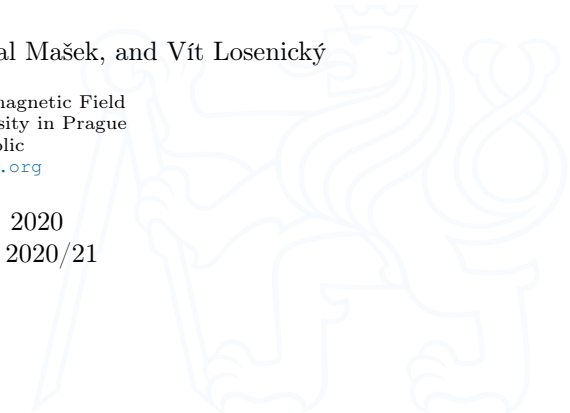
Lecture 7: Visualization

BE0B17MTB – Matlab

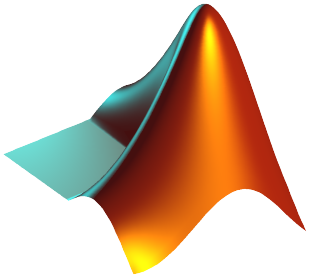
Miloslav Čapek, Viktor Adler, Michal Mašek, and Vít Losenický

Department of Electromagnetic Field
Czech Technical University in Prague
Czech Republic
matlab@elmag.org

November 11, 2020
Winter semester 2020/21



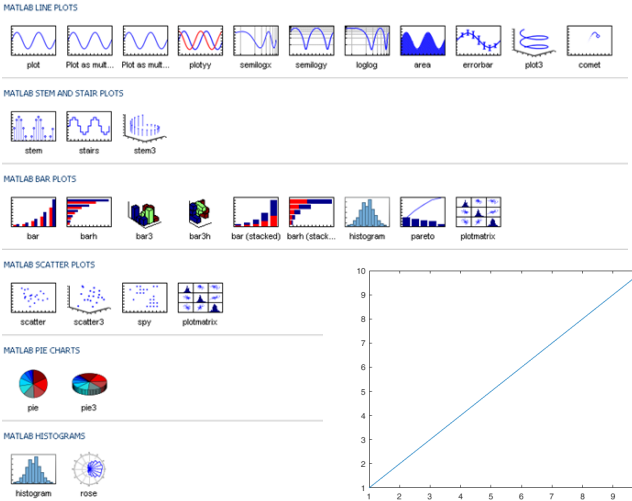
Outline



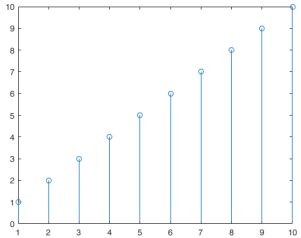
Introduction to Visualizing

- We have already got acquainted (marginally) with some of MATLAB graphs.
  - `plot`, `stem`, `semilogx`, `pcolor`
- In general, graphical functions in MATLAB can be used as:
  - higher level
    - Access to individual functions, object properties are adjusted by input parameters of the function.
    - The first seven weeks of the semester.
  - lower level
    - Calling and working with objects directly.
    - Knowledge of MATLAB handle graphics (OOP) is required.
    - Opens wide possibilities of visualization customization.
- Details to be found in help:
  - MATLAB ← Graphics

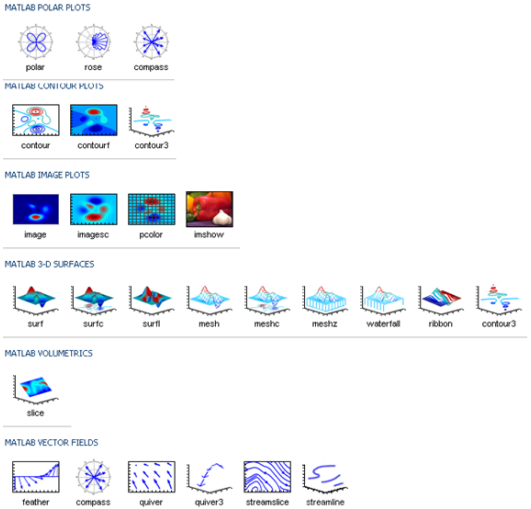
Selected Graphs I.



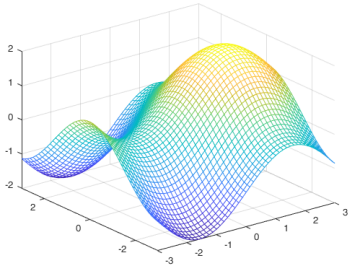

```
plot(linspace(1,10,10));
stem(linspace(1,10,10));
% ... and others
```

Selected Graphs II.




```
x = -3:0.125:3;
y = x.';
z = sin(x) + cos(y);
mesh(x,y,z);
axis([-3 3 -3 3 -2 2]);
```



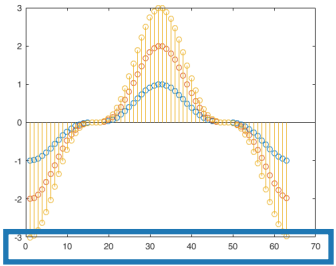
Visualization

Function `figure`

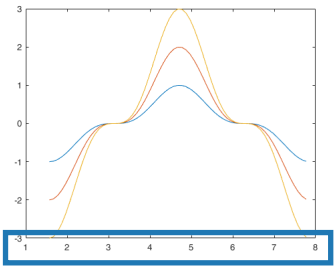
- `figure` opens empty figure to plot graphs.
  - The function returns object of class `matlab.ui.Figure`.
  - It is possible to plot matrix data (column-wise).
  - Don't forget about x-axis data!

```
x = (0:0.1:2*pi) + pi/2;
fx = -[1 2 3].'*sin(x).^3;
```

```
figure;
stem(fx.');
```




```
figure;
plot(x, fx);
```



LineSpec – Customizing Graph Curves I.

- What do `plot` function parameters mean?
  - See `>> doc LineSpec`.
  - The most frequently customized parameters of graph's lines:
    - Color (can be entered also using matrix [R G B], where R, G, B vary between 0 a 1),
    - marker shape,
    - line style.

line color	
`'r'`	red
`'g'`	green
`'b'`	blue
`'c'`	cyan
`'m'`	magenta
`'y'`	yellow
`'k'`	black
`'w'`	white

marker	
`'+'`	plus
`'o'`	circle
`'*'`	asterisk
`'.'`	dot
`'x'`	x-cross
`'s'`	square
`'d'`	diamond
`'^'`	triangle
and others	`>> doc LineSpec`

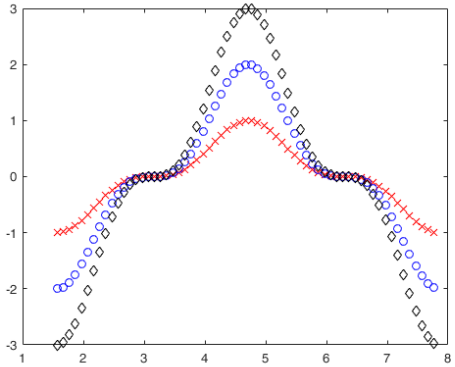
```
plot(x,f,'bo-');
plot(x,f,'g*--');
```

line style	
`'-'`	solid
`'--'`	dashed
`':'`	dot
`'-.'`	dash-dot
`'none'`	no line

Function hold on

- Function `hold on` enables to plot multiple curves in one axis.
- It is possible to disable this feature by typing `hold off`.
- Functions `plot`, `plot3`, `stem` and others enable to add optional input parameters (as strings).

```
x = (0:0.1:2*pi) + pi/2;
fx = -[1 2 3].'*sin(x).^3;
figure;
plot(x, fx(1, :), 'xr');
hold on;
plot(x, fx(2, :), 'ob');
plot(x, fx(3, :), 'dk');
```



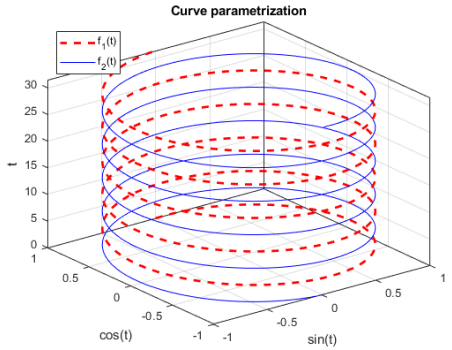
Selected Functions for Graph Modification

- Graphs can be customized in many ways, the basic ones are:

function	description
`title`	title of the graph
`xlabel, ylabel, zlabel`	label axes
`grid on, grid off`	turns grid on / off
`hold on`	enables to add another graphical elements while keeping the existing ones
`xlim, ylim, zlim`	set axes' range
`legend`	display legend
`subplot`	create more axes in one figure
`yyaxis`	create chart with two y-axes
`box on`	display axes outline
`text`	adds text to graph
and others	

Visualizing

- The example below shows plotting a spiral and customizing plotting parameters.
- It is possible to use additional name-value pair arguments with majority of plotting functions.

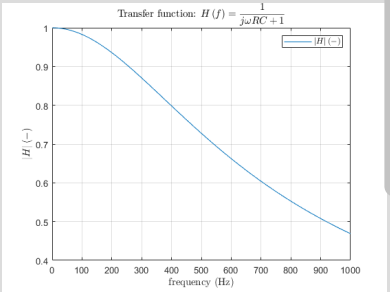



```
figure('color', 'w');
t = 0:0.05:10*pi;
plot3(sin(t), cos(t), t, 'r--', ...
  'LineWidth', 2);
hold on;
plot3(-sin(t), -cos(t), t, 'b')
box on;
grid on;
xlabel('sin(t)');
ylabel('cos(t)');
zlabel('t');
title('Curve parametrization');
legend('f_1(t)', 'f_2(t)', ...
  'Location', 'northwest');
```

LaTeX in Figures

- Labels and titles in figure have `Interpreter` property.
- Possible values are `'tex'`, `'latex'` and `'none'`.
- Font is default LaTeX font.

```
figure;
f = 1:1e3; R = 100; C = 3e-6;
Hf = abs(1./(1j*2*pi*f*R*C + 1));
plot(f, Hf);
grid on;
xlabel('frequency (Hz)', 'Interpreter', 'latex');
ylabel('$$\left| H \right|\left( - \right)$$', ...
  'Interpreter', 'latex');
title(['Transfer function: $$H\left( f \right)', ...
  ' = \frac{1}{{j\omega RC + 1}}$$'], ...
  'Interpreter', 'latex');
hL=legend('$$\left| H \right|\left( - \right)$$');
hL.Interpreter = 'latex';
```

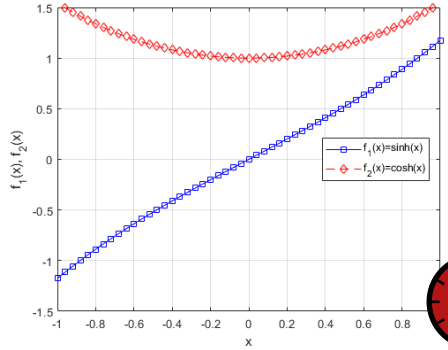


LineSpec – Customizing Graph Curves II.a

- Evaluate following two functions in the interval $x \in [-1, 1]$ for 51 values:

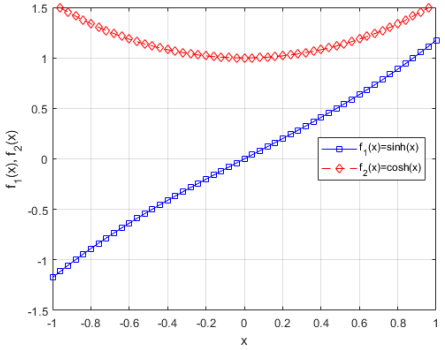
$$f_1(x) = \sinh(x), \; f_2(x) = \cosh(x)$$

- Use the function `plot` to depict both f_1 and f_2 so that:
  - both functions are plotted in the same axis,
  - the first function is plotted in blue with $\square$ marker as solid line,
  - the other function is plotted in red with $\diamondsuit$ marker and dashed line,
  - limit the interval of the y-axis to $[-1.5, 1.5]$,
  - add a legend associated to both functions,
  - label the axes (x-axis: x, y-axis: f_1(x), f_2(x)),
  - apply grid to the graph.



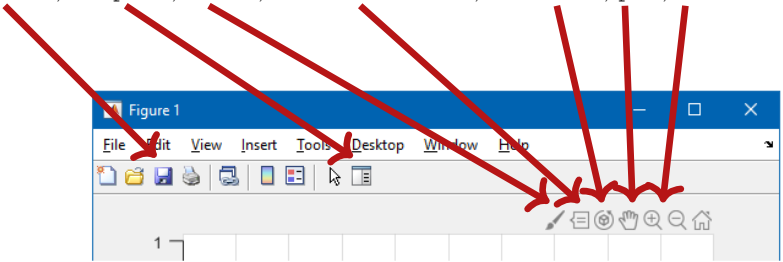
LineSpec – Customizing Graph Curves II.b

$$f_1(x) = \sinh(x), \; f_2(x) = \cosh(x)$$



Visualizing – Plot Tools

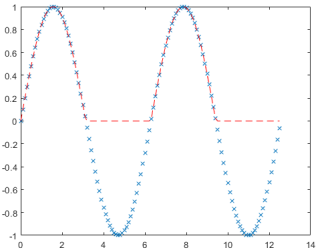
- It is possible to keep on editing the graph by other means.
- All operations can be carried out using MATLAB functions.
  - `saveas`, `inspect`, `brush`, `datacursormode`, `rotate3d`, `pan`, `zoom`



- Properties of all graphical objects can be set programmically (see later).
  - Preferred for good-looking graphs with lot of graphical features.

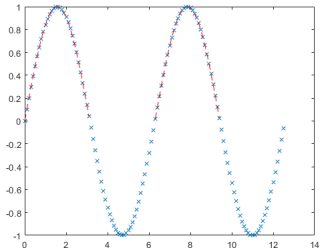
Visualizing — Use of NaN Values

- NaN values are not depicted in graphs.
  - It is quite often needed to distinguish zero values from undefined values.
  - Plotting using NaN can be utilized in all functions for visualizing.



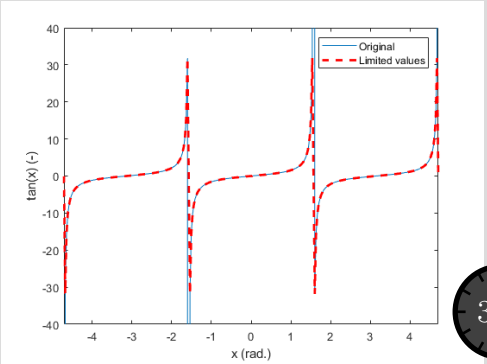
```
x = 0:0.1:4*pi;
fx = sin(x);
figure;
plot(x, fx, 'x');
hold on;
fx2 = fx;
fx2(fx < 0) = 0;
plot(x, fx2, 'r--');
```

```
% ...
fx2(fx < 0) = NaN;
% ...
```



Rounding

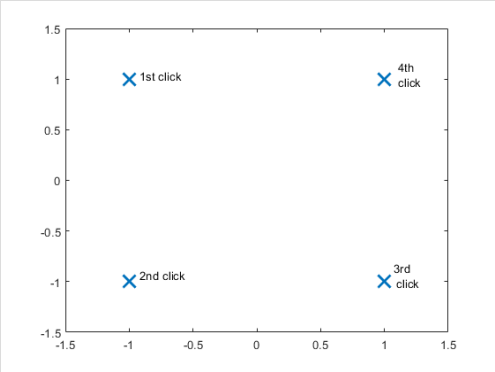
- Plot function $\tan(x)$ for $x \in [-3/2\pi, 3/2\pi]$ with step $pi/2$.
- Limit depicted values by ± 40.
- Values of the function with absolute value greater than $1 \cdot 10^{10}$ replace by 0.
  - Use logical indexing.
- Plot both results and compare them.



Function gtext

- Function gtext enables placing text in graph.
  - The placing is done by selecting a location with the mouse.

```
plot([-1 1 1 -1], [-1 -1 1 1], ...
  'x', 'MarkerSize', 15, ...
  'LineWidth',2);
xlim(3/2*[-1 1]); ylim(3/2*[-1 1]);

gtext('1st click');
gtext('2nd click');
gtext({'3rd'; 'click'});
gtext({'4th', 'click'});
```

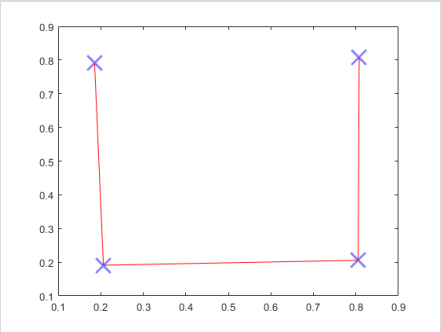


Function ginput

- Function `ginput` enables selecting points in graph using the mouse.
  - We either insert requested number of points (`P = ginput(x)`) or terminate by pressing Enter.

```
P = ginput(4);
```

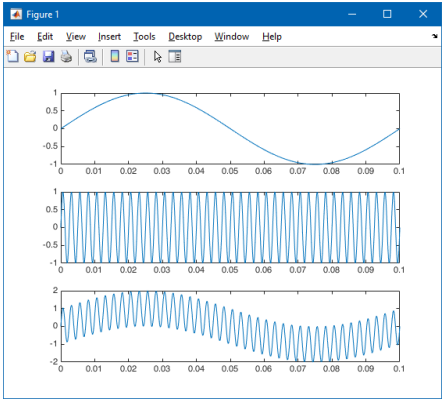
```
plot(P(:, 1), P(:, 2), ...
   'LineStyle', 'none', ...
   'LineWidth', 2, ...
   'Color', [0.5 0.5 1], ...
   'Marker', 'x', ...
   'MarkerSize', 20);
hold on;
plot(P(:, 1), P(:, 2), 'r');
```



More Graphs in a Figure I. – subplot

- Inserting several different graphs in a single window figure.
  - Function `subplot(m, n, p)`:
  - `m` is number of rows,
  - `n` is number of columns,
  - `p` is position.

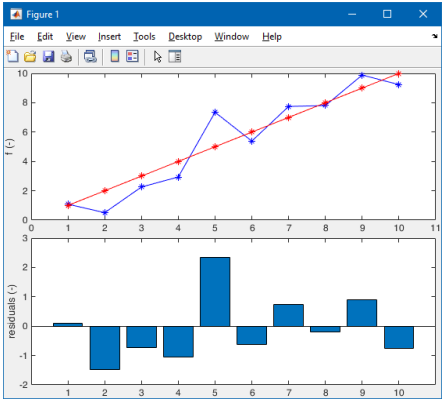
```
t = linspace(0, 0.1, 0.1*10e3);
f1 = 10; f2  = 400;

y1 = sin(2*pi*f1*t);
y2 = sin(2*pi*f2*t);
y3 = y1 + y2;

figure('color', 'w');
subplot(3, 1, 1); plot(t, y1);
subplot(3, 1, 2); plot(t, y2);
subplot(3, 1, 3); plot(t, y3);
```



More Graphs in a Figure II. – `tiledlayout`, `nexttile`

- `tiledlayout` creates invisible grid for advanced axes placement.
  - Properties `TileSpacing` and `Padding` set grid spacing and edges.

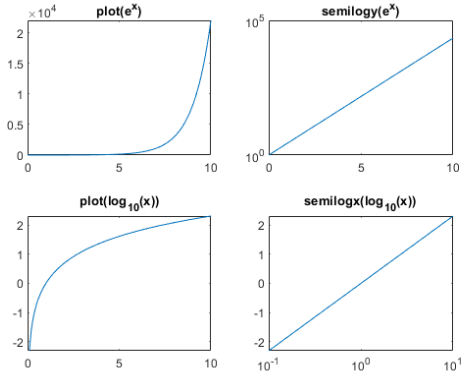
```
x = 1:10;
f = x + randn(size(x));

figure;
tiledlayout(2, 1, ... grid 2x1
  'TileSpacing', 'none', ...
  'Padding', 'none');
nexttile;
plot(x, f, '*-b', x, x, '*-r');
xlim([0 11]);
ylabel('f (-)');

nexttile;
bar(x, f - x); xlim([0 11]);
ylabel('residuals (-)');
```



Logarithmic Scale

- Functions `semilogy`, `semilogx`, `loglog`.

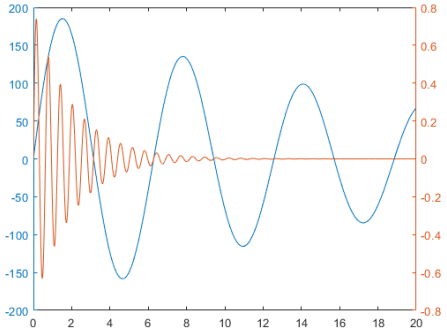
```
x  = 0:0.1:10;
y1 = exp(x);
y2 = log(x);

figure('color', 'w');
subplot(2, 2, 1); plot(x, y1);
title('plot(e^x)');

subplot(2, 2, 2); semilogy(x, y1);
title('semilogy(e^x)');

subplot(2, 2, 3); plot(x, y2);
title('plot(log_1_0(x))');

subplot(2, 2, 4); semilogx(x, y2);
title('semilogx(log_1_0(x))');
```



Double y Axis — yyaxis I.

- Enable to draw more curves to a single graph with two y axis with different ranges.

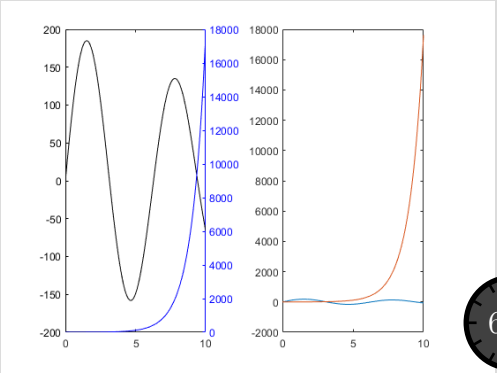
```
x  = 0:0.01:20;
y1 = 200 * exp(-0.05*x) .* sin(x);
y2 = 0.8 * exp(-0.5*x) .* sin(10*x);

figure('color', 'w');
yyaxis left; plot(x, y1);
yyaxis right; plot(x, y2);
```



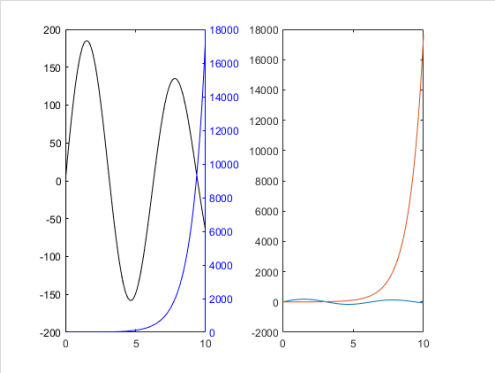
Double y Axis — `yyaxis` II.

- Compare plot and `yyaxis` in one figure object (using `subplot`) for functions shown below.
  - In the object created by `yyaxis` change default colors of individual lines to blue and black (don't forget about the axes).

```
x  = 0:0.1:10;
y1 = 200 * exp(-0.05*x) .* sin(x);
y2 = 0.8 * exp(x);
```



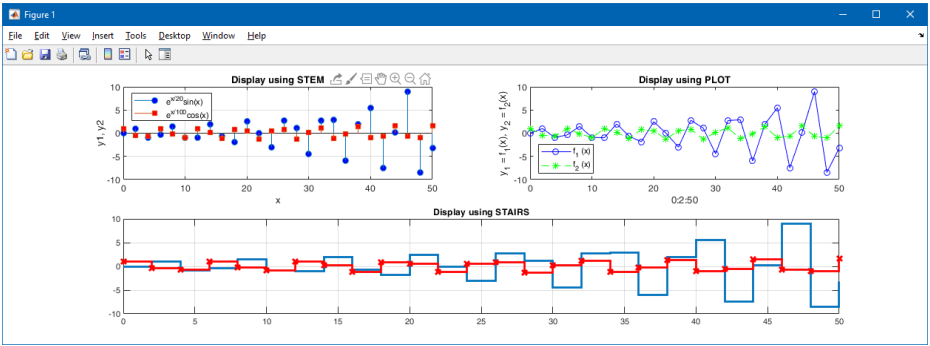
Double y Axis — `yyaxis` II.



Functions `stem`, `stairs`

- Try to imitate the figure where functions `y1` and `y2` are defined below.
  - See documentation of `stem` and `stairs` function.

```
x  = 0:2:50;
y1 = exp(0.05*x).*sin(x);
y2 = exp(0.01*x).*cos(x);
```

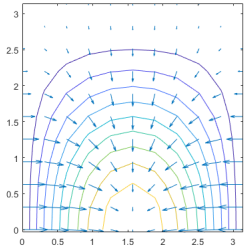


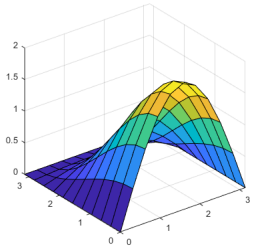
stem, stairs

Plotting 2-D Functions

- `contour`, `quiver`, `surf`

```
x = 0:pi/10:pi;
y = x.';
z = sin(x) + cos(y).*sin(x);
[gx, gy] = gradient(z);

figure('Color', 'w');

subplot(1, 2, 1);
contour(x, y, z);
hold on;
quiver(x, y, gx, gy);

subplot(1, 2, 2);
surf(x, y, z);
```

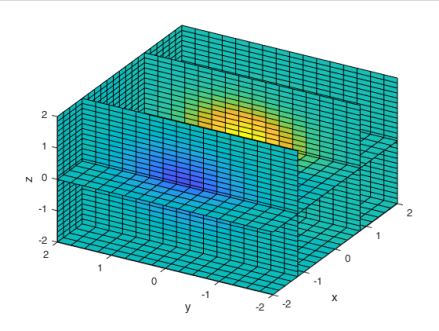




Volumetric Visualizing

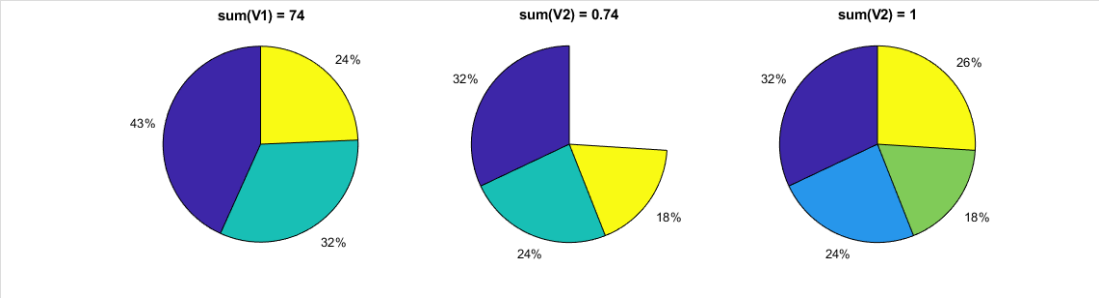
- Function `slice`.
  - Draw slices for the volumetric data.

```
x = -2:0.2:2;
y = (-2:0.25:2).';
z = shiftdim(-2:0.16:2, -1);

v = x.*exp(-x.^2 - y.^2 - z.^2);

xSlice = [-1.2, 0.8, 2];
ySlice = 2;
zSlice = [-2, 0];

figure('Color', 'w');
slice(x, y, z, v, xSlice, ySlice, zSlice);
xlabel('x'); ylabel('y'); zlabel('z');
% view(azimuth, elevation)
view(-60, 40);
```



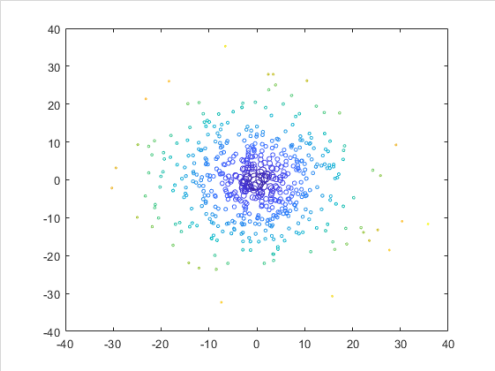
Functions pie, pie3

```
V1 = [32 24 18]; % sum(V1) = 74
V2 = V1/100; % sum(V2) = 0.74
V3 = [V2 1-sum(V2)]; % sum(V3) = 1

figure('Color', 'w');
subplot(1, 3, 1); pie(V1); title('sum(V1) = 74');
subplot(1, 3, 2); pie(V2); title('sum(V2) = 0.74');
subplot(1, 3, 3); pie(V3); title('sum(V2) = 1');
```



Function scatter

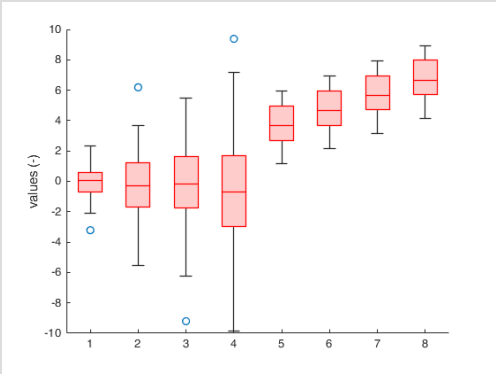
```
x = 10*randn(500, 1);
y = 10*randn(500, 1);
c = hypot(x, y);

figure('color', 'w');
scatter(x, y, 100./c, c);
box on;
```



Box Plot – `boxchart` (2020a)

- Box plot shows basic statistical properties of random data.
  - Median, lower and upper quartiles, outliers and minimal/maximal values (outside outliers).

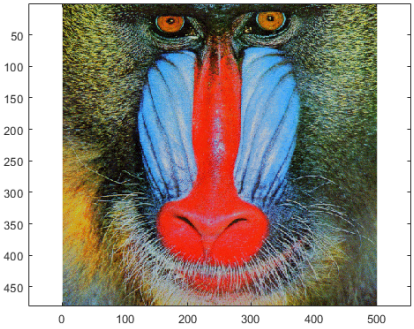
```
nSamples = 1e2;
data = [randn(nSamples, 4).*(1:4), ...
  5*rand(nSamples, 1) + (1:4)];

figure
boxchart(data, 'BoxFaceColor', 'r')
ylabel('values (-)')
```



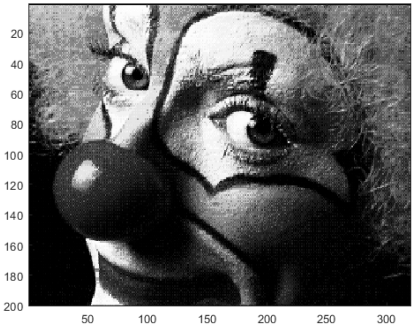
Picture Depiction

- Function `image`, `imagesc`, `colormap`.

```
load mandrill
image(X)
colormap(map)
axis equal
```

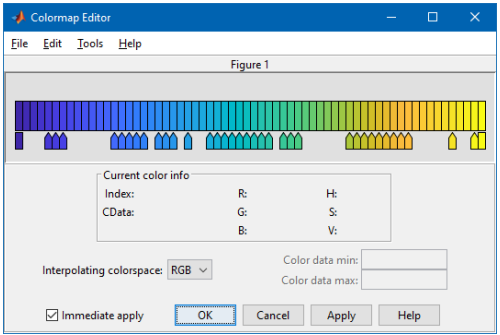



```
load clown
imagesc(X)
colormap(gray)
```



Function `colormap` I.

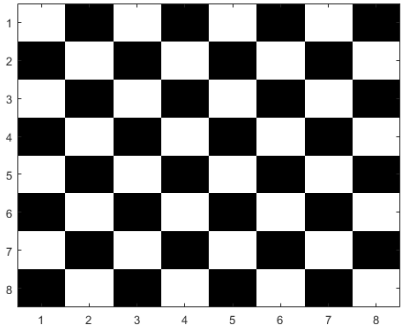
- Determines the scale used in picture color mapping.
- It is possible to create/apply an own one: `colormapeditor`.



Colormap Name	Color Scale
parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
gray	
bone	
copper	
pink	
lines	
colorcube	
prism	
flag	
white	

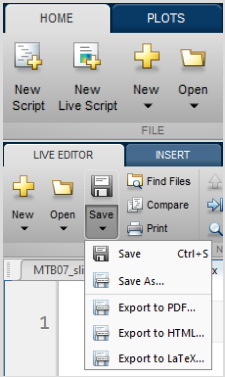
Function `colormap` II.

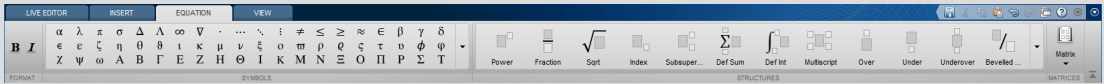
- Create a chessboard as shown in the figure.
  - The picture can be drawn using the function `imagesc`.
  - Consider `colormap` setting.



Live Script I.

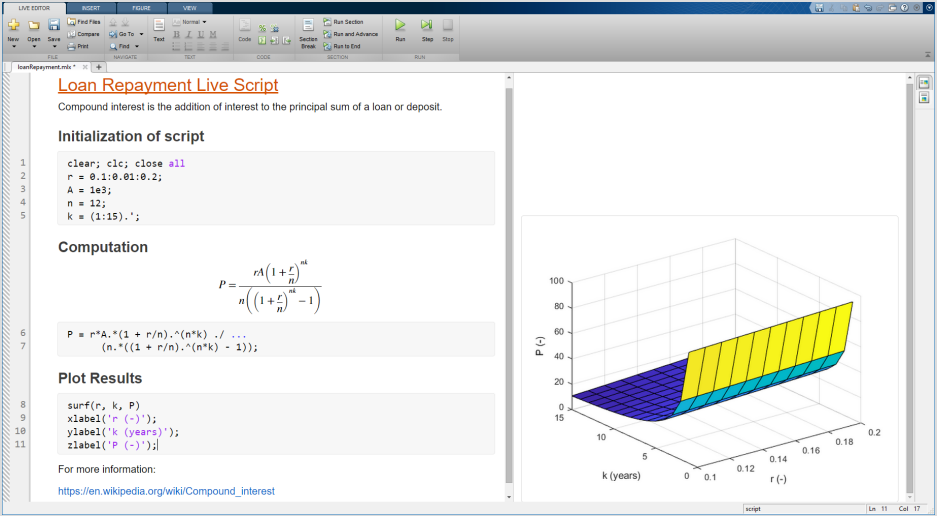
- Live script can contain code, generated output, formatted text, images, hyperlinks, equations, ...
  - It is necessary to use Live Editor.
  - From MATLAB window: HOME → New Live Script.
  - From editor: EDITOR → New → Live Script
  - Editor creates `*.mlx` files.
- Export options: PDF, HTML, LaTeX.
- Internal extensive equation editor.





Visualization

Live Script II.



Object Handles I.

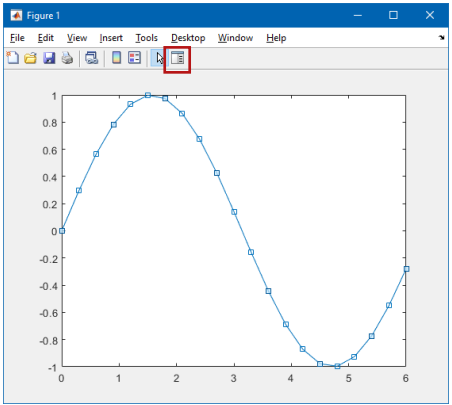
- Each individual graphical object has its own pointer ('handle' in Matlab terms).
- These handles are practically a reference to an existing object.
- Handle is always created by MATLAB, it is up to the user to store it.
- One handle can be saved to several variables but they refer to a single object.
- All graphical objects inherit superclass `handle`.
  - Inherits several useful methods (`set`, `get`, `delete`, `isvalid`, ...).

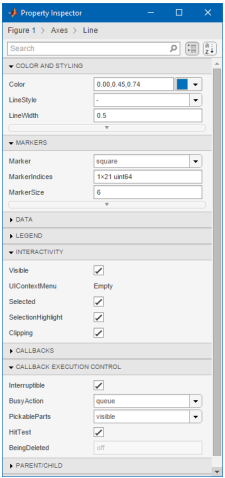
```
hFig = figure;
hAx = axes('Parent', hFig);
hLine1 = line('Parent', hAx);
```

- Graphical objects respect specific hierarchy.
- See help for list of properties (`>> doc Figure Properties`, `>> doc Axes Properties`, `>> doc Line Properties`, ...)

Object Handles II.

- Property inspector (`inspect`).





Object Handles III.

- The way of setting handle object properties.
  - Using functions `set` and `get`.
    - It is not case sensitive.

```
myLineObj = plot(1:10);
get(myLineObj, 'color')
```

```
set(myLineObj, 'color', 'r')
```

  - Dot notation.
    - It is cAsE sEnSiTiVe.

```
myLineObj = plot(1:10);
myLineObj.Color
```

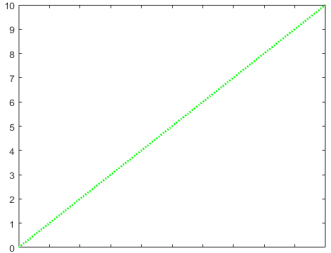
```
myLineObj.Color = 'r';
myLineObj.Color
```

Functions `get` and `set`

- Create a graphic object in the way shown. Then using functions `get` and `set` perform following tasks.

```
myLineObj = plot(0:10);
```

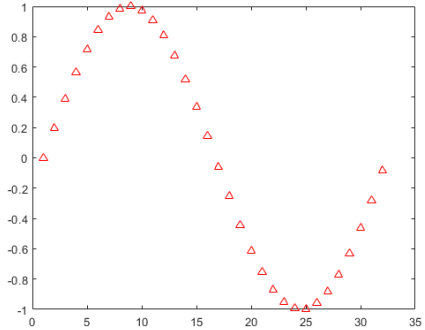
  - Find out the thickness of the line and increase it by 1.5.
  - Set the line color to green.
  - Set the line style to dotted.



60

Dot Notation Application

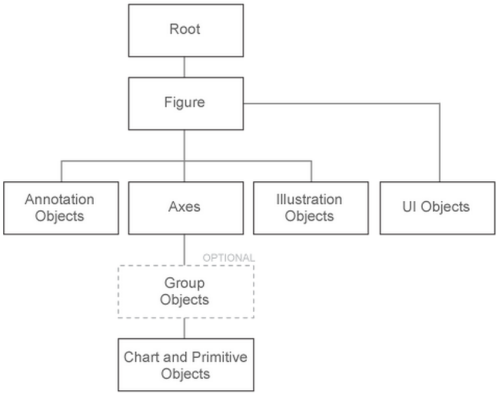
- Using dot notation change the initial setting of the function shown to get plot as in the figure.




```
myLineObj = plot(sin(0:0.2:2*pi));
```

Graphics Object Hierarchy

- All graphical objects are connected in the hierarchy via `Children` and `Parent` properties.




```
hRoot = groot;
hFig = figure('Parent', hRoot);
hAx = axes('Parent', hFig);
hLine = line('Parent', hAx, ...
  'XData', -10:10, ...
  'YData', (-10:10).^3);
hTitle = title(hAx, 'Cubic fcn.');
```

```
hRoot.Children % ans = hFig
hFig.Children % ans = hAx
hAx.Children % ans = hLine
hLine.Children
% ans = 0x0 GraphicsPlaceholder
hTitle.Parent % ans = hAx
```

```
hRoot.Children.Children.Color = 'y';
```

LineSpec — Default Setting

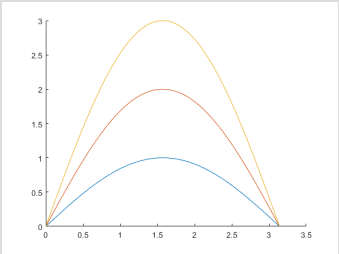
- Colors in given order are used when plotting more lines in one axis.
- It is not necessary to set color of each curve separately when using `hold on`, nor plotting matrix columns.

```
close all; clear; clc;
x = (0:0.01:pi).';
figure;
hold on;
plot(x, 1*sin(x));
plot(x, 2*sin(x));
plot(x, 3*sin(x));
```

```
figure, plot(x, 1:3*sin(x));
```

```
set(groot, 'defaultAxesColorOrder', ...
  myColors)
```

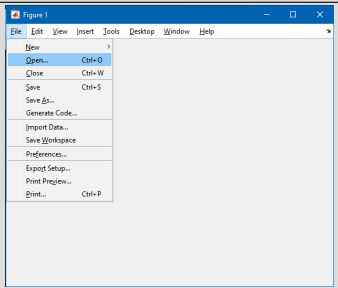
```
>> get(groot, 'DefaultAxesColorOrder')
% ans =
%         0    0.4470    0.7410
%    0.8500    0.3250    0.0980
%    0.9290    0.6940    0.1250
%    0.4940    0.1840    0.5560
%    0.4660    0.6740    0.1880
%    0.3010    0.7450    0.9330
%    0.6350    0.0780    0.1840
```

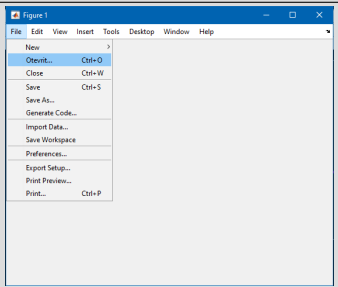


What Is Handle Good For?

- When having a handle, one can entirely control given object.
- The example below returns all identifiers existing in window `figure`.
- In this way we can, for instance, change item 'Open...' to 'Otevrit...'.
  - Or anything else (*e.g.* callback of file opening to callback of window closing :)).

```
hFig = figure('Toolbar', 'none');
allFigHndl = guihandles(hFig);
set(allFigHndl.figMenuOpen, 'Label', 'Otevrit...');
```



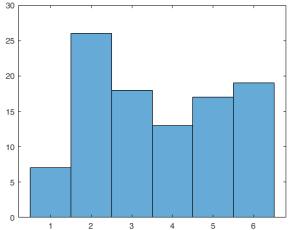


Exercises

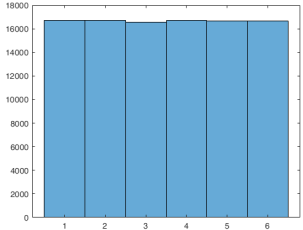
Exercise I.

- Create a script to simulate `L` roll of the dice.
  - What probability distribution do you expect?
  - Use `histogram` to plot the result.
  - Consider various number of tosses `L` (from tens to millions).

```
L = 1e2;
```



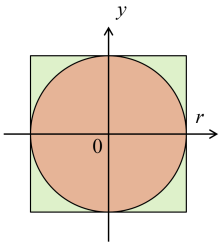

```
L = 1e5;
```



Exercise II.a

- Use Monte Carlo method to estimate the value of π.
  - Monte Carlo is a stochastic method using pseudo-random numbers.
- The procedure is as follows:
  - 1. Generate points (uniformly distributed) in a given rectangle.
  - 2. Compare how many points there are in the whole rectangle and how many there are inside the circle.

$$\frac{S_{\circ}}{S_{\square}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} \approx \frac{\text{hits}}{\text{shots}}$$



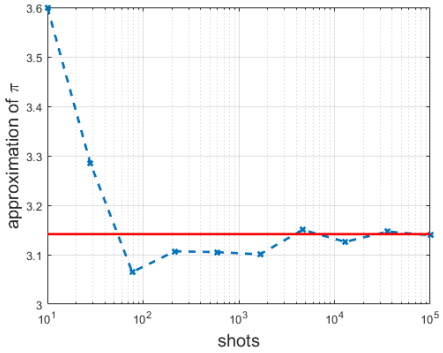
- Write the script in the way that the number of points can vary.
  - Notice the influence of the number of points on accuracy of the solution.

Exercise II.b

- Resulting code (circle radius $r = 1$):

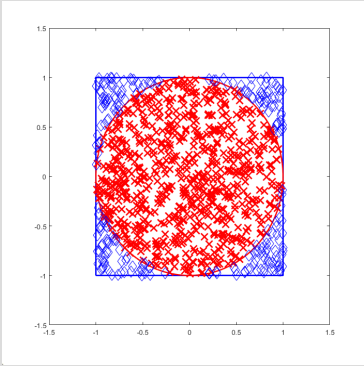
Exercise II.c

- Approximation of Ludolph's number - visualization:



Exercise II.d

- Visualization of the task:

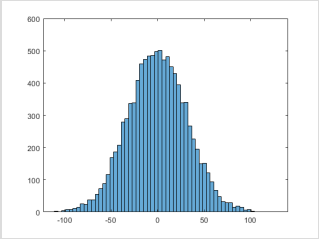


Exercise III.a

- Create a script to simulate `N` series of trials, where in each series a coin is tossed `M` times (the result is either head or tail).
  - Generate a matrix of tosses (of size `M` × `N`).
  - Calculate how many times head was tossed in each of the series (a number between 0 and `M`).
  - Calculate how many times more (or less) the head was tossed than the expected average (given by uniform probability distribution).
  - What probability distribution do you expect?
  - Plot resulting deviations of number of heads.
    - Use function `histogram`.

Exercise III.b

- Mean and standard deviation of `nOnesOverAverage`:



$$\mu = \frac{1}{N}\sum_i x_i \approx 0$$

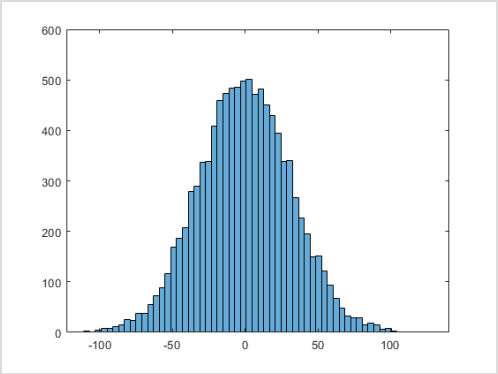
$$\sigma = \sqrt{\frac{\sum_i (\mu - x_i)^2}{N}} = \sqrt{1000} \approx 31.62$$

Exercise III.c

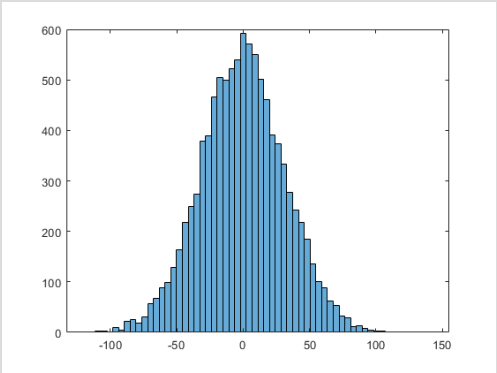
- To test whether we get similar distribution for directly generated data:

```
figure(2);
histogram(0 + 31.62*randn(N,1), 60);
```

Coin toss:



Directly generated data:

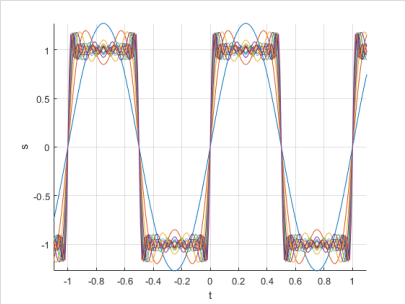


Exercise IV.

- Fourier series approximation of a periodic rectangular signal with zero direct component, amplitude A and period T is

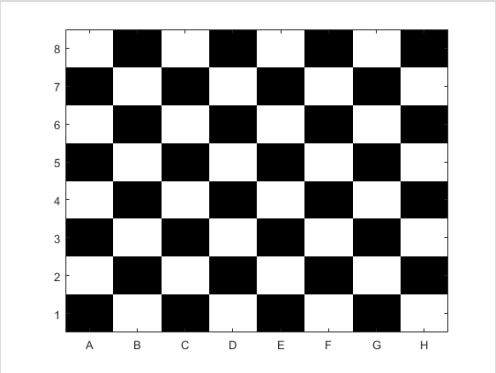
$$s(t) = \frac{4A}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin\left(\frac{2\pi t\,(2k+1)}{T}\right).$$

- Plot resulting signal $s(t)$ approximated by one to ten harmonic components in the interval $t \in [-1.1;\, 1.1]\,\mathrm{s}$; use $A = 1\,\mathrm{V}$ and $T = 1\,\mathrm{s}$.



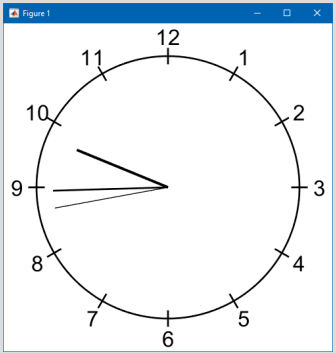
Exercise V.

- Modify the axes of the chessboard so that it corresponded to reality:



Exercise VI.a

- Create a script which shows a figure with a clock face showing actual time.
- To determine actual time use function `clock`.



Exercise VI.b

Questions?

BE0B17MTB – Matlab
matlab@elmag.org

November 11, 2020
Winter semester 2020/21

Acknowledgement: Filip Kozak, Pavel Valtr.