

MATLAB Introduction

- What is MATLAB
 - Matrix calculations (MATrix LAB)
 - Powerful scientific calculator
 - Programming language
 - Data analysis tool
 - Dynamic system modelling and simulation
 - Control of dynamic systems
 - Plotting device

MATLAB Introduction

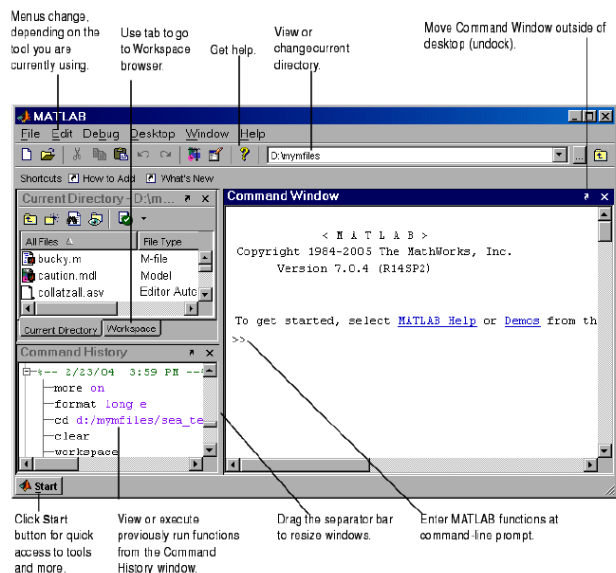
- Start MATLAB through the desktop solution
(desktop.hials.no)
 - Install the VMware Horizon View Client
 - VMware Horizon View HTML access
- Getting to know MATLAB and the MATLAB GUI
- Storing files in separate folders

MATLAB Basics

- Useful commands
 - **doc** bring up the built-in documentation
 - **help** <cmd> for help with a command (e.g. **help sqrt**)
 - **clc** clears the command window
 - **clear** remove all variables from the workspace
 - **close** closes the current figure
 - **pwd** presents working directory
 - **dir** or **ls** lists the current directory
 - **what** lists the MATLAB specific files
 - **cd** changes current directory
 - **path** or **matlabpath** lists the MATLAB search path
 - **addpath** adds a directory to the search path

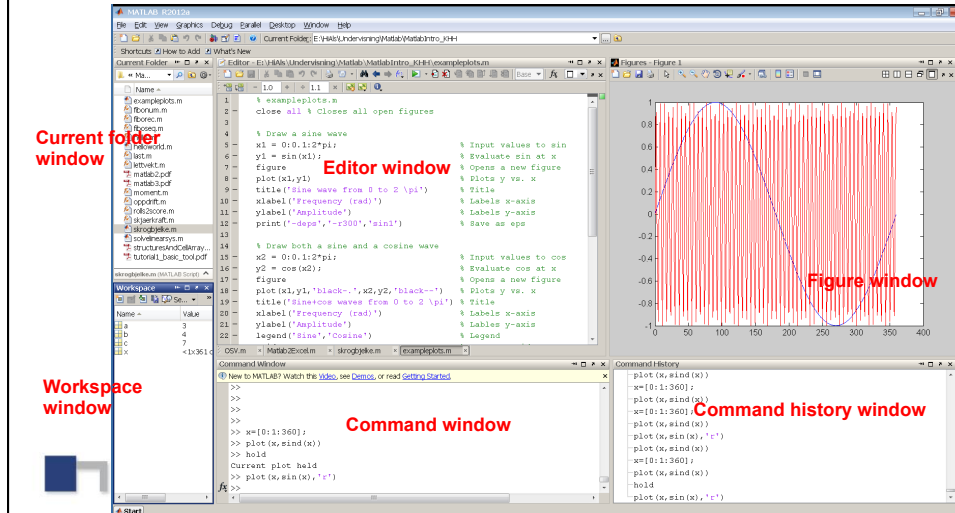
MATLAB Basics

- The GUI can be customized in various ways
- The most important windows are
 - Command window
 - Editor window
 - Figure window



MATLAB Basics

- The GUI can be customized in various ways



MATLAB Basics – Pseudo-code and syntax

- Pseudo code: average age of the class
 - Collect data from all students
 - Sum all the ages
 - Divide by the number of studentss
- Which “Syntax” is it?
- Do you approve this syntax?

<ul style="list-style-type: none"> Samle studenter inn fra alle data Alle summere aldrane Studenter på antal dele 	<ul style="list-style-type: none"> Samle inn data fra alle studenter Summere alle aldrane Dele på antal studenter
--	--

MATLAB Basics – Pseudo-code and syntax

- Are you OK with this syntax?
 - Coletar a idade de todos os estudantes
 - Somar todos os valores
 - Dividir pelo numero de estudantes
- Another language, another syntax!

MATLAB Basics – Algorithm, program and code

- **Algorithm:**
A set of instructions or procedures for solving a problem
- **Program:**
The set of instructions within a computer which enables it to perform the various tasks required
- **Code:**
Any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text.

Human code vs MATLAB code

- Collect data from all students
`all_nam = {'aa','bb','cc'}`
`all_age = {21, 23, 27}`
- Sum all the ages
`sum_age = sum(all_age)`
- Divide by the number of students
`num_stud = length(all_age)`
`average = sum_age/num_stud`

MATLAB Basics – Variables

- Scalar Variables: `>> a = 3; b=4;`
- Arithmetic operations: `>> c = a+b; d= c*b;`
`>> sin(2*pi)+exp(-3/2)`
`>> sind(90)`
- Exponentiation: `>> 4^2;`
`>> (3+4*a)^2`
- Complicated expressions: `>> ((2+3)*b)^0.1`
- Multiplication is **not** implicit given by parantheses `>> 3(1+0.7) → gives error`
- Clear command window `>> clc`
- There is a huge number of built-in functions, the list is too long to include here, but note that `atan(x)` and `log10(x)` corresponds to $\tan^{-1}(x)$ and $\log_{10}(x)$, respectively.

MATLAB Basics – Variables

- Examples of elementary functions:

```
>> x = 9;
>> sqrt(x), exp(x), log(sqrt(x)), log10(x^2+6)
ans = 3
ans = 8.1031e+03
ans = 1.0986
ans = 1.9395
>> x=5*cos(pi/6); y = 6*sin(pi/6);
>> acos(x/5), asin(y/6)
ans = 0.5236
ans = 0.5236
>> pi/6
ans = 0.5236
```



MATLAB Basics – Exercise 1

Compute the following expressions

1. Arithmetic operations

$$\frac{2^5}{2^5 - 1} \quad \left(1 - \frac{1}{2^5}\right) \quad \frac{\sqrt{5} - 1}{(\sqrt{5} + 1)^2}$$

(Answers: 1.0323, 1.0323, 0.1180)

2. Exponentials and logarithms

$$e^3 \quad \ln(e^3) \quad \log_{10}(e^3) \quad \log_{10}(10^5)$$

(Answers: 20.0855, 3, 1.3029, 5)

3. Trigonometric operations

$$\sin\left(\frac{\pi}{6}\right) \quad \cos(\pi) \quad \tan\left(\frac{\pi}{2}\right) \quad \sin^2\left(\frac{\pi}{6}\right) + \cos^2\left(\frac{\pi}{6}\right)$$

(Answers: 0.5, -1, 1.6331E16, 1)



MATLAB Basics

- Arrays

1. Matrix of numbers (double or complex)
2. Cell array of objects (more advanced structures)

**MATLAB makes vectors easy!
That's its power!**



MATLAB Basics – Vectors

- Row vectors

- Variables inserted between brackets and separated by commas or spaces

```
>> row = [1 2 5.4 -6.6]
>> row = [1, 2, 5.4, -6.6];
```

- Command window:

```
>> row = [1 2 5.4 -6.6]

row =
    1.0000    2.0000    5.4000   -6.6000
```

MATLAB Basics – Vectors

- Column vectors

- Variables inserted between brackets and separated by semicolons

```
>> column = [4;2;7;4]
```

- Command window:

```
>> column = [4;2;7;4]
```

```
column =  
      4  
      2  
      7  
      4
```



MATLAB Basics – Vectors

- You can tell the difference between a row and a column vector by:

- Looking in the workspace
- Displaying the variable in the command window
- Using the size function

```
>> size(row)           >> size(column)
```

```
ans =  
     1     4           ans =  
     4     1
```

- To get a vector's length, use the length function

```
>> length(row)         >> length(column)
```

```
ans =  
     4                 ans =  
     4
```



MATLAB Basics – Vectors

- The transpose operator: `'`

```
>> x = [1 2 3];      >> y = [4; 5; 6];
>> x'              >> y'
ans =              ans =
     1             4     5     6
     2
     3
```

- Vectors can be created more effectively using the `'colon'` operator or the `'linspace'` function

```
>> z = 8:1:10
z =
     8     9    10

>> v = linspace(0,10,5)
v =
     0   2.5000   5.0000   7.5000  10.0000
```



MATLAB Basics – Vectors

- Addition and subtraction of vectors

```
>> x = [1 2 3];      >> y = [4; 5; 6];
>> x + 2            >> x + y
ans =              ans =
     3     4     5         5     7     9
```

- Multiplication of vector elements

```
>> x*y
??? Error using → mtimes
Inner matrix dimensions must agree
```

Here we break the rules for matrix multiplications.
If we meant to multiply the two vectors element by element,
i.e. $x(1)*y(1)$, we must use the special Matlab syntax `','.*'`

```
>> x.*y            >> y.^2
ans =              ans =
     4    10    18         16    25    36
```



MATLAB Basics – Vectors

- The dot product:

```
>> a = [1 2 3]; b = [4 5 6];
>> c = dot(a,b)
c =
    32
```

The dot product of two vectors $\mathbf{a} = [a_1, a_2, \dots, a_n]$ and $\mathbf{b} = [b_1, b_2, \dots, b_n]$ is defined as:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

- The cross product

```
>> d = cross(a,b)
d =
   -3    6   -3
```

The definition of the cross product can be represented as:

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$

This determinant can be computed as

$$\mathbf{u} \times \mathbf{v} = (u_2 v_3 - u_3 v_2) \mathbf{i} + (u_3 v_1 - u_1 v_3) \mathbf{j} + (u_1 v_2 - u_2 v_1) \mathbf{k}$$



MATLAB Basics – Matrices

- Matrices

- You can make matrices like vectors, element by element

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

- Or by concatenating vectors and matrices
(note that size matters!)

```
>> a = [1 2];
>> b = [3 4];
>> c = [5;6];

>> d = [a;b];
>> e = [d c];
>> f = [[e e];[a b a]];

```



MATLAB Basics – Matrices

Matrix multiplication between:

- A **matrix** and a **real number**

$$\lambda = 2, \quad \mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

$$2\mathbf{A} = 2 \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 2 \cdot a & 2 \cdot b \\ 2 \cdot c & 2 \cdot d \end{pmatrix} = \begin{pmatrix} a \cdot 2 & b \cdot 2 \\ c \cdot 2 & d \cdot 2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} 2 = \mathbf{A}2.$$

- A **matrix** and a **vector**

$$\mathbf{AB} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

- **Two matrices**

$$\mathbf{AB} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a \times e + b \times g & a \times f + b \times h \\ c \times e + d \times g & c \times f + d \times h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}.$$



MATLAB Basics – Matrices

- A **matrix** multiplies a **vector**

$$\mathbf{AB} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

$$\begin{bmatrix} m \\ n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} m' \\ n' \end{bmatrix}$$

$$\begin{cases} m = m' \cdot \cos\theta - n' \cdot \sin\theta \\ n = m' \cdot \sin\theta + n' \cdot \cos\theta \end{cases}$$

Practice :

$\theta = \pi/6$, $m'=4$, $n'=2$, calculate the value of $[m,n]$.



MATLAB Basics – Matrix operations

- Matrix operations

- Matrix multiplication operator `*`

`a*b`

- Element-by-element multiplication can be carried out by using the dot together with the multiplication

`a.*b`

- The element by element operation can also be used together with division and exponentiation

COMMAND	DESCRIPTION
<code>.*</code>	Element-by-element multiplication
<code>./</code>	Element-by-element division
<code>.^</code>	Element-by-element exponentiation

```
>> a = [2 3; 4 5]
```

```
ans =
     2     3
     4     5
```

```
>> b = [4 7; 9 6]
```

```
ans =
     4     7
     9     6
```

```
>> a*b
```

```
ans =
    35    32
    29    41
```

```
>> a.*b
```

```
ans =
     8    21
    45     6
```

MATLAB Basics – Matrix operations

- You can access individual elements, entire rows and columns, and subsets of matrices

```
>> w = [1 2 3 4; 5 6 7 8;
        9 10 11 12]
```

```
ans =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

```
>> w(1,1)
```

```
ans =
     1
```

```
>> w(3,:)
ans =
     9    10    11    12
```

```
>> v = w(1:2,2:3)
```

```
v =
     2     3
     6     7
```

```
>> w(2,4)=13
```

```
w =
     1     2     3     4
     5     6     7    13
     9    10    11    12
```

MATLAB Basics – Matrices

Matrix multiplication example:

- Beef pies cost \$3 each
- Chicken pies cost \$4 each
- Vegetable pies cost \$2 each

They are sold in 4 days:

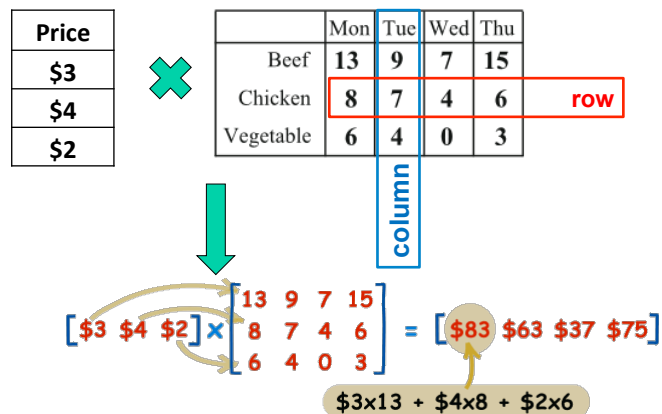
	Mon	Tue	Wed	Thu
Beef	13	9	7	15
Chicken	8	7	4	6
Vegetable	6	4	0	3

the **value of sales** for Monday is calculated as:

$$\begin{aligned}
 & \text{Beef pie value} + \text{Chicken pie value} + \text{Vegetable pie value} \\
 & = \$3 \times 13 \quad + \$4 \times 8 \quad + \$2 \times 6 \quad = \$83 \\
 & = (\$3, \$4, \$2) \cdot (13, 8, 6) = \$3 \times 13 + \$4 \times 8 + \$2 \times 6 \quad = \$83
 \end{aligned}$$

Calculate how much sales the shop makes on each day in matrix operations.

MATLAB Basics – Matrices



No. of **columns** of the **1st** matrix = No. of **rows** of the **2nd** matrix
 What if the price vector is placed after the quantity matrix?

MATLAB Basics – Exercise 2

- The following matrix is defined

$$M = \begin{bmatrix} 6 & 9 & 12 & 15 & 18 & 21 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 2 & 1 & 0 & -1 & -2 & -3 \\ -6 & -4 & -2 & 0 & 2 & 4 \end{bmatrix}$$

- Evaluate the following expressions without using MATLAB, then check your results with MATLAB

- `A = M([1,3], [2,4])`
- `B = M(:, [1,4:6])`
- `C = M([2,3], :)`



MATLAB Basics – Strings

- A variable does not need to be a number. We can assign a textstring to a variable

```
>> s1 = 'Hello world!'; % This is a comment
>> s2 = 'My name is Bond, James Bond';
>> s12 = [s1 s2];
>> disp(s12) % displays the s12 string
```

- The output from this will be

```
>> Hello World! My name is Bond, James Bond
```



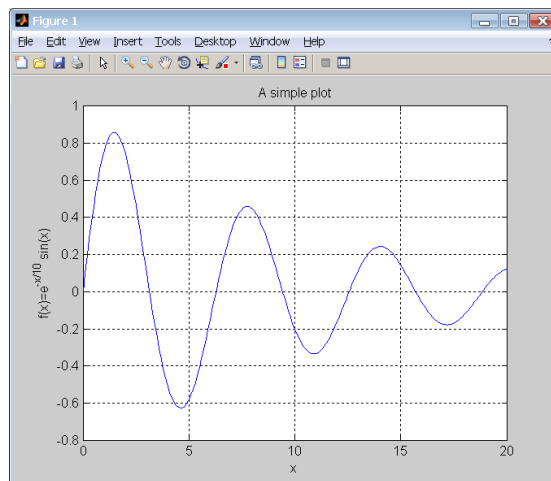
MATLAB Basics – Plotting

- MATLAB is a very powerful tool for producing both 2D and 3D plots. You may create and manipulate the plots interactively or by commands.
- MATLAB can offer a great number of different formats for exporting the plots (e.g. eps, pdf, jpeg)
- The simplest and most commonly used plotting command is `plot(x,y)`, where `x` and `y` are simply vectors containing the `x` and `y` coordinates of a data set
- Example:

```
>> x = 0:0.1:20;  
>> y = exp(-x/10).*sin(x);  
>> plot(x,y), grid on, xlabel('x'),...  
ylabel('f(x) = e^{-x/10} sin(x)'),...  
title('A simple plot')
```

MATLAB Basics – Plotting

- The vectors containing the `x` and `y` data must have the same length
- The plot command can be used to plot multiple sets, e.g.
`plot(x1,y1,x2,y2)`
- The dot-dot-dot (...) notation is used to indicate that the command line is broken into two lines
- 'Grid on' displays the grid in the plot



MATLAB Basics – Plotting

- `xlabel('My x-axis label')`, `ylabel('My y-axis label')`, and the `title('My title')` can be used to label the plot. The labels must be enclosed by single quotes to denote the string format
- `Legend('Data1','Data2')` is used to place a legend and label the data sets when you have multiple data sets in your plot
- You can specify line style and colour within the `plot` command e.g. `plot(x1,y1,'b-',x2,y2,'r-')`. This command would make the first data set a solid blue line, and the second data set a dashed red line.
- Common line styles and colours:

STRING SPECIFIER	LINE STYLE	STRING SPECIFIER	LINE COLOUR
—	Solid line (default)	r	Red
--	Dashed line	g	Green
:	Dotted line	b	Blue (default)
-.	Dash-dot line	w	White
		k	Black

MATLAB Basics – Plotting

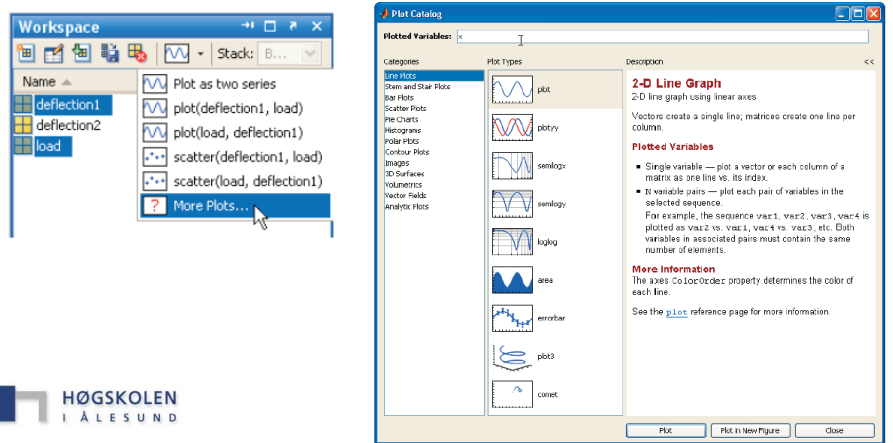
- Plot properties can also be manipulated interactively by clicking the *Show Plot Tools* icon in the Figure Window toolbar



- Useful advice for producing good and informative plots
 - Give your plot an informative title, e.g. `title('Stress vs. strain of steel')`
 - Label your axes and remember to include units where appropriate, e.g. `xlabel('Strain')`, `ylabel('Stress (Mpa)')`
 - Use line colours and styles carefully so that multiple data sets can easily be distinguished, e.g. `plot(x1,y1,'b-',x2,y2,'r-')`
 - Remember to insert a legend when you are plotting multiple data sets in one plot, e.g. `legend('Carbon steel', 'Stainless steel')`

MATLAB Basics – Plotting

- MATLAB has many built-in plot types, and a easy way to get a quick overview of the different plot types is to select some variables in the Workspace Browser, click on the disclosure triangle next to the *plot* toolbar icon and select *More plots...*



MATLAB Basics – Plotting

Multiple plots in one Figure Window

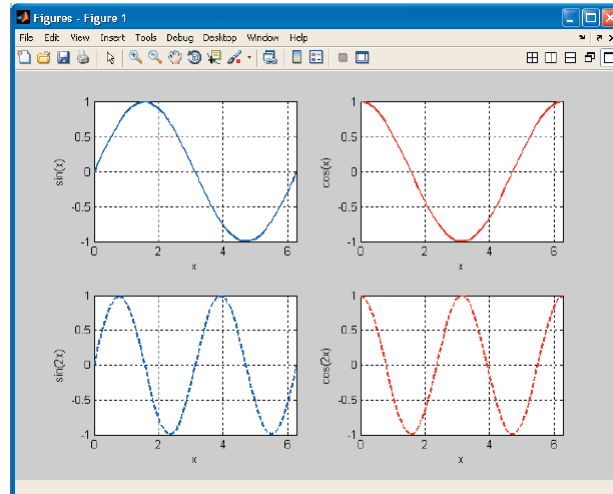
- The subplot command can be used to display a number of different plots in a single Figure Window.
- The subplot command `subplot(2,2,1)` specifies that the window should be divided into two rows and two columns of plot, and selects the first subplot to plot into.

Example:

```
>> x = linspace(0,2*pi,50);
>> subplot(2,2,1), plot(x,sin(x)), xlabel('x'),
ylabel('sin(x)');
>> subplot(2,2,2), plot(x,cos(x)), xlabel('x'),
ylabel('cos(x)');
>> subplot(2,2,3), plot(x,sin(2*x)), xlabel('x'),
ylabel('sin(2x)');
>> subplot(2,2,4), plot(x,cos(2*x)), xlabel('x'),
ylabel('cos(2x)');
```

MATLAB Basics – Plotting

This figure shows the results of the commands on the previous page



MATLAB Basics – Plotting

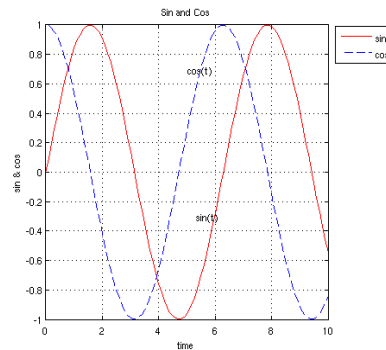
title	– set graph title
xlabel	– set X-axis label
ylabel	– set Y-axis label
text	– text annotation
gtext	– place text with mouse
grid on/off	– set grid lines
legend	– display legend
axis	– control axis scaling and appearance

MATLAB Basics – Plotting

```
t=0:0.1:10;  
y1=sin(t);  
y2=cos(t);  
plot(t,y1,'r',t,y2,'b--');
```

```
x=[1.7*pi;1.6*pi];  
y=[-0.3; 0.7];  
s=['sin(t)';'cos(t)'];
```

```
text(x, y, s); % Add comment at (x,y)  
title('Sin and Cos'); % Title  
legend('sin','cos') % Add legend  
xlabel('time') % the name of X-axis  
ylabel('sin & cos') % the name of Y-axis  
grid on % Add grid  
axis square % set figure as a shape of square
```



HØGSKOLEN
I ÅLESUND

MATLAB Basics – Exercise 3

1. Plot the following functions (choose your own appropriate range for x):

- a) $y = 1/x$, with a blue dashed line
- b) $y = \sin(x) \cos(x)$, with a red dotted line
- c) $y = 2x^2 - 3x + 1$, with red cross markers

Turn the grid on in all your plots, and remember to label axes and use a title

2. Given the following function

$$s = \arccos(\phi) + \sqrt{b^2 - (a \sin(\phi) - c)^2}$$

plot s as a function of angle ϕ when $a = 1$, $b = 1.5$, $c = 0.3$ and $0^\circ < \phi < 360^\circ$



HØGSKOLEN
I ÅLESUND

MATLAB Basics – Plotting

Curve-fitting of data

- Fitting of data can be done in several ways with MATLAB. One simple approach is by using the functions `polyfit` and `polyval`.

```
coeff = polyfit(xdata,ydata,n);
```

- Following this command, `coeff` will now be a vector containing the coefficients for the polynomial of best fit. `xdata` and `ydata` are vectors containing the independent and dependent variables, and `n` is the degree of the polynomial to be fitted.

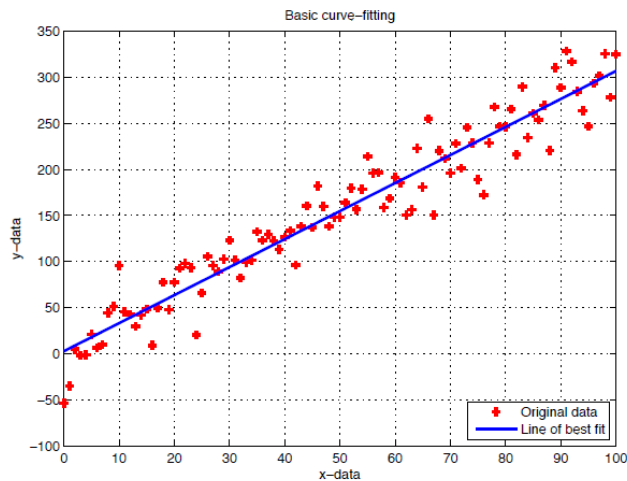
- Example:

```
>> coeff = polyfit(x,y,1);  
>> y_fit = polyval(coeff,x);  
>> plot(x,y,r+,x,y_fit), grid on, xlabel('x-data'),... ylabel('y-  
data'), title('Basic curve-fitting'),...  
legend('Original data','Line of best fit','Location','SouthEast')
```



MATLAB Basics – Plotting

This figure shows the results of the commands on the previous page

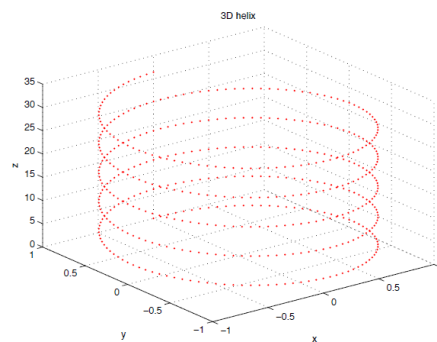


MATLAB Basics – Plotting

3D-plotting using `plot3` and `surf`

- MATLAB provides a number of built-in functions for producing different types of 3D plots.
- The 2D function `plot` becomes `plot3(x, y, z)` for plotting points and lines in 3D space

```
>> t=0:pi/50:10*pi;  
>> plot3(sin(t), cos(t), t, ...  
'r.', grid on, xlabel('x'), ...  
ylabel('y'), zlabel('z'), ...  
title('3D helix'))
```

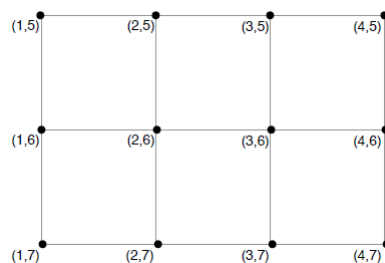


MATLAB Basics – Plotting

3D-plotting of surfaces and contours using `surf` and `mesh`

- Syntax of these functions: `surf(x, y, z)` and `mesh(x, y, z)`
- The function `meshgrid` must be used to define a grid of points which the surface will be plotted onto.

```
>> x=[1 2 3 4];  
>> y=[5 6 7];  
>> [xx, yy] = meshgrid(x,y)  
xx =  
    1    2    3    4  
    1    2    3    4  
    1    2    3    4  
yy =  
    5    5    5    5  
    6    6    6    6  
    7    7    7    7
```



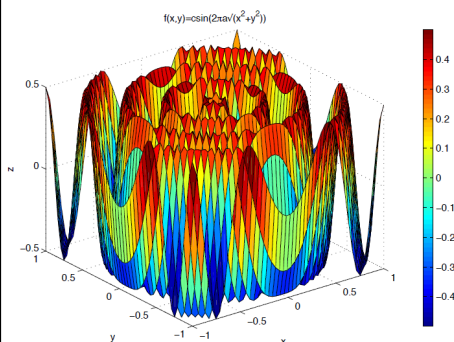
MATLAB Basics – Plotting

- Given the function $z = c \cdot \sin(2\pi a \sqrt{x^2 + y^2})$, where $a = 3$, $c = 0.5$, $-1 < x < 1$ and $-1 < y < 1$
- We use the function `surf` and `mesh` to plot this function

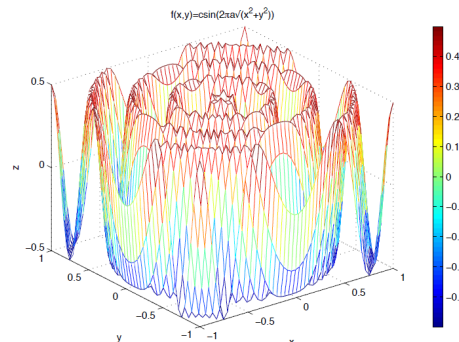
```
>> x=linspace(-1,1,50);
>> y=x;
>> a=3
>> c=0.5
>> [xx, yy] = meshgrid(x,y);
>> z = c*sin(2*pi*a*sqrt(xx.^2+yy.^2));
>> surf(xx,yy,z), colorbar, xlabel('x'), ylabel('y'),
zlabel('z'),title('f(x,y)=c sin(2 \pi a \surd(x^2+y^2))')
>> figure;
>> mesh(xx,yy,z), colorbar, xlabel('x'), ylabel('y'),
zlabel('z'), title('f(x,y)=c sin(2 \pi a \surd(x^2+y^2))')
```

MATLAB Basics – Plotting

surf function



mesh function



MATLAB Basics – Exercise 4

- Plot the following 3D curves using the **plot3** function
 - Spherical helix
$$x = \sin\left(\frac{t}{2c}\right) \cos(t)$$
$$y = \sin\left(\frac{t}{2c}\right) \sin(t)$$
$$z = \cos\left(\frac{t}{2c}\right)$$
where $c = 5$ and $0 < t < 10\pi$
 - Sine wave on a sphere
$$x = \cos(t) \sqrt{b^2 - c^2 \cos^2(at)}$$
$$y = \sin(t) \sqrt{b^2 - c^2 \cos^2(at)}$$
$$z = c \cdot \cos(at)$$
where $a = 10$, $b = 1$, $c = 0.3$, and $0 < t < 2\pi$
- Plot the following 3D curves using the **surf** function
 - Sine surface
$$x = \sin(u)$$
$$y = \sin(v)$$
$$z = \sin(u + v)$$
where $0 < u < 2\pi$ and $0 < v < 2\pi$
 - Elliptic torus
$$x = [1 - r_1 \cos(v)] \cos(u)$$
$$y = [1 - r_1 \cos(v)] \sin(u)$$
$$z = r_2 \cdot \left[\sin(v) + \frac{tu}{\pi} \right]$$
where $r_1 = r_2 = 0.5$, $t = 1.5$, $0 < u < 10\pi$ and $0 < v < 10\pi$



MATLAB Basics – Saving variables

- You can save your variables in the following way

```
>> save myFile a b f
>> save('myFile', 'a', 'b', 'f')
```
- This command will save the variables (matrices) a, b and f to the file `myFile.mat` in your present working directory
- Include the full path of your file if you want to store the file in a particular directory
- You can remove variables from the environment by

```
>> clear a b f
```

Now, the variables a, b and f will be gone from the workspace
- You can load variables back to the environment by

```
>> load myFile
```

Now, if you look at the workspace, you will see that a, b and f are back

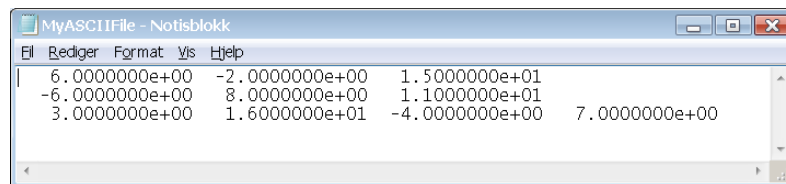


MATLAB Basics – Using I/O files

- You can also save (or load) data in the ASCII format, e.g. if you want to use the data as input in another program (or if you want to process data from another program)

```
>> V=[3 16 -4 7]; % create a 1x4 vector
>> A=[6 -2 15; -6 8 11]; % create a 2x3 matrix
>> save -ascii myASCIIFile
```

- This file can now be opened by e.g. Notepad



```
6.000000e+00 -2.000000e+00 1.500000e+01
-6.000000e+00 8.000000e+00 1.100000e+01
3.000000e+00 1.600000e+01 -4.000000e+00 7.000000e+00
```

MATLAB Basics – Using I/O files

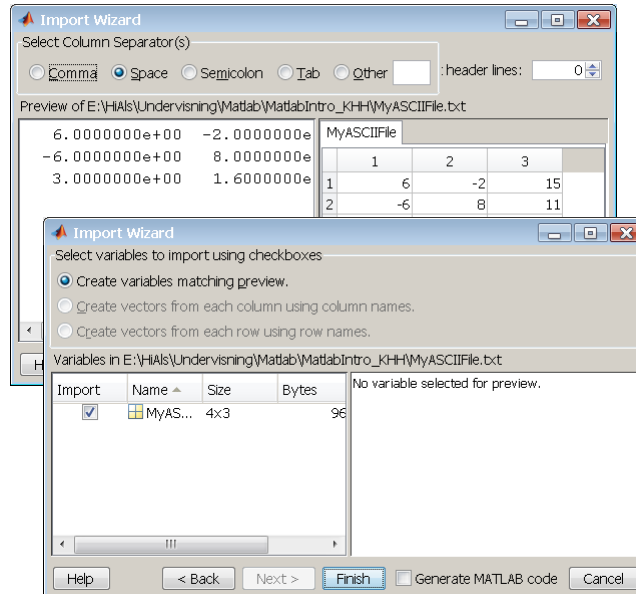
Import wizard

- Data can be opened through the **Import Wizard**. This is the most flexible way of importing data, since you do not need to know the format of the data.
- You start the Import Wizard by selecting Import data in the File menu of the Command Window.
- The Import Wizard then opens a file selection window which shows all the data files recognized by the Wizard.
- The Wizard opens the selected file and displays a portion of the data in the file, for the user to identify the content, see next slide

MATLAB Basics – Using I/O files

Import wizard

- The Wizard will try to process the data
- You will be asked to specify such things as column separator
- The final processed data will show up with a variable name in the Workspace



MATLAB Basics – Using I/O files

Reading from Excel files - `xlsread`

- Importing data from Excel is done with the `xlsread` command. In the simplest form we write the command

```
>> A=xlsread('input.xls')
```
- If the Excel file has more than one sheet, data will be imported from the first sheet.
- To import data from an Excel file with several sheets, we can use

```
>> B=xlsread('input.xls','Sheet2')
```
- Another option is to specify the parts of the sheet where the data should be gathered

```
>> C=xlsread('input.xls','Sheet3','C2:E5')
```
- Now, ('C2:E5') is a 4 by 3 region consisting of rows 2,3,4 and 5 and columns C,D and E

MATLAB Basics – Using I/O files

Writing to Excel files - `xlswrite`

- Exporting data from MATLAB to an Excel spreadsheet is done by using the `xlswrite` command. In the simplest form this is

```
>> xlswrite('output.xls',D)
```

where D is the variable name of the data we want to export to Excel.
- If we want to put the data in a particular sheet, we can use

```
>> xlswrite('output.xls',D,'Sheet2')
```
- And if we want to put the data in the variable D into a particular part in a particular sheet of the Excel file, we can use

```
>> xlswrite('output.xls',D,'Sheet3','C2:E5')
```
- Now, D must be a 4x3 matrix in order to fit into the specified range ('C2:E5')

MATLAB Basics – Scripts and functions

- In the 'save' command you save all your previous MATLAB commands in a separate **.mat** file. If you 'load' this file at a later stage, you will be back to the same situation as at the time of saving.
- MATLAB commands can also be stored in another format, the so-called **m-files** (extension **.m**). We can distinguish between two separate versions of the m-files.
 - **Script files**
Useful when you want to repeat a set of commands, and only want to change the value of some variables every time. The variables will be available in the workspace after you have run the script file
 - **Function files**
In the function file, the variables are local to the function, and not available in the workspace after you have run the function file.
A function file will always begin with a function definition line. This specifies the input and output variables used in the function

MATLAB Basics – Scripts

- A script file can be written in the editor window, and may be thought of as a small computer program. You can build your own algorithms in a script file, and carry out almost any mathematical operation within the script.
- We strongly encourage to make use of the commenting options MATLAB allows for. In the present example, we have a script version of the 'surf' plot

```
% my_surf.m           Script to plot a surface
% Variables:         x, y   Vectors of ranges used to plot the function z
%                   a, c   Coefficients used in the function z
%                   xx, yy Matrices generated by meshgrid to defined points
%                   z      Definition of function to plot

clear all; clc; % Clear all variables and clear command window
x = linspace(-1,1,50); % create vector x
y = x; % create vector y
a = 3; c = 0.5;
[xx, yy] = meshgrid(x,y); % Generate xx & yy arrays for plotting
z = c*sin(2*pi*a*sqrt(xx.^2+yy.^2)); % Calculate z
surf(xx,yy,z), xlabel('x'), ylabel('y'), zlabel('z'), ...
Title('f(x,y)=c sin(2 \pi a \surd (x^2+y^2))') % plots filled-in surface
```

MATLAB Basics – Functions

- A function always begins with a function definition line. This line specifies the input and output variable that the function will use, and defines the name of the function.
- Below is an example of the definition of a function to calculate an area, and an example of how to apply the function in a small program (a 'script').
- Note that a script will make use of the variables found in the workspace, whereas a function will only make use of the variables entered as input to the function and the variables defined in the function itself

```
Function area = calculateArea(x,y)
% Function to calculate an area of a rectangle (x,y)
% Variables:      x, y   lengths in two different directions
%               area   the area of the rectangle
Area = x*y; % Calculates area
```

Commands in Command window:

```
>> x = 5; y = 10;
>> area = calculateArea(x,y)
area =
    50
```

Alternatively:

```
>> length1 = 25; length2 = 100;
>> newArea = calculateArea(length1,length2)
newArea =
    2500
```

MATLAB Basics – Functions

FUNCTION DEFINITION	FILENAME	INPUT VARIABLES	OUTPUT VARIABLES	NOTES
<code>function [rho, H, F] = motion(x, y, t)</code>	<code>motion.m</code>	<code>x, y, t</code>	<code>rho, H, F</code>	
<code>function [theta] = angleTH(x, y)</code>	<code>angleTH.m</code>	<code>x, y</code>	<code>theta</code>	
<code>function theta = THETA(x, y)</code>	<code>THETA.m</code>	<code>x, y</code>	<code>theta</code>	If there is only one output variable the square brackets can be omitted
<code>function [] = circle(r)</code>	<code>circle.m</code>	<code>r</code>	None	
<code>function circle(r)</code>	<code>circle.m</code>	<code>r</code>	None	If there are no output variables the square brackets and the equals sign can be omitted

MATLAB Basics – Scripts and functions

- It is extremely useful to include comments in your MATLAB code. This can explain what the code is supposed to do, and what the meaning of the different variables is. A comment is preceded by a percent sign (%). Anything placed after a percent sign on a line will not be executed.
- Script file names cannot contain spaces (replace spaces with e.g. an underscore), start with a number, be names of built-in functions or be variables names.
- It is a good idea to use the commands `clear all` and `clc` as the first commands in your script file. This ensures that any confusion with existing variables is avoided.
- This will be further explored in next weeks session on MATLAB programming
 - For loops
 - While loops
 - If-then-else loops