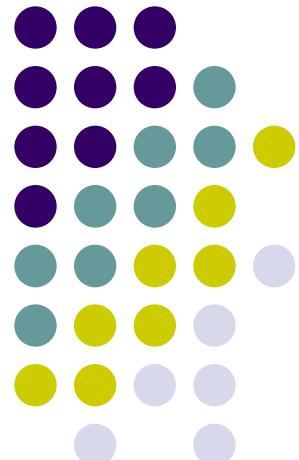


# Angewandte Mathematik

Studiengang Informatik  
Duale Hochschule BW Karlsruhe

Kaori Nagato-Plum

[kaori.nagatou@kit.edu](mailto:kaori.nagatou@kit.edu)





# Lösungsvorschlag für die Aufgaben 2.1-2.4

2.1 Berechne Sie alle partiellen Ableitungen erster und zweiter Ordnung der Funktionen:

$$f(x, y) = x^2 e^y + e^{xy}$$

$$g(x, y) = \sin^2(xy)$$

$$h(x, y) = e^{\cos x + y^3}$$

Lösungsvorschlag:

$$f_x(x, y) = 2xe^y + ye^{xy}, f_y(x, y) = x^2e^y + xe^{xy}$$

$$f_{xy}(x, y) = f_{yx}(x, y) = 2xe^y + (xy+1)e^{xy}, f_{xx}(x, y) = 2e^y + y^2e^{xy}, f_{yy}(x, y) = x^2e^y + x^2e^{xy}$$

$$g_x(x, y) = 2y \sin(xy) \cos(xy) = y \sin(2xy), g_y(x, y) = 2x \sin(xy) \cos(xy) = x \sin(2xy),$$

$$g_{xy}(x, y) = g_{yx}(x, y) = 2xy \cos(2xy) + \sin(2xy),$$

$$g_{xx}(x, y) = 2y^2 \cos(2xy), g_{yy}(x, y) = 2x^2 \cos(2xy)$$

$$h_x(x, y) = -e^{\cos x + y^3} \sin(x), h_y(x, y) = 3y^2 e^{\cos x + y^3}$$

$$h_{xy}(x, y) = h_{yx}(x, y) = -3y^2 \sin(x) e^{\cos x + y^3},$$

$$h_{xx}(x, y) = (\sin^2 x - \cos x) e^{\cos x + y^3}, h_{yy}(x, y) = (9y^4 + 6y) e^{\cos x + y^3}$$

## 2.2 Berechnen Sie die Hesse-Matrizen der Abbildungen:



$$f(x, y) = e^{xy} + \cos^2 y$$

$$g(x, y) = xy - e^{x+y}$$

$$h(x, y) = \sqrt{x^2 + y^2 + 1}$$

Lösungsvorschlag:

$$f(x, y) = e^{xy} + \cos^2 y$$

$$H(x, y) = \begin{bmatrix} f_{xx}(x, y) & f_{xy}(x, y) \\ f_{yx}(x, y) & f_{yy}(x, y) \end{bmatrix} = \begin{bmatrix} y^2 e^{xy} & e^{xy} + xye^{xy} \\ e^{xy} + xye^{xy} & x^2 e^{xy} - 2 \cos(2y) \end{bmatrix}.$$

$$g(x, y) = xy - e^{x+y}$$

$$H(x, y) = \begin{bmatrix} g_{xx}(x, y) & g_{xy}(x, y) \\ g_{yx}(x, y) & g_{yy}(x, y) \end{bmatrix} = \begin{bmatrix} -e^{x+y} & 1 - e^{x+y} \\ 1 - e^{x+y} & -e^{x+y} \end{bmatrix}.$$

$$h(x, y) = \sqrt{x^2 + y^2 + 1}$$

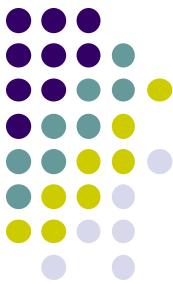
$$H(x, y) = \begin{bmatrix} h_{xx}(x, y) & h_{xy}(x, y) \\ h_{yx}(x, y) & h_{yy}(x, y) \end{bmatrix} = \begin{bmatrix} \frac{y^2 + 1}{(x^2 + y^2 + 1)\sqrt{x^2 + y^2 + 1}} & -\frac{xy}{(x^2 + y^2 + 1)\sqrt{x^2 + y^2 + 1}} \\ -\frac{xy}{(x^2 + y^2 + 1)\sqrt{x^2 + y^2 + 1}} & \frac{x^2 + 1}{(x^2 + y^2 + 1)\sqrt{x^2 + y^2 + 1}} \end{bmatrix}.$$

## 2.3 (vergangene Klausur Aufgabe (10 Punkte))

Bestimmen Sie die Tangentialebene an die Fläche

$$z = f(x, y) = x^2 - y - xe^y$$

im Punkt  $(1, 0, f(1, 0))$ .



Lösungsvorschlag:

$$f_x(x, y) = 2x - e^y, \quad f_y(x, y) = -1 - xe^y$$

$$\begin{aligned} \Rightarrow g(x, y) &= f(1, 0) + f_x(1, 0)(x - 1) + f_y(1, 0)(y - 0) \\ &= x - 2y - 1 \end{aligned}$$



## 2.4 Bestimmen und klassifizieren Sie alle Extrema der Funktion

$$f(x, y) = (1 + 2x - y)^2 + (2 - x + y)^2 + (1 + x - y)^2.$$

Lösungsvorschlag:

$$\begin{cases} f_x(x, y) = 2 + 12x - 8y = 0 \\ f_y(x, y) = -8x + 6y = 0 \end{cases}$$

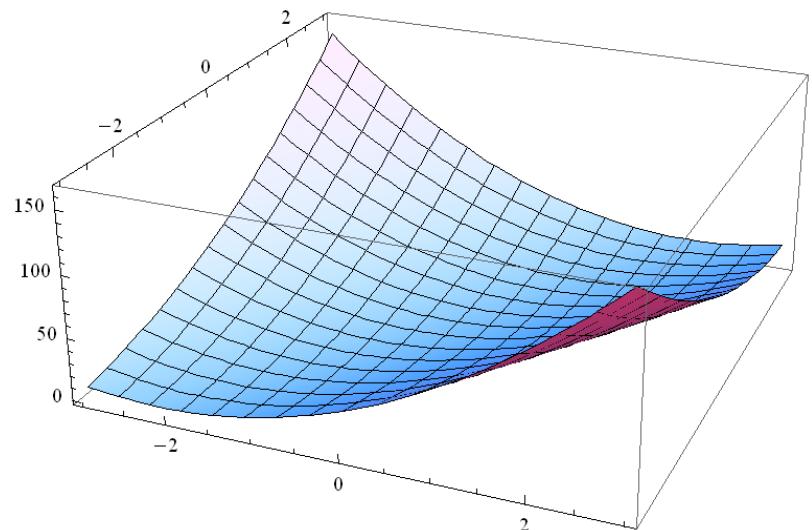
$$\Rightarrow (x_0, y_0) = \left( -\frac{3}{2}, -2 \right)$$

$$H(x, y) = \begin{bmatrix} f_{xx}(x, y) & f_{xy}(x, y) \\ f_{yx}(x, y) & f_{yy}(x, y) \end{bmatrix} = \begin{bmatrix} 12 & -8 \\ -8 & 6 \end{bmatrix}$$

$$\left| H\left(-\frac{3}{2}, -2\right) \right| = 8 > 0$$

$$f_{xx}\left(-\frac{3}{2}, -2\right) = 12 > 0$$

Das Minimum der Funktion liegt bei  $x = -\frac{3}{2}, y = -2$ .  $f\left(-\frac{3}{2}, -2\right) = 4,5$





# MATLAB Overview

<http://www.mathworks.com/products/matlab/>

MATLAB® is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

- All data are treated as array
- Any declaration for variable or array is not needed as in C(C++) or Fortran
- Many functions are available
- Beautiful graphics are obtained without writing complicated code
- Many useful toolbox

# MATLAB's Main Window



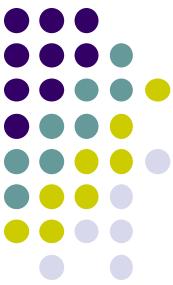
The screenshot shows the MATLAB R2011b interface with the following components:

- Current Folder**: A browser window showing files like `matlab`, `timetest.m`, `interface_eigenvalues.m`, and `completetest.m`.
- File details**: A panel showing details for the selected file `interface_eigenvalues.m`.
- Command Window**: Displays the command `>> a=1; b=2; c=a+b` and its output `c = 3`.
- Workspace**: Shows variables `a`, `b`, and `c` with their assigned values.
- Command History**: Lists the commands `a=1; b=2; c=a+b` run on 20.09.2013 at 18:24.

**Command Window:** Execution of calculations,  
to call commands and functions

**Workspace:**  
Shows assigned  
variables

**Command History:**  
Lists all previously  
called commands



## Example : multiplication of a matrix A and a vector x

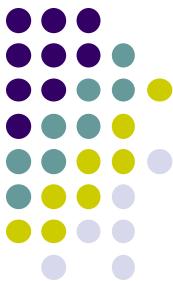
C language ...

(After some declarations and initialization)

```
for(i=0; i<n; i++){  
    for(j=0; j<n; j++){  
        y[i]+=A[i][j]*x[j];  
    }  
}
```

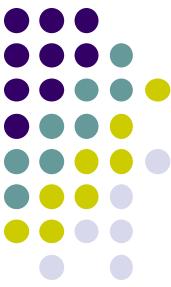
MATLAB ...

```
y = A*x;
```



# Variables

- The first character of a name of a variable should be an alphabetic character and the variable-name should be different from the name of MATLAB functions (or predefined variables)
- Names of variables are case sensitive, i.e. a capital letter is distinguished from a small letter
- Types of variables do not need to be specified: In assignment, MATLAB does not distinguish between integers and real numbers with decimal point
- All initialized variables are listed in Workspace-window: Variables can easily be saved for later sessions or imported from saved data



## Basics: In- and Output

```
>> a=1; b=2; c=a+b ↵  
c =  
3
```

↵ : Enter Key

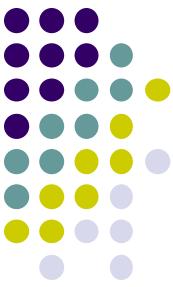
Variables **a** and **b** are initialized (no prior declaration), sum of **a** and **b** is calculated and stored in variable **c**.

Semicolon after formula: Result is not displayed  
Without semicolon the result is displayed

```
>> a=1, b=2, c=a+b ↵  
a =  
1  
b =  
2  
c =  
3
```

```
>> a=1; b=2; c=a+b; ↵  
    >>
```

In case that all formulas end with semicolon, nothing is displayed.



By adding three dots in the end of a line, input can be continued to the next line.

```
>> a1=1; a2=2; a3=3; a4=4; ...      ↘  
a5=5; a6=6; a7=7; a8=8; ...      ↘  
a1+a2+a3+a4+a5+a6+a7+a8      ↘  
ans =  
36
```

If no variable is initialized for a result, the expression “ans =“ is displayed . You can use the command “disp” if you want to display a result without “ans=”.

```
>> a1=1; a2=2; a3=3; a4=4; ...      ↘  
a5=5; a6=6; a7=7; a8=8; ...      ↘  
disp(a1+a2+a3+a4+a5+a6+a7+a8)  ↘  
36
```



## Special variables

In order to express some special numerical values or special matrices, some names are fixed as follows:

Name	Meaning
pi	$\pi$
Inf, inf	infinity $\infty$
i, j	imaginary unit $\sqrt{-1}$
eps	machine epsilon (minimum $\epsilon$ satisfying $1+\epsilon>1$ )
eye(n)	unit matrix of dimension n
ones(n,m)	matrix (n rows, m columns) whose all elements are 1
zeros(n,m)	matrix (n rows, m columns) whose all elements are 0
NaN, nan	Not a Number
realmin	the smallest positive floating point number which is expressible in a computer
realmax	the greatest positive floating point number which is expressible in a computer



※ If you substitute a different value for these special variables, then it is maintained until you use the “clear” command. Especially i, j are often used as variables in loops, so you must take care in case you use the imaginary unit.

```
>> pi ↪  
ans =  
    3.14159265358979  
>> pi=6 ↪  
pi =  
    6  
>> 2*pi ↪  
ans =  
    12  
>> clear pi ↪  
>> 2*pi ↪  
ans =  
    6.28318530717959
```

## complex variables



```
>> z1=1+2*i ↪  
z1 =  
    1 + 2i  
>> z2=3+4*j ↪  
z2 =  
    3 + 4i  
>> z1+z2 ↪  
ans =  
    4 + 6i
```

Once a value is substituted for  $i$ , then you cannot use  $i$  for imaginary unit until you use the “clear” command in order to restore it.

```
>> i=2; z2=5+6*i ↪  
z2 =  
    17  
>> clear i ↪  
>> z2=5+6*i ↪  
z2 =  
    5 + 6i
```

## Changing of displayed accuracy

【Note that this does not affect computational accuracy】



```
>> x=1/3 ↪  
x =  
0.3333  
>> format short e; x ↪  
x =  
3.3333e-001  
>> format long; x ↪  
x =  
0.33333333333333  
>> format long e; x ↪  
x =  
3.33333333333333e-001
```

short (default)	fixed point number with 5-digit number
short e	floating point number with 5-digit number
long	fixed point number with 15-digit number
long e	floating point number with 15-digit number

If you change the displayed accuracy using “format”, it is maintained unless you change it again.

# Arrays



vector ··· one dimensional array

matrix ··· two dimensional array

An index of an array must be a positive integer starting from 1.

```
>> x=[1 2 3]    ↘  
x =  
    1    2    3  
>> x'  
ans =  
    1  
    2  
    3  
>> y=[1;2;3]    ↘  
y =  
    1  
    2  
    3
```

row vector (also  $x=[1,2,3]$ )

column vector is expressed adding ' '.

Since semicolon expresses the end of a row, a column vector can also be generated like this.



There are other methods to generate vectors.

■ Specify the initial value, increment and last value with colon

```
>> x=[1:2:7] ↘  
x =  
    1   3   5   7  
>> x=[1:4] ↘  
x =  
    1   2   3   4
```

From 1 to 7 incrementing by 2

If the increment is 1 then it can be omitted.

■ Specify the initial value, last value and the number of element using the function “linspace”.

```
>> y=linspace(1,3,5) ↘  
y =  
    1       1.5      2       2.5      3
```

Generate 5 elements from 1 to 3 with equal space.



```
>> A=[1 2 3; 4 5 6] ↪
```

```
A =
```

```
1 2 3  
4 5 6
```

```
>> A' ↪
```

```
ans =
```

```
1 4  
2 5  
3 6
```

```
>> a=A(:) ↪
```

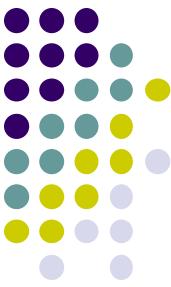
```
a =
```

```
1  
4  
2  
5  
3  
6
```

matrix with 2 rows  
and 3 columns

A' stands for the transposed  
matrix of A (if A is real).  
(A complex: A' is the complex  
conjugate transpose of A)

transformation of two  
dimensional array to one  
dimensional array.  
Columns of the matrix are  
written into vector sequentially



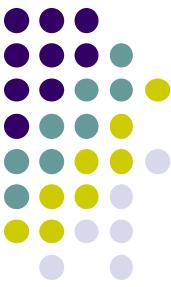
## Access to single elements of an array

$x(k)$ : k-th element of one dimensional array x.

$A(i,j)$ : (i,j)-element of two dimensional array A.

```
>> x=[1 2 3]    ↘  
x =  
    1    2    3  
>> x(2)    ↘  
ans =  
    2  
>> A=[1 2 3;4 5 6]    ↘  
A =  
    1    2    3  
    4    5    6  
>> A(2,2)    ↘  
ans =  
    5  
>> A(5,5)    ↘  
??? The index exceeds the dimension of array
```

An error code is displayed if you called a non-existing element.



## Access to subarrays

$x(k:l)$ : subvector  $\left( x_i \right)_{k \leq i \leq l}$

$A(k:l, n:m)$ : submatrix  $\left( A_{ij} \right)_{\begin{subarray}{l} k \leq i \leq l \\ n \leq j \leq m \end{subarray}}$

$A(:,m)$ : gives the m-th column of A (column vector)

$A(n,:)$ : gives the n-th row of A (row vector)

```
>>x=[1:7]
x =
    1    2    3    4    5    6    7
>>x(3:5)
ans =
    3    4    5
```

```
>>A=[1,2,3;4,5,6]
A =
    1    2    3
    4    5    6
>>A(:,1)
ans =
    1
    4
>>A(1,:)
ans =
    1    2    3
```



# Useful command and control key

## ■ help command

You can check how to use some *command* as follows:

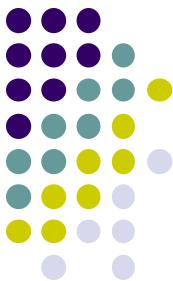
`>> help command`

```
>> help format ↵
format Set output format.
format with no inputs sets the output format to the default appropriate
for the class of the variable. For float variables, the default is
format SHORT.

format does not affect how MATLAB computations are done. Computations
on float variables, namely single or double, are done in appropriate
floating point precision, no matter how those variables are displayed.
Computations on integer variables are done natively in integer. Integer
variables are always displayed to the appropriate number of digits for
the class, for example, 3 digits to display the INT8 range -128:127.
format SHORT and LONG do not affect the display of integer variables.

format may be used to switch between different output display formats
of all float variables as follows:
format SHORT   Scaled fixed point format with 5 digits.
format LONG    Scaled fixed point format with 15 digits for double
                and 7 digits for single. ....
```

how to use  
“format”



The statement which starts with % in your original M-file can be also displayed by the help command.

For example, if the file flow.m has the following content,

```
% Program to plot the solution  
function [f] = flow(K);  
...
```

then help command gives us the following output.

```
>> help flow ↪  
Program to plot the solution
```

Add some appropriate comment in the beginning of your M-file in order to remember of which kind of program it is.



## ■ lookfor command

You can search for a keyword in predefined matlab functions and commands. For example, if you want to find functions and commands involving max then you may input “lookfor max” in the command line.

```
>> lookfor max ↩  
cgminmaxexpr           - Expression that chooses min/max of expression  
HueSaturationValueExample - Compute Maximum Average HSV of Images with MapReduce  
MaxMapReduceExample      - Find Maximum Value with MapReduce  
maxArrivalDelayMapper   - Mapper function for the MaxMapreduceExample.  
maxArrivalDelayReducer  - Reducer function for the MaxMapreduceExample.  
cummax                  - Cumulative largest component.  
max                     - Largest component.  
flintmax                - Largest consecutive integer in floating point format.  
intmax                  - Largest positive integer value.  
realmax                 - Largest finite floating point number.  
maxNumCompThreads       - controls the maximum number of computational threads  
namelengthmax           - Maximum length of MATLAB function or variable name.  
bitmax                  - Maximum double precision floating point integer.  
nzmax                   - Amount of storage allocated for nonzero matrix elements.  
rtwdemo_minmax_script  - Optimizing Generated Code Using Specified Minimum and Maximum Values  
warnAboutBusElementMinMaxValidity - calls from bus element dialog and bus editor  
dndsmi                  - (non)-synonymous substitution rates by the maximum likelihood method.  
graphmaxflow             - calculates the maximum flow in a directed graph. ...
```



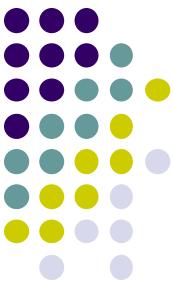
Some functions or commands are displayed with simple explanations.



## ■ arrow key and control key

In MATLAB the command-history is preserved in the memory, you can use the arrow keys to call previous commands.

↑, ↓	search in command history
x( ↑,↓	search in command history for commands starting with x( only
→, ←	Move right /left for one letter
Delete	Delete the letter to the left of cursor
Ctrl-D	Delete the letter to the right of cursor
Ctrl-A	Go to to beginning of the line
Ctrl-E	Go to to end of the line
Ctrl-U	Delete current line
Ctrl-K	Delete all letters from current position to the end of the line



# Operations, Mathematical functions

## operations

formula	expression in MATLAB
$a + b$	<b>a + b</b>
$a - b$	<b>a - b</b>
$ab$	<b>a * b</b>
$a / b$	<b>a / b</b>
$a^b$	<b>a^b</b>



comparison operation → true: 1 false: 0

---

formula expression in MATLAB

---

$a < b$       **a < b**

$a \leq b$       **a <= b**

$a > b$       **a > b**

$a \geq b$       **a >= b**

$a = b$       **a == b**

$a \neq b$       **a ~= b**

and      **&**

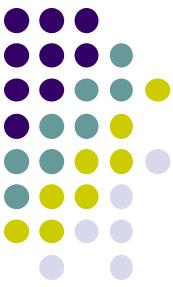
or      **|**

---



## Example:

```
>> a=1; b=[2;3]; A=[1 2;3 4]; B=[5 2; 3 5];  
>> a>0  
ans =  
1  
>> b==2  
ans =  
1  
0  
>> A==B  
ans =  
0 1  
1 0  
>> A<B  
ans =  
1 0  
0 1  
>> A~=B  
ans =  
1 0  
0 1
```



## element-wise operations

For example, for two matrices  $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ ,  $B = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$

```
>> A*B ↵
```

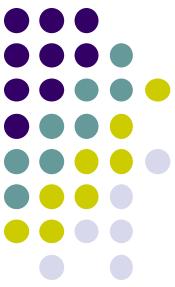
gives  $AB = \begin{pmatrix} 5 & 4 \\ 11 & 10 \end{pmatrix}$

If you want to compute the matrix  $\begin{pmatrix} a_{ij} b_{ij} \end{pmatrix}$  (element-wise operation)  
then replace `'*'` by '.*' as follows:`

```
>> A .* B ↵
```

```
ans =
```

```
1 4  
6 4
```



## Example for vectors

$$\mathbf{a} = (a_1, \dots, a_n), \mathbf{b} = (b_1, \dots, b_n)$$

---

formula	expression in MATLAB
$(a_1 + b_1, \dots, a_n + b_n)$	$\mathbf{a} + \mathbf{b}$
$(a_1 b_1, \dots, a_n b_n)$	$\mathbf{a} .* \mathbf{b}$
$(a_1 / b_1, \dots, a_n / b_n)$	$\mathbf{a} ./ \mathbf{b}$
$(a_1^{b_1}, \dots, a_n^{b_n})$	$\mathbf{a} .^{\wedge} \mathbf{b}$

---



# Mathematical function

Various mathematical functions are available in MATLAB.  
There are also many mathematical functions in each toolbox.  
Use the “help” command to understand their handling.

## Basic mathematical functions

abs	absolute value
sqrt	square root
log	natural logarithm
exp	exponential function
real	real part of a complex variable
imag	imaginary part of a complex variable
angle	argument of a complex variable
conj	conjugate complex variable
...	



## Basic statistical functions

sum	sum of elements
mean	mean value
cov	covariance
corrcoef	correlation coefficient
sort	sort in ascending or descending order
prod	product of elements
cumsum	cumulative sum
cumprod	cumulative product
...	

## trigonometric functions

sin	$\sin(x)$
cos	$\cos(x)$
tan	$\tan(x)$
asin	$\sin^{-1}(x)$
acos	$\cos^{-1}(x)$
atan	$\tan^{-1}(x)$
sinh	$\sinh(x)$
cosh	$\cosh(x)$
...	

## matrix functions

... etc

inv	inverse matrix
det	determinant
lu	LU decomposition
chol	Cholesky decomposition
norm	(various) norms
cond	condition number
eig	eigenvalue
...	



# Script and function

M-files can be “scripts” and “functions”, both are named \*\*\*.m.

script file ··· consisting of some commands

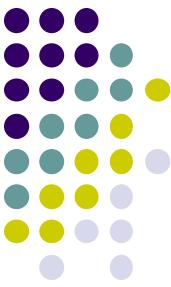
You can execute it by typing its file-name.

function file ······ consisting of definition for function

You can use a function/script without any declaration if the folder containing the m-file is on your MATLAB-path.

To add a folder:

**File** → **Set Path ...** → **Add Folder**

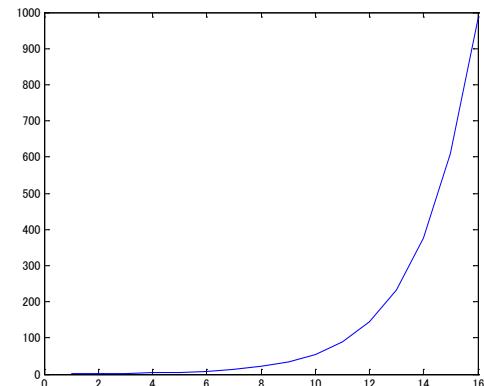


## Example of a script

```
% An M-file to calculate Fibonacci numbers  
f=[1 1]; i=1;  
while f(i)+f(i+1)<1000  
    f(i+2)=f(i)+f(i+1);  
    i=i+1;  
end  
disp(f)  
plot(f)
```

Size of array is increased automatically  
→ **fibo.m**

```
>> fibo ↩  
Columns 1 through 11  
1 1 2 3 5 8 13 21 34 55 89  
Columns 12 through 16  
144 233 377 610 987
```





## Definition of function

In the beginning of M-file, declare the function with in- and output parameters as follows:

**function [list of output] = function-name(list of input)**

## Example of function

```
% Function to work out the sum and remainder  
% of two real numbers  
function [wa,sa]=cal(a,b)  
wa=a+b;  
sa=a-b;
```

→ **cal.m**

```
>> x=1; y=2; ↪  
>> [c,d]=cal(x,y) ↪  
c =  
    3  
d =  
   -1
```



## Global variables

The common variables between script and function can be declared as global variables.

### Example

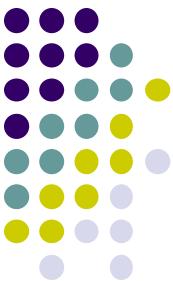
```
global a b c  
a=1; b=2; c=3;  
x=1;  
y=func(x);  
disp(y)
```

→ **cal2.m**

```
function y=func(x)  
global a b c  
y=a*x^2 + b*x +c;
```

→ **func.m**

>> cal2 ↪  
6



# Branch, Loop

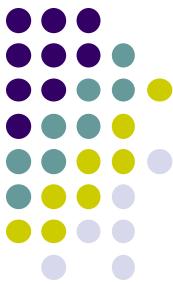
## if statement

```
if expression1  
    statement1  
else if expressions2  
    statement2  
else  
    statement3  
end
```

or

```
if expression1, statement1  
else if expressions2, statement2  
else statement3  
end
```

※ “else if” and “else” can be omitted.

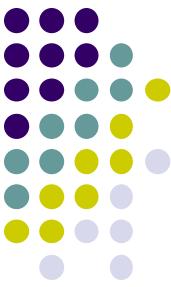


## Example

```
>> x=-1;  
>> if x>0, y=1;  
    elseif x==0, y=0;  
    else y=-1;  
    end  
>> y  
y =  
-1
```

If x does not have any value then the following error message appears:

??? Undefined function or variable 'x'.



# Loop

Several program statements can be repeatedly executed in a **loop**. The variable controlling the repetition is called **loop variable**.

There are two types of loops.

- To specify the number of iteration

————→ **for**

- To repeat a group of statements as long as a certain condition is satisfied

————→ **while**

# for



```
for loop variable = range of the loop variable  
    statement  
end
```

```
>> s=0;  
>> for i=1:10  
    s=s+i;  
end;  
>> disp(s)  
55
```

i: loop variable, iteration number is 10

```
>> s=0;  
>> for i=10:-1:1  
    s=s+i;  
end;  
>> disp(s)  
55
```

If you want to execute the iteration conversely then these expressions are available.

A loop variable can be incremented by a predefined value in each step:

```
>> s=0;  
>> for i=2:2:10  
    s=s+i;  
end;  
>> disp(s)  
30
```

Loop variable i is set as even number.  
( i starts at 2, is incremented by 2 in each step and ends at 10 )

```
>> s=0;  
>> for i=10:-2:2  
    s=s+i;  
end;  
>> disp(s)  
30
```

Converse version

```
>> v=[1 10 100];  
>> for k=v  
    disp(k)  
end  
1  
10  
100
```

vector can also serve as range for loop variable



The **continue** command is used when you want to go to the next iteration of the loop, and the **break** command is used when you want to terminate the execution of the loop.

```
>> s=0;  
>> for k=[1 -1 2 -2 3 -3 4 -4 5 -5]  
    if k<0, continue;  
    end  
    s=s+k;  
    if s>5, break;  
    end  
    disp(s);  
end  
1  
3
```



If k is negative then go to next iteration.

If s is greater than 5, quit the loop.



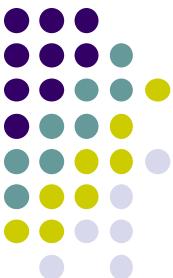
## while

```
while condition  
    statement  
end
```

```
>> a=10;  
>> while a>=1  
    a=a/2;  
    disp(a)  
end  
5  
2.50000000000000  
1.25000000000000  
0.62500000000000
```

The commands within the loop are executed while  $a$  is greater than or equal to 1.

“break” and “continue” can be used as in the “for”-loop.



# Measurement of elapsed time

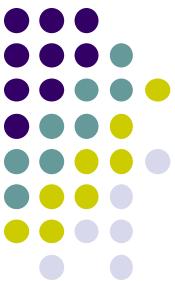
The elapsed time between “tic” and “toc” can be measured.

```
>> tic;  
A=rand(1000,1000);  
toc;  
elapsed_time = 0.078000
```

```
>> tic;  
A=rand(1000,1000);  
time=toc;  
disp(time);  
0.078000000000000
```



elapsed time can be stored  
in a variable.



## A small trick can speed up computations!

```
>> clear;  
tic;  
for j=1:1000  
    for k=1:1000  
        A(j,k) = j+k;  
    end  
end  
toc;  
Elapsed time is 0.314685 seconds.
```



Try to reduce computation time.



```
>> clear;
tic;
A=zeros(1000,1000);
for j=1:1000
    for k=1:1000
        A(j,k) = j+k;
    end
end
toc;
Elapsed time is 0.032468 seconds.
```



Add this line

In MATLAB arrays do not need to be declared in advance. However, resizing an array in a loop needs time, due to the allocation of memory in each iteration. In the above example, `zeros(1000,1000)` reserves all needed memory at once (before the loop).



# Graphics in 2 dimensions

■ **ezplot** ••• displays the graph of function

**ezplot('f(x)')**       $\longrightarrow$     Plot  $f(x)$  in  $-2\pi \leq x \leq 2\pi$

**ezplot('f(x)',[a,b])**       $\longrightarrow$     Plot  $f(x)$  in  $a \leq x \leq b$

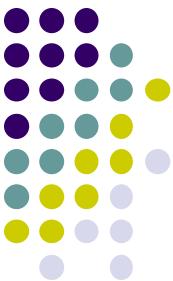
**ezplot('f(x,y)')**       $\longrightarrow$     Plot  $f(x,y)=0$  in  $-2\pi \leq x, y \leq 2\pi$

**ezplot('f(x,y)',[a,b])**     $\longrightarrow$     Plot  $f(x,y)=0$  in  $a \leq x, y \leq b$

**ezplot('f(x,y)',[a,b,c,d])**     $\longrightarrow$     Plot  $f(x,y)=0$  in  $a \leq x \leq b, c \leq y \leq d$

**ezplot('x(t), y(t)')**       $\longrightarrow$     Plot  $x=x(t), y=y(t)$  in  $0 \leq t \leq 2\pi$

**ezplot('x(t), y(t)',[a,b])**     $\rightarrow$     Plot  $x=x(t), y=y(t)$  in  $a \leq t \leq b$

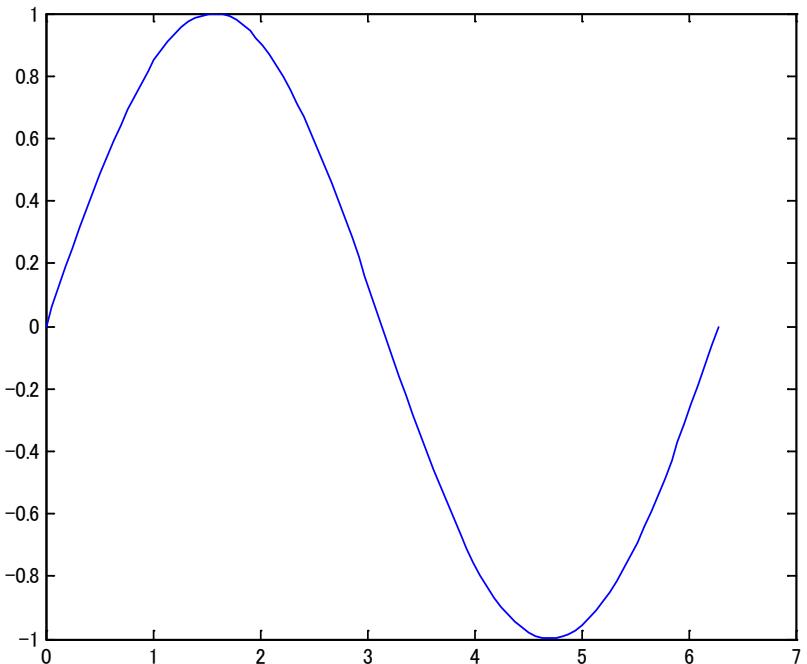


## ■ plot ••• 2D plot with vector data

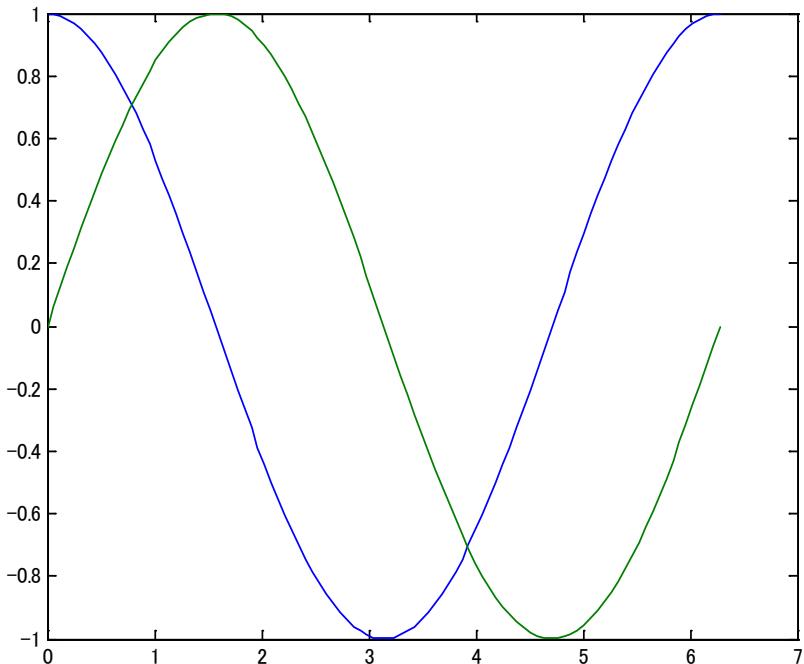
plot(x,y) plots vector x versus vector y

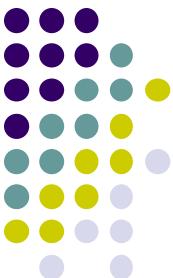
Example:

```
>> x=linspace(0, 2*pi, 100);  
>> plot(x, sin(x));
```



```
>> x=linspace(0, 2*pi, 100);  
>> plot(x, sin(x), x, cos(x));
```





When `ezplot` or `plot` is executed,  
the graph is displayed in a window named “Figure 1”.

→ Another call of `ezplot/plot` erases the graph.

```
>> x=linspace(0, 2*pi, 100);  
>> figure(1); plot(x, sin(x));  
>> figure(2); plot(x, cos(x));
```



In this way, two graphs  
are displayed in Figure 1  
and Figure 2 respectively.

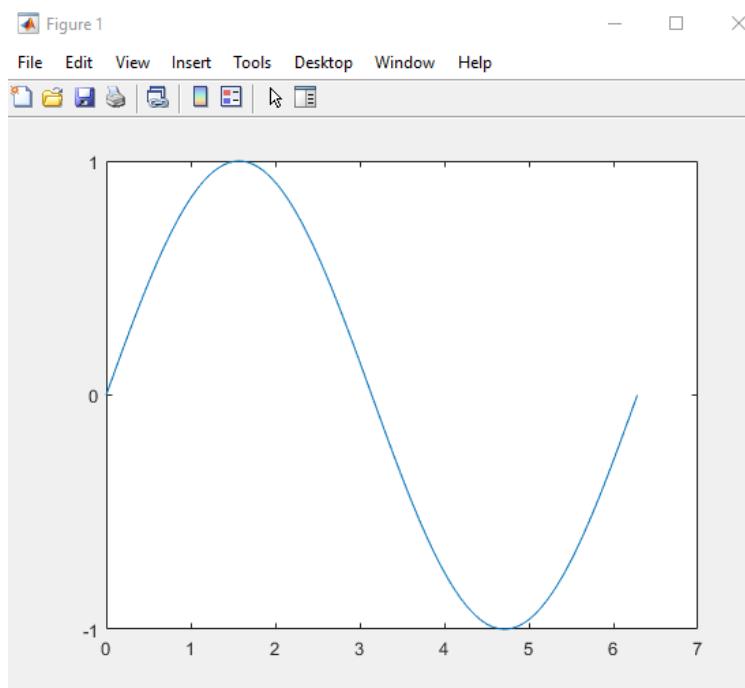


Figure 1

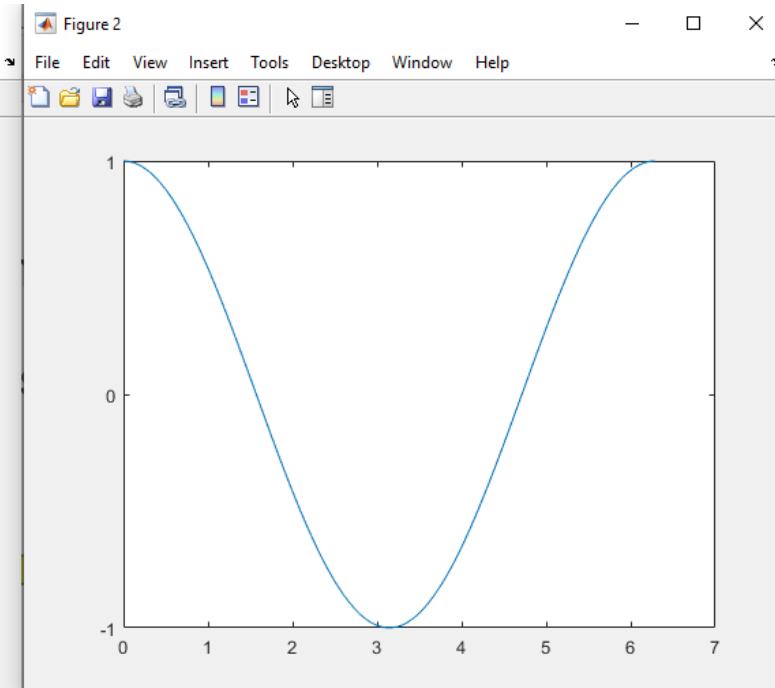


Figure 2

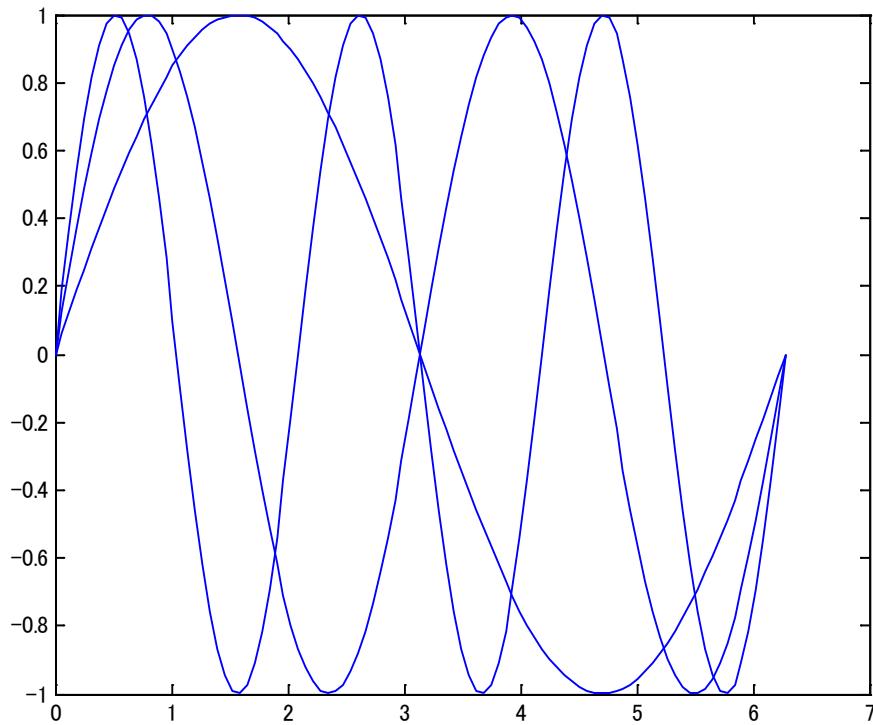


The command “hold on” is useful when you want to display some graphs in the same Figure.  
(The command “hold off” is used to quit it.)

```
>> x=linspace(0, 2*pi, 100);
>> plot(x,sin(x));
>> hold on;
>> plot(x,sin(2*x));
>> plot(x,sin(3*x));
```



The graph of  
 $\sin(x)$ ,  $\sin(2x)$ ,  $\sin(3x)$   
are displayed in one Figure.



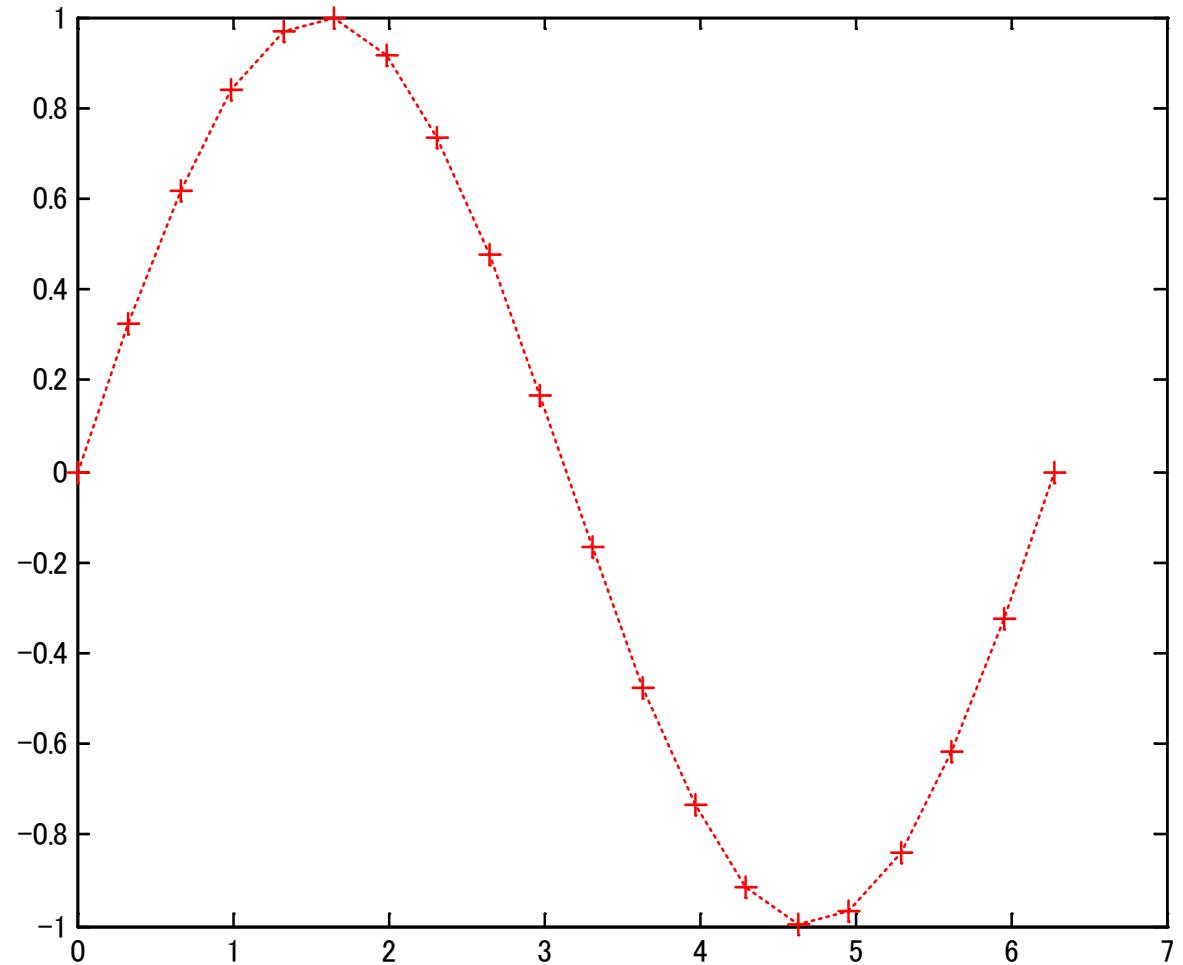
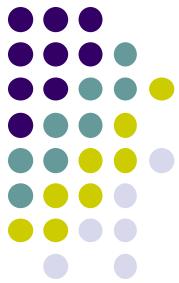


## Some options in graphics

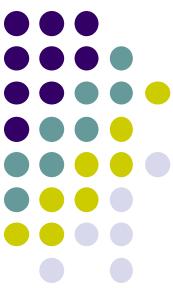
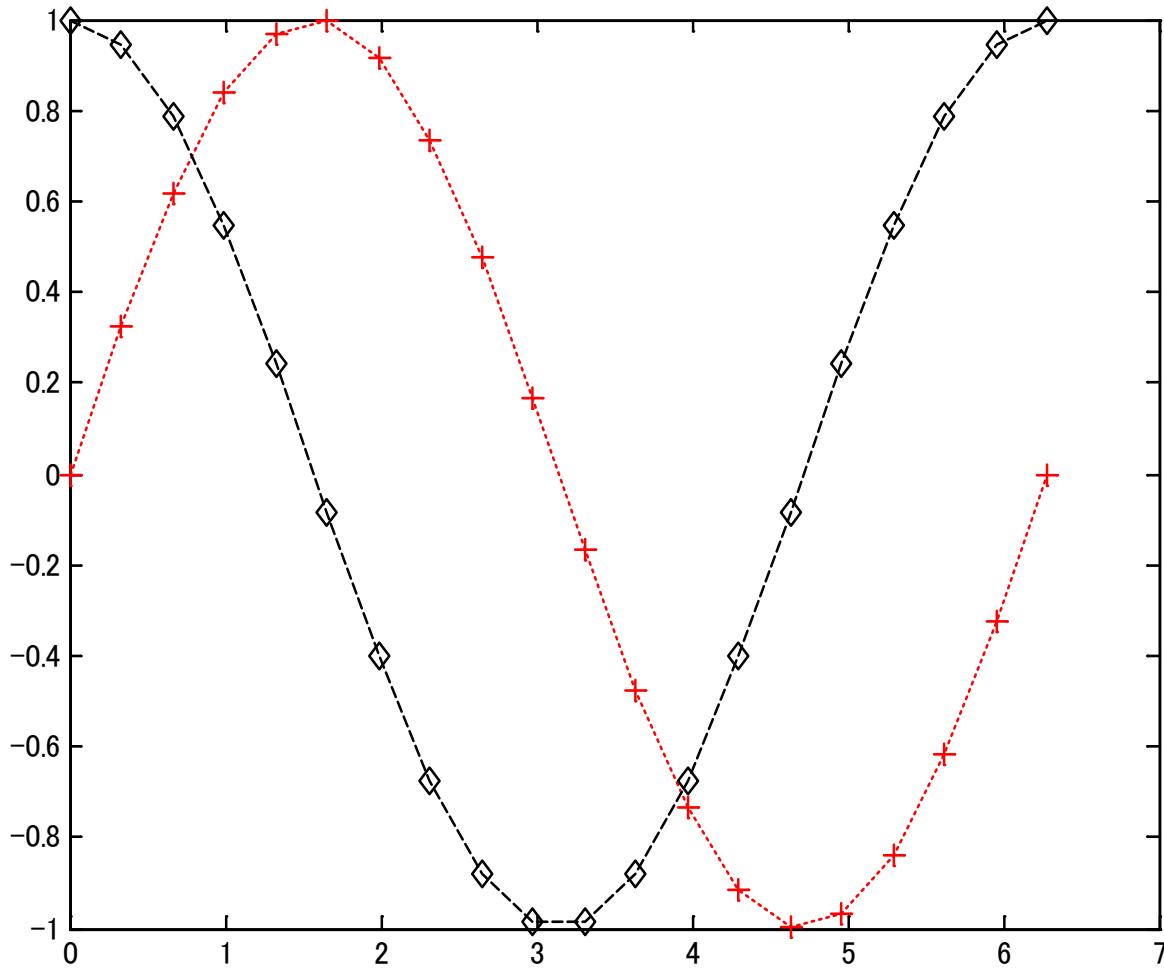
`plot(x, y, 's')` ··· 's' is specified as follows:

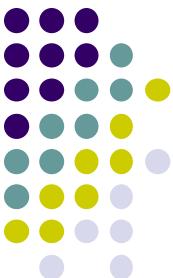
b	blue	.	dot	-	solid line
g	green	o	circle shape	:	dotted line
r	red	x	x shape	-.	chained line
c	cyan	+	plus shape	--	broken line
m	magenta	*	star shape	(none)	no line
y	yellow	s	square		
w	white	d	diamond shape		
k	black	v	triangle		
		^	inverted triangle		
		<	triangle (left direction)		
		>	triangle (right direction)		
		p	pentagon		
		h	hexagon		

```
>> x=linspace(0, 2*pi, 20);
>> plot(x, sin(x), 'r+:'')
```



```
>> x=linspace(0, 2*pi, 20);
>> plot(x, sin(x), 'r+:', x, cos(x), 'kd--')
```

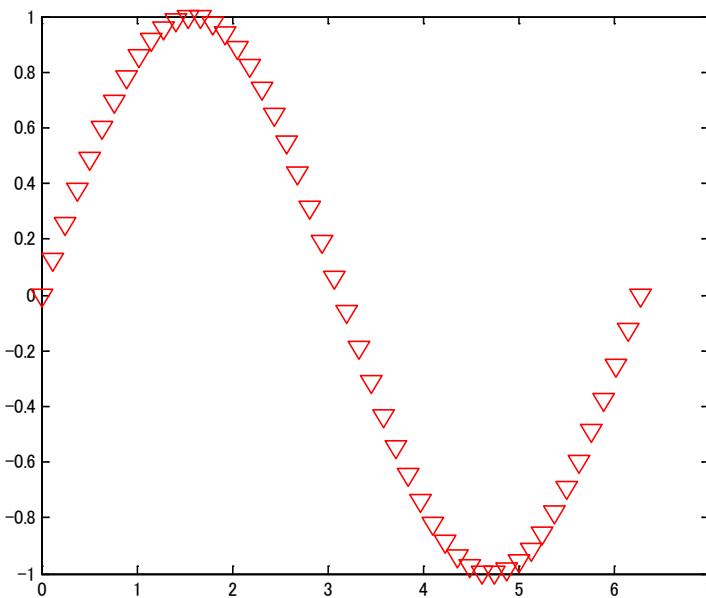
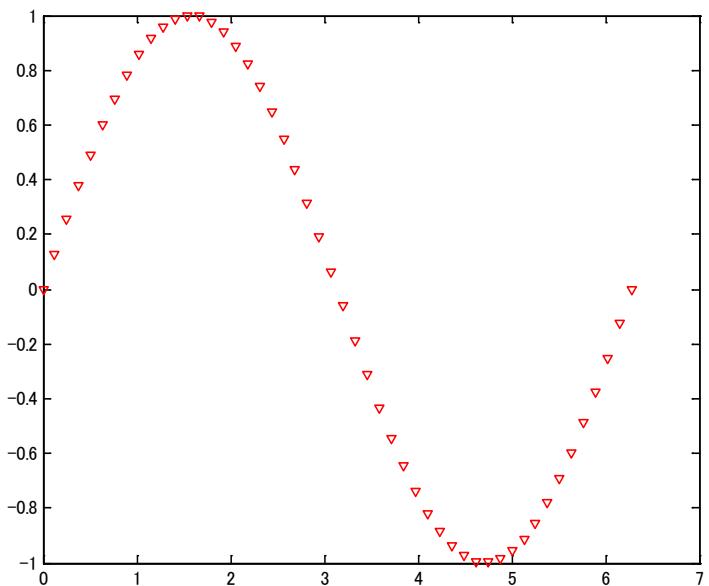




The size of marker can be specified by “MarkerSize”:

```
>> x=linspace(0, 2*pi, 50);  
>> plot(x, sin(x), 'rv', 'MarkerSize', 5)
```

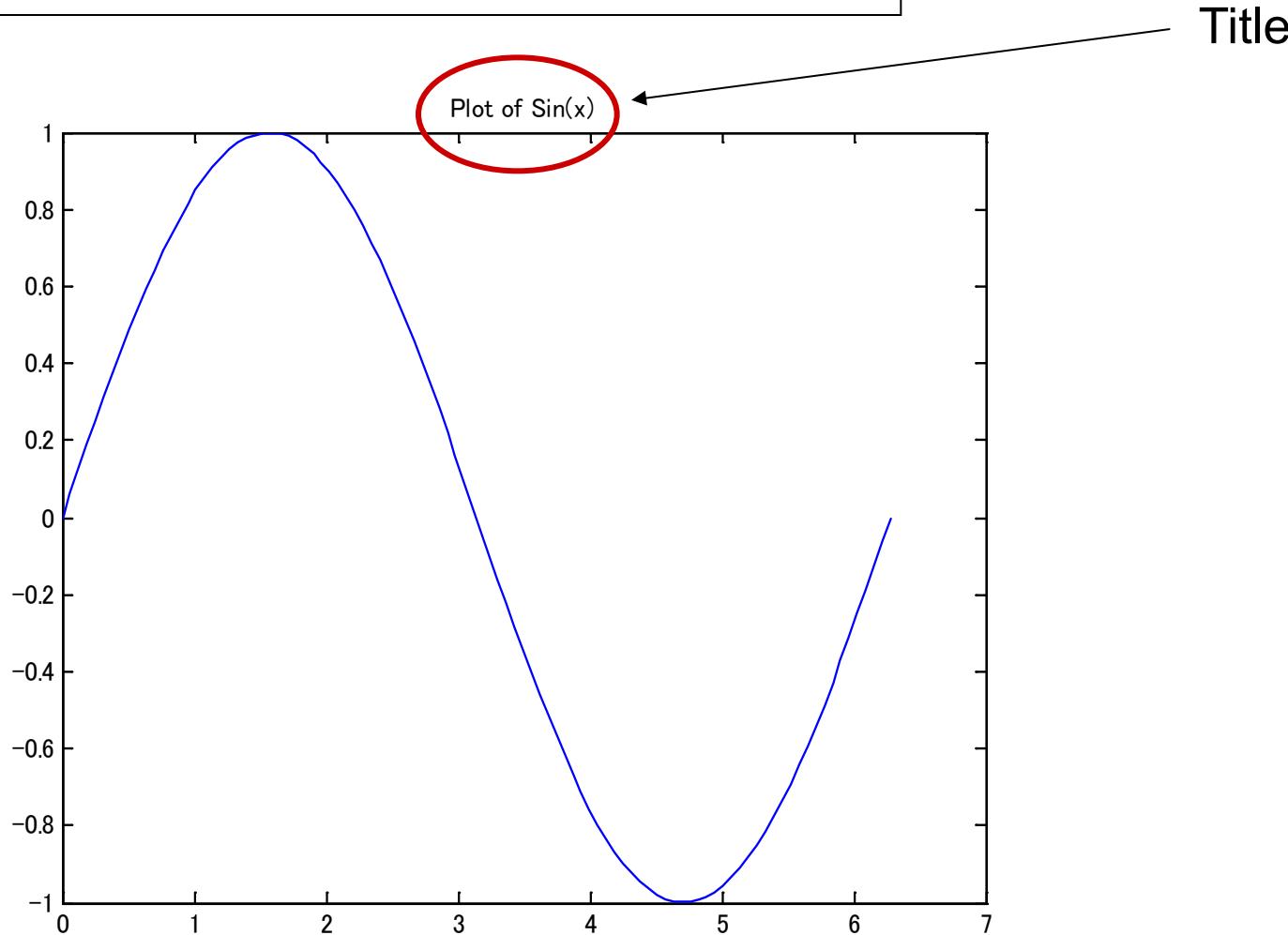
```
>> x=linspace(0, 2*pi, 50);  
>> plot(x, sin(x), 'rv', 'MarkerSize', 10)
```

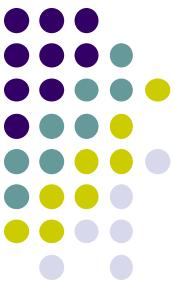




The title can be added by `title('text')` in upper part of the graph:

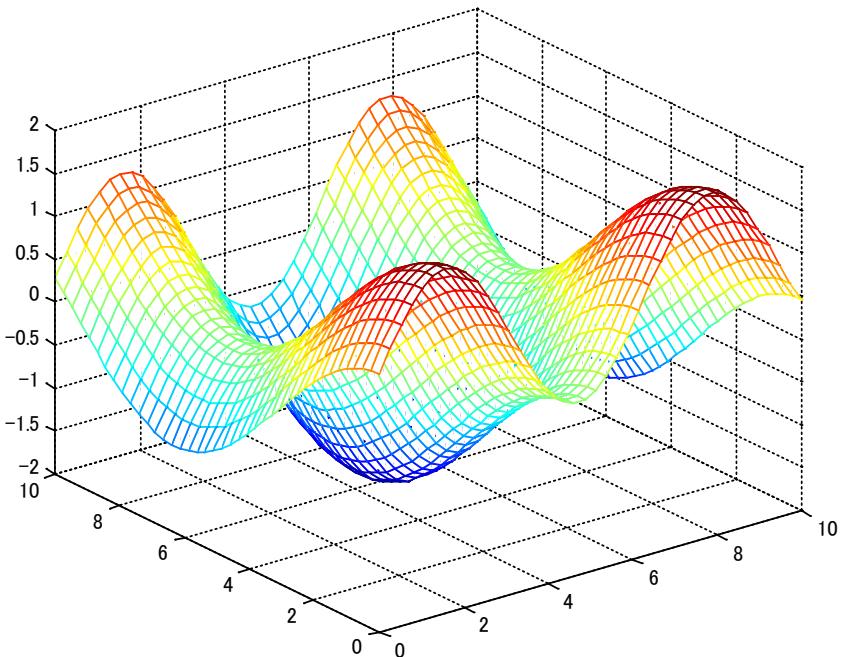
```
>> x=linspace(0, 2*pi, 100);  
>> plot(x, sin(x)); title('Plot of Sin(x)');
```





# Graphics in 3 dimensions

```
>> t=linspace(0,10,40);
>> [x,y]=meshgrid(t,t);
>> z=sin(x)+cos(y/2);
>> mesh(x,y,z);
```



Additionally ezplot3 etc.  
(Check via “help ezplot3”.)



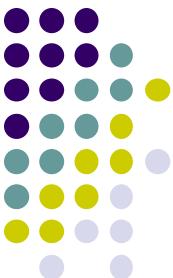
# Making use of pause command

When pause command appears in MATLAB script, the execution is interrupted there, and Return key is used to go on. Moreover, e.g. pause(2) can interrupt the execution for two seconds.

## Example :

```
%A demo file for graphics  
  
x=linspace(0,2*pi,100);  
for a=10:-1:1  
    plot(x, a*sin(x));  
    grid on; ←  
    hold on;  
    pause(2);  
end
```

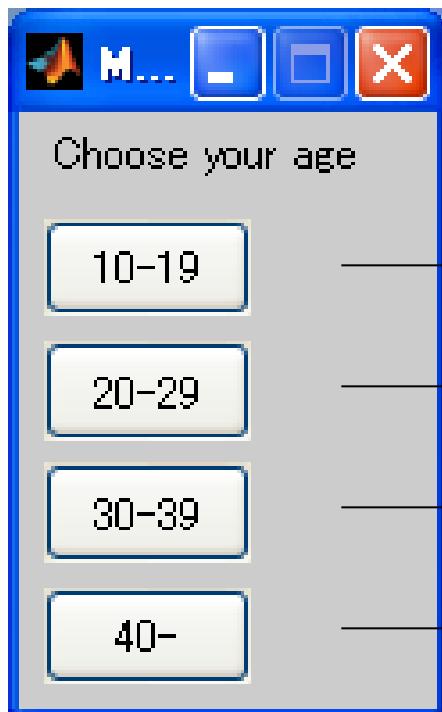
to display the grid lines in the graph



# Interface with the screen

## ■ “menu” function

```
>> item=menu('Choose your age', '10-19', '20-29', '30-39', '40-')
```



header

Items of menu

click

10-19

item = 1

20-29

item = 2

30-39

item = 3

40-

item = 4

If none of menus was chosen then item = 0.



For example, if you want to choose some formulae to calculate definite integrals for function “f”, then you can make the following script after preparing for each functions such as “midpoint”.

```
method=menu('Choose integration formula', ...
'midpoint formula', 'trapezoid formula', 'Simpson formula')

if method==1      S=midpoint(f)    % Calculation by midpoint formula
elseif method==2 S=trape(f)        % Calculation by trapezoid formula
elseif method==3 S=simpson(f)     % calculation by Simpson formula
end
```



click → **method = 1**  
→ **method = 2**  
→ **method = 3**



## ■ input function

Using “input” function you can input some value when you execute a MATLAB script.

**Example:**

```
>> N=input('The number of data ?')
```

If you execute above then you have the following question

The number of data ?

So if you enter 100 via keyboard then

```
N =  
100
```

appears and N is set as 100. Here N could be a vector or matrix.



## Variables: Save and Load

To save variables you can use

```
>> save filename
```

A file named *filename.mat* is created which saves all variables from workspace. Using

```
>> load filename
```

variables in *filename.mat* are imported into workspace

\*If you omit the file name after “save” command then all variables are saved in a file named *matlab.mat* („load“ restores data from *matlab.mat*)

Alternatively you can use the buttons „Save“ and „Import data“ on top of workspace window

e.g.

```
>> whos
```

Name	Size	Bytes	Class
A	100x100	80000	double array
B	200x200	320000	double array
fid	1x1	8	double array
i	1x1	8	double array
j	1x1	8	double array
z	1x1	16	double array (complex)

Grand total is 50004 elements using 400040 bytes

```
>> save variable
```

```
>> clear
```

```
>> whos
```

```
>> load variable
```

```
>> whos
```

Name	Size	Bytes	Class
A	100x100	80000	double array
B	200x200	320000	double array
fid	1x1	8	double array
i	1x1	8	double array
j	1x1	8	double array
z	1x1	16	double array (complex)

Grand total is 50004 elements using 400040 bytes





## Beispiel 1: Integralrechnung

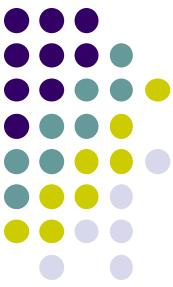
$$f : [a, b] \rightarrow \mathbb{R}$$

Eine Zerlegung  $P : a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$

$$\underline{f}_k := \min_{x \in [x_{k-1}, x_k]} f(x), \quad \bar{f}_k := \max_{x \in [x_{k-1}, x_k]} f(x)$$

Riemann-Darboux'sche Unter- bzw. Obersumme

$$\underline{S}_P := \sum_{k=1}^n \underline{f}_k \cdot (x_k - x_{k-1}), \quad \bar{S}_P := \sum_{k=1}^n \bar{f}_k \cdot (x_k - x_{k-1})$$



$$\delta_P := \max_{k=1,\dots,n} (x_k - x_{k-1})$$

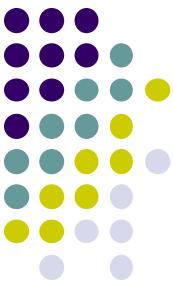
$$\lim_{\delta_P \rightarrow 0} \underline{S}_P = \lim_{\delta_P \rightarrow 0} \bar{S}_P \quad \Rightarrow \quad \int_a^b f(x) dx := \lim_{\delta_P \rightarrow 0} \underline{S}_P = \lim_{\delta_P \rightarrow 0} \bar{S}_P$$

$f$  ist Riemann-integrierbar auf  $[a,b]$

## Mittelwertsatz der Integralrechnung

Zu einer stetigen Funktion  $f : [a,b] \rightarrow \mathbb{R}$  gibt es ein  $z \in [a,b]$  mit

$$\int_a^b f(x) dx = f(z)(b-a)$$



# 1. Hauptsatz der Differenzial - und Integralrechnung

Die Funktion  $F : [a, b] \rightarrow \mathbb{R}$  mit

$$F(x) := \int_a^x f(t) dt$$

zu einer stetigen Funktion  $f : [a, b] \rightarrow \mathbb{R}$  ist differenzierbar auf  $(a, b)$  und es gilt

$$F'(x) = f(x) \quad \text{für } x \in (a, b).$$



## Definition der Stammfunktion

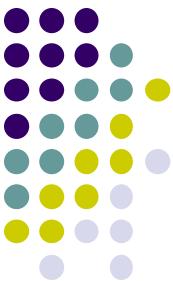
Sei  $f$  eine auf einem offenen Intervall  $(a, b)$  definierte Funktion.

Jede differenzierbare Funktion  $F : (a, b) \rightarrow \mathbb{R}$  mit  $F'(x) = f(x)$   
heißt Stammfunktion von  $f$ .

Schreibweise:  $F(x) = \int f(x) dx$

$$\int (f(x) + g(x)) dx = \int f(x) dx + \int g(x) dx$$

$$\int cf(x) dx = c \int f(x) dx \quad \text{für alle } c \in \mathbb{C}$$

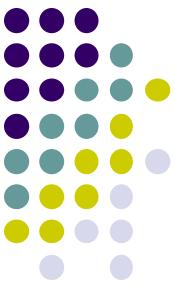


## 2. Hauptsatz der Differenzial - und Integralrechnung

Wenn  $F : [a, b] \rightarrow \mathbb{R}$  eine stetige und auf  $(a, b)$

stetig differenzierbare Funktion ist mit integrierbarer Ableitung  $F'$ ,  
dann gilt

$$\int_a^b F'(x) dx = F(b) - F(a).$$



# Grundlegende Stammfunktionen

$$\int x^\alpha \, dx = \frac{x^{\alpha+1}}{\alpha+1} \quad \text{für } \alpha \neq -1$$

$$\int \frac{1}{x} \, dx = \ln |x|$$

$$\int e^x \, dx = e^x$$

$$\int \ln x \, dx = x \ln x - x$$

$$\int \cos x \, dx = \sin x, \quad \int \sin x \, dx = -\cos x, \quad \int \tan x \, dx = -\ln |\cos x|$$

$$\int \cosh x \, dx = \sinh x, \quad \int \sinh x \, dx = \cosh x$$

$$\left. \begin{aligned} \cosh(x) &= \frac{1}{2} (e^x + e^{-x}), & \sinh(x) &= \frac{1}{2} (e^x - e^{-x}) \end{aligned} \right\}$$



# Weitere Stammfunktionen

$$\int \frac{1}{1+x^2} dx = \arctan x$$

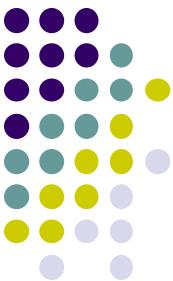
$$\int \frac{1}{1-x^2} dx = \frac{1}{2} \ln \left| \frac{1+x}{1-x} \right|$$

$$\int \frac{1}{\cos^2 x} dx = \tan x$$

$$\int \frac{1}{\sqrt{1-x^2}} dx = \arcsin x$$

$$\int a^x dx = \frac{a^x}{\ln a}$$

...



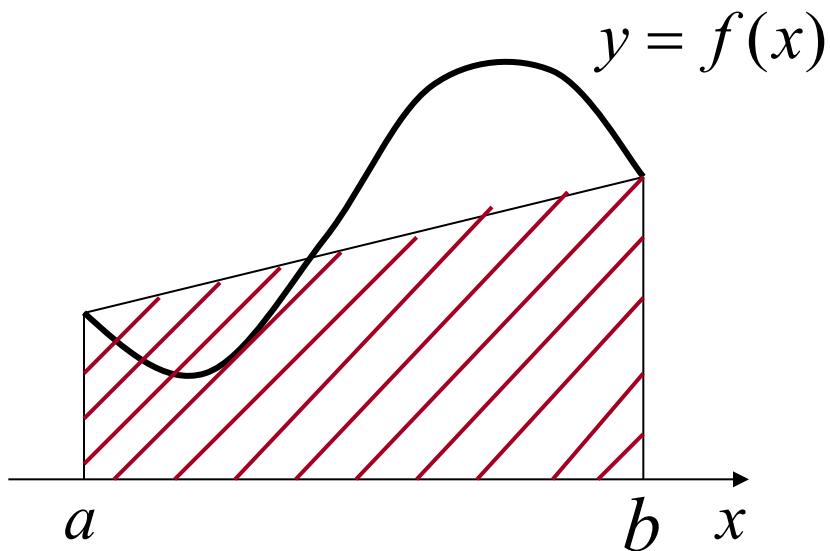
# Approximationsformel und Fehler

Sei  $f : [a, b] \rightarrow \mathbb{R}$  eine stetige Funktion.

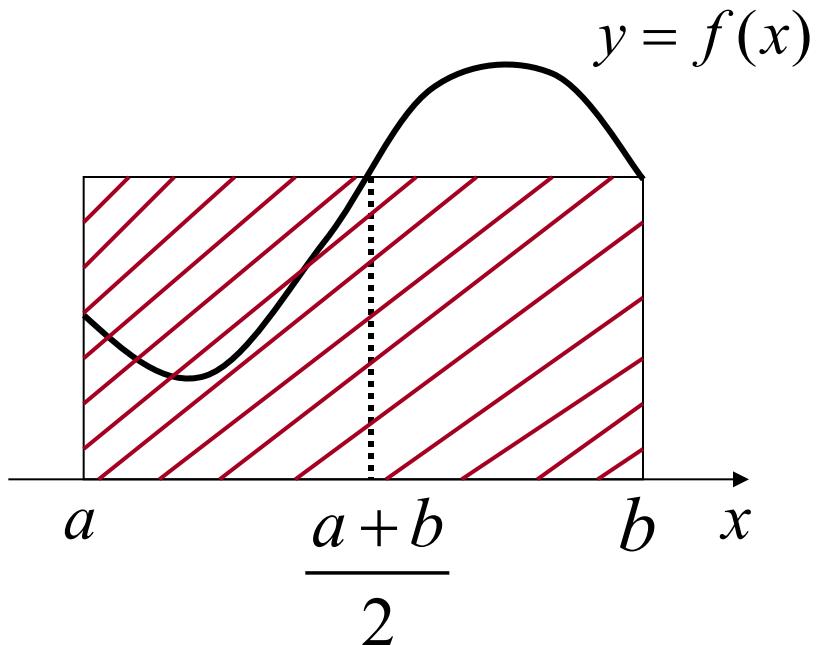
$\int_a^b f(x) dx$  wird wie folgt approximiert:

$$\frac{b-a}{2} \{f(a) + f(b)\},$$

$$(b-a)f\left(\frac{a+b}{2}\right)$$



Trapezregel



Mittelpunktsregel



## Fehler der Approximationsregeln

Sei  $f : [a, b] \rightarrow \mathbb{R}$  eine zweimal stetig differenzierbare Funktion

$$M = \max_{a \leq x \leq b} |f''(x)|$$

### Fehler der Trapezregel

$$\left| \int_a^b f(x) dx - \frac{b-a}{2} \{f(a) + f(b)\} \right| \leq \frac{(b-a)^3}{12} M$$

### Fehler der Mittelpunktsregel

$$\left| \int_a^b f(x) dx - (b-a) f\left(\frac{a+b}{2}\right) \right| \leq \frac{(b-a)^3}{24} M$$



Fehler der Trapezregel : Fehler der Mittelpunktsregel = 2:1

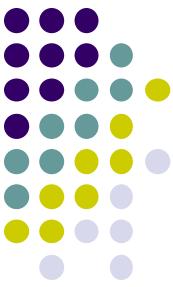
[Trapezregel]                            [Mittelpunktsregel]

$$\rightarrow \frac{1}{3} \cdot \frac{b-a}{2} \left\{ f(a) + f(b) \right\} + \frac{2}{3} \cdot (b-a) f\left(\frac{a+b}{2}\right)$$
$$= \frac{b-a}{6} \left\{ f(a) + f(b) + 4 \cdot f\left(\frac{a+b}{2}\right) \right\}$$

**Simpson-Regel**

## Fehler der Simpson-regel

$$\left| \int_a^b f(x) dx - \frac{b-a}{6} \left\{ f(a) + f(b) + 4 \cdot f\left(\frac{a+b}{2}\right) \right\} \right| \leq \frac{(b-a)^3}{36} M$$



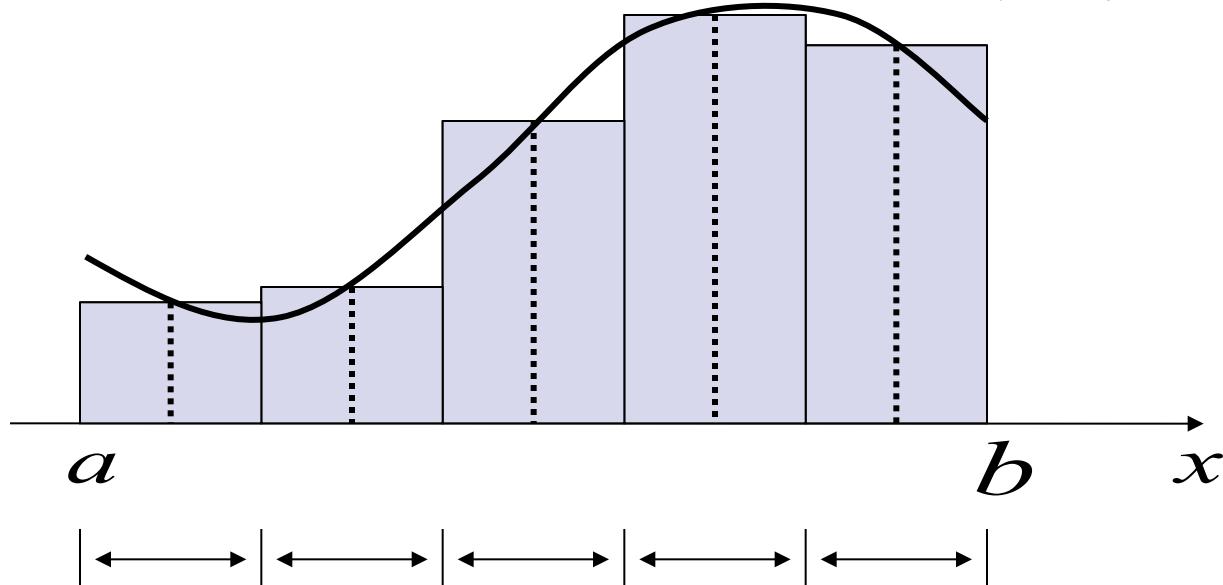
Eine Zerlegung  $P: a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$

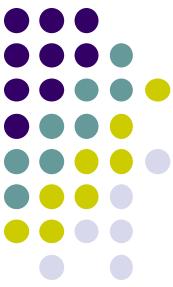
→ Anwendung jeder Regeln auf  $[x_k, x_{k+1}]$

$n$ : Anzahl der Zerlegungsintervalle

Beispiel:  $n = 5$

$$y = f(x)$$





Eine gleichmäßige Zerlegung  $P : a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$

$$\frac{b-a}{n} = x_k - x_{k-1} \quad (k=1, \dots, n)$$

### 【Trapezregel】

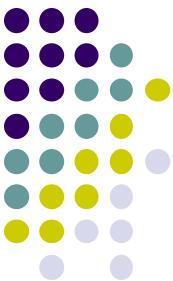
$$\int_a^b f(x) dx \approx \frac{b-a}{2n} \left\{ f(a) + f(b) + 2 \sum_{k=1}^{n-1} f\left(a + \frac{k}{n}(b-a)\right) \right\}$$

### 【Mittelpunktsregel】

$$\int_a^b f(x) dx \approx \frac{b-a}{n} \sum_{k=1}^n f\left(a + \frac{2k-1}{2n}(b-a)\right)$$

### 【Simpson-Regel】

$$\int_a^b f(x) dx \approx \frac{b-a}{6n} \left\{ f(a) + f(b) + 2 \sum_{k=1}^{n-1} f\left(a + \frac{k}{n}(b-a)\right) + 4 \sum_{k=1}^n f\left(a + \frac{2k-1}{2n}(b-a)\right) \right\}$$



**Beispiel:**  $\int_0^1 \frac{1}{1+x^2} dx$

trapez.m

```
n=8  
a=0; b=1;  
  
sum=0;  
for k=1:n-1  
    sum=sum+f(a+(k/n)*(b-a));  
end  
  
I_trapez = ((b-a)/(2*n))*(f(a)+f(b)+2*sum)
```

f.m

```
function [a] = f(x)  
a=1/(1+x^2);  
end
```

$$\frac{b-a}{2n} \left\{ f(a) + f(b) + 2 \sum_{k=1}^{n-1} f\left(a + \frac{k}{n}(b-a)\right) \right\}$$

sum



## Beispiel 2: Nullstellenbestimmung

Nichtlineare Gleichung  $f(x) = 0$

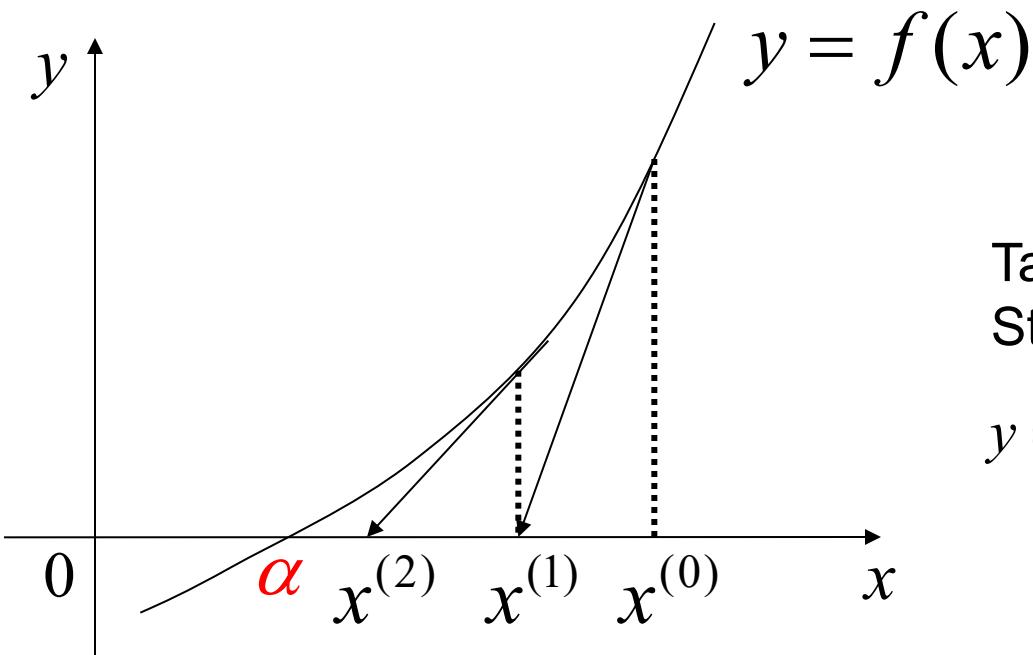
Anfangswert  $x^{(0)} \rightarrow x^{(1)} \rightarrow x^{(2)} \rightarrow \dots \rightarrow x^{(N)} \approx \alpha$

Die Strategie: Eine Nullstelle wird durch eine Folge von linearen Approximationen der Funktion bestimmt.



# Newton Verfahren

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$$

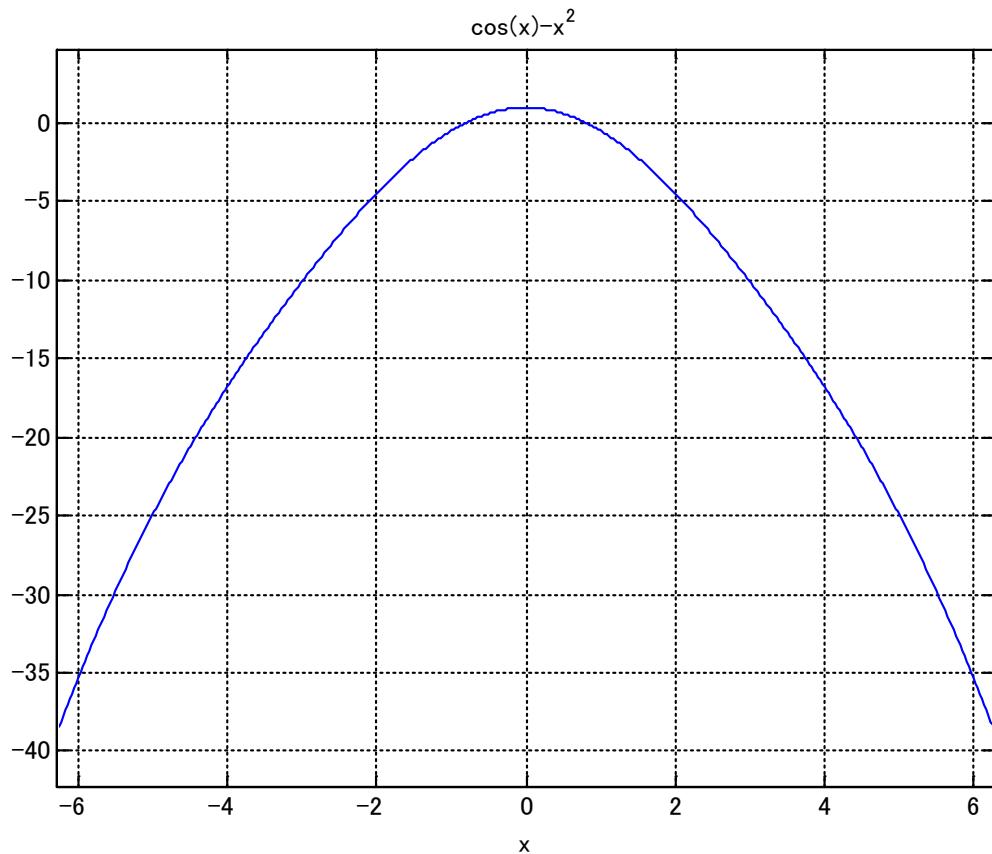


Tangente Gerade an der Stelle  $(x^{(n)}, f(x^{(n)}))$ :

$$y = f'(x^{(n)})(x - x^{(n)}) + f(x^{(n)})$$

# Beispiel

$$\cos(x) - x^2 = 0$$



```
>> ezplot('cos(x)-x^2'); grid on
```

```
function [a] = f(x)
a=cos(x)-x^2;
end
```

f.m

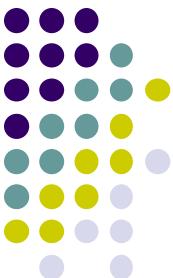
```
function [b] = df(x)
b=-sin(x)-2*x;
end
```

df.m

## newton.m



```
x0=0.5
IRMAX=30
eps=1.0E-6
for k=1:IRMAX
    f0=f(x0);
    df0=df(x0);
    if df0~=0
        x1=x0-f0/df0
    else
        error('df=0')
    end
    if(abs(x1-x0)<eps*abs(x0))
        disp('Converged:')
        disp(x1)
        break
    end
    x0=x1
end
if k==IRMAX
    error('failed to converge')
end
```



# Aufgaben

3.1. Für  $x = 192119201$ ,  $y = 35675640$  rechnen Sie

$$z = \frac{1682xy^4 + 3x^3 + 29xy^2 - 2x^5 + 832}{107751}$$

mit Matlab.

3.2. Berechnen Sie folgende Ausdrücke:

1)  $a = 1 + 2i$ ,  $b = 2 + 3i$ ,  $a + b$

2)  $| -2 + 3i |$

3)  $\cos^2(\pi / 3)$

4)  $a = 2$ ,  $b = 3$ ,  $\sqrt{a^2 + b^3}$



3.3. Berechnen Sie folgende Ausdrücke für  $a = \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix}$ ,  $b = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix}$

1)  $a + b$

2)  $a \cdot b$

3)  $|a|_2 + |b|_2$

4)  $a \times b$

$$\left. \begin{array}{l} a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \\ |a|_2 = \sqrt{a_1^2 + a_2^2 + a_3^2}, a \times b = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix} \end{array} \right\}$$

3.4 Bestimmen Sie eine Nullstelle von

1)  $\sin(x) - x^3 + 1$

2)  $e^x + \cos(x)$