MATLAB Tutorial

Aly El-Osery

October 27, 2004

This tutorial is meant to be a quick start to get used to MATLAB. It is by far not comprehensive. The examples included in this tutorial are meant to make you aware of some useful functions. Even those examples don't explore the extend of the used MATLAB functions but it provides you with a sample of how you might use them. I strongly recommend that you do `help <function-name>` in order to learn more about those functions. Last but not least, your feedback is greatly appreciated.

http://www.ee.nmt.edu/~elosery/matlab
Printable version

Contents

1 Data generation

- Generating a sequence of integers from 1 to 10.

```
>> n=1:10

n =

     1     2     3     4     5     6     7     8     9    10
```

- Generating numbers from 0 to 1 with increments of 0.1.

```
>> t=0:0.1:1

t =

   0  0.1000  0.2000  0.3000  0.4000  0.5000  0.6000  0.7000  0.8000  0.9000  1.0000
```

- Generating a row vector with 10 ones.

```
>> a=ones(1,10)

a =

     1     1     1     1     1     1     1     1     1     1
```

- Generating a column vector with 5 zeros.

```
>> b=zeros(5,1)

b =

     0
     0
     0
     0
     0
```

- Generating a 3x3 matrix with all elements equal to 3.

```
>> c=3*ones(3,3)

c =

     3     3     3
     3     3     3
     3     3     3
```

- Generating 4 linearly spaced points between 5 and 10.

```
>> linspace(5,10,4)

ans =

    5.0000    6.6667    8.3333   10.0000
```

2 Array operations

- An example of a row vector

```
>> a=[1 2 3]

a =

     1     2     3
```

- An example of a column vector

```
>> a=[1;2;3]

a =

   1
   2
   3
```

- If you need to multiply, square, divide, etc. any array you need to proceed the operation by a '.'.

```
>> n=0:5

n =

     0     1     2     3     4     5

>> n^2
??? Error using = > ^
Matrix must be square.

>> n.^2

ans =

      0     1     4     9    16    25
```

- Be careful when multiplying two arrays.

```
>> a=[1 2 3]

a =

     1     2     3

>> b=[4;5;6]

b =

     4
     5
     6

>> a*b

ans =

    32


>> a'.*b

ans =

     4
     10
     18
```

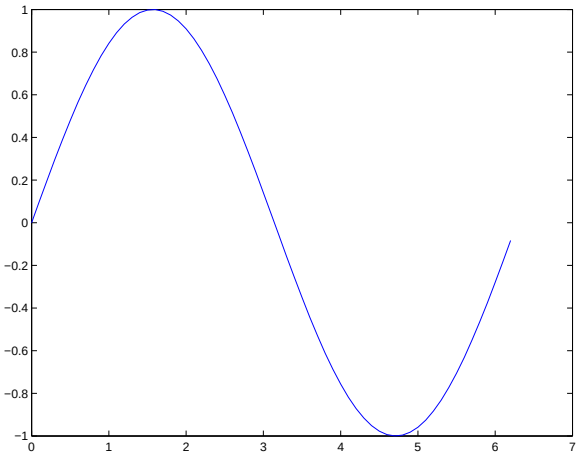

Figure 1: Sin wave example

Since `a` is a row vector and `b` is a column vector, if you use `a*b` you will have performed a dot product. On the other had if you want to multiply every element in `a` by every element in `b` to generate a new vector, then you need to transpose one of the two vectors by using `'` and then multiply the two vectors using `.*`.

3 Matrix operations

- To extract elements from a matrix.

```
>> a=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> a(2:4,3:4)

ans =

     7     8
    11    12
    15    16
```

In the above example the elements in rows 2, 3 and 4, and columns 3 and 4 are extracted.

4 Plotting

4.1 Plot function

```
>> t=0:0.1:2*pi;
>> y=sin(t);
>> plot(t,y)


>> xlabel('t')
>> ylabel('y')
>> title('Sample Plot')
```

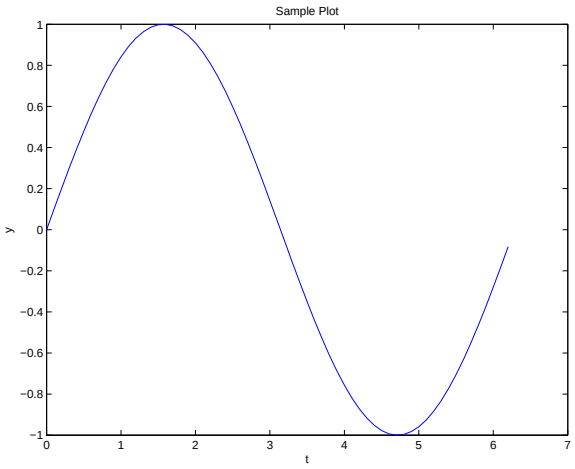

Figure 2: Adding a title to a figure

4.2 Subplot function

You can place multiple plots on the same figure using the `subplot(m,n,p)` command. `m` specifies how many rows you will have, `n` is the number of columns, and `p` is where you want to place the figure in this `m x n` matrix.

```
>> t=0:0.1:2*pi;
>> y1=sin(t);
>> y2=cos(t);
>> y3=tan(t);
>> y4=y1+y2;
>> y5=y2+y3;
>> y6=y1+y3;
>> subplot(2,3,1)
>> plot(t,y1)
>> title('y1')
>> subplot(2,3,2)
>> plot(t,y2)
>> title('y2')
>> subplot(2,3,3)
>> plot(t,y3)
>> title('y3')
>> subplot(2,3,4)
>> plot(t,y4)
>> title('y4')
>> subplot(2,3,5)
>> plot(t,y5)
>> title('y5')
>> subplot(2,3,6)
>> plot(t,y6)
>> title('y6')
```

4.3 Few other plotting commands

```
>> n=-10:10;
>> y=n.^2;
>> subplot(1,2,1)
>> stem(n,y)
>> title('stem function')
>> subplot(1,2,2)
>> stairs(n,y)
```

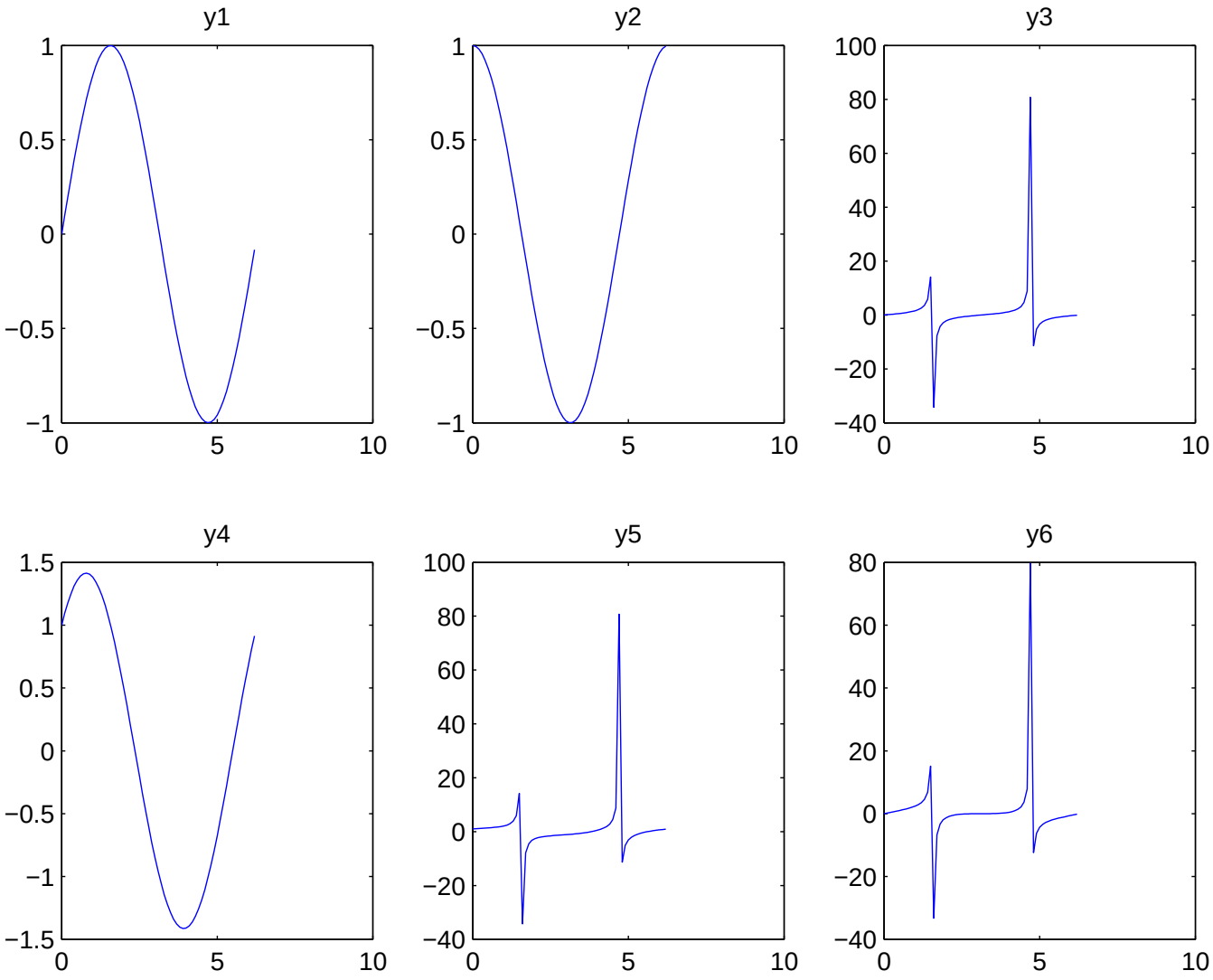

Figure 3: `subplot` example

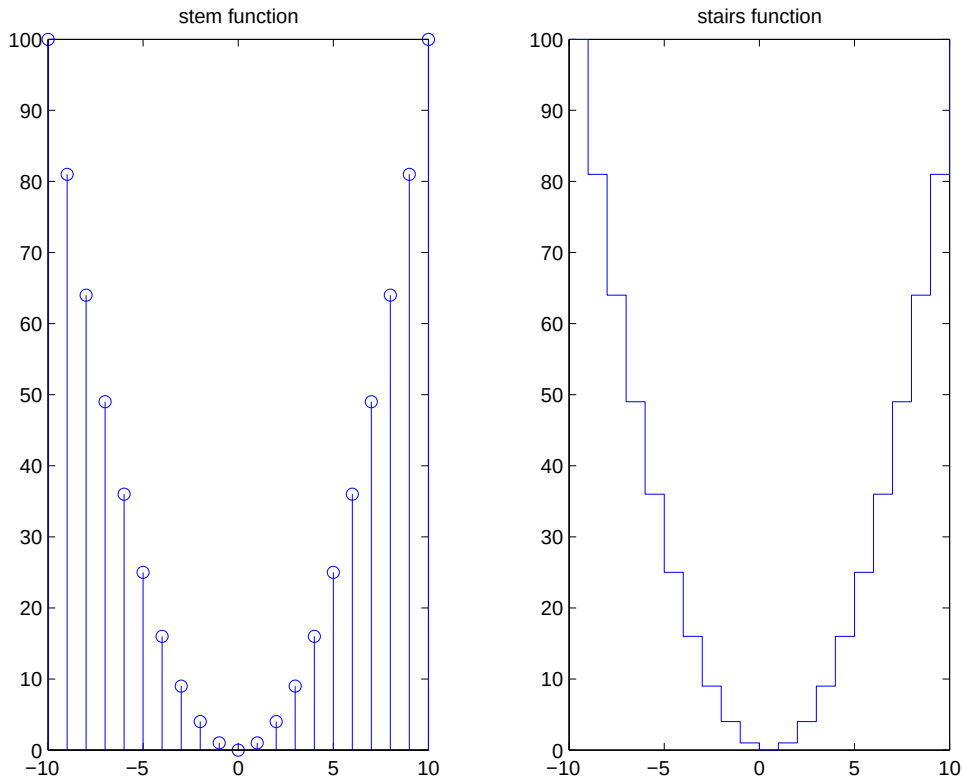

Figure 4: `stem` and `stairs` example

```
>> title('stairs function')
```

5 System creation

There are several ways you can create and enter a system. You can also convert from one system representation to another.

5.1 Transfer function

In order to enter a transfer function you need to specify a numerator and a denominator. For example:
Given the transfer function,

$$H(z) = \frac{1 + 2z^{-2}}{1 + 3z^{-1} - 0.5z^{-2}}$$

you may enter the system in the following way:

```
>> num=[1 0 2]

num =

     1     0     2

>> den=[1 3 -0.5]

den =

    1.0000    3.0000   -0.5000

>> sys=tf(num,den,-1)
```

```
Transfer function:
    z^2 + 2
---------------
z^2 + 3 z - 0.5

Sampling time: unspecified
```

Note that while entering the data for the numerator, we had to specifically put zero as the coefficient for z^{-1}. Also, in the function `tf`, if we don't have a specific sampling rate, you need to put a -1 as the last argument.

5.2 Zeros, poles and gain

You may also enter the system as poles, zeros and gain. For example:
Given a system with double zeros located at -1 and 1, and double poles located at j and $-j$

```
>>  z=[-1;-1;1;1]

z =

    -1
    -1
     1
     1

>> p=[-j;-j;j;j]

p =

        0 - 1.0000i
        0 - 1.0000i
        0 + 1.0000i
        0 + 1.0000i

>> k=0.5

k =

    0.5000

>> sys=zpk(z,p,k,-1)

Zero/pole/gain:
0.5 (z+1)^2 (z-1)^2
-------------------
   (z^2  + 1)^2

Sampling time: unspecified
```

5.3 System representation coversion

You can convert from transfer function to zero-pole representation and vica versa using the following commands

```
>> [n,d]=zp2tf(z,p,k)

n =

    0.5000         0   -1.0000         0    0.5000
```

```
d =

     1     0     2     0     1

>> [z,p,k]=tf2zp(num,den)

z =

        0 + 1.4142i
        0 - 1.4142i


p =

   -3.1583
    0.1583


k =

     1
```

6 Transfer functions

Let's assume that we have the following transfer function

$$H(z) = \frac{1 + z^{-1}}{1 - z^{-1} + 0.5^{-2}} = \frac{B(z)}{A(z)}$$

- To enter the numerator

  ```
  >> b=[1 1]

  b =

       1     1
  ```

- To enter the denominator

  ```
  >> a=[1 -1 0.5]

  a =

      1.0000   -1.0000    0.5000
  ```

  NOTE: if, for example the middle coef. was zero, you still have to enter it as zero, you should not ignore it otherwise the denominator will only be first order.

- To find the zeros and poles

  ```
  >> [z,p]=tf2zp(b,a)

  z =

      -1


  p =
  ```

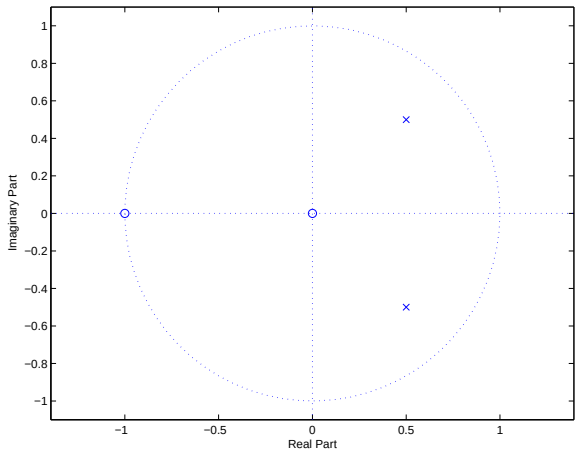

Figure 5: Pole-zero plot

```
   0.5000 + 0.5000i
   0.5000 - 0.5000i
```

where `z` and `p` are the vectors representing the zeros and poles of the system, respectively.

- To plot the zeros and poles

```
>> zplane(b,a)
```

or

```
>> zplane(z,p)
```

- To get the partial fraction expansion

```
>> [R,P,K]=residuez(b,a)

R =

   0.5000 - 1.5000i
   0.5000 + 1.5000i


P =

   0.5000 + 0.5000i
   0.5000 - 0.5000i


K =

     []
```

The above is equivalent to

$$H(z) = \frac{R(1)}{1 - P(1)z^{-1}} + \frac{R(2)}{1 - P(2)z^{-1}}$$

7 DSP functions

7.1 Impulse response

Given the following transfer function:

$$H(z) = \frac{(1 - z^{-1})(1 + z^{-1})}{1 - 1.1314z^{-1} + 0.64z^{-2}}$$

- To enter the system infomration

```
>> num=conv([1 -1],[1 1])

num =

     1     0    -1

>> den=[1 -1.1314 0.64]

den =

    1.0000   -1.1314    0.6400
```

The `conv` function will multiply the to first-order polynomials in the numerator to give you your second-order numerator polynomial.

- To plot the impulse response

```
>> h=dimpulse(num,den);
>> stem(0:length(h)-1,h)
```

If you only used `dimpulse` without anything on the left, the function will automatically plot the impulse response, but it uses the `stairs` function. To store the values and plot them using the impulse function, you need to store the output into some variable. The first argument of the `stem` function, `0:length(h)`, is used to generate the values 0, 1, 2, ..., to the length of `h`.

7.2 Frequency response

Given the following transfer function:

$$H(z) = \frac{(1 - z^{-1})(1 + z^{-1})}{1 - 1.1314z^{-1} + 0.64z^{-2}}$$

You can enter the information in the same way as in the previous subsection.

- Using the `freqz` function

```
>> freqz(num,den)
```

Notice the frequency scale. Also, the magnitude response is in dB.

- Another way to use `freqz`

```
>> [H,w]=freqz(num,den);
>> subplot(2,1,1)
>> plot(w,abs(H))
>> grid on
>> xlabel('freq. (rad/sec)')
>> ylabel('mag. response')
>> subplot(2,1,2)
>> plot(w,angle(H))
>> grid on
>> xlabel('freq. (rad/sec)')
>> ylabel('phase response')
```

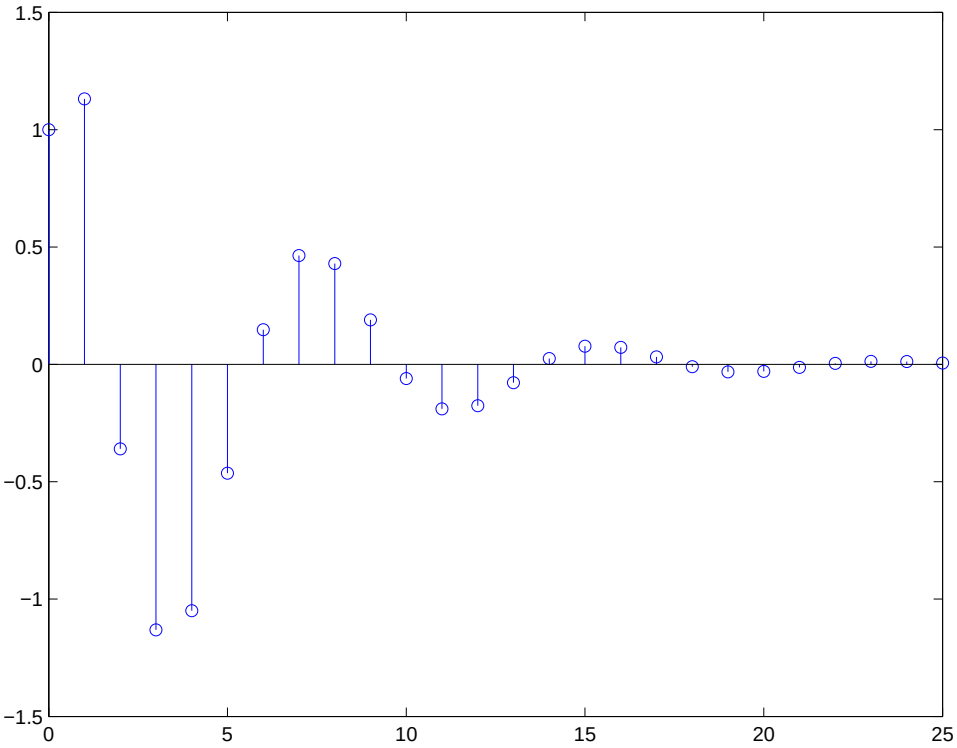

Figure 6: Impulse response

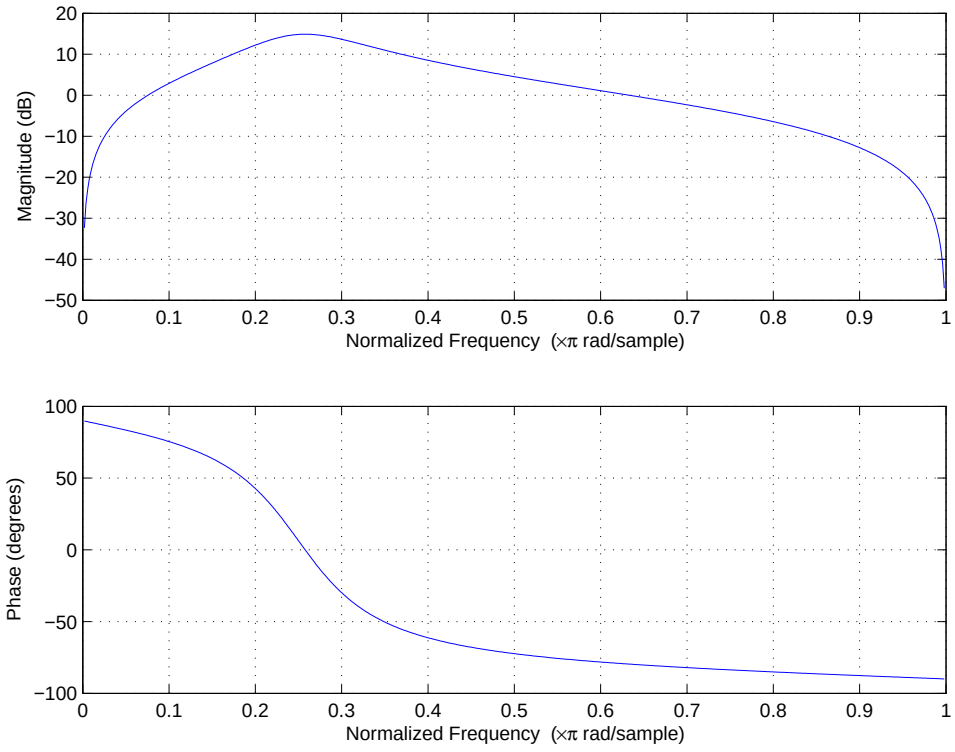

Figure 7: Frequency response

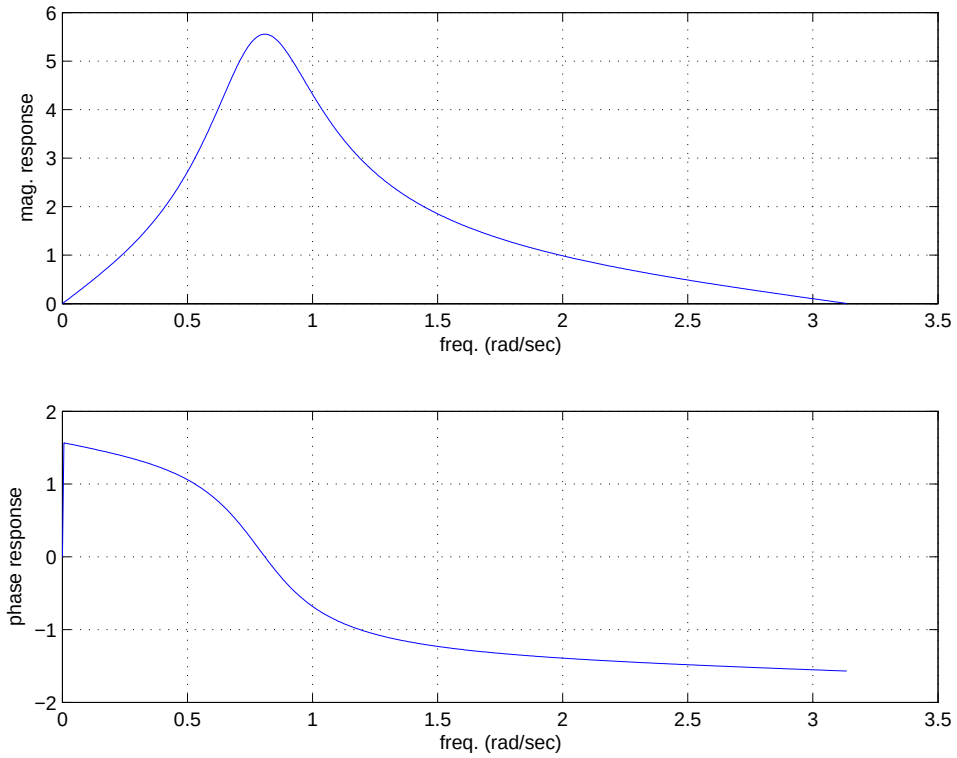

Figure 8: Frequency response (example 2)

7.3 Fast Fourier transform (FFT)

Best way to explain that is using an example. Assume that we have a sinusoidal signal that we want to determine its Fourier transform.

```
>> fs=100;
>> t=0:1/fs:1;
>> y=cos(2*pi*t);
>> plot(t,y)
>> xlabel('t')
>> ylabel('y(t)')
```

Use the following commands to compute the FFT, find its length and plot the magnitude of Y.

```
>> Y=fft(y);
>> length(Y)

ans =

   101

>> plot(abs(Y))
```

The above plot doesn't show many details since it is only using 101 points. We can specify the length of the FFT to be longer by

```
>> N=1024;
>> Y=fft(y,N);
>> plot(abs(Y))
```

Now we can see more details, but still we can't extract a lot of information from the plot. If we compute the Fourier transform of a sine or a cosine function we expect to see two impulses at the frequency of the sine or the cosine. But, that's not what we are seeing. Here is how we can modify the plot to see something more familiar.

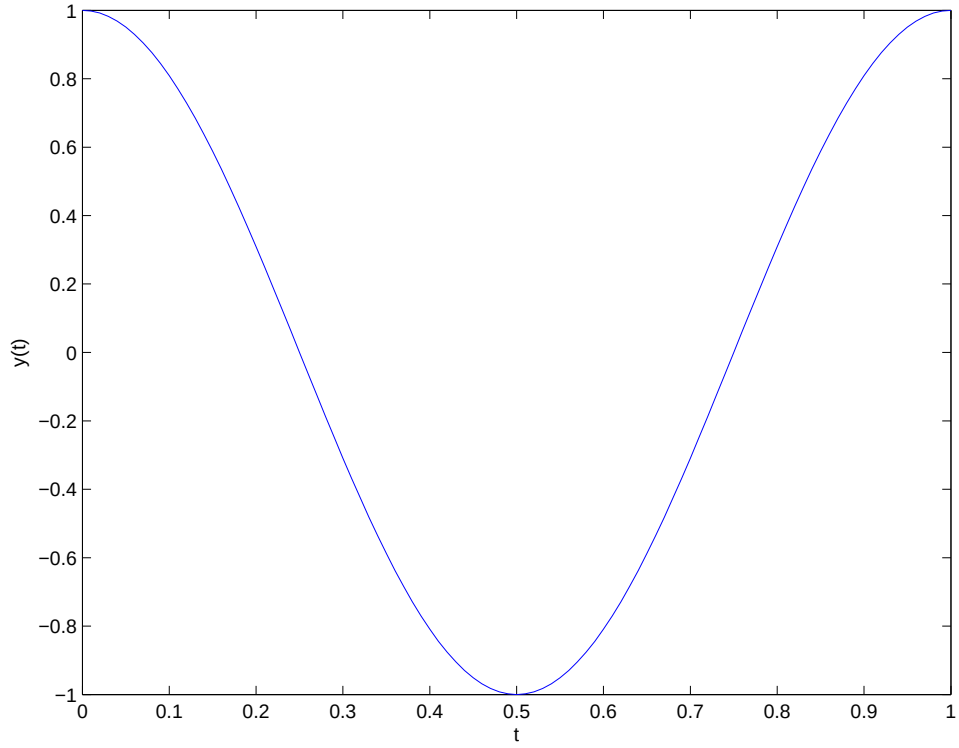

Figure 9: 1sec sinwave of 1Hz

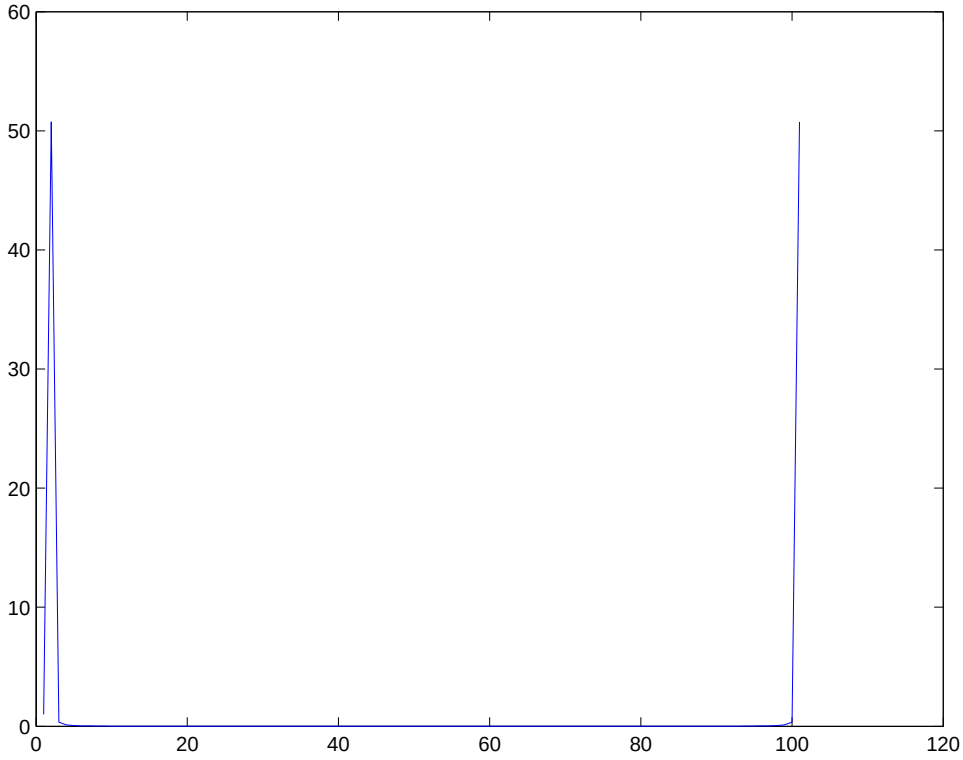

Figure 10: 101-point FFT

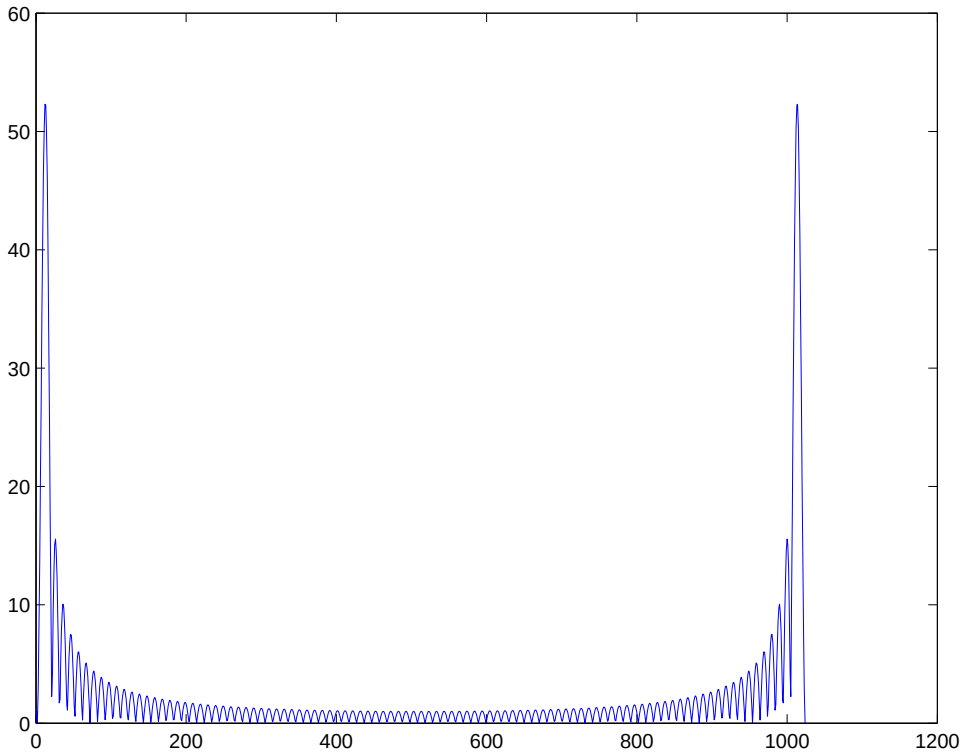

Figure 11: 1024-point FFT

```
>> plot(fftshift(abs(Y)))
```

If we zoom around the center we get something familiar.

But, it is not really the impulses that we expected. We have all those ripples and also the x-axis does not match the frequency of the cos. Let's deal with the second problem first.

We need to map the x-axis value to reflect frequency in Hz. To do that all you need to remember that the length of the FFT corresponds to the sampling rate F_s for continuous frequencies and corresponds to 2π for discrete frequency. Therefore to get the positive and negative frequencies you need to get the following.

```
>> k=-N/2:N/2-1;
>> plot(k,fftshift(abs(Y)))
```

Now we get the axis containing positive and negative values. All we have left to do is to map it to actual frequencies.

```
>> plot(k*fs/N,fftshift(abs(Y)))
```

Now are getting something meaningful. We have the two peaks at $\pm$1Hz. But, they are not really impulses like we wanted. The reason is we really wanted an infinitly long sinusoide to get the impulses, due to the fact that we can't have an infinitly long sinusoide to process, we have effectively multiplied the sinusoid by a rectangular window of length 1sec. If we multiply in the time domain by a rectangular window, we convolve in the frequency domain with a `sinc` function. That is what we are getting, two `sinc` functions centered around $\pm$1Hz.

If we increase the length of the window, we expect the `sinc` function to look more and more like an impulse. To do that lets generate a sinusoidal signal that last for 10sec instead.

```
>> fs=100;
>> t=0:1/fs:10;
>> y=cos(2*pi*t);
>> N=4096;
>> Y=fft(y,N);
>> k=-N/2:N/2-1;
>> plot(k*fs/N,fftshift(abs(Y)))
```

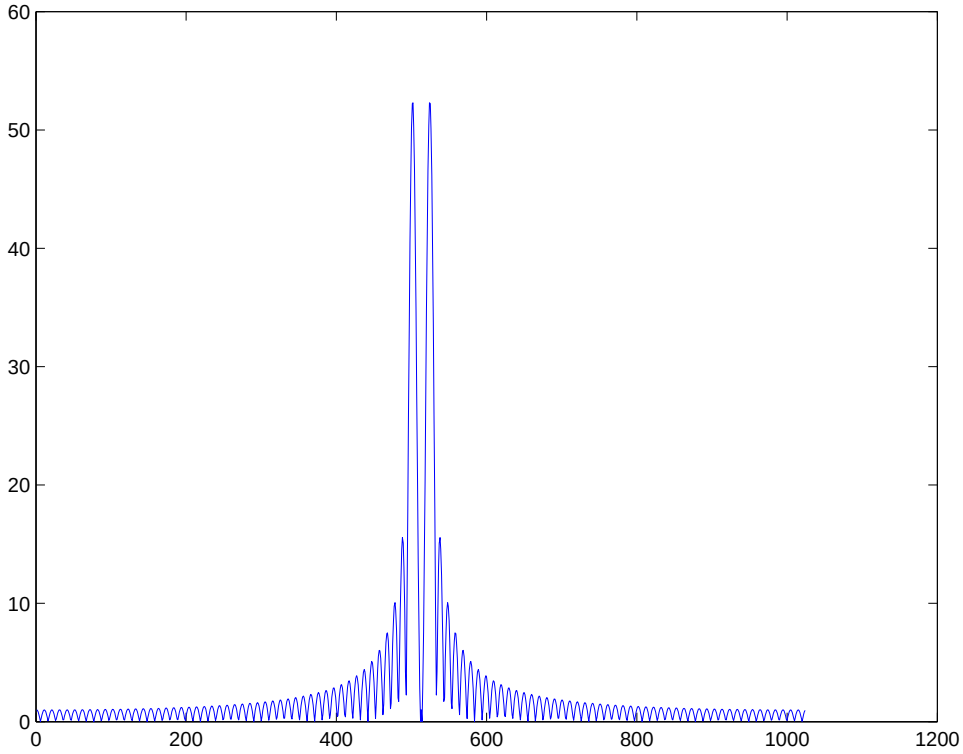

Figure 12: `fftshift`

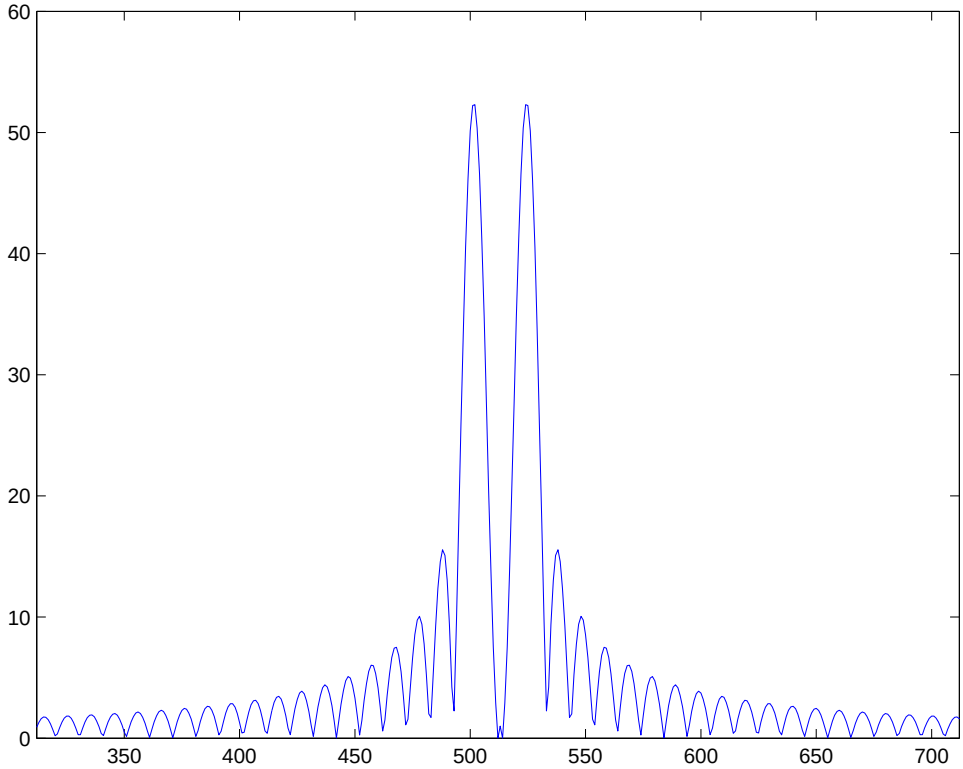

Figure 13: `fftshift` zoomed

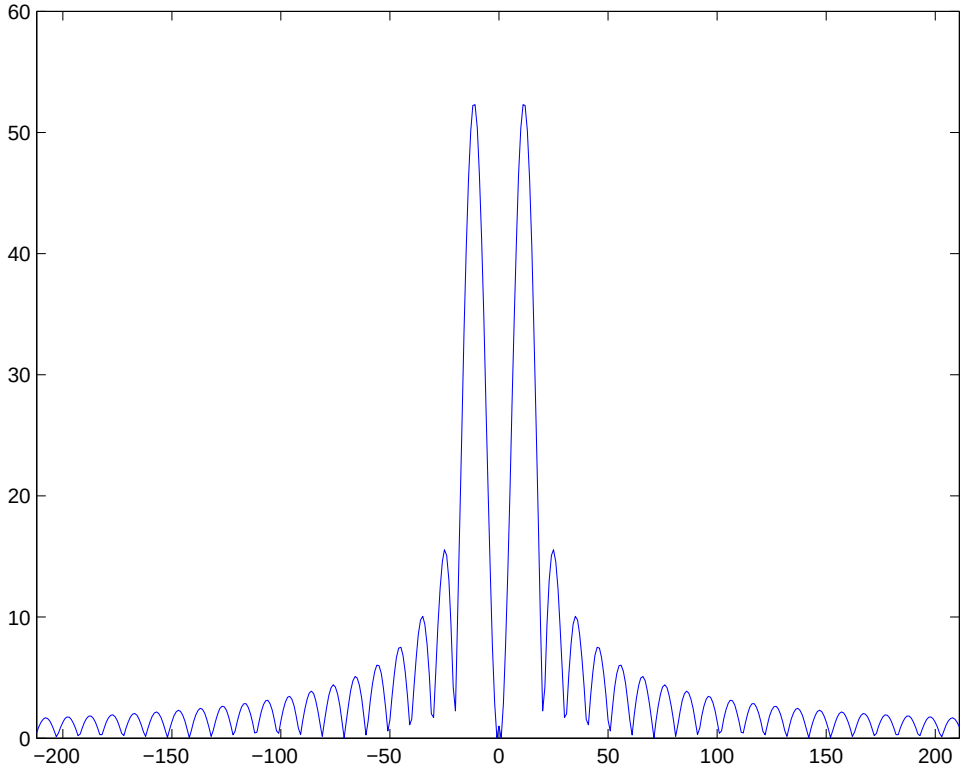

Figure 14: `fftshift` with $k = -N/2 : N/2 - 1$

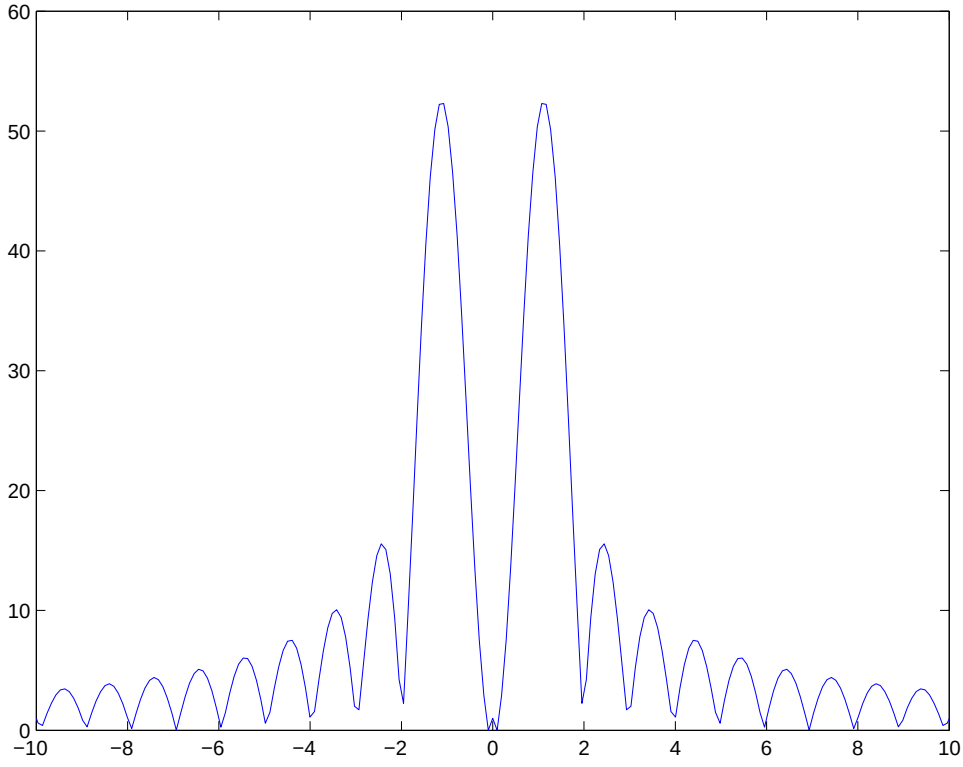

Figure 15: Frequency mapping

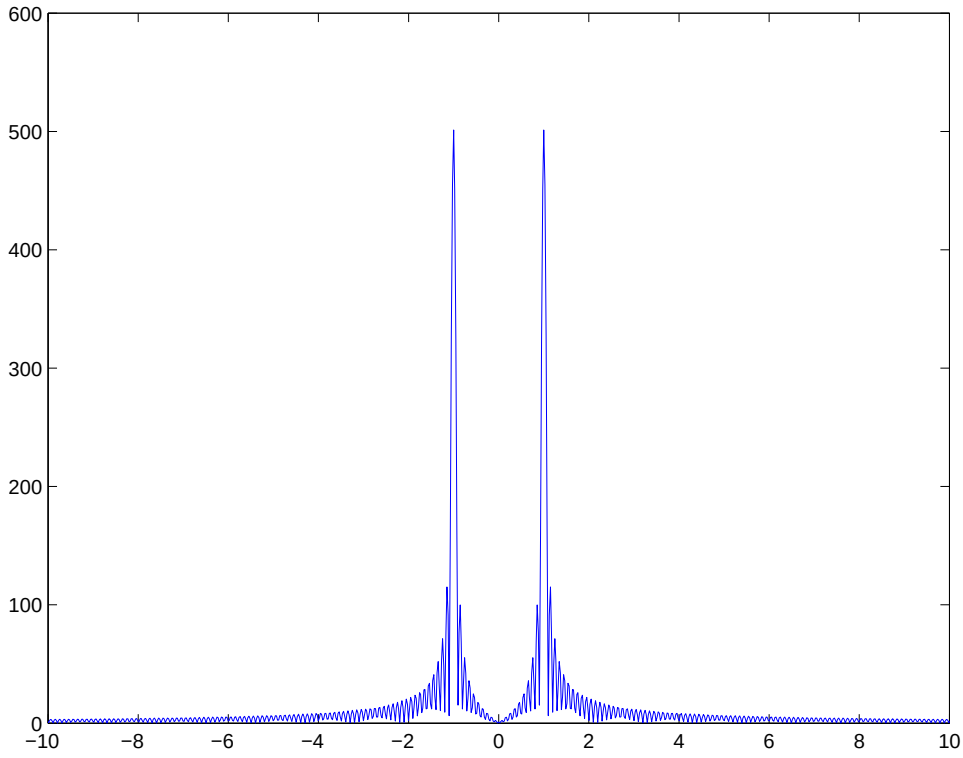

Figure 16: FFT of a 10sec 1Hz sinwave

8 Misc. commands

- `clear`: clears all the variables.
- `close all`: closes all the plots.
- `clc`: clears the display in the workspace area.
- `diary` ¡*filename*¿: this will save all subsequent command window input and most of the output to ¡*filename*¿. `diary off` suspends the command and `diary on` resumes it.

9 Helpful hints

- If you want to edit a figure at a later time make sure that you save it as a `.fig` file.
- When writing m-files you may want to add the semicolon at the end of some of the commands to prevent them from displaying the result to the screen. This will make the program finish faster. Of course you may want to see the results of some of those commands for debugging reasons, so it is your call.