Universität Rostock

Traditio et Innovatio

Digital Communications

Matlab Tutorial

– SS 2023 –

Universität Rostock
Institut für Nachrichtentechnik
Prof. Dr.-Ing. Volker Kühn
Albert-Einstein-Straße 26
18059 Rostock

http://www.int.uni-rostock.de

Daniel Franz
Room 306
Tel.: 0381/498-7314
daniel.franz@uni-rostock.de

Version from
April 3, 2023

Contents

General Information

- **Do not hesitate to ask questions!!!** (German or English)
- In this project you will have to transmit data using the Lyrtech Hardware provided in the ComLab. For this you have to implement an Orthorgonal Frequency Division Multiplexing (OFDM) system in Matlab. You can form small groups of two or three students and solve the tasks jointly.
- In each exercises there might be a brief repetition of required knowledge. Be aware, that this cannot be a compensation for a missed lecture. The main part of the project will be programming tasks using Matlab. The weekly exercises will be used to answer your questions and to introduce the next tasks.
- Since you will have to write your own scripts in Matlab you should have at least some basic knowledge up to now. There will be a (very) short introduction to Matlab in the first two exercises. If this is not sufficient, there are tons of Matlab books available at the library.
- Before you start,
  - Login to Stud.IP (`https://studip.uni-rostock.de`)
  - Search for the module number "24196" and add it to your own courses. (Note that there are separate Studip-Courses for Vorlesung/Lecture and Projekt/Project.)
  - In the documents section ("*Dateien/Files*") you can find a *Allgemeiner Dateienordner/-General Folder* with the project tasks.
- You can download a framework with many useful functions from Stud.IP. You can always check your implementations by comparing your own results to the results of the functions in provided framework. Please download the framework from the *General Folder*, extract the content and start Matlab. **Before you can use the functions of the framework, you have to run `framework/add_paths.m` script once.** If you forget, the framework functions will be unknown to Matlab.
- For students Matlab is available at the Remote-Desktop-Server of the University of Rostock. If you use Windows, run `mstsc.exe` and connect to `unicomp.uni-rostock.de`. If your studentID is `xx001`, your login name should be `rechenzentrum\xx001`. See also `https://www.itmz.uni-rostock.de/en/internet-services/application-server-itmz/application-ser` You might also download Matlab from the Software Server of the University of Rostock `https://softsrv.uni-rostock.de/distsoft/mathworks` for your own PC.

1 General Remarks

MATLAB is a commercial software of the company *The MathWorks, Inc.* to solve mathematical problems. It is used in both industry and universities especially for numerical simulation as well as the collection, analysis, and evaluation of data. In principal, MATLAB is designed for numerical calculations by means of matrices from which it has its name *MATrix LABoratory*.

At the university of Rostock MATLAB is provided on the corresponding application server as well as for private installation, whereby the latter requires a VPN connection to the university's net. In principal, licenses for MATLAB are costly. However, there are less expensive versions especially for private use of students as well as an open source solution called Octave (`https://www.gnu.org/software/octave/`) which accurately reproduces the functionality of MATLAB.

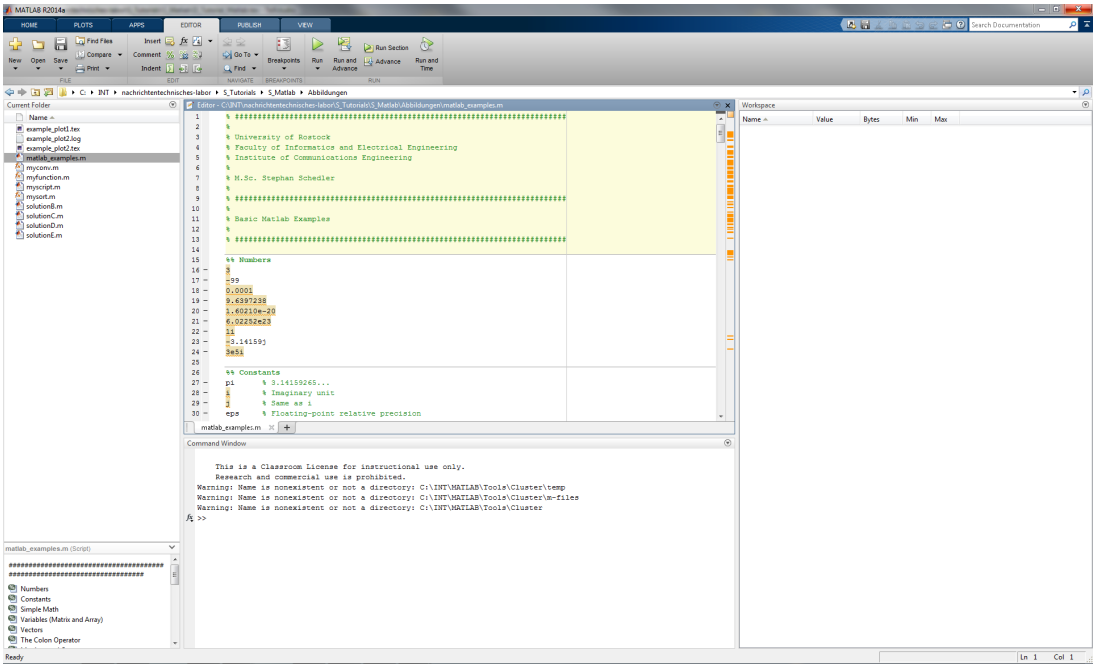


Figure 1: Desktop environment consisting of: Working directory (left), Editor (middle), Command line (below), Workspace (right)

2 Fundamentals

Commands can be typed straightforward in the command line:

```
% This is a comment

a = 1; % terminated command
a+1   % non terminated command (with output)
b = 3, % terminated command with output

c = a + b; c + 1
```

- MATLAB is an interpreter language meaning that scripts and functions are executed line by line. That is the reason why MATLAB was previously known to very inefficiently execute loops. Since that time this issue has been hugely improved. Anyhow, compared to optimized code in C, loops in MATLAB are still considered as inefficient. Fortunately, there are straightforward mechanisms to integrate C code into MATLAB. The MATLAB own functions are usually very efficient.
- A huge benefit of an interpreter language is the possibility to interrupt scripts at any point while execution so that variables can be evaluated and modified. As a consequence, debugging of MATLAB code is straightforward and painless.
- MATLAB is case sensitive.
- To execute commands again, arrow keys ↑ and ↓ can be used to go through previously used commands.
- There are three possibilities to terminate a command:
  - Semicolon ; (Output of the command is suppressed.)
  - comma , (Result of the command will be outputted on command line.)
  - line break (Result of the command will be outputted on command line.)
- Comments start with percent symbol % and are valid for the whole line.
- Running scripts or functions can be straightforward canceled by pressing `[Strg]+[C]`.

Numbers

```
3
-99
0.0001
9.6397238
1.60210e-20
6.02252e23
```

Similar to other languages `XeY` means $X10^Y$.

Predefined Constants

```
pi      % 3.14159265...
i       % Imaginary unit
j       % Same as i
eps     % Floating-point relative precision
realmin % Smallest floating-point number
realmax % Largest floating-point number
Inf     % Infinity
NaN     % Not-a-number

huge    = realmax  % biggest number representable in MATLAB
too_big = huge *2

% Not-a-number
2/0
```

Simple Arithmetic Operations

```
1 + 1
12/3 + 7 *(5 - 1)
2^10
```

Complex Numbers

```
c = 1 + 2i

imag(c)
real(c)

conj(c)
abs(c)
angle(c) *360 / (2*pi)

% Attenetion: "i" is not protected
1*i
i = 1
1*i
1i
```

Variables

In MATLAB data types of variables are adapted dynamically. By default numbers have type double.

```
a = 12    % Numeric (Default: "double")
b = true  % Logical (0 = false, 1 = true)
c = 'ABCD' % String
```

Logical Operators

```
1 > 3
1 <= 3
1 == 3
1 ~= 3

a = 1 == 3

x = 1:5;
x > 2

x < 4 | x > 2
x < 4 & x > 2
true || false
true && false

A = [1 2 3 4];
B = [1 3 2 4];
isequal(A,B)% Compares whole matrix
A == B     % Compares matrix elementwise
```

Strings

```
myText = 'Hello, world'
otherText = 'You''re right'

size(myText)
myText([4:5, 11])

% Concatenate strings
longText = [myText,' -',otherText]

% Convert number <-> string
d = 65;
tempText = ['Temperature is ', d,' C']
tempText = ['Temperature is ',num2str(d),' C']
tempZahl = str2num('123.456')
```

3 Matrices

In MATLAB every variable is a matrix. Scalar values are a matrix with dimension 1×1. The same holds for vectors which are either column vectors with $m \times 1$ or line vectors with $1 \times n$.

Scalars

```
a = 1
size(a)
```

Vectors

```
a = [1, 2, 3, 4]
size(a)

b = [1; 2; 3; 4]
size(b)

% Generate sequences
1:10
1:2:10
100:-7:50
0:pi/4:pi
```

Matrix Operations

```
A = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
size(A)
A = [1 2 3; 4 5 6; 7 8 9]
size(A)

A + 10
A - 10
```

```
A  * A % matrix multiplication
A .* A % elementwise multiplication of matrices

% further operators
A^2
A.^2

A  / A
A ./ A

A'
A.'

X = floor(10*rand(2,2) + 10*rand(2,2) *1i)
X'
X.'
```

Addressing Matrix Elements

```
A = [1 2 3; 4 5 6; 7 8 9]
A(3,2)  % 3. row, 2. column
A(:, 2)
A(:, end)
A(2:3,2)

A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
A(2,1)
A(4)
A(1,:)
A(A > 7)
A(A == 4) = -1

A(4,5) = 17
A(1:3,2)
A(3,:)
A(:, end)
```

Removing Matrix Elements

```
x = [2.1 1.7 1.6 1.5 NaN 1.9 1.8 3.1];

x(7) = []

% cancel NaN values
x = [2.1 1.7 1.6 1.5 NaN 1.9 1.8];
isfinite(x)
x(isfinite(x))

% delete rows or columns
a = [1; 2; 3; 4; 5]
a(3) = []

A(2, :) = []
A(:, 4) = []
```

Concatenate Matrices

```
[A, A]
[A; A]
A = [A, A; A, A]
```

Helpful Functions

```
A = [1 3 5];
B = [10 6 4];

max(A)
max(A,B)

maxA = max(A);

help('max')
max(1,2) % alternatively, choose function name and press [F1]

disp('hello world');

diag([1 2 3; 4 5 6; 7 8 9])

sum(A)

sort([1 5 2 7 -3 6])

find([0 0 1 0])
find(isnan(x))
indices = find(x > 1.6)

clc

clear B
clear all
```

Functions to Generate Special Matrices

```
Z = zeros(2,4)    % Matrix, filled with zeros
F = 5*ones(3,3)   % Matrix, filled with 5

E = eye(4)
E = eye(3)
repmat(E, 2, 1)
repmat(E, 1, 2)

A = 1:30
reshape(A, 3, 10)
```

Matrix Properties

```
A = [1 2 3 4];
B = [1 2 3 5];
C = [1 2; 3 4; 5 6];
```

```
size(C)    % Size of array.
length(C)  % Length of vector.
numel(C)   % Number of elements.
disp(C)    % Display matrix or text.
isempty(C) % True for empty array.
isempty([])
```

Random Numbers

```
U = rand(1,10) % uniform distribution in [0,1]
N = randn(1,10) % standard normal distribution (Mittelwert: 0, Varianz: 1)
```

Rounding

```
x = -1:0.2:1
fix(x)     % round toward zero
floor(x)   % round toward minus infinity
ceil(x)    % round toward plus infinity
round(x)   % round to closest integer
sign(x)    % extract sign
```

Trigonometric Functions

```
%    sin        - Sine.
%    sind       - Sine of argument in degrees.
%    sinh       - Hyperbolic sine.
%    asin       - Inverse sine.
%    asind      - Inverse sine, result in degrees.
%    asinh      - Inverse hyperbolic sine.
%    cos        - Cosine.
%    ...
%    tan        - Tangent.
%    ...
%    cot        - Cotangent.
%    ...
```

Exponential Functions

```
%    exp        - Exponential.
%    expm1      - Compute exp(x)-1 accurately.
%    log        - Natural logarithm.
%    log1p      - Compute log(1+x) accurately.
%    log10      - Common (base 10) logarithm.
%    log2       - Base 2 logarithm and dissect floating point number.
%    pow2       - Base 2 power and scale floating point number.
%    realpow    - Power that will error out on complex result.
%    reallog    - Natural logarithm of real number.
%    realsqrt   - Square root of number greater than or equal to zero.
%    sqrt       - Square root.
%    nthroot    - Real n-th root of real numbers.
%    nextpow2   - Next higher power of 2.
```

4 Plots

2D Plots

```
% Line Plots
x = 0:(pi/100):2*pi;
y = sin(x);
plot(x,y)
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')
axis([0, max(x), min(y), max(y)])

plot(x,y,'r--')

x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)

grid on

hold on
y2 = cos(x);
plot(x,y2,'r:')
legend('sin','cos')

% Discrete Plots
x = 0:pi/5:2*pi;
y3 = cos(x);
stem(x,y3,'g')

bar(x,y3,'g')
```

3D Plots

```
x = -2:.2:2;
y = -2:.2:2;
[X,Y] = meshgrid(x,y);
Z = X .* exp(-X.^2 -Y.^2);
surf(X,Y,Z)

t = 0:pi/10:10*pi;
plot3(t, t.*sin(t), t.*cos(t))
grid on

% Subplots
t = 0:pi/10:2*pi;
subplot(2,2,1); plot(t,sin(t),'r-d')
subplot(2,2,2); plot(t,cos(t),'b:^')
subplot(2,1,2); plot(t,log(t),'b:^')
```

5 Control Structures

For-Loop

```
n = randn(1,1000000);
sum_n = 0
for k = 1:numel(n)
    sum_n = sum_n + n(k);
end
mittelwert = sum_n / numel(n)

sum_n2 = 0;
for (k = 1:numel(n)) % <- brackets are optional
    sum_n2 = sum_n2 + (n(k) -mittelwert)^2;
end
varianz = sum_n2 / numel(n)
```

While-Loop

```
x = 0
while x < 10
    x = x + 1
end
```

IF-Then-Else

```
n = rand(1,1)
 if n < 0.4
    disp('Random variable is less than 0.4.')
 elseif (n > 0.6)
    disp('Random variable is greater than 0.6.')
else
    disp('Random variable is between 0.4 and 0.6.')
end
```

Switch-Case

```
X = floor(4 *rand(1))
switch X
   case 0
      disp('Zero')
   case 1
      disp('One')
   otherwise
      disp('greater One')
end
```

6 Generate Custom Functions

Scripts

Scripts do not have inputs or outputs. They simply execute its listed commands one after another.

Task 6.1

Make a new file `myscript.m` in your working directory with the following content:

```
figure(1)
hist(randn(1,100000),100);

figure(2)
hist(rand(1,100000),100);
```

```
edit myscript % Opens m-File in editor

myscript

path = pwd(); % save current working directory
cd ..         % go directory up
myscript

cd(pwd)       % restores saved working directory
```

Functions

Task 6.2

Make a new file `myfunction.m` with following content in your working directory:

```
function [output1, output2] = myfunction(input1, input2)

  if nargin == 2
    output1 = input1 + input2;
    output2 = input1 *input2;
  elseif nargin == 1
    output1 = 'single input';
    output2 = input1;
  end
end
```

```
[output1, output2] = myfunction(10, 5)
[output1] = myfunction(10)
```

7 Advanced Topics

Errors and Warnings

```
A = 1:16
reshape(A, 4, 4)
reshape(A, 4, 3)

try
  reshape(A, 4, 3)
catch MyError
  disp('an error occurred')
end

error('Custom Error')

A = [1 0 0; 0 2 0; 0 0 3]
inv(A)

A = [1 0 0; 0 2 0; 0 3 0]
inv(A)

warning('Custom Warning')
```

Structures

```
S.name = 'Max';
S.score = 83;
S.grade = 'B+'

S(2).name = 'Paul';
S(2).score = 91;
S(2).grade = 'A-';

S(3) = struct('name','John',...
'score',70,'grade','C')

S(1)
S(2)
S(3)
```

8 Exercises

Task 8.1

Find an as short as possible command to generate the following matrix in MATLAB:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 9 & 7 & 5 & 3 & 1 & -1 & -3 & -5 \\ 4 & 8 & 16 & 32 & 64 & 128 & 256 & 512 \end{pmatrix}$$

Task 8.2

Generate a file `mysort.m` in your working directory with the following content:

```
function list = mysort( list )
```

```
for x = 1:(numel(list)-1)

  for y = 1:(numel(list)-1)
    if list(y) == list(y+1)
      temp = list(y+1);
      list(y+1) = list(y);
      list(y) = temp;
    end

end
```

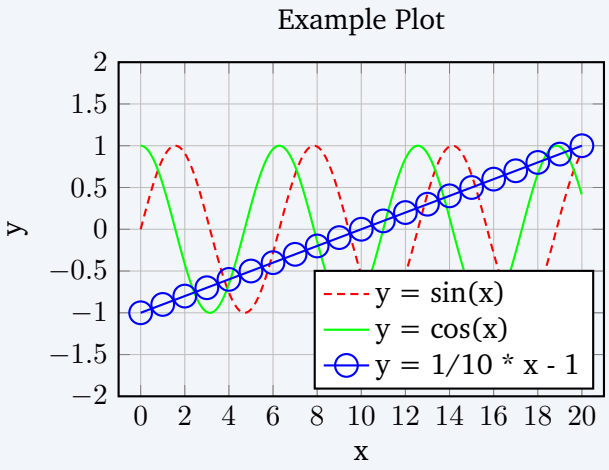
This is an implementation of the well known bubble-sort algorithm. Unfortunately is the given code erroneous. There are 1 × syntax error and 1 × logical error! Find the errors and correct them!

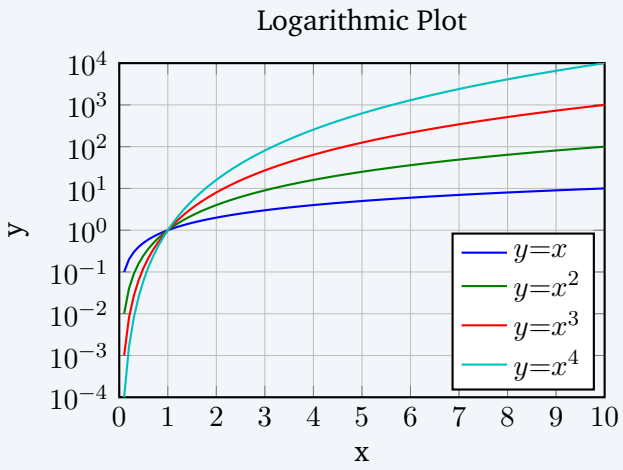
Hints:

- Use the debugger in MATLAB to find the logical error.
- To check whether your algorithm is working, compare the result of `mysort([3 4 1 2 3])` with the standard-function `sort([3 4 1 2 3])`.

Task 8.3

Write a script which generates the following graphics:






```
axis(), figure(), grid(), hold(), legend(), plot(), semilogy(), title(),
xlabel(), ylabel()
```

In MATLAB, on one hand, the function 'conv' can be used to convolve two vectors, on the other hand a convolution can be calculated by simple matrix multiplication, where one of the two vectors is transformed into a convolutional matrix: $y = h * x \rightarrow y = Hx$ with convolutional matrix H having toeplitz structure.

Task 8.4

Implement the convolution of two vectors representing rectangular signals of lengths 8 and 5. Try both above described approaches and compare as well as plot the results. What do you expect?
Hints:

- Use the function 'toeplitz' to create a convolutional matrix in MATLAB.

Task 8.5

Complete the following function for the discrete convolution in MATLAB.

```
function w = myconv(u, v)
% ...
end
```

Check your result by using this function for the convolution of two rectangular signals and plotting its outcome.
Hints:

- $w(t) = u(t) * v(t) = \sum_k u(k)v(t-k+1)$

Task 8.6

Assume a sampling frequency of $F_s = 100$ Msps and generate a sine wave with a frequency of $f = 100$KHz. Plot the signal and its corresponding spectrum by use of the FFT-function of MATLAB. Vary the length of the signal and observe its influence on the signal spectrum delivered by the FFT.
Hints:

- $x[k] = \sin(2\pi f k T_s)$
- Type 'doc fft' to find the help of the FFT-function especially the examples therein are very enlightening

Task 8.7

Generate another sine with frequency 2MHz and add this to the one from the prior task. Again, display the signal in time and frequency.

Task 8.8

Generate a chirp which starts at 1MHz and ends at 2MHz. Display the signal as stated above!
Hints:

- Signal $x(t) = e^{jw(t)t}$ has time dependent frequency
- Linear function $f(t) = f_0 + \frac{k}{2}t$ with slope $k = \frac{f_1 - f_0}{t_1 - t_0}$