Course unit title:
Basics of Information Systems
Course unit code:
NIRIA1SEND

Matlab basics II.

Dr. habil. Levente Kovács
associate professor

Obuda University,
John von Neumann Faculty of Informatics,
Department of Information Systems,
Physiological Controls Group

17.09.2013.

Review – last lecture

- m-files
- Importing Data
- Arithmetic Operators
- Relational Operators
- Logical Operators
- who, whos, clear, clc
- disp
- Creating and Manipulating Vectors
- Creating and Manipulating Matrices
- zeros, ones, eye, rand
- Elements of a matrix
- if, elseif, else, end
- for, while

Contents

Create 2-D line plot

```
plot(X,Y)
```

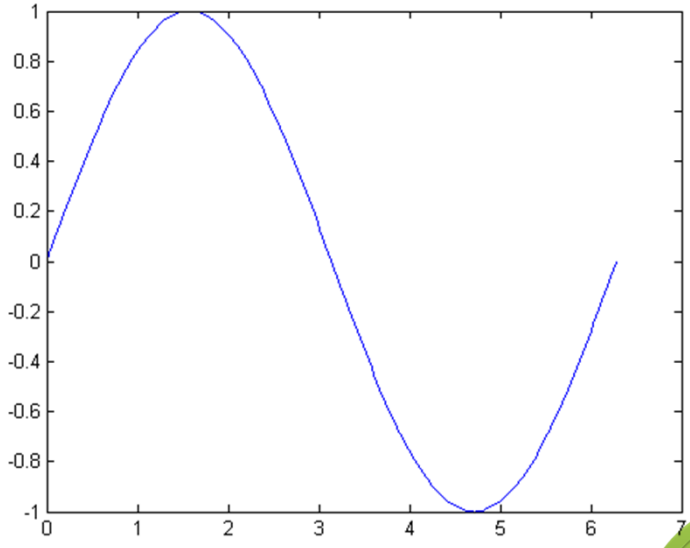
• Creates a 2-D line plot of the data in Y versus the corresponding values in X.

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

• Define *x* as a vector of linearly spaced values between 0 and 2π.
• Use an increment of π/100 between the values.
• Define *y* as sine values of *x*.

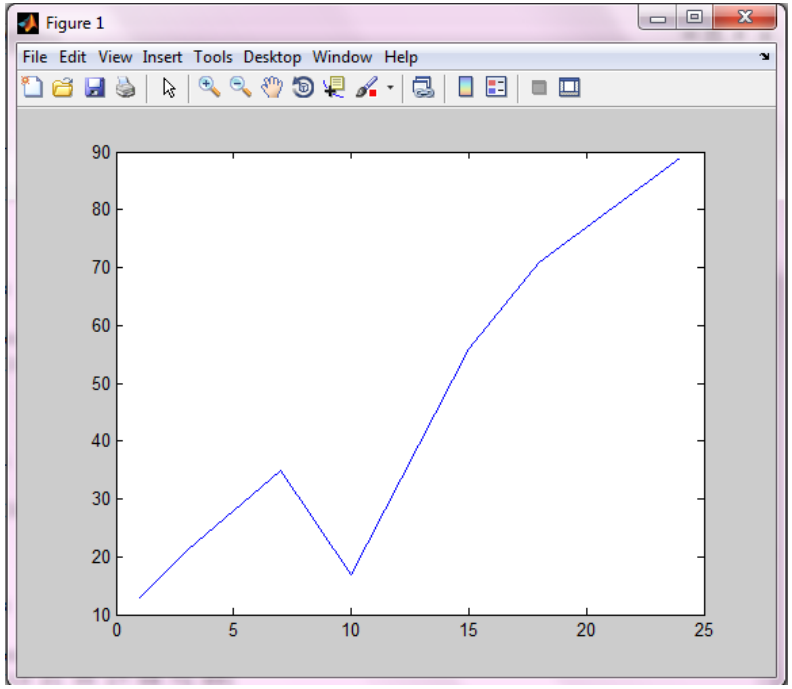
```
figure % opens new figure window
plot(x,y)
```

• Create a line plot of the data.
• If *x* and *y* are both vectors, then they must have equal length and MATLAB plots *y* versus *x*
• If *x* and *y* are both matrices, then they must have equal size and MATLAB plots columns of *y* versus columns of *x*

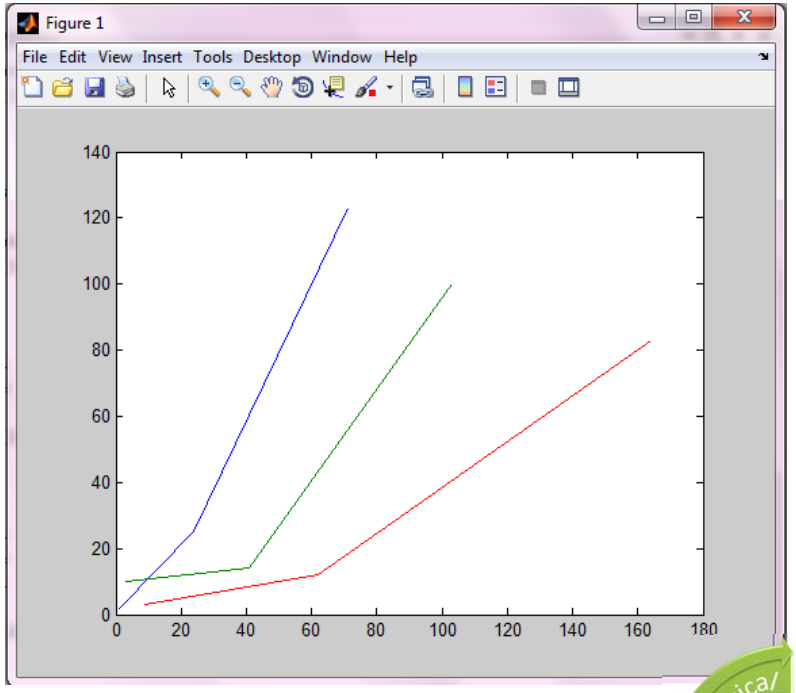


Create line plot – examples

```
x=[1 3 7 10 15 18 24];
y=[13 21 35 17 56 71 89];
figure
plot(x,y)
```




```
X=[1 3 9; 24 41 62; 71 103 164];
Y=[2 10 3; 25 14 12; 123 100 83];
figure
plot(x,y)
```



LineSpec

```
plot(X,Y,LineSpec)
```

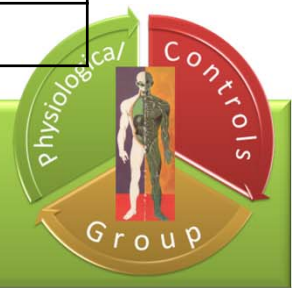
• Line style, marker symbol, and color, specified as a string.

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Specifier	Color
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

Specifier	Marker
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram

Example: `'--or'` is a red dashed line with circle markers

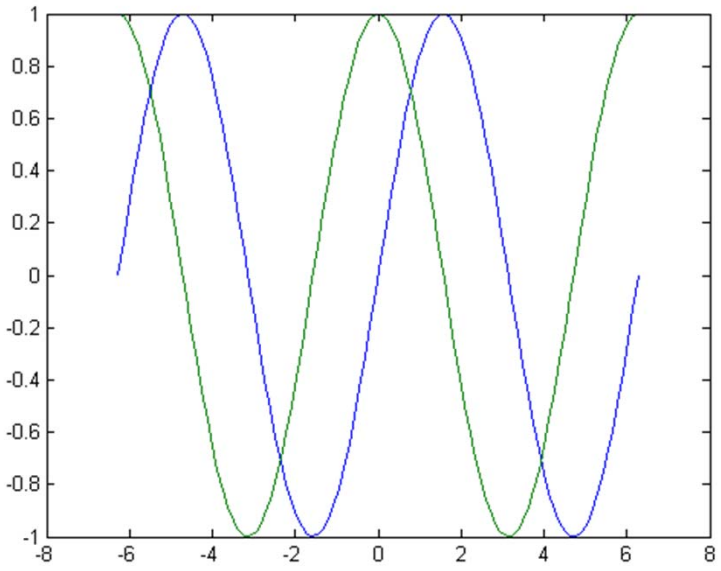


Plot multiple lines

```
plot(X1,Y1,...,Xn,Yn)
```

- Plots multiple *X*, *Y* pairs using the same axes for all lines

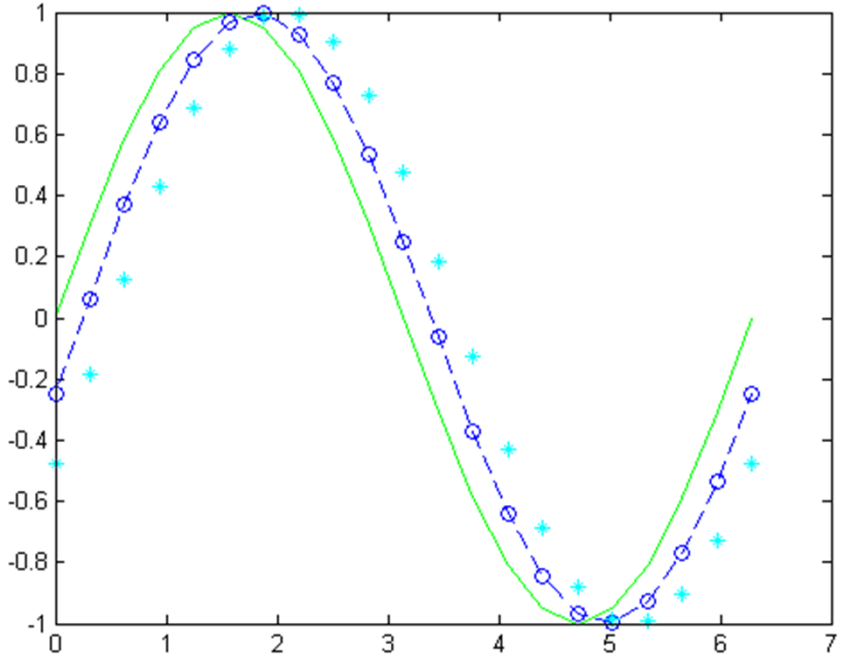
```
x = linspace(-2*pi,2*pi);
y1 = sin(x);
y2 = cos(x);
figure
plot(x,y1,x,y2)
```



- Define *x* as 100 linearly spaced values between -2π and 2π.
- Define *y1* and *y2* as sine and cosine values of *x*.
- Create a line plot of both sets of data.

Plot multiple lines – example

```
x = 0:pi/10:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);
figure plot(x,y1,'g',x,y2,'b--o',x,y3,'c*')
```

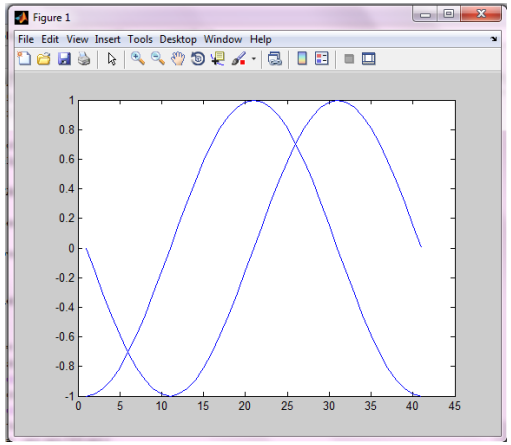


hold

The `hold` function controls whether MATLAB clears the current graph when you make subsequent calls to plotting functions (the default), or adds a new graph to the current graph.

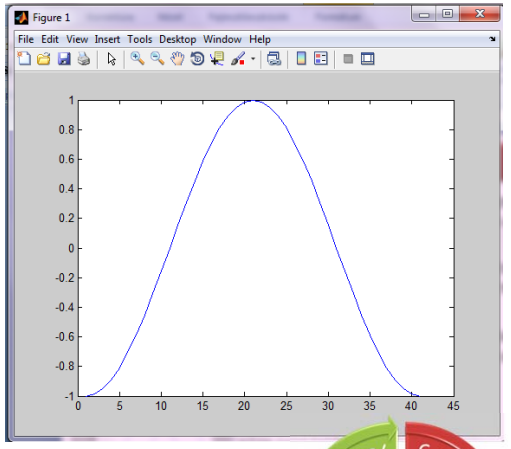
`hold on` retains the current graph and adds another graph to it. MATLAB adjusts the axes limits, tick marks, and tick labels as necessary to display the full range of the added graph.

`hold off` resets hold state to the default behavior, in which MATLAB clears the existing graph and resets axes properties to their defaults before drawing new plots




```
x = -pi:pi/20:pi;
plot(sin(x))
hold on
plot(cos(x))
hold off
```

```
x = -pi:pi/20:pi;
plot(sin(x))
plot(cos(x))
```

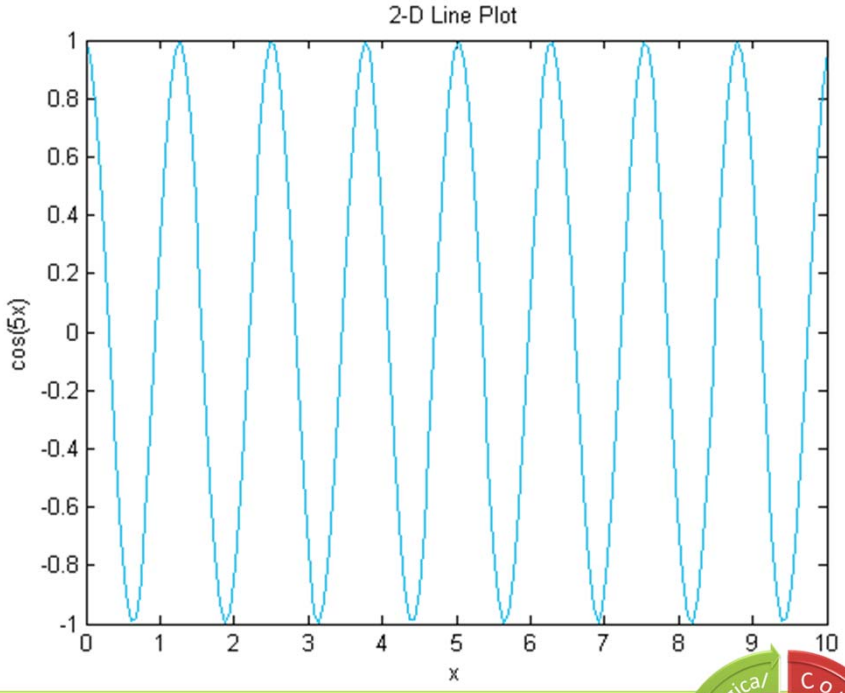


Add title and axis labels

```
x = linspace(0,10,150);
y = cos(5*x);
figure
plot(x,y,'Color',[0,0.7,0.9])
title('2-D Line Plot')
xlabel('x')
ylabel('cos(5x)')
```

`linspace` function defines *x* as a vector of 150 values between 0 and 10

`Color:` Change the line color using an RGB color value



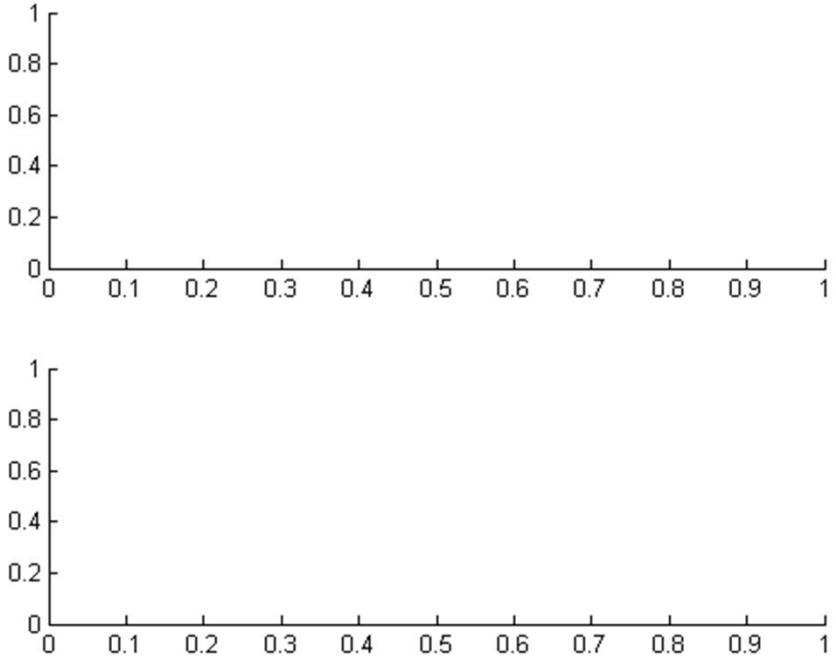

```
title('')
xlabel('')
ylabel('')
```

Subplot

```
subplot(m,n,p)
```

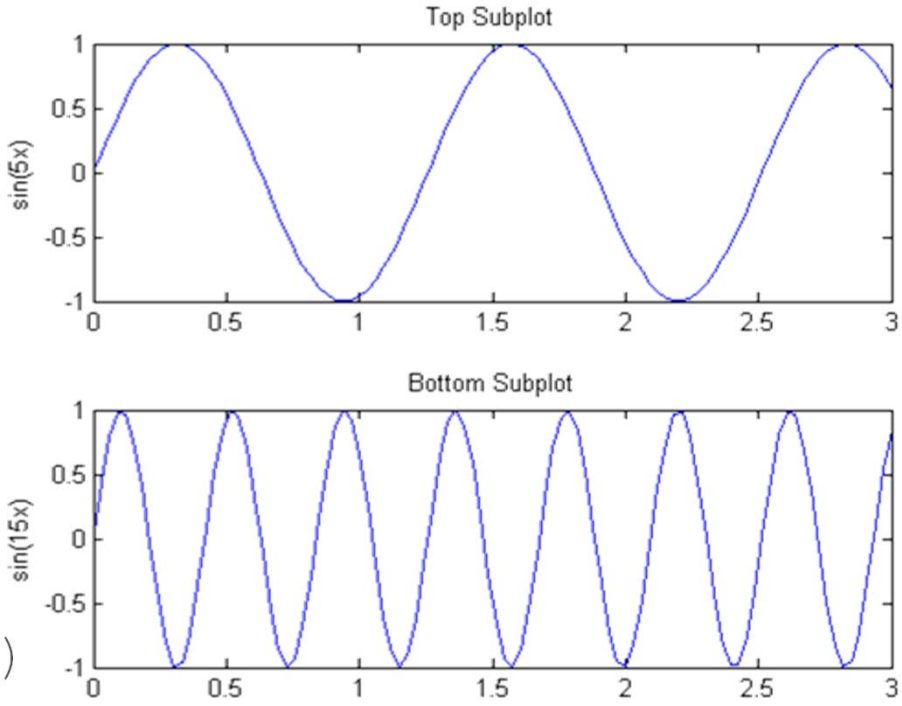
- Divides the current figure into an *m*-by-*n* grid and creates an axes in the grid position specified by *p*

```
figure;
% new figure

s(1) = subplot(2,1,1);
% top subplot

s(2) = subplot(2,1,2);
% bottom subplot
```

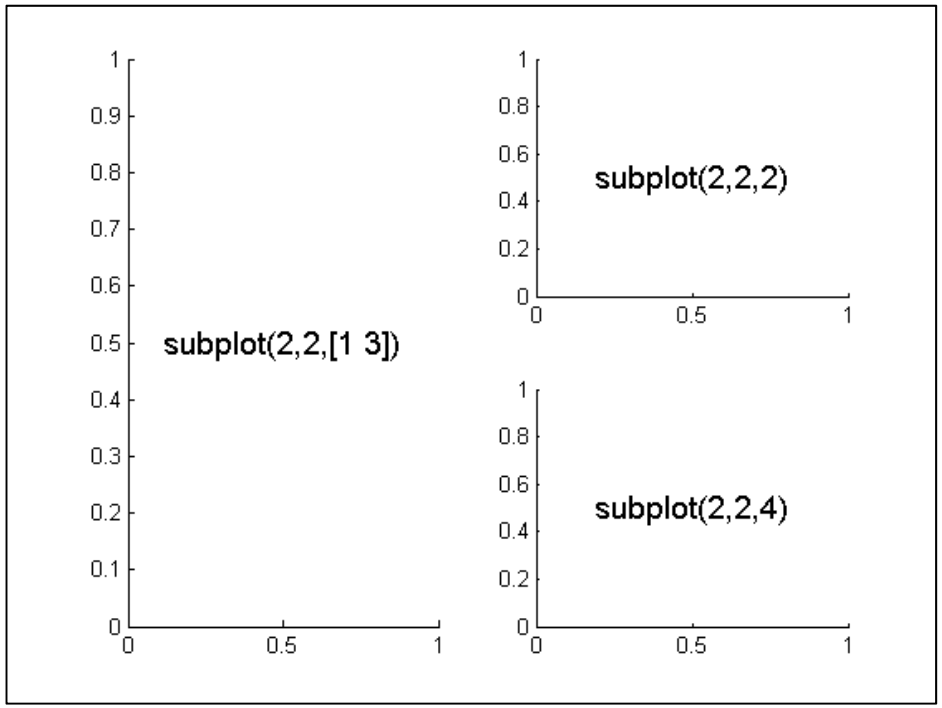


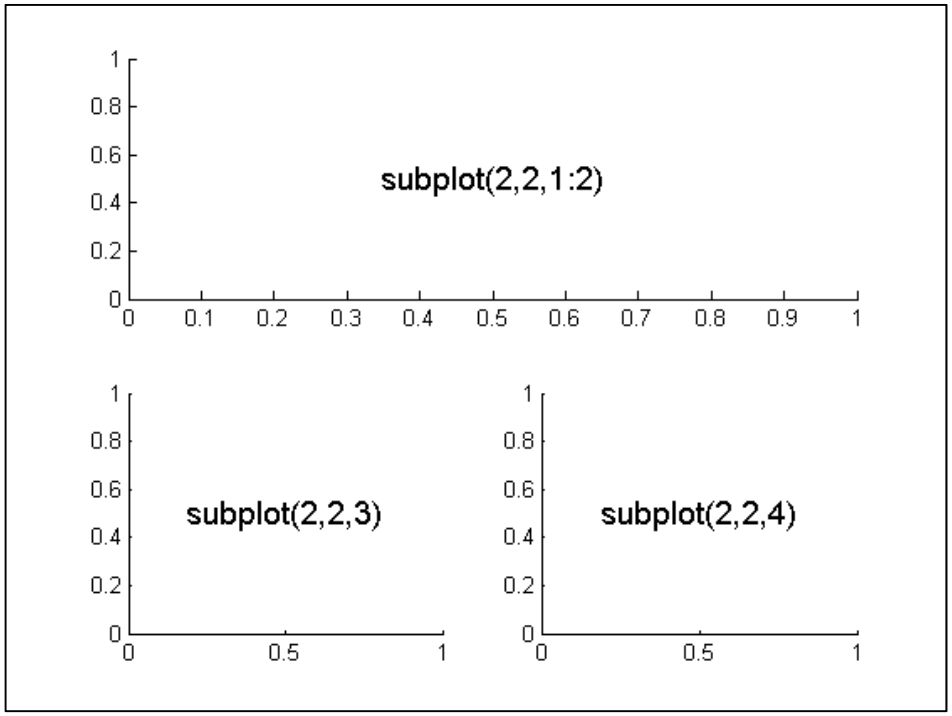
Subplot – example 1

```
x = linspace(0,3);
y1 = sin(5*x);
y2 = sin(15*x);

plot(s(1),x,y1)
title(s(1),'Top Subplot')
ylabel(s(1),'sin(5x)')

plot(s(2),x,y2)
title(s(2),'Bottom Subplot')
ylabel(s(2),'sin(15x)')
```



Subplot – example 2



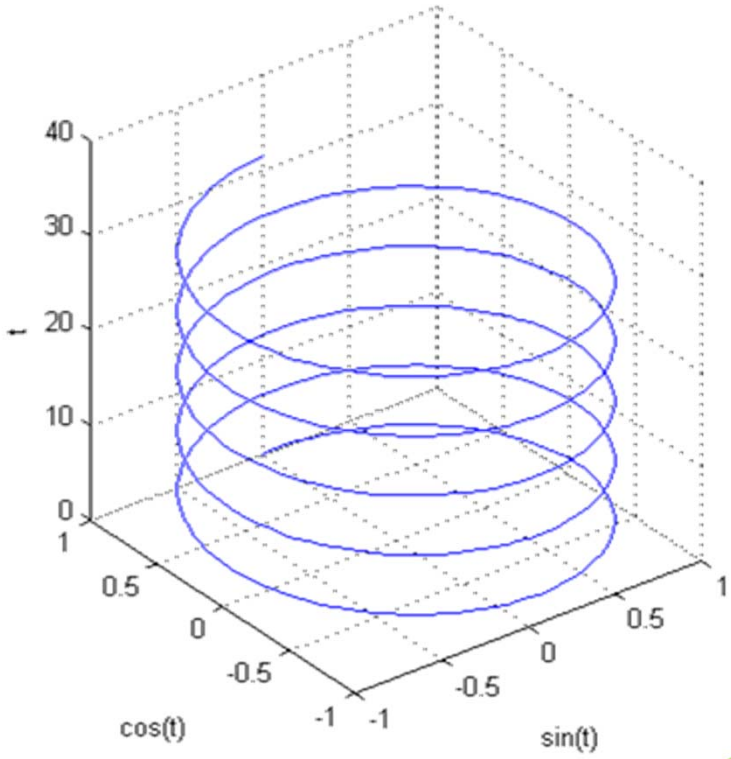


Create 3-D line plot

```
plot3(X1,Y1,Z1,...)
```

• Plots one or more lines in three-dimensional space through the points whose coordinates are the elements of *X1*, *Y1*, and *Z1*

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
xlabel('sin(t)')
ylabel('cos(t)')
zlabel('t')
grid on
axis square
```



• Plot a three-dimensional helix.

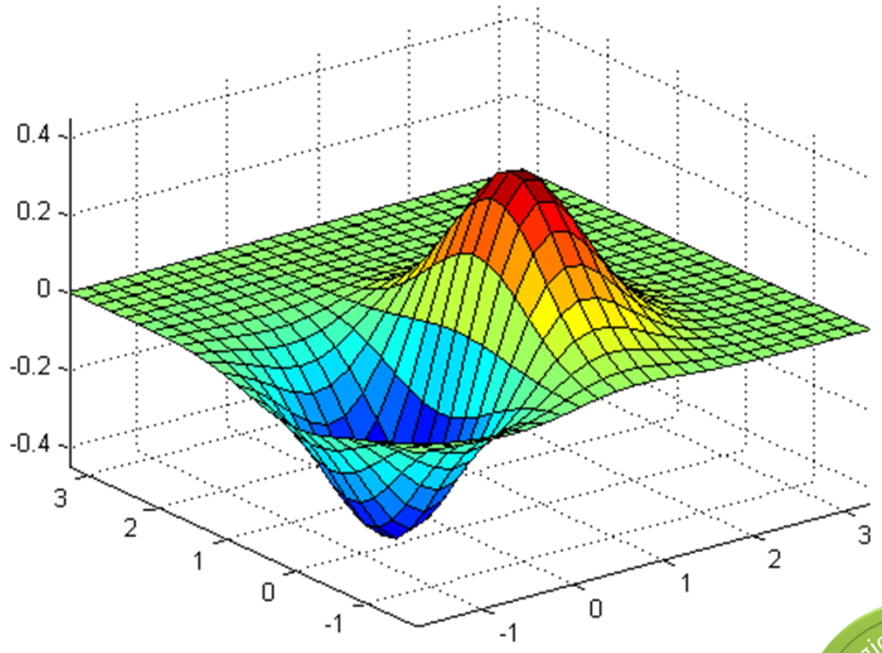
Create 3-D shaded surface plot

```
surf(X,Y,Z)
```

• `surf(X,Y,Z)` uses Z for the color data and surface height. X and Y are vectors or matrices defining the x and y components of a surface

```
[x,y] = meshgrid([-1.75:.2:3.25]);
z = x.*exp(-x.^2-y.^2);
surf(x,y,z)
xlim([-1.75 3.25])
ylim([-1.75 3.25])
zlim([-0.45 0.45])
```

`meshgrid` creates *X* and *Y* matrices for arbitrary domains



`xlim, ylim, zlim` set the *x*-, *y*- and *z*-axis limits to match the actual range of the data

Reading sound files

```
[y, Fs] = wavread(filename)
```

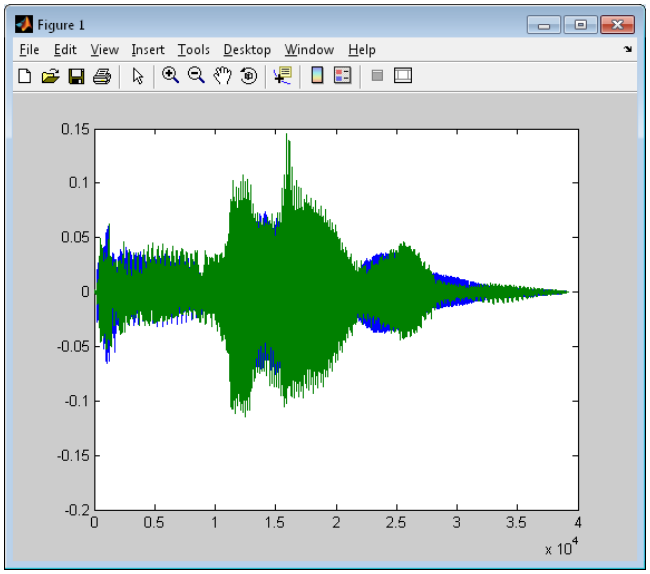
- loads a WAVE file specified by the string *filename*,
- returns the sampled data in *y*
- returns the sample rate (*Fs*) in Hertz used to encode the data in the file.
- if *filename* does not include an extension, wavread appends *.wav*

```
sound(y,Fs)
```

- convert matrix of signal data to sound
- sends audio signal *y* to the speaker at sample rate *Fs*

Reading sound files – examples

```
[y, Fs] = wavread('win.wav');
sound(y, Fs);
plot(y);
```




```
fs = 44100; % sampling frequency
f0 = 200; % base frequency
T = 2; % duration
n = [0:fs*T]; % time axis(sample)
x = sin(2*pi*f0*n/fs); % signal
plot(n/fs, x)
sound(x, fs)
```

Reading image from graphics file

```
A = imread(filename, fmt)
```

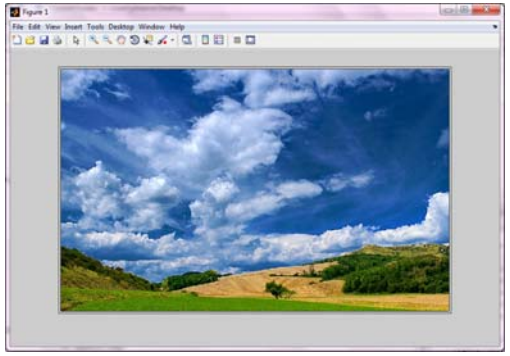
- reads a grayscale or color image from the file specified by the string *filename*,
- the text string *fmt* specifies the format of the file by its standard file extension
- the return value *A* is an array containing the image data
  - ✓ if the file contains a grayscale image, *A* is an M-by-N array
  - ✓ if the file contains a truecolor image, *A* is an M-by-N-by-3 array

```
imshow(I)
```

- displays the image *I* in a Handle Graphics figure, where *I* is a grayscale, RGB (truecolor), or binary image

Reading image from graphics file – example 1

```
img = imread('cloud.jpg');
imshow(img)
size(img)
ans =
      628   1000    3
```

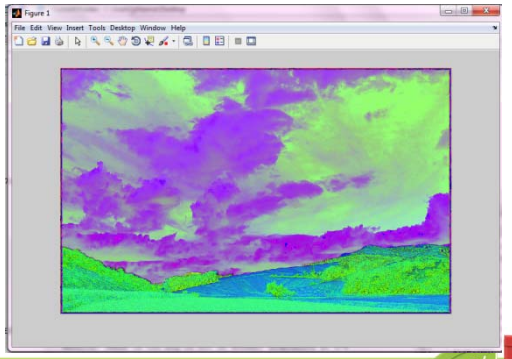


```
img2 = rgb2gray(img);
imshow(img2)
```

Convert RGB image or colormap to grayscale

```
img3 = rgb2hsv(img);
imshow(img3)
```

Convert RGB colormap to HSV colormap



Reading image from graphics file – example 2

```
figure
subplot(3,1,1)
imshow(img(:,:,1))
subplot(3,1,2)
imshow(img(:,:,2))
subplot(3,1,3)
imshow(img2(:,:,3))
```

Loading Data

```
load count.dat
```

- import data into MATLAB using the `load` function
- loading this data creates a 24-by-3 matrix called count in the MATLAB workspace

```
[n,p] = size(count)
n =
     24
p =
     3
```

- get the size of the data matrix
- *n* represents the number of rows, and *p* represents the number of columns

Exercise 1

Calculating π with Leibniz series

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

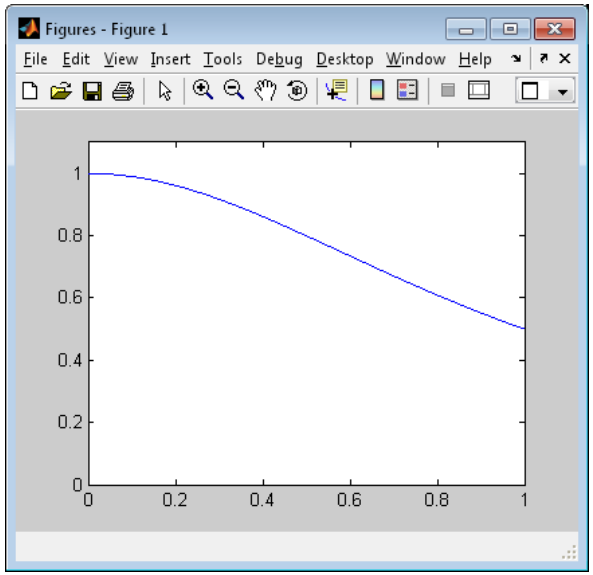
```
format long
iterPi = 10000000; % iteration
erLeibniz = 1; %1. element of the series
for i = 3:4:iterPi
    erLeibniz = erLeibniz - (1.0 / i) + (1.0 / (i +
2));
end

disp('Eredmény: ')
erLeibniz=erLeibniz*4

disp('Pi:')
pi
```

Exercise 2

Calculating π with $\frac{1}{1+x^2}$ area under the function

$$\int_0^1 \frac{1}{1+x^2}\,dx = \frac{\pi}{4} \approx 0.785398$$

```
x=[0:0.001:1];
y= 1./(1+x.^2);
plot(x,y)
xlim([0 1]);
ylim([0 1.1]);
```



Exercise 2

Calculating π with $\frac{1}{1+x^2}$ area under the function

```
myfun.m:
      function y = myfun(x)
      y= 1./(1+x.^2) *4;

>> quad(@myfun,0,1)

ans =

   3.141592682924567
```

Exercise 2

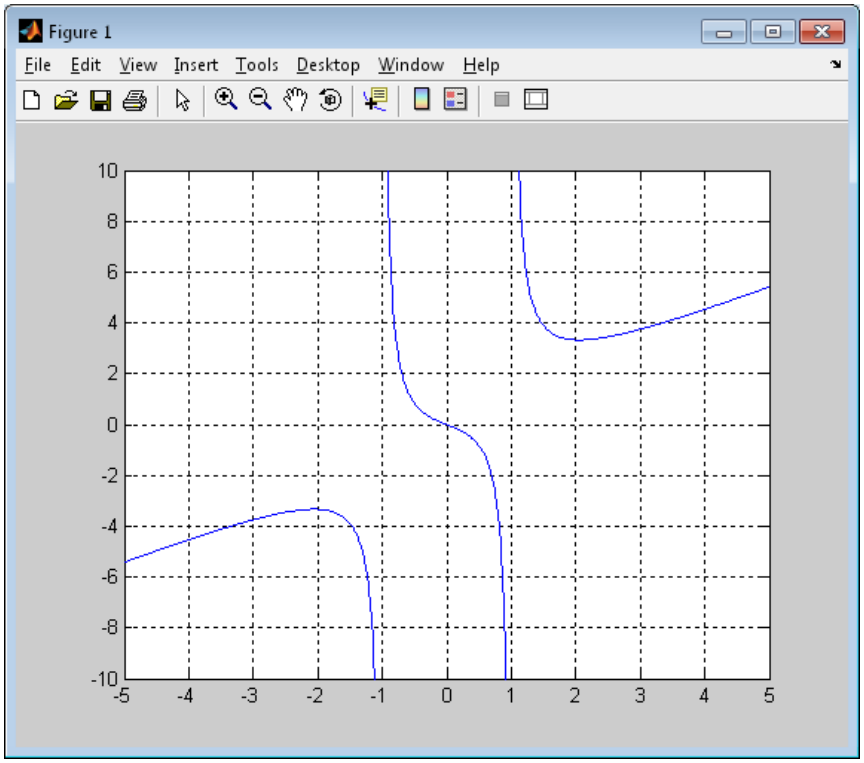
Calculating π with $\frac{1}{1+x^2}$ area under the function

```
format long
iterPi = 10000000; % iteration
x=0;
width =  1 / iterPi;
erFuggvenyTer = 0;

for  i = 0:iterPi
    x = (i + 0.5) * width;
    erFuggvenyTer = erFuggvenyTer + 4 / (1 + x * x);
end

disp('1/(1+n^2) alatti terület : ')
erFuggvenyTer*width

disp('Pi:')
pi
```

Exercise 3

Plotting $f(x) = x + \dfrac{2x}{x^2 - 1}$

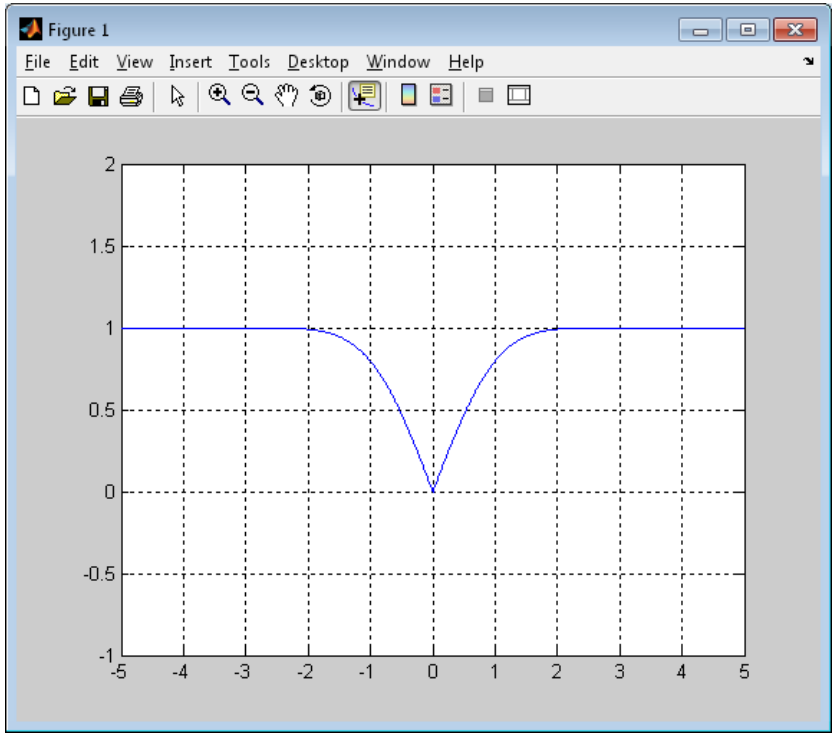
```
x=[-5:0.01:5]
y=x + (2*x./ (x.*x-1) )
plot(x,y)
Grid
xlim([-5 5])
ylim([-10 10])
```



Exercise 4

Plotting $f(x) = \sqrt{1 - e^{-x^2}}$

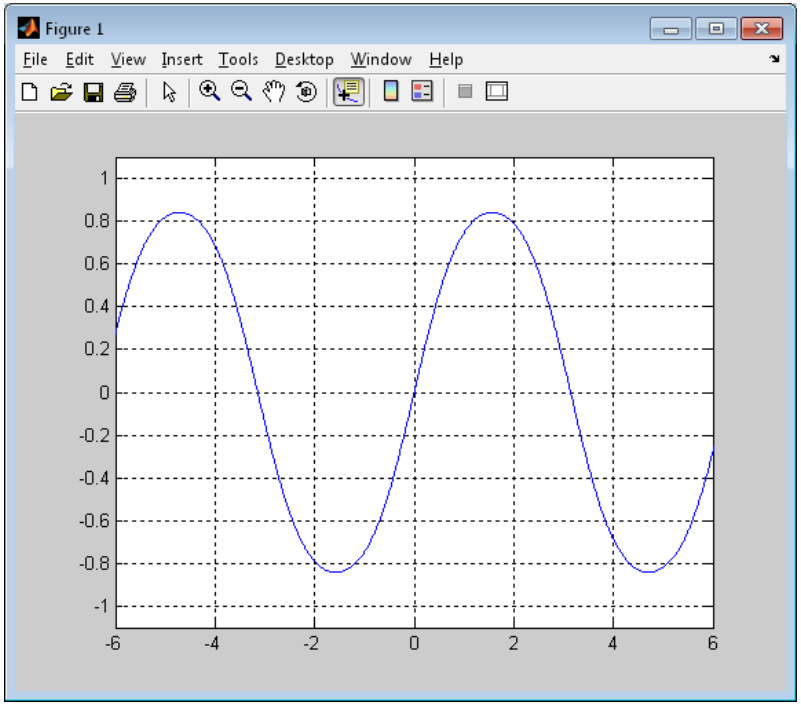
```
x=[-5:0.01:5]
y=sqrt(1-exp(1).^(-x.^2))
plot(x,y)
Grid
xlim([-5 5])
ylim([-1 2])
```



Exercise 5

Plotting $f(x) = \sin(\sin(x))$

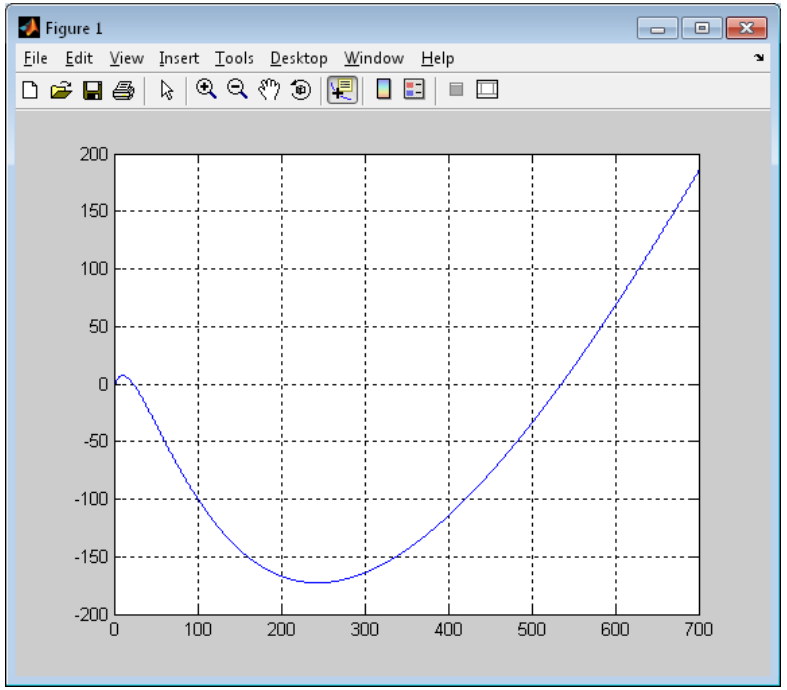
```
x=[-10:0.01:10]
y=sin(sin(x))
plot(x,y)
Grid
xlim([-6 6])
ylim([-1.1 1.1])
```



Exercise 6

Plotting $f(x) = x \cdot \sin(\ln(x))$

```
x=[0:0.1:700]
y=x.*sin(log(x));
plot(x,y)
grid
```



Course unit title:
Basics of Information Systems
Course unit code:
NIRIA1SEND

Thank you for your attention!

kovacs.levente@nik.uni-obuda.hu

17.09.2013.