Vrije Universiteit Brussel

PRACTICAL INTRODUCTION TO MATLAB

Péter Zoltán, CSURCSIA
MSc in EE & IT

Outline

I. Introduction
II. Matrix algebra in Matlab
III. Statistics
IV. Graphical tools
V. Programming in Matlab
VI. Signals and systems
VII. Example

I. Preface

This practical lecture is designed for doctoral students to get familiar with Matlab. The course is optimized for the spring school on Identification of Nonlinear Dynamic Systems.

This presentation is downloadable from www.commodos.hu/matlab

On this website you can find some source code examples.

We wish you a great success and a good luck with Matlab.

Labor-instructors of the doctoral school ELEC:

Dhammika
Koen
Maarten
Péter

I. Important operators and brackets

Character	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Power
sqrt(number)	Square root of a number
(and)	Parenthesis for mathematical groping and functions
[and]	Squared brackets for vectors and matrices

Examples

```
>> 10+3*(3*2^10)

ans =

        9226
```

```
>> sqrt(10+3*(3*2^10))

ans =

  96.052069212484952
```

I.Important constants

name	description
pi	π
i and j	$\sqrt[2]{-1}$ **I strongly advise** to use the form "1i" or "1j" and not the form of "1*j" or "1*i" because i and j are often used as variables for "while" or "for" loops. See example below.
realmin, realmax	the smallest and the largest (float) number
inf	Infinity value (IEEE representation)
NaN	Not a Number

Examples

```
>> pi+3j

ans =

   3.1416 + 3.0000i
```

```
>> pi+3i

ans =

   3.1416 + 3.0000i
```


I.Important instructions

- **help**, example:

Name of the instruction

```
>> help sum
 SUM Sum of elements.
    S = SUM(X) is the sum of the elements of the vector X. If
    X is a matrix, S is a row vector with the sum over each
    column. For N-D arrays, SUM(X) operates along the first
    non-singleton dimension.
```

- **why**, examples:

```
>> why
He wanted it that way.
>> why
The programmer suggested it.
>> why
Barney suggested it.
>> why
To please a very terrified and smart and tall engineer.
>> why
The tall system manager obeyed some engineer.
```

I.Variables: creation and clear

```
>> a=3
a =

     3
>> b=2+3i
b =

   2.0000 + 3.0000i
>> c=a+b
c =

   5.0000 + 3.0000i
>> d=c*sin(pi)
d =

  6.1232e-016 +3.6739e-016i
```

To create variables a,b,c,d

The result can be expressed in exponential form

```
>> who
Your variables are:

a  b  c  d
```

To check the available variables

```
>> clear all
```

To clear all variables

To clear variable a

```
>> clear a
```


I.Displaying numbers

```
>> format short
>> pi
ans =

    3.1416
```

To change to short displaying format

```
>> format long
ans =

   3.141592653589793
```

To change to long displaying format

```
>> ceil(pi)
ans =

     4
```

To get the higher integer part of a number

```
>> floor(pi)
ans =

     3
```

To get the lower integer part of a number

```
>> round(pi)
ans =

     3
```

To round a number to the nearest integer

II. Vectors and matrices: creation

row1 row2 row3

```
>> [2 4; -1 1+i; 0 2-i]
```

The elements of a vector/matrix must be in square brackets

the result of the operation

```
ans =

    2.0000         4.0000
   -1.0000         1.0000 + 1.0000i
         0         2.0000 - 1.0000i
```

```
>> A=[1 3; 2 4];
```

Closing an operation with semicolon
▶no "ans =....." result

The name of variable to store the matrix

II. Vectors and matrices: operations

```
>> A=[1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B=[1 1 1; 2 2 2; 0 0 0]

B =

     1     1     1
     2     2     2
     0     0     0
```

Example matrices

```
>> A+B

ans =

     2     3     4
     6     7     8
     7     8     9
```

To add A and B

```
>> A*B

ans =

     5     5     5
    14    14    14
    23    23    23
```

To multiply A with B

```
>> A'

ans =

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

Complex conjugate of A.
For simple transpose use **A.'**

II. Vectors and matrices: size and length

```
>> A=[1 2 3 4 5];B=[1;2;3];
>> C=B*A

C =

     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15

>> size(C)
ans =

     3     5

>> size(C,2)
ans =

     5

>> length(C)
ans =

     5
```

To create row vector A and column vector B

To create matrix C as product of B and A

The size (dimensions) of matrix C

The number of columns of matrix C

The (maximum) dimension of matrix C

II. Vectors and matrices: referring to elements 1.

Example Matrix

$$\mathbf{A} = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

Element of row *i* and column *j* ► A(i,j)	A(3,1) ►7
Element at index *k* ► A(k)	A(1) ►1 | A(2) ►4 | A(5) ►5
Elements of row *i* ► A(i,:)	A(2,:) ►[4 5 6]
Elements of column *j* ► A(:,j)	A(:,3) ►$[3\ 6\ 9]^T$
2nd and 3rd elements of row 1	A(1,[2 3])►[2 3]
Each second element	A(1:2:end)►[1 7 5 3 9]

From the first element — *Step size* — *Till end*

II.Vectors and matrices: referring to elements 2.

Example Matrix

A =

1	2	3
4	5	6
7	8	9

The middle of the matrix

The 'middle': 9/2=4.5 is not an integer number

```
>> A(end/2:end)
Warning: Integer operands are required for colon operator when used as index

ans =

     5     8     3     6     9
```

```
>> norm(A,2)
ans =
  16.8481
>> norm(A)
ans =
  16.8481
```

It returns the second norm of A

It is same as the norm(A,2)

II.Increasing/decreasing sequences

```
>> x=linspace(1,5,3)

x =
     1     3     5
```

To create 3 elements of a sequence from 1 to 5

```
>> x=1:0.5:5

x =

    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000
```

To create an increasing sequence from 1 to 5 with step size of 0.5

```
>> x=5:-2:-5

x =

     5     3     1    -1    -3    -5
```

To create a decreasing sequence from 5 to -5 with step size of -2

II. Matrices: operations 1.

```
A =

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

Matrix A

```
>> rank(A)
ans =

     2
```

To compute the rank of A

```
>> cond(A)
ans =

   3.8131e+016
```

To compute the condition number of A

```
>> inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.541976e-018
ans =

  1.0e+016 *

   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504
```

The inverse is "wrong", because **A** is ill-conditioned

```
>> det(A)

ans =

   6.6613e-016
```

To compute the determinant of A

```
>> pinv(A)
ans =

   -0.6389   -0.1667    0.3056
   -0.0556    0.0000    0.0556
    0.5278    0.1667   -0.1944
```

To compute the pseudo inverse of A.
It is the "best" solution for inverse of A.

II. Matrices: operations 2.

```
A =

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

Definition of Matrix A

```
>> A^2 % is equal to A*A
    30    36    42
    66    81    96
   102   126   150
```

Matrix A to the power two

```
>> A.^2 % element by element powering
     1     4     9
    16    25    36
    49    64    81
```

The dot operator is the **element by element operation.** Here it is the powering

```
>> sum(A)

    12    15    18
```

To compute sums of columns

```
>> prod(A)

    28    80   162
```

To compute products of columns

If the input parameter is a vector, then it returns a scalar value

II.Matrices: special types

```
>> zeros(2,2)

ans =

     0     0
     0     0
```

To create a null matrix

```
>> ones(2,2)

ans =

     1     1
     1     1
```

To create a matrix of ones

```
>> eye(2,2)

ans =

     1     0
     0     1
```

To create an identity matrix

II.Matrices: remove a row/column

```
>> X=[2 -1 0 -3;5 6 7 8]

X =

     2    -1     0    -3
     5     6     7     8
```

Definition of Matrix X

```
>> X(:,3)=[]

X =

     2    -1    -3
     5     6     8
```

To remove the third column

II.Matrices: add/change a row/column

```
>> X=[2 -1  0 -3; 5 6 7 8]

X =

     2    -1     0    -3
     5     6     7     8
```

Definition of Matrix X

```
>> X=[X; 0 0 0 0]

X =

     2    -1     0    -3
     5     6     7     8
     0     0     0     0
```

To add a new row after the last one

```
>> X(2,:)=[9 9 9 9]

X =

     2    -1     0    -3
     9     9     9     9
     0     0     0     0
```

To change the second row

II.Matrix: a complex example

```
>> A=1:2:3;
>> B=zeros(1,3);
>> C=rand(2);
>> D=randn(2,3);
>> E=[A B;C D]

E =

   1.0000    3.0000         0         0         0
   0.9501    0.6068   -0.4326    0.1253   -1.1465
   0.2311    0.4860   -1.6656    0.2877    1.1909
```

III. Statistical functions

Function	Description
max(V) or max(M)	▪ max/min/mean/median(V) means that we compute this quantities for the vector V and the result is a scalar value ▪ max/min/mean/median(M) means that we compute this quantities for the matrix M and the result is a row vector which consists of the max/min/mean/median of the columns of M
min(V) or min(M)	
mean(V) or mean(M)	
median(V) or median(M)	
std(V) or std(V,1)	▪ std(V)/var(V) means the unbiased standard deviation/variance of the vector V. ▪ std(V,1)/var(V,1) is the biased standard deviation/variance of the vector V.
var(V) or var(V,1)	
cov(V) or cov(M1,M2)	▪ cov(V) is equal to var(V) ▪ cov(M1,M2) is the covariance matrix of matrices M1 and M2
rand(N,M)	▪ rand(N,M): returns a matrix with N rows and M columns with uniformly distributed random values between 0 and 1 ▪ randn(N,M): returns a matrix with N rows and M columns with standard normally distributed random values
randn(N,M)	

III. Statistical functions: example 1

```
>> n=1+2.*randn(10000,1);
>> mean(n)

ans =

    1.0097

>> std(n)

ans =

    1.9814

>> var(n)

ans =

    3.9258

>> cov(n)

ans =

    3.9258
```

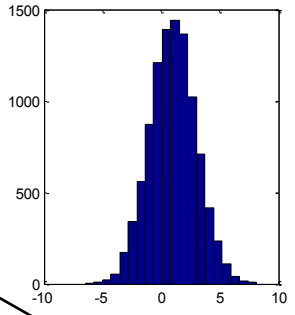
To create a normally distributed random sequence with mean of 1 and standard deviation of 2.
Remark: 2.*randn(10000,1) means that we get a column vector of 10000 random values and we multiply them 'element by element' with 2

To compute the mean value

To compute the unbiased standard deviation

To compute the unbiased variance with the help of the covariance function

The empirical histogram of the vector **n** plotted in 20 intervals



>> hist(n,20)

III. Statistical functions: example 2

```
>> n=2*pi+(4*pi-2*pi).*rand(100000,1);
>> var(n)

ans =

    3.2815

>> std(n)

ans =

    1.8115

>> mean(n)

ans =

    9.4240

>> median(n)

ans =

    9.4305
```

To create a uniformly distributed random sequence between [2π ,4π]

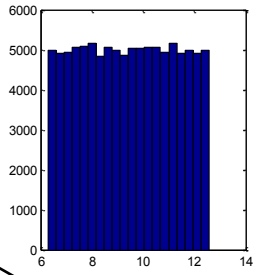
To compute the unbiased variance with the help of the covariance function

To compute the unbiased standard deviation

To compute the mean value

To compute the median value

The empirical histogram of the vector **n** plotted in 20 intervals



>> hist(n,20)

IV. Graphical functions 1.

Function	Description
plot(Y) plot(X,Y) plot(X,Y,s)	▪ plot(Y) plots the values of vector Y ▪ plot(X,Y) plots vector Y versus vector X ▪ plot(X,Y) plots vector Y versus vector X with visualization parameters, see later on
stem(Y) stem(X,Y) stem(X,Y,s)	▪ They plot the data sequence Y as stems from the x axis terminated (with circles) for the data value.
figure figure(number_identifier)	▪ figure: create a new window for plotting ▪ figure(number_identifier): a new window with this identifier in the title part of the window
hist(V) hist(V,n)	▪ To make a histogram of vector V ▪ To make a histogram of vector V in n bins
hold on	▪ To hold the actual canvas for other new plots ***default***: if you plot a new function the old one will be erased

IV.Graphical functions 2.

Function	Description
subplot(n,m,p)	▪ It breaks the actual canvas into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.
title(string)	▪ It sets a title of the current plot
xlabel(string)	▪ To set the label text of axis X and Y
ylabel(string)	
axis([XMIN XMAX YMIN YMAX])	▪ It sets scaling for the x- and y-axes on the current plot.
legend(string1,string2,...)	▪ It puts a legend on the current plot.
grid on	▪ To set the grid on the current plot
semilogx(X,Y,s)	▪ They are the same as plot() except a logarithmic (base 10) scale is used for the X/Y/X and Y-axis.
semilogy(X,Y,s)	
loglog(X,Y,s)	
close all	▪ To close all the figure-windows
mesh(X,Y,Z)	▪ To plot 3D figures
surf(X,Y,Z)	

IV.Graphical functions -Plot

Various line types, plot symbols and colors may be obtained with PLOT(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:

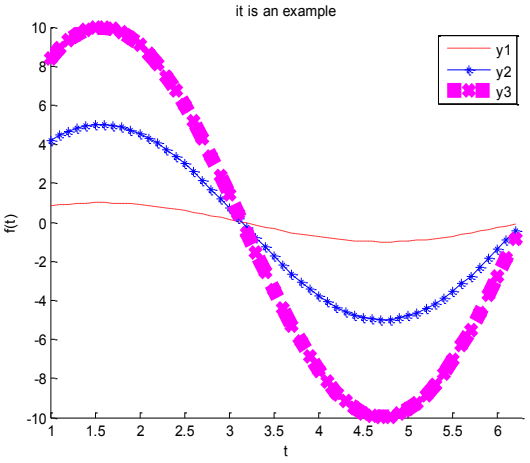
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus at each data point; PLOT(X,Y,'bd') plots blue diamond at each data point but does not draw any line.

IV.Graphical functions –Example 1.

DO IT YOURSELF
LINE BY LINE AND SEE THE RESULTS

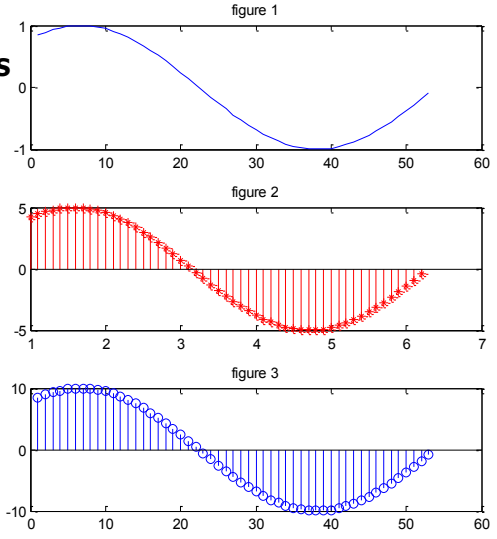
```
figure(10);
x=[1:0.1:2*pi];
y1=sin(x);
y2=5*sin(x);
y3=10*sin(x);
hold on;
plot(x,y1,'r--');
plot(x,y2,'b*-');
plot(x,y3,'mx:','LineWidth',10);
legend('y1','y2','y3');
title('it is an example');
xlabel('t');
ylabel('f(t)');
axis([1 2*pi -10 10]);
```



IV. Graphical functions –Example 2.

DO IT YOURSELF
LINE BY LINE AND SEE THE RESULTS

```
subplot(3,1,1);
plot(y1);
title('figure 1');
subplot(3,1,2);
stem(x,y2,'r*');
title('figure 2');
subplot(3,1,3);
stem(y3);
title('figure 3');
```



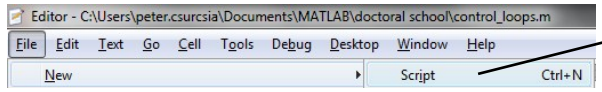
V.Programming in Matlab

- The programming language of Matlab is similar to C
- You can easily define scripts and functions
- You can use ***if/for/while/switch*** which should be closed by ***end***

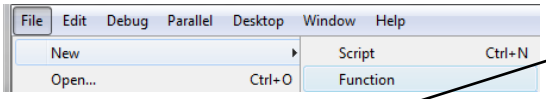
- To get an input data from the keyboard use the instruction ***input***
- To display an information use ***display*** and ***sprintf***
- To make an online debugging in a function/script use the instruction ***keyboard***

- ***See the examples later on***

V.Create a function/script



It behaves like a normal script written in the console



The file name of the function must be the same as the name of the function!

```
function [ output_args ] = NameOfFunction( input_args )
% NameOfFunction Summary of this function goes here
% Detailed explanation goes here
 ...
end
```

EXAMPLE:

```
function [ c ] = Add(a,b)
% ADD To add two numbers together
% Add(a,b) will return with c=a+b
c=a+b;
end
```

```
>> Add(2,3)

ans =

     5
```

```
>> help Add
  ADD To add two numbers together
  Add(a,b) will return with c=a+b
```

V.Control structures

if (expression1)
    instructions…
elseif (expression2)
    instructions…
else
    instructions…
end

Example

```
y=rand(1,1)
if(y<0)
   display('it is negative');
else
   display('it is positive');
end;
```

```
>> y=rand(1,1)

y =

    0.3852

>> if(y<0)
display('it is negative');
else
display('it is positive');
end;
it is positive
```

The result of the script

V.Loops

switch(*variable*)
case value1
    instructions…
case value2
    instructions…
otherwise
    instructions…
end

while(*expression to be true*)
    instructions…
end

for *variable=starter_value:incrementer:end_value*
    instructions…
end

Modifiers
continue: it jumps to the beginning of while/for
break: it leaves immediately the loop

V.Conditional operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Logical Operation	Equivalent Function
A & B	and(A, B)
A | B	or(A, B)
~A	not(A)

Use them for vectors

Use them in expressions of for/while

Operator	Description
&&	Returns logical 1 (true) if both inputs evaluate to true, and logical 0 (false) if they do not.
||	Returns logical 1 (true) if either input, or both, evaluate to true, and logical 0 (false) if they do not.

V.Source code example – the code

```
display('Please choose one of the following possibilities');
display('Type 1 if you want to see a while loop with integer numbers');
display('Type 2 if you want to see a for loop with float numbers');
display('Type something else to exit');
menu=input('your preference:');
switch(menu)
   case 1
      i=0;
      while(i<10)
         display(sprintf('Actual value of i is now %i',i));
         i=i+1;
      end;
   case 2
      for i=0:0.1:1
         display(sprintf('Actual value of i is now %f',i));
      end;
   otherwise
         display('Bye-bye my friend');
end;
```

To input a number

It sets to 'int'

It sets to 'float'

V. Source code example - result

Please choose one of the following possibilities
Type 1 if you want to see a while loop with integer numbers
Type 2 if you want to see a for loop with float numbers
Type something else to exit

```
your preference:1
Actual value of i is now 0
Actual value of i is now 1
Actual value of i is now 2
Actual value of i is now 3
Actual value of i is now 4
Actual value of i is now 5
Actual value of i is now 6
Actual value of i is now 7
Actual value of i is now 8
Actual value of i is now 9
```

```
your preference:2
Actual value of i is now 0.000000
Actual value of i is now 0.100000
Actual value of i is now 0.200000
Actual value of i is now 0.300000
Actual value of i is now 0.400000
Actual value of i is now 0.500000
Actual value of i is now 0.600000
Actual value of i is now 0.700000
Actual value of i is now 0.800000
Actual value of i is now 0.900000
Actual value of i is now 1.000000
```

```
your preference:3
Bye-bye my friend
```


V.Load and save the workspace

% Save the workspace (all variables) from the workspace to test.mat:

save test.mat

% loads the variables from test.mat

load(test.mat)

%saves only the specified variables

save(test.mat, variable_name)

%loads only the specified variables

var_new = load(test.mat, variable_name)

V. Operator Precedence

1. Parentheses ()
2. Transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
3. Unary plus (+), unary minus (-), logical negation (~)
4. Multiplication (.*), right division (./), left division (.\), matrix multiplication (*), matrix right division (/), matrix left division (\)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8. Element-wise AND (&)
9. Element-wise OR (|)
10. Short-circuit AND (&&)
11. Short-circuit OR (||)

VI. Signals and Systems in Matlab 1.

Function	Description
SYS = tf(num,den)	▪ tf(num,den) creates a continuous-time transfer function SYS with numerator num and denominator den ▪ tf(num,den,Ts) creates a discrete-time transfer function SYS with numerator num and denominator den with sampling time Ts.
SYS = tf(num,den,Ts)	
impulse(SYS)	▪ it returns/plots the impulse response of the dynamic system SYS
step(SYS)	▪ it returns/plots the step response of the dynamic system SYS
lsim(SYS,U,T)	▪ it returns/plots the time response of the dynamic system SYS to the input signal described by U and T
bode(SYS)	▪ it draws the Bode plot of the dynamic system SYS.
db(x)	▪ 20*log10(abs(x))

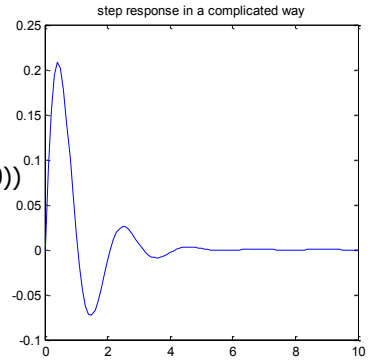
VI.Signals and Systems in Matlab 2.

Function	Description
filter(B,A,u)	▪ it returns filtered data to the input u. The filter described by vectors A and B
[X]=fft(x)	▪ they compute the discrete Fourier/ inverse Fourier transform (DFT) of vector X ▪ Fft: the first element of returned array is the DC value!
[x]=ifft(X)	
[Gest]=tfestimate(u,y)	▪ it estimates the transfer function, u and y are time-discrete input and output data (vectors)
[H,W] = freqz(B,A,n)	▪ it returns the n-point complex frequency response vector H and the N-point frequency vector W in radians/sample in freqz case and radians/sec in freqs case of the filter. See also filter
[H,W] = freqs(B,A)	
[Sxy]=cpsd(x,y)	▪ It returns the cross power spectral density function of time discrete data x and y
repmat(A,m,n)	▪ creates a large matrix consisting of an m-by-n tiling of copies of A

VI.Example: step response of a system

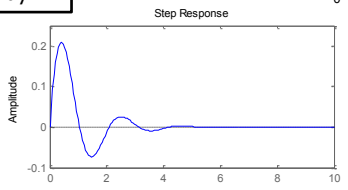
```
>> h=tf([1 0],[1 2 10])
 Transfer function:
      s
--------------
s^2 + 2 s + 10

plot(0:0.1:10,lsim(h,ones(1,length(0:0.1:10)),0:0.1:10))
title('step response in a complicated way');
```

Complicated way



Easy way

```
>>step(h,10)
```



VI.Example: Estimating a transfer function

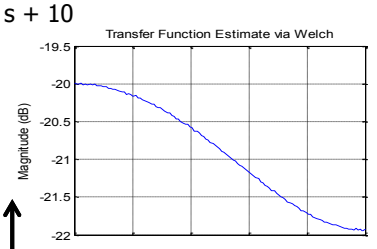
Transfer function:

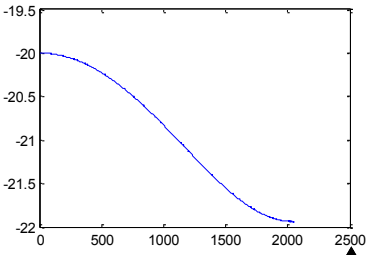
1

---

s + 10

The system to demonstrate in 's' domain






```
h=tf([1],[1 10]) %to define h
u=randn(10440,1);%to generate the input signal
u_sim=repmat(u, 2); %to make two periods, it is equivalent to u=[u;u]
y_sim=lsim(h,u_sim,0:length(u_sim)-1); %to compute on the whole domain
y_meas=y_sim(end/2+1:end); %to skip the first period which may contain transients
tfestimate(u,y_meas); %to plot the frequency response function FRF estimate
%plot(db(cpsd(u,y_meas)./cpsd(u,u))) // THE NORMAL WAY TO COMPUTE THE FRF
```


VI.Example: Estimating a transfer function cont'd.

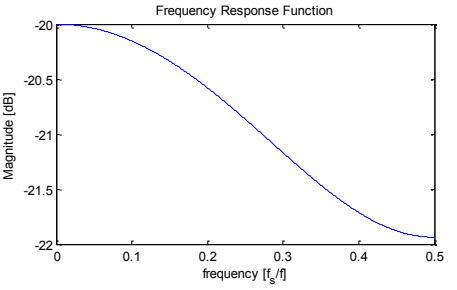
```
H=db(fft(y_meas)./fft(u));
%H=20*log10(abs(fft(y_meas)./fft(u)));

plot(linspace(0,0.5,length(u)/2),H(1:end/2));

xlabel('frequency [f_s/f]'); ylabel('Magnitude [dB]');

title('Frequency Response Function');
```

To compute the FRF in the 'shorter' way

To compute the FRF in the 'normal' way

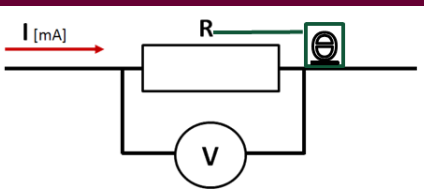
This solution can be used with restrictions only!

Think it over e.g. what happens if the U (fft(u)) contains zero value



VI. Example: Least Squares solution

This example is solved by LS in matrix form.
The parameter to estimate is θ **(R).**
The observation matrix is X (I) and the system output is Y (V).



X: I	Y: V
10 mA	10,5 V
20 mA	20,0 V
30 mA	30,5 V
40 mA	39,0 V

```
>> X=[10/1000 20/1000 30/1000 40/1000]';
>> Y=[10.5 20 30.5 39]';
```

$$\hat{\theta}_{LS} = \hat{R}_{LS} = [X^T X]^{-1} X^T Y$$

Inverse can be problematic…

```
>> inv(X'*X)*X'*Y
ans =

  993.3333
```

\ also solves the equation but with help of QR decomposition and it is better conditioned

```
>> X\Y
ans =

  993.3333
```

The analytical way

$$\hat{R}_{LS} = \frac{\sum XY}{\sum X^2} = \frac{\sum VI}{\sum I^2}$$

```
>> (X'*Y)/(X'*X)
ans =
  993.3333
```


„It is not because things are difficult that we do not dare, it is because we do not dare that they are difficult.”
Seneca

/commodos.hu/matlab

csurcsia@mit.bme.hu