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

All right reserved. This material can be used only for the department ELEC and for the BME MIT (Budapest). Source is used for this material: www.mathworks.com

I. Important operators and brackets

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

Examples

```
>> 10+3*(3+2*10)
ans =

9226
```

```
>> sqrt(10+3*(3+2*10))
ans =

96.052069212454952
```
I. Important constants

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi</td>
<td>( \pi )</td>
</tr>
<tr>
<td>i and j</td>
<td>( \sqrt{-1} ) I strongly advise to use the form “1i” or “1j” and not the form of “1<em>i” or “1</em>j” because i and j are often used as variables for “while” or “for” loops. See example below.</td>
</tr>
<tr>
<td>realmin, realmax</td>
<td>the smallest and the largest (float) number</td>
</tr>
<tr>
<td>inf</td>
<td>Infinity value (IEEE representation)</td>
</tr>
<tr>
<td>NaN</td>
<td>Not a Number</td>
</tr>
</tbody>
</table>

Examples

```
>> pi+3j
ans =
3.1416 + 3.0000i
```

```
>> pi+3i
ans =
3.1416 + 3.0000i
```

I. Important instructions

- **help**, example:

  ```
  >> 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
  Re 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

To create variables a, b, c, d:

```
>> 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 +3.0678e-016i
```

The result can be expressed in exponential form.

To check the available variables:

```
>> WHO
Your variables are:
a b c d
```

To clear all variables:

```
>> clear all
```

To clear variable a:

```
>> clear a
```

I. Displaying numbers

To change to short displaying format:

```
>> format short
>> pi
ans =
3.1416
```

To change to long displaying format:

```
>> format long
ans =
3.141592653589793
```

To get the higher integer part of a number:

```
>> ceil(pi)
ans =
4
```

To get the lower integer part of a number:

```
>> floor(pi)
ans =
3
```

To round a number to the nearest integer:

```
>> round(pi)
ans =
3
```
II. Vectors and matrices: creation

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

```matlab
>> [2 4; -1 1+i; 0 2-i]
ans =
    2.0000    4.0000
   -1.0000   1.0000 + 1.0000i
     0   2.0000 - 1.0000i
```

Closing an operation with semicolon:
► no "ans = ......" result.

The name of variable to store the matrix.

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

II. Vectors and matrices: operations

- To add A and B
- To multiply A with B

Example matrices

```
>> 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
```

```
>> A+B
ans =
    2     3     4
    6     7     8
    7     8     9

>> A*B
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

To create row vector A and column vector B:

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

To create matrix C as product of B and A:

```matlab
C =
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
```

The size (dimensions) of matrix C:

```matlab
>> size(C)
ans =
3 5
```

The number of columns of matrix C:

```matlab
>> size(C,2)
ans =
5
```

The (maximum) dimension of matrix C:

```matlab
>> length(C)
ans =
5
```

II. Vectors and matrices: referring to elements 1.

Example Matrix

\[
A = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\]

Element of row \(i\) and column \(j\) \(\rightarrow\) \(A(i,j)\)

\(A(3,1)\) \(\rightarrow\) 7

Element at index \(k\) \(\rightarrow\) \(A(k)\)

\(A(1)\) \(\rightarrow\) 1 \(\mid\) \(A(2)\) \(\rightarrow\) 4 \(\mid\) \(A(5)\) \(\rightarrow\) 5

Elements of row \(i\) \(\rightarrow\) \(A(i,:)\)

\(A(2,:)\) \(\rightarrow\) [4 5 6]

Elements of column \(j\) \(\rightarrow\) \(A(:,j)\)

\(A(:,3)\) \(\rightarrow\) [3 6 9]^{T}

2\textsuperscript{nd} and 3\textsuperscript{rd} elements of row 1

\(A(1,[2 3])\) \(\rightarrow\) [2 3]

Each second element

\(A(1:2:end)\) \(\rightarrow\) [1 7 5 3 9]

From the first element

Step size

Till end
II. Vectors and matrices: referring to elements 2.

Example Matrix

\[
A = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\]

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

II. Vectors and matrices: referring to elements 2.

\[
\text{II. Increasing/decreasing sequences}
\]

\[
\text{To create 3 elements of a sequence from 1 to 5}
\]

\[
\text{To create an increasing sequence from 1 to 5 with step size of 0.5}
\]

\[
\text{To create a decreasing sequence from 5 to -5 with step size of -2}
\]
II. Matrices: operations 1.

Matrix $A$

To compute the rank of $A$

To compute the condition number of $A$

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

The inverse is "wrong", because $A$ is ill-conditioned.

To compute the determinant of $A$

II. Matrices: operations 2.

Definition of Matrix $A$

Matrix $A$ to the power two

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

To compute products of columns

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

To compute sums of columns

To compute products of columns
II. Matrices: special types

To create a null matrix:

```matlab
>> zeros(2,2)
ans =
     0     0
     0     0
```

To create a matrix of ones:

```matlab
>> ones(2,2)
ans =
     1     1
     1     1
```

To create an identity matrix:

```matlab
>> eye(2,2)
ans =
     1     0
     0     1
```

II. Matrices: remove a row/column

Definition of Matrix X:

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

To remove the third column:

```matlab
>> X(:,3)=[]
X =
    2   -1    -3
    5     6     8
```
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

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

>> X(2,:)=[9 9 9 9]
X =
  2  -1   0  -3
  9   9   9   9
  0   0   0   0
```

Definition of Matrix X

To add a new row after the last one

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max(V) or max(M)</td>
<td>* max/min/mean/median(V) means that we compute this quantities for the vector V and the result is a scalar value</td>
</tr>
<tr>
<td>min(V) or min(M)</td>
<td>* 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</td>
</tr>
<tr>
<td>mean(V) or mean(M)</td>
<td></td>
</tr>
<tr>
<td>median(V) or median(M)</td>
<td></td>
</tr>
<tr>
<td>std(V) or std(V,1)</td>
<td>* std(V)/var(V) means the unbiased standard deviation/variance of the vector V.</td>
</tr>
<tr>
<td>var(V) or var(V,1)</td>
<td>* std(V,1)/var(V,1) is the biased standard deviation/variance of the vector V.</td>
</tr>
<tr>
<td>cov(V) or cov(M1,M2)</td>
<td>* cov(V) is equal to var(V)</td>
</tr>
<tr>
<td>rand(N,M)</td>
<td>* rand(N,M): returns a matrix with N rows and M columns with uniformly distributed random values between 0 and 1</td>
</tr>
<tr>
<td>randn(N,M)</td>
<td>* randn(N,M): returns a matrix with N rows and M columns with standard normally distributed random values</td>
</tr>
</tbody>
</table>

#### III. Statistical functions: example 1

```matlab
>> n=1:2.*randn(10000,1);  
>> mean(n)
ans =
  1.0097
>> std(n)
ans =
  1.8814
>> var(n)
ans =
  3.6258
>> cov(n)
ans =
  3.0258
>> hist(n,20)
```

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

The empirical histogram of the vector n plotted in 20 intervals

To compute the unbiased variance with the help of the covariance function
III. Statistical functions: example 2

To create a uniformly distributed random sequence between \([2\pi, 4\pi]\):

```
>> n=2*pi*(4*pi-2*pi).*rand(100000,1):
>> var(n)
ans =
 3.2815
```

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

```
>> std(n)
ans =
 1.8115
```

To compute the unbiased standard deviation:

```
>> mean(n)
ans =
 5.4260
```

To compute the mean value:

```
>> median(n)
ans =
 9.4305
```

The empirical histogram of the vector \(n\) plotted in 20 intervals:

```
>> hist(n,20)
```

IV. Graphical functions 1.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot(Y)</td>
<td>• plot(Y) plots the values of vector Y</td>
</tr>
<tr>
<td>plot(X,Y)</td>
<td>• plot(X,Y) plots vector Y versus vector X</td>
</tr>
<tr>
<td>plot(X,Y,s)</td>
<td>• plot(X,Y) plots vector Y versus vector X with visualization parameters, see later on</td>
</tr>
<tr>
<td>stem(Y)</td>
<td>• They plot the data sequence Y as stems from the x axis terminated (with circles) for the data value.</td>
</tr>
<tr>
<td>stem(X,Y)</td>
<td>• stem(X,Y) plots the data sequence X as stems from the y axis terminated (with circles) for the data value.</td>
</tr>
<tr>
<td>stem(X,Y,s)</td>
<td>• stem(X,Y) plots the data sequence X as stems from the y axis terminated (with circles) for the data value.</td>
</tr>
<tr>
<td>figure</td>
<td>• figure: create a new window for plotting</td>
</tr>
<tr>
<td>figure(number_identifier)</td>
<td>• figure(number_identifier): a new window with this identifier in the title part of the window</td>
</tr>
<tr>
<td>hist(V)</td>
<td>• To make a histogram of vector V</td>
</tr>
<tr>
<td>hist(V,n)</td>
<td>• To make a histogram of vector V in n bins</td>
</tr>
<tr>
<td>hold on</td>
<td>• To hold the actual canvas for other new plots</td>
</tr>
<tr>
<td></td>
<td><strong>default:</strong> if you plot a new function the old one will be erased</td>
</tr>
</tbody>
</table>
### IV. Graphical functions 2.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subplot(n,m,p)</td>
<td>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.</td>
</tr>
<tr>
<td>title(string)</td>
<td>It sets a title of the current plot</td>
</tr>
<tr>
<td>xlabel(string)</td>
<td>To set the label text of axis X and Y</td>
</tr>
<tr>
<td>ylabel(string)</td>
<td></td>
</tr>
<tr>
<td>axis([XMIN XMAX YMIN YMAX])</td>
<td>It sets scaling for the x- and y-axes on the current plot.</td>
</tr>
<tr>
<td>legend(string1,string2,...)</td>
<td>It puts a legend on the current plot.</td>
</tr>
<tr>
<td>grid on</td>
<td></td>
</tr>
<tr>
<td>semilogx(X,Y,s)</td>
<td>They are the same as plot() except a logarithmic (base 10) scale is used for the X/Y/X and Y-axis.</td>
</tr>
<tr>
<td>semilogy(X,Y,s)</td>
<td></td>
</tr>
<tr>
<td>loglog(X,Y,s)</td>
<td></td>
</tr>
<tr>
<td>close all</td>
<td>To close all the figure-windows</td>
</tr>
<tr>
<td>mesh(X,Y,Z)</td>
<td>To plot 3D figures</td>
</tr>
<tr>
<td>surf(X,Y,Z)</td>
<td></td>
</tr>
</tbody>
</table>

### 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

```matlab
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

```matlab
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');
```
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

```matlab
>> y = rand(1,1)
    y =
    0.3852
```

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

The result of the script:

It is positive
```

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Logical Operation</th>
<th>Equivalent Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
<td>A &amp; B</td>
<td>and(A, B)</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
<td>A</td>
<td>or(A, B)</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
<td>~A</td>
<td>not(A)</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>==</code></td>
<td>Equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>~=</code></td>
<td>Not equal to</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use them in expressions of for/while.

## V. Source code example – the code

```plaintext
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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS = tf(num,den)</td>
<td>• tf(num,den) creates a continuous-time transfer function SYS with numerator num and denominator den</td>
</tr>
<tr>
<td>SYS = tf(num,den,Ts)</td>
<td>• tf(num,den,Ts) creates a discrete-time transfer function SYS with numerator num and denominator den with sampling time Ts.</td>
</tr>
<tr>
<td>impulse(SYS)</td>
<td>• it returns/plots the impulse response of the dynamic system SYS</td>
</tr>
<tr>
<td>step(SYS)</td>
<td>• it returns/plots the step response of the dynamic system SYS</td>
</tr>
<tr>
<td>lsim(SYS,U,T)</td>
<td>• it returns/plots the time response of the dynamic system SYS to the input signal described by U and T</td>
</tr>
<tr>
<td>bode(SYS)</td>
<td>• it draws the Bode plot of the dynamic system SYS.</td>
</tr>
<tr>
<td>db(x)</td>
<td>• 20*log10(abs(x))</td>
</tr>
</tbody>
</table>
### VI. Signals and Systems in Matlab 2.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>filter(B,A,u)</code></td>
<td>• it returns filtered data to the input u. The filter described by vectors A and B</td>
</tr>
<tr>
<td><code>[X]=fft(x)</code></td>
<td>• they compute the discrete Fourier/ inverse Fourier transform (DFT) of vector X</td>
</tr>
<tr>
<td><code>[x]=ifft(X)</code></td>
<td>• <code>Fft</code>: the first element of returned array is the DC value!</td>
</tr>
<tr>
<td><code>[Gest]=tfestimate(u,y)</code></td>
<td>• it estimates the transfer function, u and y are time-discrete input and output data (vectors)</td>
</tr>
<tr>
<td><code>[H,W] = freqz(B,A,n)</code></td>
<td>• 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</td>
</tr>
<tr>
<td><code>[H,W] = freqs(B,A)</code></td>
<td></td>
</tr>
<tr>
<td><code>[Sxy]=cpsd(x,y)</code></td>
<td>• It returns the cross power spectral density function of time discrete data x and y</td>
</tr>
<tr>
<td><code>repmat(A,m,n)</code></td>
<td>• creates a large matrix consisting of an m-by-n tiling of copies of A</td>
</tr>
</tbody>
</table>

### VI. Example: step response of a system

```matlab
>> 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));
figure
title('step response in a complicated way');
```

```matlab
>> step(h,10)
    Complicated way
```

```matlab
    Easy way
```

![Step Response](image-url)
VI. Example: Estimating a transfer function

Transfer function:
\[ \frac{1}{s + 10} \]

The system to demonstrate in 's' domain

% to define h
h=tf([1],[1 10]);
% to generate the input signal
u=randn(10440,1);
% to make two periods, it is equivalent to u=[u;u]
u_sim=repmat(u,2);
% to compute on the whole domain
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 plot the frequency response function FRF estimate
tfestimate(u,y_meas);
% to plot the normal way to compute the FRF

\[
H=\text{db}(\text{fft}(y_\text{meas})./\text{fft}(u));
\]

To compute the FRF in the 'shorter' way

% \( H = 20 \times \log_{10}(\text{abs}(\text{fft}(y_\text{meas})./\text{fft}(u))) \); 

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

To compute the FRF in the 'normal' way

xlabel('frequency [f_s/f]');
ylabel('Magnitude [dB]');
title('Frequency Response Function');

\textit{This solution can be used with restrictions only!}

Think it over e.g. what happens if the \( \text{U}(\text{fft}(u)) \) contains zero value.
VI. Example: Least Squares solution

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

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

Inverse can be problematic...

\[
\text{also solves the equation but with help of QR decomposition and it is better conditioned}
\]

\[
\text{The analytical way}
\hat{R}_{LS} = \frac{\sum X Y}{\sum X^2} = \frac{\sum V I}{\sum I^2}
\]

\[
\text{\textbackslash also solves the equation but with help of QR decomposition and it is better conditioned}
\]

\[
\text{>> X=[10/1000 20/1000 30/1000 40/1000];,}
\]
\[
\text{>> Y=[10.5 20 30.5 39];}
\]
\[
\text{\textbackslash also solves the equation but with help of QR decomposition and it is better conditioned}
\]

\[
\text{>> X\Y}
\]
\[
\text{ans =}
\]
\[
993.3333
\]
\[
\text{>> inv(X'*X)*X'*Y}
\]
\[
\text{ans =}
\]
\[
993.3333
\]

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

Thanks...

http://commodos.hu/matlab

csurcsia@mit.bme.hu