Introduction to Matlab **(CSE)**

Emily Mo-Hellenbrand

Scientific Computing in Computer Science
Technische Universität München

October 13^{th}-14^{th}, 2015

Schedule of the next two days

Tuesday, October 13^{th}:
09:00 – 11:30 interactive lecture
11:30 – 12:45 lunch break
12:45 – 15:00 interactive lecture
15:00 – 15:15 break
15:15 – 17:30 supervised individual work

Wednesday, October 14^{th}:
09:00 – 12:00 interactive lecture
12:00 – 13:00 lunch break
13:00 – 15:00 supervised individual work
15:00 – 15:15 break
15:15 – 16:30 supervised individual work

Why MATLAB? From organizational perspective

- Matlab is needed for at least two lectures Scientific Computing Lab and Numerical Analysis
- For Scientific Computing you will need to build teams of two (or three) and submit programs in Matlab
- anybody intends to use other software (Octave, R, Phyton , ...)?,

What is MATLAB® and why do we use it?

- Matlab is a technical computing environment for high-performance numerical computations and visualisation.
- The name Matlab stands for *matrix laboratory*.
- Matlab provides a high-level programming language and an interactive technical computing (and debuging) environment.

- Simulation pipeline: Modeling , Discretization, Computation , Visualization
- Fast prototyping tool for: Algorithm development, Data analysis and visualisation, Numerical computations

Industries using Matlab (selection)

- Aerospace
- Automotive
- Bio-chem, Pharmaceutical, Medical
- Communication
- Financial Industry
- Electronics
- Semiconductors
- ...

Experience with programming languages

- Matlab
- C/C++
- Java, C#
- Visual Basic
- Ada
- Pascal
- Fortran
- Mathematica, Maple
- Shell (Unix, dos)
- Perl
- others

Technical Preparations

Launch matlab on rayhalle:

- Open Terminal and type in "matlab"
- You should have it in your program list (Scientific $\longrightarrow$ Matlab)
- If does not work then:

```
$ /mount/applic/packages/matlab32/matlab/bin/matlab
```

Use Matlab on your Nootebook and Desktop

- Windows 7 (and other Microsoft product): https://prod.maniac.tum.de/ManiacGUI
- Matlab student version: 90$ http://www.mathworks.de/academia/student_version/index.html
- Matlab TUM Student version(free, but you need password, "MatlabForStudents"): http://www.ma.tum.de/bin/view/Studium/MatlabForStudents
- Octave (also with free packages) http://www.gnu.org/software/octave/ http://octave.sourceforge.net/packages.php
- QtOctave as GUI(rather primitive): http://packages.debian.org/testing/math/qtoctave

Technical Preparations

- Open a Linux terminal
- Create directoryies and download files there

```
$ cd ~ #change to your home directory

$ mkdir matlab #creates directory for your M-files

$ mkdir slide #creates directory for the
Introduction slide

$ firefox & #Download this slides and the example
files from the course web page
```

Outline Part I

1. Accessing MATLAB
2. Entering matrices
3. Matrix operations, array operations
4. Statements, expressions, variables; saving a session
5. Matrix building functions
6. M-files I
7. Control constructs: For, while, if
8. Scalar functions
9. Graphics I

Accessing MATLAB


```
($ matlab)
>> % do some work
>> cd <your HOME directory>
>> quit
```

Getting help

Matlab documentation
http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
Matlab Primer
http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html

```
>> help
>> help demo
>> help lookfor
>> help doc
```

command completion
TAB;
previous command
UP;
next command in the history
DOWN;

Entering matrices

scalar

```
>> n = 8;
>> n
```

vector

```
>> x = [1 2 3 4 5 6 7 8]
```

matrix

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

or

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

Entering matrices cont.


```
>> a = [1 2]
>> b = [3 4]
>> B = [a;b]
>> B(1,1) = 5;
>> B
```

load from file

```
>> !echo "1 2 3">C.dat; echo "4 5 6">>C.dat;
echo "7 8 9" >>C.dat
>> save BFile B
>> load('C.dat')
>> D = load('C.dat')
>> load BFile
```

Matrix operations, array operations


```
>> at = a'
>> B = [1 1 1;2 2 2;3 3 3];
>> A^2
>> A + B
>> A * B
>> A.* B
>> A.^2
>> (n = 8)
>> n * A
>> n + A

>> F = [1 2; 3 4]
>> c = [2; 2]
```

Exercise 1

Calculate the solution vector x of the the system $Fx = c$. Use the left division operator '\'. Verify your result.

Statements, expressions, variables; saving a session

- An expression is a combination of values, functions, and variables, that are interpreted (evaluated) according to the rules of matlab.
- A statement is the minimal unit of structuring in matlab.
- All variables are created dynamically. There is no declaration or definition.

```
>> 1>2
>> a*b
>> ans
>> c1 = c;
>> ans
```

Statements, expressions, variables; saving a session cont.

existing variables

```
>> i
>> j
>> eps
>> pi
```

saving a session

```
>> save session1
>> who
>> whos
>> clear
>> who
>> load session1
>> whos
>> !head session1.mat
```

Matrix building functions


```
>> E = eye(3)
>> M = rand(3)
>> Z = zeros(3,2)
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]; % defined
already
>> A(6) %(?)
>> A(9) %(?)
>> A(10) %(?)
```

Exercise 2

Build a 6×3 -Matrix out of A and E !
Build a 3×6 -Matrix out of A and E !

M-files I

- a sequence of commands can be stored in a script file (% comment line)
- files are called 'M-files' (extension of the files is '.m')
- two types of M-files: script files and function files
- store the M-files in the directory ∼/matlab.
- The script will be executed if you call it in the Matlab command line.

```
>> edit % start the matlab editor
```

Exercise 3

Write your solutions of the previous exercises in M-files and execute them!

Control constructs: For, while, if

conditions

```
d = 3.7; e = rand(1);
if (e ~= 0.0)
 f = d/e;
end
 (e ~= 0.0) % 1 -> true, 0 -> false

if (e ~= 0.0)
 f = d/e;
else
 f = 0;
end
 a=1; (a~= 1.001); (a~=1.0000000000000001);
```

Control constructs: For, while, if cont.


```
if (e < 0.5)
 f = -1;
elseif (e > 0.5)
 f = 1;
else
 f = 0;
end
clear f;
```

loops

```
z = [];
for (k=1:10)
 z = [z, rand];
end
z
clear z;
```

Control constructs: For, while, if cont.

insertion: indent lines

```
z = [];
for (k=10:-1:1)
   z = [z, rand];
end
z
clear z;

z = 9.7;
n = 0.0;
while (n+1 <= z)
   n = n + 1;
end
n
clear n z;
```

Control constructs: For, while, if cont.

breaking loops

```
n = 10;
   z = rand(1,n);
l = -1;
for (k=1:n)
   if (z(k)<0.5)
      l = k;
      break;
   end
end
l
clear n z l;
```

Exercise 4

Write an M-file that computes the factorial ($n!$) of a given integer number n !

Scalar functions


```
pi_4 = atan(1.);
sin(pi_4)
exp(1.)
```

insertion: Colon notation

```
[1:5]
[1:3:15]
clear x;
x = [1:3:15];
z = rand(1,10);
z2 = z(1:2:10)
clear z z2;
```

Graphics I


```
f = sin(0:0.1:2*pi);
plot(f)
clear f;
z = 0:0.1:2*pi;
f = sin(z);
plot(z,f)
clear f z;

plot(sin(0:0.1:2*pi));
hold on
plot(cos(0:0.1:2*pi));
hold off
z = 0:0.1:2*pi;
plot(z,sin(z),'r-',z,cos(z),'b--')
clear z;
```

Graphics I cont.


```
z = -2*pi:0.1:2*pi;
plot(z,sin(z),'r-',z,cos(z),'b--')
title('Sine and Cosine');
xlabel('angle');
ylabel('value');
legend('sine','cosine');
grid on
axis([-pi pi -1.5 1.5]);
clear z;
```

Exercise 5

Work though matlab graphics demo 2-D Plots, Line Plotting, and Axes Properties!

Anonymous functions & Graphics I


```
f1 = @(x)(x.*x + 3);
f2 = @(x,y)(2*y - x);
x = 0:0.1:2;
y = 2:-0.1:0;
fr1 = f1(x);
fr2 = f2(x,y);
plot(x,fr1);
figure;
plot(x,fr2);
clear all;
```

Outline Part II

Vector functions


```
>> clear all
>> x = [2 8 3 4 -5 -3 7 -1]
>> y = [3 8 2 1 4 11 8 1.2]
>> A = [6 2 3;1 8 -9]
>> max(x)
>> z = max(x,y)
>> max(A)
>> max(A,[],1)
>> max(A,[],2)
>> [v,ii] = max(x',[],1);
>> v
>> ii
>> x(ii)
>> max(A,4)
```

Vector functions


```
>> sum(x)
>> sum(A)
>> sum(A,1)
>> clear z v ii;
```

Exercise 6

Write an M-file that multiplies the elements in the rows of an 3×3-matrix (each row with a diffrent scalar, the scalars are in a vector) and stores the results in a new 3×3-matrix !

Matrix functions


```
>> B = [x; y]
>> size(B)
>> max(size(B))
>> C = zeros(length(B));
>> whos
>> clear B C;
>> B = [x(1:3); y(3:2:length(y)); A(2,:)]
>> eig(B)
>> [V,D] = eig(B);
>> V
>> D
>> det(B)
>> rank([x;y;x])
```

Exercise 7

Write an M-file that calculates the inverse of a 3×3-matrix! Verify your result! (Verify the result from the [V D] = eig(B);)

Sub-matrices and colon notation


```
>> B
>> B(1:2,2:3)
>> B(:,1)
>> B(2,:)
>> A
>> A2 = A(1:2,1:2);
>> A(1:2,1:2) = eye(2)
>> A(1:2,1:2) = A2(1:2,1:2);
>> C = [1 3; 2 4]
>> z = C(:)
>> n = B(3)
>> B(4)
>> clear n A2 z
```

Text strings, error messages, input


```
>> s = 'Hello World!'
>> s(1)
>> s(1:2:length(s))
>> s1 = s(1:6)
>> s2 = s(7:12)
>> s3 = [s1 s2]
>> s4 = [s1; s2]
>> s(1:12) = 'Hi everybody'
>> s = [s4 '!']
>> clear s s1 s2 s3 s4
```

Text strings, error messages, input cont.


```
>> disp('Hello world!');
>> disp ('The value of pi is:'), disp (pi)
>> val = input('Please enter a number:')
>> error('Sorry, the value is out of range!')
>>
```

Exercise 8

Write a M-file that counts the number of elements between two blanks resp. the begin and the end of a given string!

M-files II (function files)


```
function a = square_area(e)
% SQUARE_AREA. Area of a square.
%   SQUARE_AREA(E) is the area of a square.
%   E is the lenght of a edge.

a = e*e;

% end of square_area
```

```
>> area = square_area(2.0)
>> help square_area
```

M-files II (function files) cont.


```
function [vol,diag] = cube_info(e,d)
% CUBE_INFO. Volume and length of the diagonal of
% a cube.
%
%   [VOLL,DIAG] = CUBE_INFO(E,D) produces the volume
%   of a cube VOL and the length of diagonal of the
%   cube.  Where E is the length of a edge of a
%   D-dimensional cube.

vol = e^d;
diag = e * sqrt(d);

% end of cube_info
```

```
>> [vol, diag] = cube_info(2.0,3)
>> vol = cube_info(2.0,3)
>> [vol, diag] = cube_info(1.5)
```

M-files II (function files) cont.


```
function [vol,diag] = cube_info(e,d)
% CUBE_INFO. Volume and length of the diagonal of
% a cube.
%
%   [VOL,DIAG] = CUBE_INFO(E) produces the volume of
%   a cube VOL and the length of diagonal of the cube.
%   Where E is the length of a edge of the cube.
%
%   [VOLL,DIAG] = CUBE_INFO(E,D) produces the volume
%   of a cube VOL and the length of diagonal of the
%   cube.  Where E is the length of a edge of a
%   D-dimensional cube.
...
```

M-files II (function files) cont.


```
...
if (nargin < 2)
  d = 3;
end

vol = e^d;
diag = e * sqrt(d);

% end of cube_info
```

```
>> [vol, diag] = cube_info(1.5)

>> type cube_info
>> type tic
>> type rank
```

Measuring the execution time: tic and toc


```
>> for n = 1:100;
    A = rand(n,n);
    b = rand(n,1);
    tic
    x = A\b;
    t(n) = toc;
>> end
>> plot(t)
>> clear n x b A t
```

Graphics II


```
close all
[X,Y] = meshgrid(-10:.2:10, -10:.2:10);
Z = sinsinc(0.0,X,Y);
surf(X,Y,Z);

function f = sinsinc (t,x,y)
r = sqrt(x.^2+y.^2) + eps;
f = cos(t)*sin(r)./r;
```

Exercise 9

Print the surface plot into an eps-file!

Graphics II cont.


```
n = 15;
inc = 2*pi/(n-1);
M=moviein(n);

for k=1:n
  t = inc*k;
  Z = sinsinc(t,X,Y);

  clf              %clear figure
  surf(X,Y,Z);
  axis([-10 10 -10 10 -1 1])

  colormap(copper)

  M(:,k) = getframe;
  pause
end
```

Graphics II cont.


```
>> movie(M)
>> movie2avi(M,'~/matlab/sinc.avi')
```

Exercise 10

Print the different figures to 'Portable Network Graphic (PNG)' files instead of creating the movie! Modify the name of the file according to the loop index.

Debuging

- For prototyping a user friendly debugging is necesarry
- Editor debuging features
- Command line debugging features

```
>> dbstop if error
>> help dbstop
```

Write in a script file and run:

```
a = [1 2 3]; i = 4;
a(i)
```