the exercises is to get started with easy problem solving in Simulink and Matlab.
But the need of remembering and writing all commands is not necessary by using the help menu in a proper you can get a lot help from Matlab. Today we are going to do some simple exercises examaning data sequences and try find out the signals hided in them.
The help menu gives us very much help if we are interested in working with signals and systems. When you are in the command window. Enter Help-> Matlab Help.
In the help browser you enter the left window denoted Help Navigator. Then look under Contents. Here you find the central for all help in Matlab. All toolboxes are listed here and by clicking here a whole new world opens up and can be used for your purpose.

Example: we will start by loading data vectors they are saved in a format (–mat) and can be found on my homepage.

```
>> load dataS&S
```

Now look in the workspace and see what variables we downloaded !
One is x1n its a vector consisting of 10000 values. Make a plot of it !

```
>> plot(x1n)
```

The plot in figure 1 shows a very noisy signal hiding one or more sine waves.

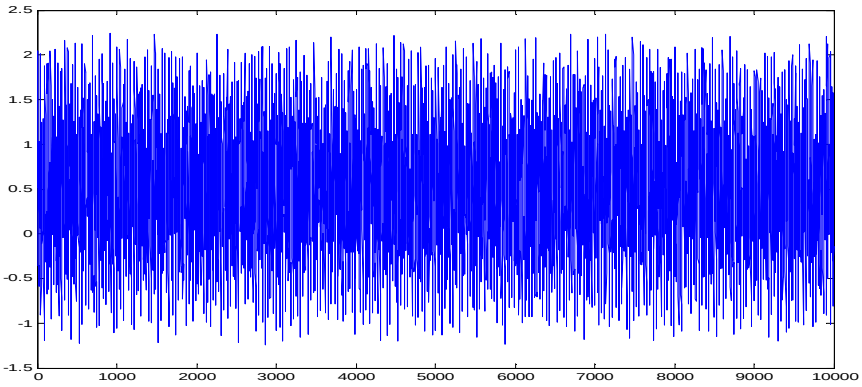


Figure 1

How can we find out the frequencies in the data sequence ? If we plot the spectrum of the signal. This could give us the information we are looking for.
We will make a Discrete-Time Fourier Transform, but in a practical case when we use matlab. This means FFT (Fast Fourier Transform).

```
>> y=fft(x1n,512);     % creates a complex vector. The number of points to calculate the
                       % DTFT is 512 points. The FFT-algorithm uses 2^N points.
>> plot(abs(y));       % plots the absolute value of y.
```

The FFT algorithms are more effective if the number $N = 2^M$, where M is an integer. If this is not true one usually adds some zeros (zeropadding) to the end of the sequence to make the algorithm faster. This is done automatically by the command fft in Matlab. If we have a sequence of length N=300 samples then we must add 212 zeros. That means that M=9. If M=8 the best length for FFT is then N=256.

Notice in figure 2 we have on the x-axis only the index of the vector. This gives us no clue of the frequency content.
You can also see that it seems to be symmetric. This is no coincidence because the DTFT is periodic and there will be a repetition of the spectrum if we continue higher in frequency. We are now only going to use the first half of the points. They correspond to the frequencies up to $f_s/2$. The other part we can skip.

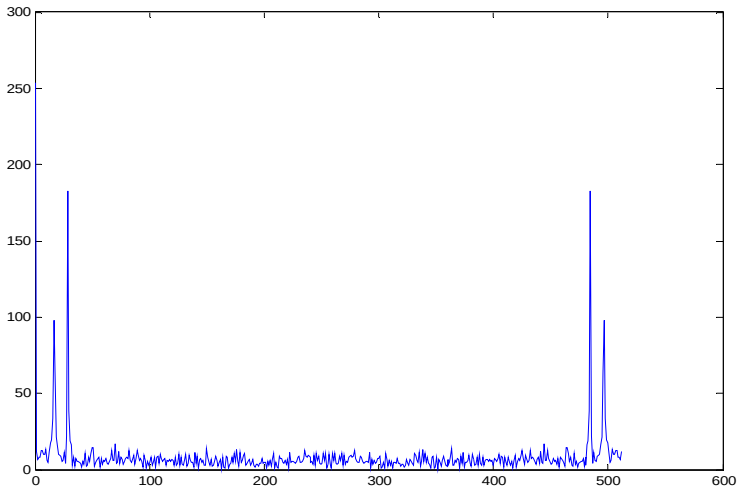


Figure 2

If we try to show how it would look like if we pretend the x-axis to be the frequency.

```
>> y1=fftshift(y);
>> plot(abs(y1))
```

Notice that the highest peak is the DC frequency and to the left negative frequencies and to the right positive frequencies.

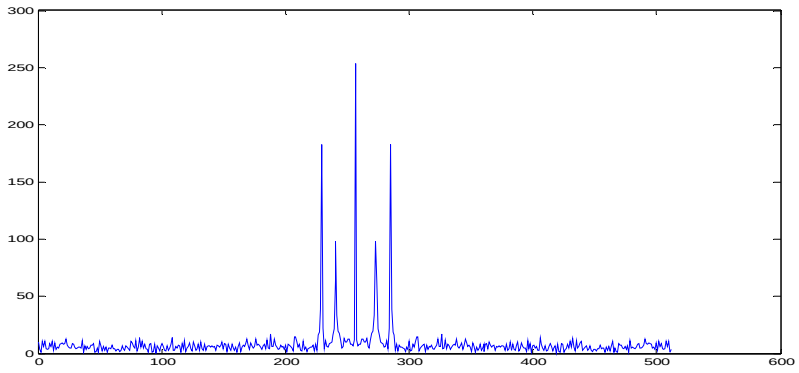


Figure 3

Return to figure 2 ! Now let's see, how should we move on ?

```
>> Pyy = y.* conj(y) / 512;    % Calculate the power of the complex vector y.
                               % Consists of 512 values.
>> f=1000*(0:256)/512;         % Makes a frequency vector from 0 to 500,
                               % with 257 values.
>> plot(f,Pyy(1:257))          % Now plot the power against frequency.
```

The figure 4 below presents the frequency content from 0 DC up to half the sampling frequency. In this case 500 Hz. What frequencies do we find in the our original signal ?

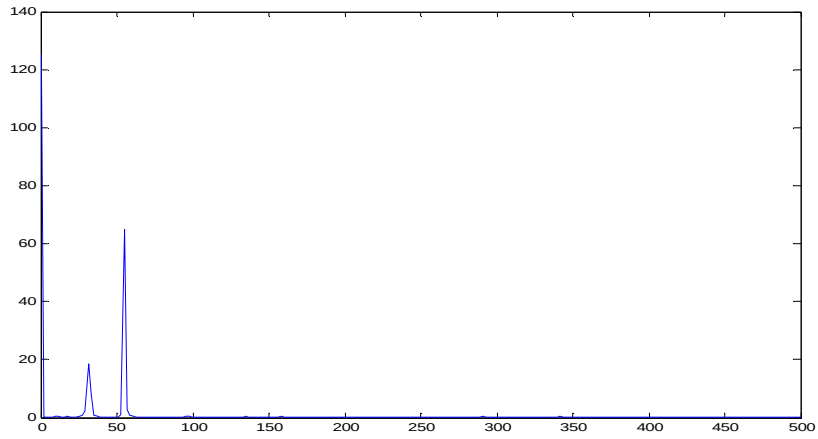


Figure 4

If we use the zoom I think we can identify the frequencies we started with.

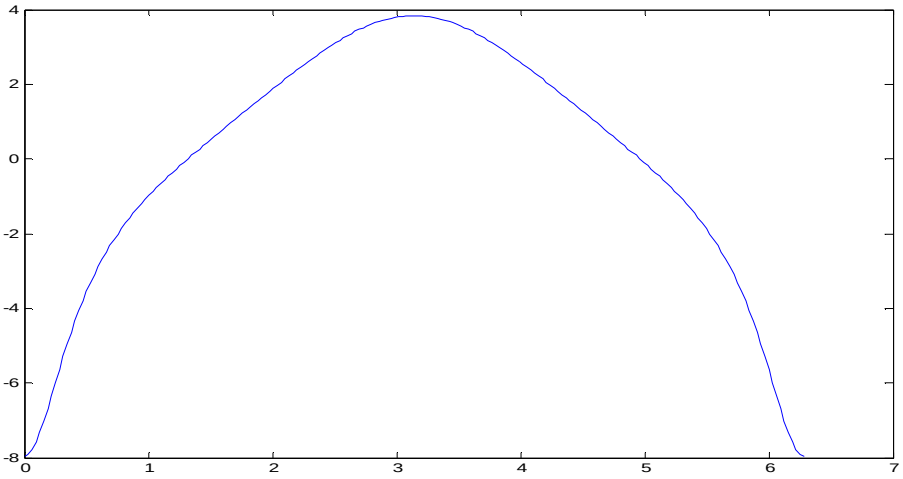
The frequency response for a discrete-time system: evaluate the response of a system described by a difference equation. Our system looks like :

$a_0 * y[n] + a_1 *y[n-1] + a_2 *y[n-2] = b_0*x[n] + b_1 *x[n-1]$

Do the folllowing in matlab command window.

```
>> a= [ 1 -1/4 -1/8];    % creates a vector with coefficients. a=[ a0 a1 a2 ]
>> b=[ -3/4 1];          % creates a vector with coefficients b=[ b0 b1]
>> w=0: pi/100:2*pi;           % frequency vector
>> H=freqz(b,a,w);    % evaluates the frequency response at different frequencies.
>> plot(w, 20*log10(abs(H)))          % plots magnitude [ dB] versus frequency.
```

Gives us the magnitude response in the frequency range 0 to 2π.



Suppose we instead would investigate a continuous-time system given by its differential equation : $d^2/dt^2(y(t)) + d/dt(y(t))-2*y(t)=2x(t)$

The transfer function expressed in the Laplace operator would look like:
$G(s)= 2/(s^2+s – 2)$
In matlabs command window we write as follows:

```
>> b=[ 2];                 % coefficients of the numerator in descending order.
>> a=[ 1 1 -2];            % coefficients of the denominator in descending order
>> w=logspace(-2,2)        % makes a logarithmic x-axis in the range 10^-2 to 10^2 rad/sec.
>> freqs(b,a,w)            % calculates and plots the frequency response.
```

Our plot is seen in figure 6 ! The upper plot gives us the magnitude response and the lower one is the phase response.

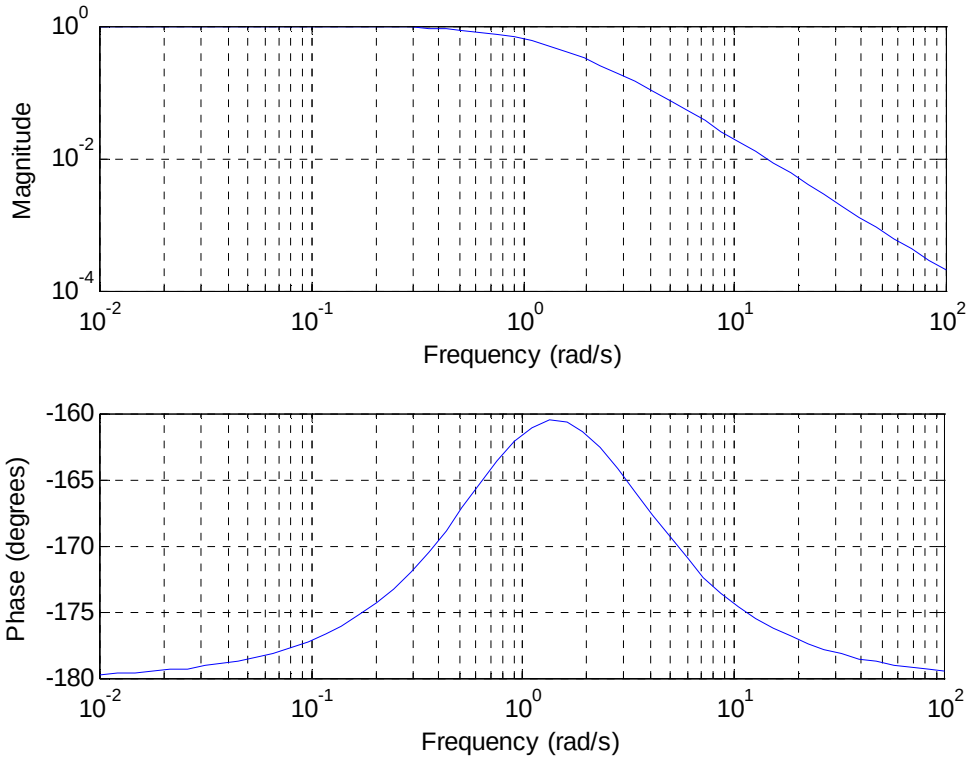


Figure 6

We can also produce this Bode plot by actually using this command in matlab, but this can be done first by calculating the transfer function from vector a and b.

```
>> G=tf(b,a)

Transfer function:
    2
-----------
s^2 + s - 2
```

The bode command wants the transfer function from a LTI-system as argument.

```
>> bode(G), grid        % Calculates and plots both magnitude and phase response.
                        % Finds the best frequency interval itself.
```

See figure 7 !

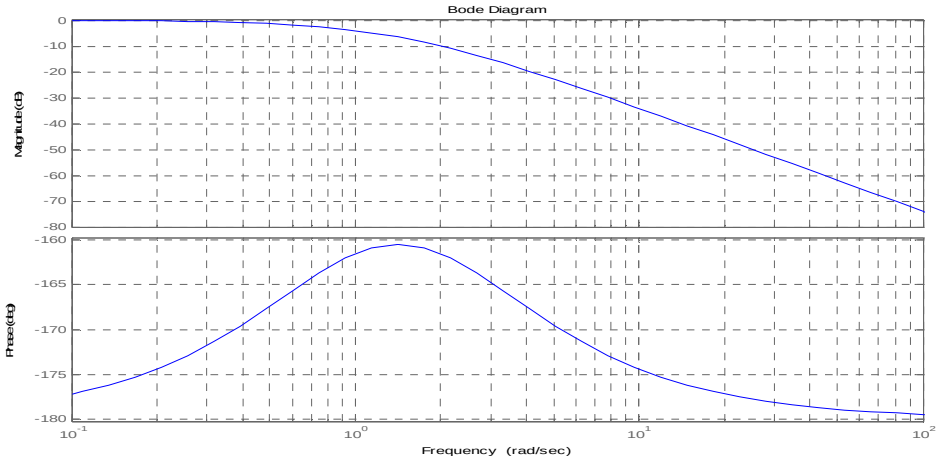


Figure 7

If we have the transfer function of this contiuous-time LTI system we could also use other commands im matlab to see how the system would react to different input signals. Try step, impulse and pzmap !

>> step(G); % gives us the step response from the system.
>> impulse(G); % impulse response

It seems that the system is unstable according the responses. Let's find where the poles are situated in the s-domain.

>> pzmap(G) % plots the zeros (circles) and poles (crosses) of the transfer function.

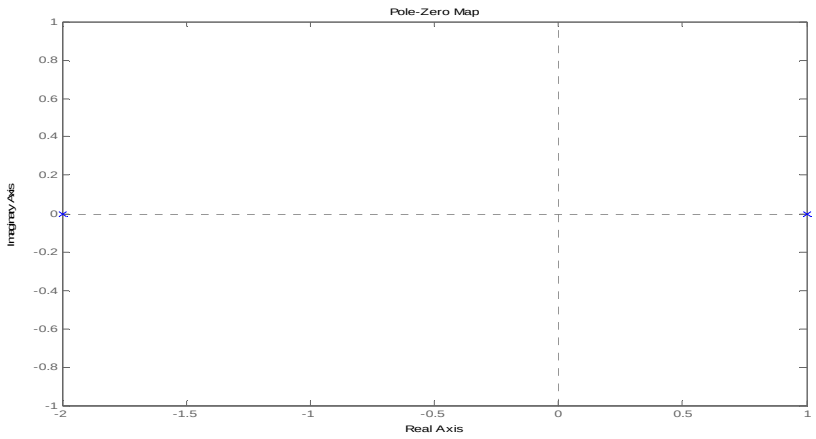


Figure 8

If you would like to find out the explicit value of the poles. Write !
>> roots(a) % calculates the roots of the polynomial given the coefficients.

There are many different ways to describe a system. We could for instance describe a system by its gain, poles and zeros. Assume a system described by two vectors containing poles and zeros. We denote them by p and z respectively.

```
>> p=[0 -1 -4];        % values of poles.
>> z=[-2];             % values of zeros.
>>k=10;                 % gain.
>> G1=zpk(z,p,k)       % creates a transfer function G1.

Zero/pole/gain:
  10 (s+2)
-------------
s (s+1) (s+4)
```

It is simple enough to find the locations of poles, zeros and the gain within a transfer function description using the commands tzero, pole and dcgain.

```
 >> zero(G1)          % gives a vector with the zeros, here: -2
>> pole(G1)          % gives a vector with the poles, here: 0,-1 and -4
>> dcgain(G1)        % Notice that this is the DC- gain (when s->0 in G1), here: Inf

>> G=tf(G1);         % The transfer function as a ratio between two polynomials.

Transfer function:
    10 s + 20
-----------------
s^3 + 5 s^2 + 4 s
```

Inverse Laplace Transform by the use of Symbolic Math Toolbox

Given a rational transform X(s) = B(s)/A(s) with the degree of B(s) less than the degree of A(s). Symbolic Math Toolbox can be used to compute and plot the Inverse Laplace Transform x(t).
If X is a symbolic function of s, then the command ”ilaplace(X)” returns the inverse Laplace transform of X(s) and finally use ”ezplot(x)” to plot x(t).

Example: our transfer function:

$$\frac{s+2}{s^3+4s^2+3s}$$

To compute x(t), use the commands

```
>> syms        X s x                % defining the symbolic variables.
>> X = (s+2)/(s^3 + 4s^2 +3s);
>> x= ilaplace(X)      % this returns x(t)
```

Then plot it by using: ezplot(x,[0,10]) % time vector goes from 0 to 10.

Sometimes we want to find the solution for an arbitrary input signal applied to a system. The system needs to be expressed as a transfer function. To compute the system output resulting from an input x(t), we need a time vector containing the values of t for which y(t) will be computed.

Example: a causal system Transfer function:

$$\frac{s^2 + 2s + 16}{s^3 + 4s^2 + 8s}$$

The response from the system to an exponential input is obtained from the following:

```
>> num=[ 1 2 16];      % numerator
>> den=[ 1 4 8 ];              % denominator
>> H=tf(num,den);
>> t= 0:10/300:10;
>> x=exp(-2*t);
>> y=lsim(H,x,t)               % simulation of continuous-time linear system.
>> plot(t,y)
```

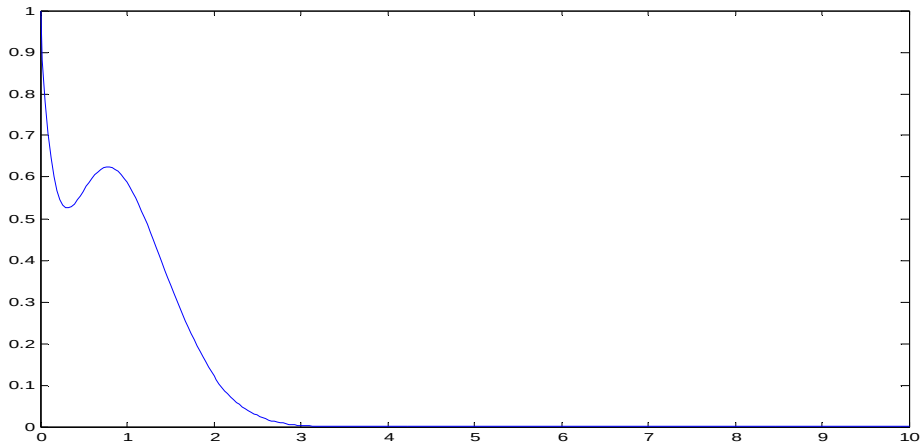


Figure 9

Frequency transformation

Starting from any lowpass filter having transfer function H(s), we can modify the cutoff frequency of the filter or construct highpass, bandpass or bandstop filters by transforming the frequency variable s. This is very easy to perform in Matlab.
The standard filter has a normalized cutoff frquency of 1 rad/sec. The resulting filter can be transformed with the use of the commands: lp2lp, lp2hp, lp2bp, lp2bs

To convert a LP-filter with a $\omega_c = \omega_1$ to a HP-filter with $\omega_c = \omega_2$ substitute s with $\omega_1\omega_2/s$.
To convert a LP-filter with a $\omega_c = \omega_1$ to a LP-filter with $\omega_c = \omega_2$ substitute s with $\omega_1 s/\omega_2$.
To convert a LP-filter with a cutoff frequency ω_c to a BP-filter with a bandwidth from

ω_1 to ω_2. Substitute s in H(s) by $\omega_c(s^2 + \omega_1\omega_2)/(s(\omega_2 - \omega_1))$!
To convert a LP-filter with a cutoff frequency ω_c to a BS-filter with a bandwidth from ω_1 to ω_2. Substitute s in H(s) by $\omega_c s(\omega_2 - \omega_1)/(s^2 + \omega_1\omega_2)$!

Example: Let us start with a lowpassfilter H(s) = 1/(s+1), ω_c=1 rad/sec and convert this into a lowpass filter with ω_c=100 rad/sec.

```
>> H=tf(1,[1 1]);
>> bode(H), grid
```

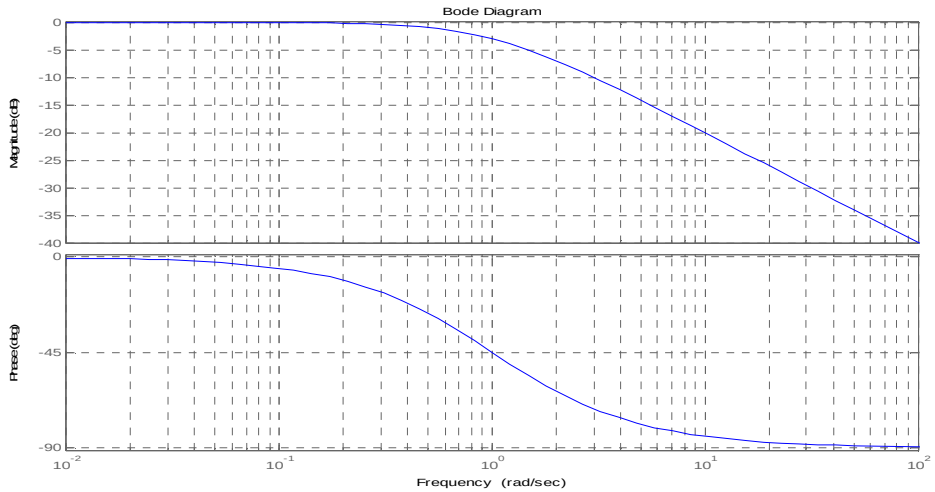


Figure 10

This is the original lowpass filter with ω_c=1 rad/sec.

```
>> [num1,den1] = lp2lp(1,[1 1],100);   %
>> bode(num1,den1),grid       % Plots the bode plot from the numerator and denominator
                              % coefficients.
```

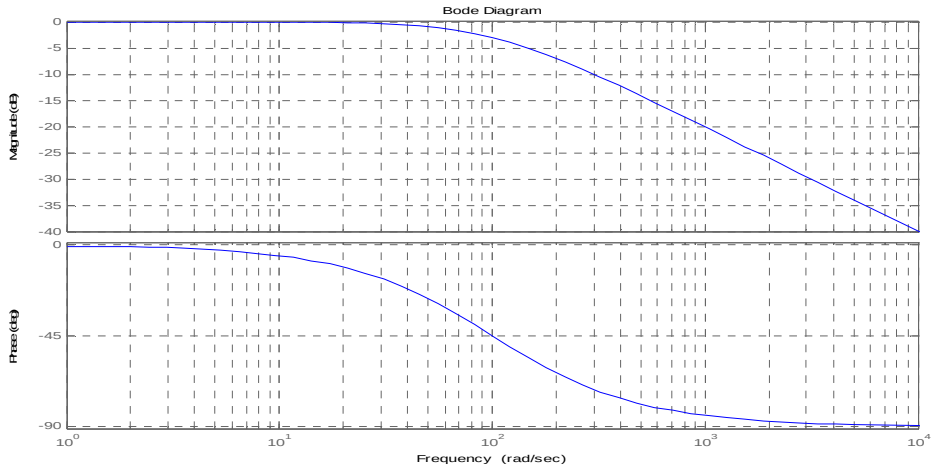


Figure 11

To create a bandpassfilter use the following transformation

```
>> [num1,den1] = lp2bp(1,[1 1],100,200);  % the last to arguments are the center
                                           % frequency and bandwidth.
>> bode(num1,den1),grid
```

See figure 12 for the corresponding Bode plot.

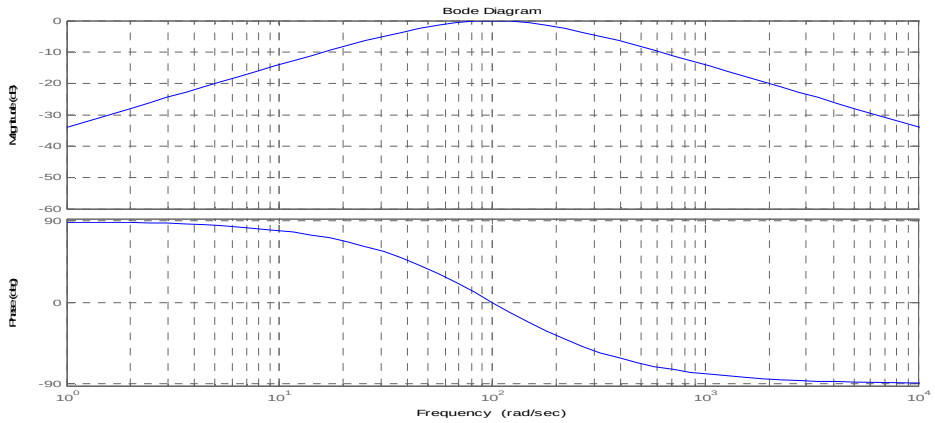


Figure 12

To create a bandstopfilter use the following transformation

```
>> [num1,den1] = lp2bs(1,[1 1],100,200);  % the last two arguments are the center
                                           % frequency and bandwidth.
>> bode(num1,den1),grid
```

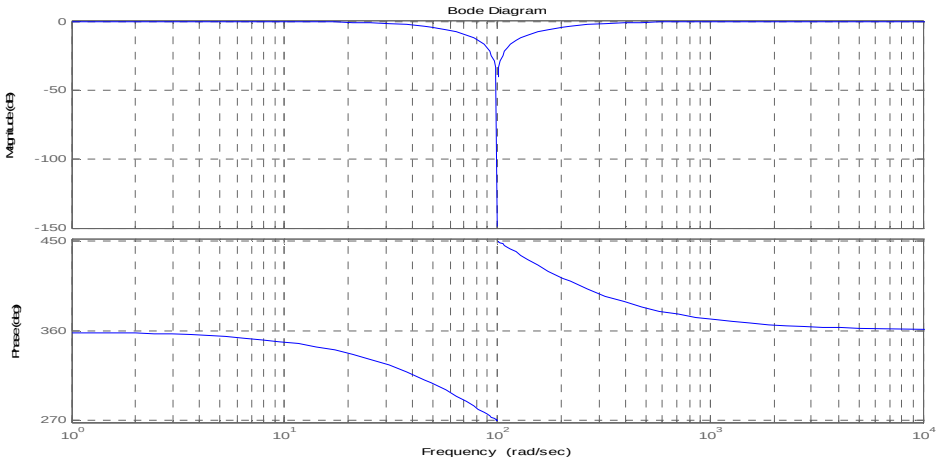
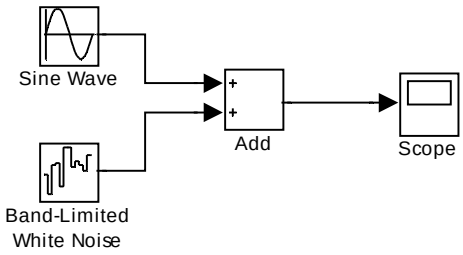


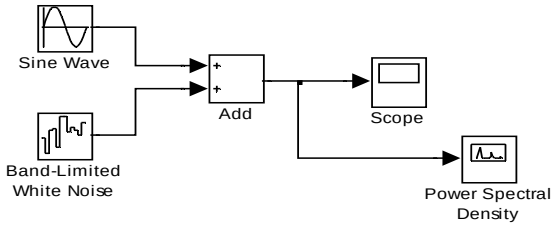
Figure 13

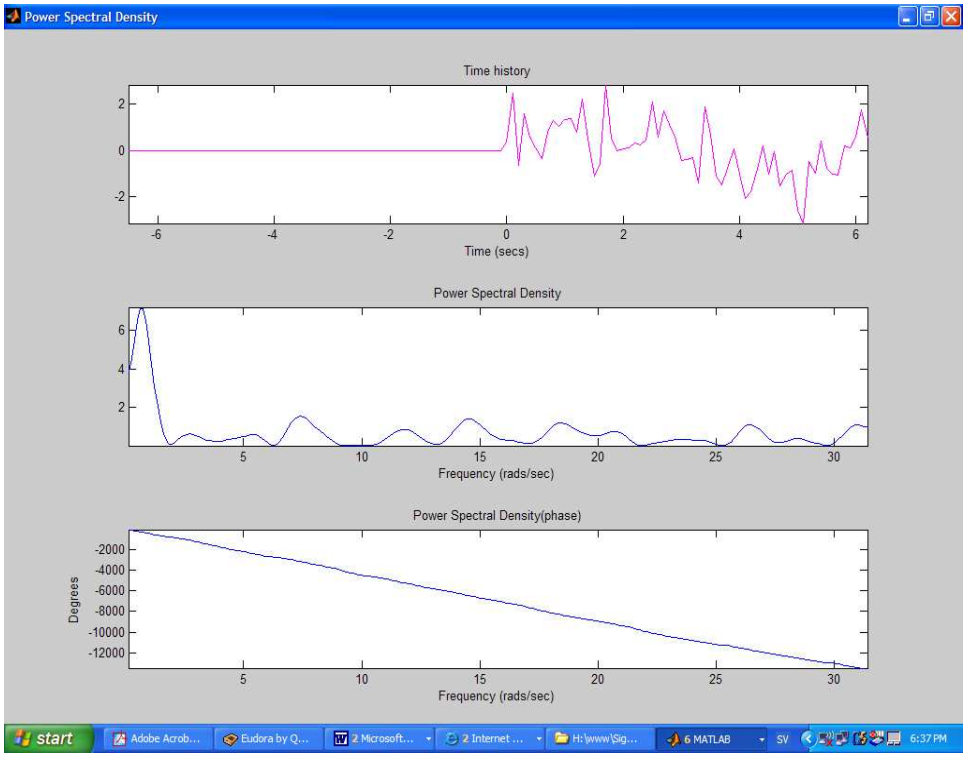
Finally we are going to examplify how we can use Simulink.

Start Simulink !
>> simulink % write in the command window and press enter.
When Simulink Library Browser opens. Open the editor window by clicking the white sheet in the left corner. Then enter Simulink-> Sources . You can here find both the Sine Wave and the White Noise.
Scope can be found in **Simulink -> Sinks** and finally Add in **Simulink-> Math Operations.**
Simulate the model by pressing the black triangle in the menu.



Change the model above by adding the block Power Spectral Density. This block can be found in the sublibrary: **Simulink Extras -> Additional Sinks.**

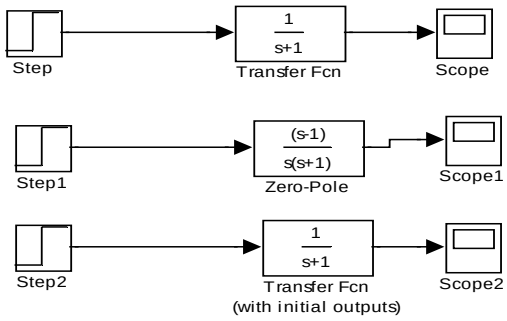




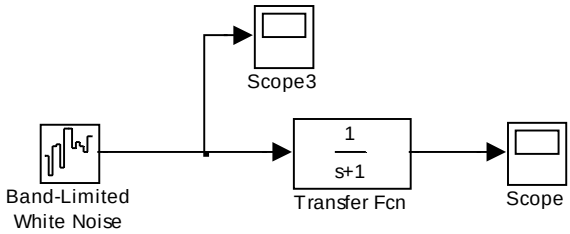
Start the simulation once again and click the block Power Spectral Density.
This window will contain three subplots. The first is the time plot and the second one is the power plotted versus time.

In this example it seems there is no particular frequency present but one.
There is only one peak that is significantly higher than the others. Around 1 rad/sec and that corresponds well to our model.

Now let's continue with transfer function within Simulink.
Open a new Edit window !
You will find the Step block in Simulink-> Sources and Scope in Simulink-> Sinks.
Transfer function and Zero-pole in the sublibrary Simulink-> Continuous.
The one with initial output can be found in Simulink Extras-> Additional Linear.



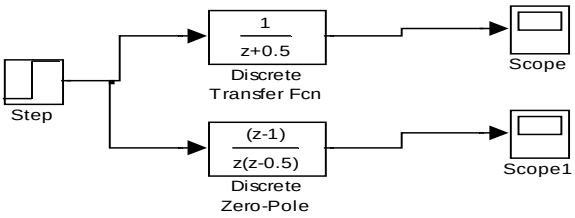
Are the systems stable ? Try to change initial value and the amplitude !
Try other Sources (actually input signals) like : Sine Wave, Ramp and the Band-Limited White Noise. See model below !



If you open the scopes. You can notice the difference in noise level due to the transfer function or is it possibly the influence of a first order LP-filter. Open the transfer function and increase the order in the denominator.Try the following poynomial: $s^3 + 11s^2 + 7s + 1$
I hope you notice the difference ! Less of the noise survives the transfer function.
We now have a LP filter of third order.

Finally we will end by also examining discrete-time signals.

Discrete-time systems: suppose we would like to simulate a system described by its transfer function. Open up a new edit window and open the sublibrary simulink-> Discrete-> Discrete Transfer Function get Discrete Zero-Pole at the same time.
Add a Scope and a step as before. It should look like below.
What is the final value for each system ?



Exercises to be solved

The exercises should be handed in as one m-file and sent to my emailadress. These exercises can be handed in individually or as a group, but notice that the group must not be larger than 2. Can give bonus points on the written exam.

1. Investigate the following system
a) $H(s)=(s+2)/(s^2+s+100)$
b) $H(s)= (s^2+2)/ (s^3+ 2s^2-s + 1)$

Your m-file should produce Bode plots of each system and present the poles and zeros in the s-domain and finally give plots presenting the step responses for each system.

2. Find the probable frequencies hiding in the file S&S_ex3.mat !
Make a m-file that produces a frequency plot where I can easily read the frequency values ! This mat-file can be found at my homepage.
In the mat-file you can find a y vector containing 100 000 values and with a sample time 1 msec.

3. Design a second-order LP-filter with only one break frequency. It should be 12 rad/sec. and with the DC-gain 0 dB.
Show a bodeplot of the system.

4. A system has the following transfer function: $H(s)=(s+2)/(s^2+10s+10)$
Decide the output for different input signals and plot the resulting output together with the corresponding input signal.
a) $x(t) = 2 \sin(3t)$
b) $x(t) = 2e^{-5t}\sin(3t)$

5. For the transfer function below decide which one

$$\frac{20}{(s+1)\,(s+8)\,(s+20)(\,s+200)}$$

of the poles dominates in the step response. Show this by a plot containing both the original 4:th order system and a first order system.

6. From the LP-filter in assignment 3. Use frequency transformations to construct a
HP-filter ω_c = 100 rad/sec of the same order,
LP-filter with ω_c = 100 rad/sec same order,
BP-filter with center frequency ω_c = 100 rad/sec and bandwidth 100 rad/sec.

Then apply a square wave signal with amplitude 1 and duty cycle 50% to these filters. Angular frequency should be 100 rad/sec. Give me one plot where the output from each filter is shown and can be recognized. You should plot at least 2-3 periods.

7. Design a highpass filter and a bandpassfilter starting from the two-pole Butterworth lowpassfilter.

a) Design the highpassfilter so that the 3-dB bandwidth runs from 10 rad/sec to ∞.
b) Design the bandpassfilter so that the 3-dB bandwidth runs from 10 to 20 rad/sec.
c) Put both filters in the same Bode plot.

Hint: use ”buttap” and ”zp2tf” for design !

8. Convert a Chebyshev lowpassfilter of degree 3 and with ripple 3dB into a bandstopfilter with center frequency 100 rad/sec and bandwidth 100 rad/sec. Show the corresponding Bode plot !
Hint: use ”cheblap” and ”zp2tf” for design !