Introduction to Matlab

Engineering Informatics I

What is Matlab?

- programming environment for
  - numeric computation
  - visualization
  - algorithm development
- for engineers, scientists, mathematicians, ...
- functionalities of Matlab:
  - matrix manipulations
  - plotting of functions and data
  - implementation of algorithms
  - creation of user interfaces
  - interfacing with programs written in other languages
  - and many more

Applications of Matlab

- **Numerical Computing:** Mathematical computation, analysis, visualization, and algorithm development
- **Math and Optimization:** Perform numeric and symbolic computation, and apply large-scale optimization techniques
- **Control Systems:** Design, test, and implement control systems
- **Digital Signal Processing:** Analyze signals, develop algorithms, and design DSP systems
- **Communications Systems:** Design and simulate complex communications systems
- **Image and Video Processing:** Acquire, process, and analyze images and video for algorithm development and system design
- **Mechatronics:** Design, optimize, and verify mechatronic systems

Why Learning Matlab?

Advantages of Matlab:

- simple programming language
- powerful
- many libraries/toolboxes for different applications
- easy visualization

Why learning Matlab now?

- you can start programming with Matlab without any programming skills
- use it during your studies for solving mathematical problems (plotting functions, evaluating mathematical expressions, . . .)
- often used by engineers, e.g. for numerical simulation

What will we learn today?

1. Getting started with Matlab
2. Existing operators, variables, and functions
3. Insertion: Vectors, matrices, and systems of linear equations
4. Dealing with vectors and matrices
5. Scripts and functions
6. Visualization with Matlab
7. A short outlook

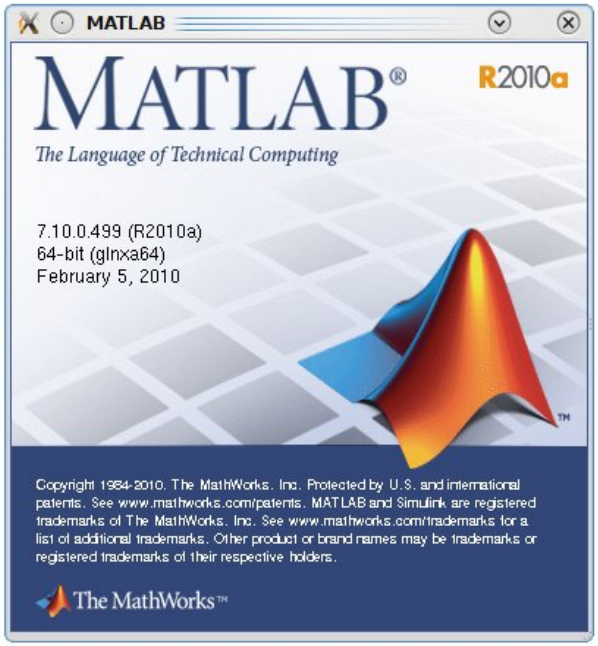
Getting Started With Matlab

- accessing Matlab:

```
>> matlab
>> matlab &
```

- getting help:

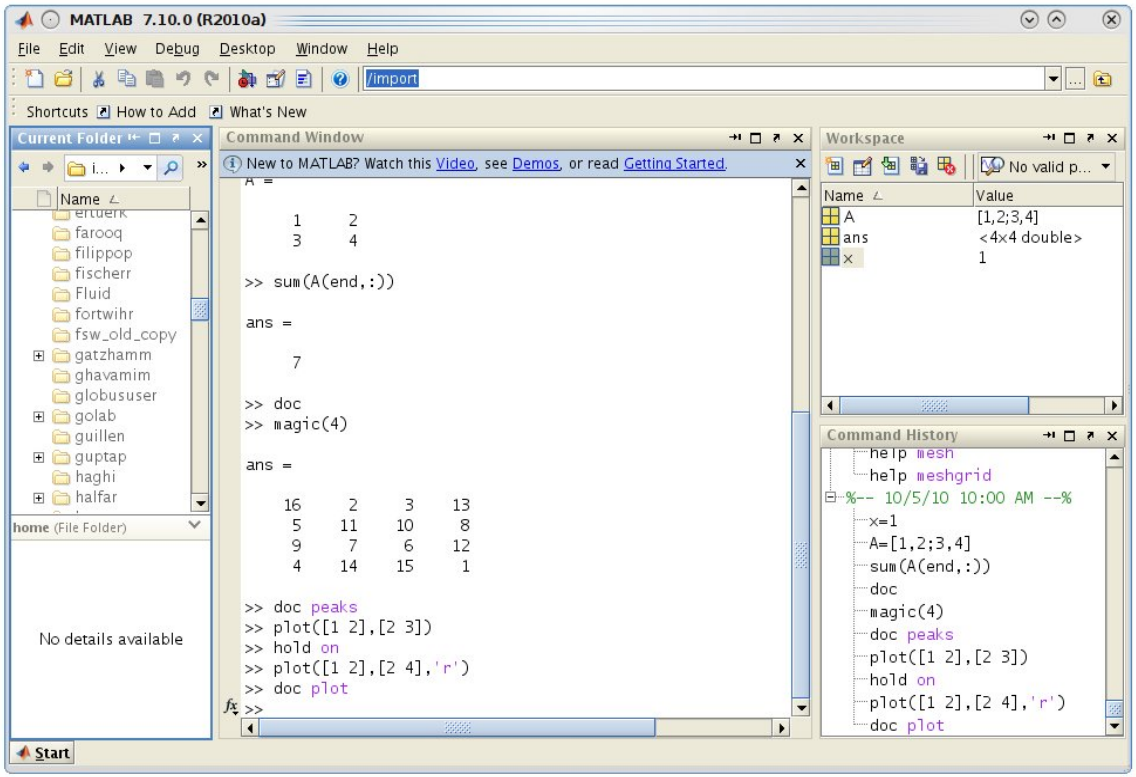
```
>> help
>> doc
>> demo 'matlab'
>> help <command>
>> doc <command>
```



- Matlab documentation in the web:
  http://www.mathworks.com/help/techdoc/
- Matlab tutorials:
  http://www.math.ucsd.edu/~bdriver/21d-s99/matlab-primer.html
  http://www-m3.ma.tum.de/m3old/ftp/matlab.pdf

Working in the Desktop Environment

- command window
- command history
- workspace
- managing files



Tips & Tricks

- command completion: `TAB`
- previous/next command (of history): `UP DOWN`
- enabling and disabling of output in console

```
>> <command>
>> <command> ;
```

- line breaks during commands

```
>> x = 1 + 2 + 3 + ...
     4 + 5 + 6;
```

- comments

```
>> x=1;  % this is a comment
```

- 'most recent answer'

```
>> 1+2+3;
>> ans
```

- clear screen: `clc`

Existing Operators, Variables, and Functions

- operators

```
+ - * / ^
```

- existing variables

```
i, pi, ...
```

- existing functions

```
exp(x), log(x), sin(x), cos(x), sqrt(x), abs(x),  ...
```

- existing functions especially for vectors

```
min(x), max(x), mean(x), sum(x), ...
```

- deleting variables from workspace

```
>> x = pi
>> clear x
```

Insertion: Vectors, Matrices, and Systems of Linear Equations

- programming with Matlab is mainly based on vectors and matrices:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \qquad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

- for example, a system of linear equations

$$\begin{array}{rcrcrcr} 3x & + & 2y & - & z & = & 1 \\ 2x & - & 2y & + & 4z & = & -2 \\ -x & + & \frac{1}{2}y & - & z & = & 0 \end{array}$$

can be represented in matrix-vector notation as follows:

$$\underbrace{\begin{pmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & \frac{1}{2} & -1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix}}_{b}$$

Defining Vectors and Matrices

- defining vectors

```
>> c = [1 -2 0]
>> b = [1; -2; 0]
>> b = [1 -2 0]'
```

$$b = \begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix} \qquad c = \begin{pmatrix} 1 & -2 & 0 \end{pmatrix}$$

- defining matrices

```
>> A = [3 2 -1; 2 -2 4; ...
        -1 0.5 -1]
>> A = [3  2 -1
        2 -2  4
       -1 0.5 -1]
>> B = A'
```

$$A = \begin{pmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & \frac{1}{2} & -1 \end{pmatrix}$$

$$B = \begin{pmatrix} 3 & 2 & -1 \\ 2 & -2 & \frac{1}{2} \\ -1 & 4 & -1 \end{pmatrix}$$

- solving a linear system of equations $Ax = b$:

```
>> x = A\b;
```

$$x = \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix}$$

The Colon Operator

- colon operator: creates vectors for scalar values

```
>> x = [1 2 3 4 5]
>> x = 1:5
```

- defining regularly spaced vectors with arbitrary step size

```
>> x = 0:0.1:1
>> y = 10:-2:0
```

- accessing parts of vectors and matrices

```
>> y(2:5)
>> A(2, :)      % second row
>> A(:, 1)      % first column
>> A(2:4, 1)
```

Accessing Matrix Elements (Slicing)

- accessing single matrix elements by their index

```
>> A(3,5)
```

- vector as index

```
>> k=2:4; b(k)
>> k = [3 2 1 2]; A(k, 2)
```

- logical indexing

```
>> x=-2:4;  x(x>0.5)
```

- ‘end’ keyword: specifies the last index

```
>> b(2:end)
>> A(2:end, end)
>> A(:,end)
```

- example: sum of the last row

```
>> sum(A(end, :))
```

Vectors and Matrices: Standard Operations

- addition and subtraction, and multiplication with a scalar

```
>> b+c          >> b-c          >> A+B
>> 3*b          >> 0.1*A        >> A/37
```

- vector operations: matrix-vector, matrix-matrix, dot product

```
>> A*b      >> A*B      >> A^2      >> (b')*b
```

- element-by-element operations: multiplication, division, and powers $\rightarrow$ mark them with '.'

```
>> b .* c          >> A .* B          >> b .^2
>> b ./ c          >> A ./ B          >> A .^3
```

- What is the difference between `b.*b` and `b'*b`?

```
>> b=1:3
>> b.*b = [1, 4, 9]'
>> b'*b = 14              % 1 + 4 + 9
```

More about Vectors and Matrices ...

- defining specific matrices and vectors

```
>> A = zeros(4, 4)
>> A = ones(3, 2)
>> A = rand(2, 2)
>> A = eye(3)
```

```
>> x = zeros(4, 1)
>> x = ones(4, 1)
>> x = rand(2, 1)
```

- combine matrices with compatible dimensions to new matrices

```
>> C = [A B]
>> D = [A(:, 1:end-1) x]   % replace last column
```

- elementwise application of built-in functions

```
>> sqrt(A)
>> exp(A)
```

```
>> sqrt(x)
>> log(x)
```

- determining the size

```
>> size(A)
```

```
>> length(x)
```

Creating Scripts & Functions

- Matlab script

```
x=10; y=20;
a=0.3;
z=a*(x+y)
```

→ just execute it

- Matlab function

```
function z = myAddition(x,y)
   a = 0.3;
   z = a*(x+y);
end
```

→ store it as `myAddition.m` (in general `<functionName>.m`)
→ call the function with the command

```
z = myAddition(10,20)
```

Programming With Matlab

- if, else, and elseif

```
if (n>0)
   disp('positive')
elseif (n<0)
   disp('negative')
else
   disp('zero')
end
```

- for-loop

```
for k = 1:5
   x(k) = 2*k;
end
```

- while-loop

```
while (k>0)
   k = k-1;
end
```

Graphics

- basic plotting commands:

```
>> x=linspace(0,4*pi, 100); y=sin(x); z=cos(x);

>> plot(x,y)
>> hold on
>> plot(x,y,'bo')
>> hold off
>> plot(x,z,'r--')

>> figure
>> plot(x,y,'b-', x,z,'r--')
>> title('trigonometric functions')
>> xlabel('x-values'), ylabel('y-values')
>> legend('sin','cos')
```

for more details use `help plot`

3D Visualization

- simple 3D plot

```
>> x = linspace(0,5, 10);
>> y = sin(x);
>> z = 1:10;
>> plot3(x,y,z, '.-')
```

- surface and contour plots

```
>> % generate 2D grid first:
>> x = -3:0.2:3;  y = -4:0.2:4;    % 1d axes
>> [X,Y] = meshgrid(x,y);          % 2d grid

>> % compute values on grid and visualize:
>> Z = sin(X).*sin(Y);
>> surf(X,Y,Z)
>> mesh(X,Y,Z)
>> plot3(X,Y,Z)
>> contour(X,Y,Z)
```

Outlook: Using Built-in Matlab Functions

- Interpolation

  `interp1, interp2, spline, ...`
- Quadrature (Evaluating Integrals)

  `quad, quadl, quad2d, ...`
- Solving Ordinary Differential Equations

  `ode23, ode45, ode15s, ...`
- Matrices & Eigenvalues

  `det, norm, cond, eig, ...`
- Fourier analysis

  `fft, fftn, ifft, ...`
- and many more ...

And now?

Let's go on with the tutorial...

...in room **MI 00.05.011** („große Rechnerhalle“)
and room **MI 00.07.023** („kleine Rechnerhalle“)

(building of the faculty of maths and informatics)