Visualisation

Visualisation

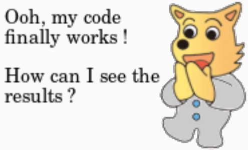
Scientific visualisation is the field that developps graphical tools to illustrate scientific results or data to help their understanding

As a general rule, any graphical representation of data should be

- Meaningful: it emphases the information present in the data
- Self-containing: the graphical output is intelligible by itself

In Matlab®, many visualisation tools are already implemented

- Vectorial tools *(2D and 3D plots, ...)*
- Statistical tools *(bar, pie, ...)*



Note: The set of instructions that extracts intelligible information from the *raw data* is called *post-processing*

Vector plots

Create a simple plot of a vector

1. Observe the behaviour of the following instructions

```
>> x=linspace(0,2*pi,200); y1=sin(x); y2=cos(x);
>> plot(y1)
>> plot(y2)
```

2. Try to plot each of the following and see what changes

```
>> plot(y1, y2)
>> plot([y1’,y2’])
>> plot([y1; y2])
```

Create a simple plot of a vector

1. Observe the behaviour of the following instructions

```
>> x=linspace(0,2*pi,200); y1=sin(x); y2=cos(x);
>> plot(y1)
>> plot(y2)
```

→ #y1 = 200 :0-200

2. Try to plot each of the following and see what changes

```
>> plot(y1, y2)
>> plot([y1',y2'])
>> plot([y1; y2])
```

$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{2\times 200}$ [: : : :]

plot (1, Y_{11}) (2, Y_{12}) ----

→ The command `plot` plots a given vector with respect to the vector's indices: from 1 to `length(y)`

→ It plots a given matrix by plotting each of its columns against the row indices

Vector plots

Create a simple plot of a vector

3. To plot against a vector of antecedents, include it in the arguments of the plot function. Plot with respect to the variable `x` by writing

```
>> x=linspace(0,2*pi,200); y1=sin(x); y2=cos(x);
>> plot(x,y1)
>> plot(x,y1,x,y2)
>> plot(x,[y1', y2'])
```

Vector plots

Create a simple plot of a vector

3. To plot against a vector of antecedents, include it in the arguments of the plot function. Plot with respect to the variable x by writing

```
>> x=linspace(0,2*pi,200); y1=sin(x); y2=cos(x);
>> plot(x,y1)
>> plot(x,y1,x,y2)
>> plot(x,[y1', y2'])
```

→ When two vectors are given, the second vector is plot against the first vector. If there is more arguments, this behaviour repeats itself until the end of the argument list.

→ When a vector and a matrix are given, each of the column vectors of the matrix are plot against the vector.

→ Pay attention to the order of the arguments!

Functions plots

Create an easy plot of a function

1. Plot an anonymous function evaluated on a vector

```
>> f = @(x) -x^2+2
>> x=linspace(0,1,10);  plot(x, f(x));
>> x=linspace(0,1,200); plot(x, f(x));
```

2. Plot an anonymous function between bounds

```
>> ezplot(f, [0, 1])    % Depreciated
>> fplot(f, [0,1])
```

Functions plots

Create an easy plot of a function

1. Plot an anonymous function evaluated on a vector

```
>> f = @(x) -x^2+2
>> x=linspace(0,1,10);  plot(x, f(x));
>> x=linspace(0,1,200); plot(x, f(x));
```

2. Plot an anonymous function between bounds

```
>> ezplot(f, [0, 1])    % Depreciated
>> fplot(f, [0,1])
```

→ Plotting by hand an anonymous function after evaluating it on a vector makes the resolution of the plot dependent on the predefined vector. Using a function designed for anonymous functions is usually preferable

Different plot scales

Use specific functions to plot on a different scale

1. Plot all the functions

$$f(x) = e^{3x},\ g(x) = x^x,\ h(x) = \log(x),\ j(x) = 3x$$

   on $[0.01, 100]$ using each of the following functions (use the command `help` to access their documentation)

   `plot` `semilogx` `semilogy` `loglog`

2. Which scale suits the best each function?

Different plot scales

Use specific functions to plot on a different scale

1. Plot all the functions
$$f(x) = e^{3x},\ g(x) = x^x,\ h(x) = \log(x),\ j(x) = 3x$$
on $[0.01, 100]$ using each of the following functions (use the command `help` to access their documentation)

```
plot    semilogx    semilogy    loglog
```

2. Which scale suits the best each function?

→ Those commands are useful when plotting logarithmic and exponential function, which are very common in error plots.

Multiple plots on one figure

Create several vectors or functions on a single plot

1. We already saw that `plot(x, y1, x, y2)` plots `y1` and `y2` with respect to `x` on the same figure. Try now

```
>> hold on
>> plot(x, y1)
>> plot(x, y2)
>> hold off
```

2. What would happen here?

```
>> close all; plot(x, y1)
>> hold on
>> for a = 1:10; plot(x, a.*x); end
>> hold off
```

Multiple plots on one figure

Create several vectors or functions on a single plot

1. We already saw that `plot(x, y1, x, y2)` plots `y1` and `y2` with respect to `x` on the same figure. Try now

```
>> hold on
>> plot(x, y1)
>> plot(x, y2)
>> hold off
```

2. What would happen here?

```
>> close all; plot(x, y1)
>> hold on
>> for a = 1:10; plot(x, a.*x); end
>> hold off
```

→ The commands `hold on` and `hold off` are useful when generating plots in a loop. Be careful where to specify them!

The attributes of a useful figure

For a figure to be meaningful and intelligible, thus *useful*, several features providing information on the plot have to be added

Title
: *The global title of your figure, telling what you are focusing on*

Colors and line typology
: *Identifies a specific plot in a figure containing many. By default, automatic colors are selected. The line typology (plain, dashed) further helps color-blinds and allows to print in black and white*

Legend
: *When several plots are in one figure, it makes clear which plot corresponds to which vector or function*

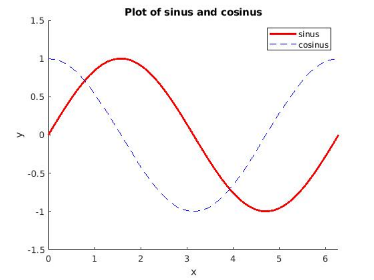
Axis labels
: *To know the used scaling, and the variable spanned on each axis*

Axis bounds
: *Crops the figure to the range of values given for each axis*

A dream plot

DreamPlot.m

```
%% Generate data  to plot
ref = 50;
x = linspace(0,2*pi,ref);
y1 = sin(x);
y2 = cos(x);


%% Creates a useful plot

% Label the figure itself
title('Plot of sinus and cosinus')

% Create the content
hold on
plot(x, y1, 'r-','linewidth',2)
plot(x, y2, 'b--')
hold off
legend('sinus', 'cosinus')

% Adapt the viewing area
axis([0 2*pi -1.5 1.5])

% Label the content
xlabel('x')
ylabel('y')
```



Defining multiple figures

When the visualization is desired in different figures, one can use the command `figure()` to draw the plot in a new figure

```
>> figure()          % Opens a new figure,
>> plot(x,y1)        % automatically indexed
```

It is also possible to select a particular figure and edit it later on

```
>> figure(1)            % Opens or jumps to the figure 1
>> plot(x,y1)
>> figure(2)            % Opens or jumps to the figure 2
>> plot(x,y2)
```

Multiple figures

Defining sub-figures

Sub-figures are used when the visualisation is wished in one single figure but should show distinct plotting spaces

`subplot(a,b,n)`: Creates and selects a sub-figure environment
`n` is the place where the plot is collocated
`a,b` are the dimension of the collocating matrix

```
>> figure(3)          % Create or jumps to the figure 3
>> subplot(1,2,1)     % Creates a 1x2 collocating matrix
>> plot(x,y1)         % and plots y1 at the position 1
>> subplot(1,2,2)     % Selects the second position
>> plot(x,y2)         % and plots y2
```

Saving figures

Saving a created figure can be done either through a graphical interaction or by script instructions

- By scripting, the main commands are `savefig` and `saveas`

```
>> fig = figure()   % Points to the desired figure
>> savefig('my filename0')
>> savefig(fig,'my filename1')
>> saveas(fig, 'my filename2')
```

To export the figure in a specific format, use an optional argument in the command `saveas`

```
>> saveas(fig,'my filename2','png')  % png, eps,...
```

- Graphically, go to File → Save As and select the format

Clearing figures

All the options to clear or close figures act on the main figures, regardless whether the figure contains sub-figures or not

- Clear the content of the figures, without closing them

```
>> clf
```

- Close the last figure

```
>> close
```

- Close all the figures

```
>> close all
```

Note: Always clear figures *after* having saved them, otherwise you will save a blank figure

3D Plots

1. Create a 3D line by entering

```
>> plot3(x,y1,y2)
```

2. Create a 3D Surface by using the commands `meshgrid` and `surf`:

```
>> [xx, yy]=meshgrid(x,x);
>> surf(xx,yy,sin(xx).*cos(yy))
```

3. What happens if we type `*` instead of `.*`?

3D Plots

1. Create a 3D line by entering

```
>> plot3(x,y1,y2)
```

2. Create a 3D Surface by using the commands `meshgrid` and `surf`:

```
>> [xx, yy]=meshgrid(x,x);
>> surf(xx,yy,sin(xx).*cos(yy))
```

3. What happens if we type `*` instead of `.*`?

→ The command `meshgrid` creates a grid of (x, y) coordinates upon two given scalar vectors

→ The command `surf` creates surface by interpolating given values at the corresponding grid points

→ We get a matrix product, which is not the desired result

Particular typologies of plots

Various of other types of plots are available *(see the Matlab® documentation)*. As a brief insight, the tools commonly used for statistics and 2D data visualisation are respectively

- Bar graphs

```
>> bar(rand(1,10))
```

- Pie chart

```
>> pie(rand(1,5))
>> pie3(rand(1,5))
```

- Scatter plot

```
>> scatter(y1, y2)
```

- Mesh visualisation

```
>> mesh(peaks)
```

- Contour visualisation

```
>> contour(peaks)
```

- Color fill

```
>> pcolor(peaks)
```

Note: The plotting functions have options, described in the help

Animation

Create your first animation!

1. Put together multiple plots and store the corresponding frames in an array

```
clear M
x=linspace(0,2*pi,100); y=sin(x);
n=120;
for k=1:n
      plot(x,y*sin(pi* k/n))
      axis([0,2*pi,-1,1])
      M(k)=getframe;
end
```

2. Launch your movie with `movie(M,2)`

Animation

Save your first animation

1. Save your first animation with `VideoWriter`

```
writerObj = VideoWriter('myvid.avi');
open(writerObj);

for k=1:n
     plot(x,y*sin(pi* k/n))
     axis([0,2*pi,-1,1])
     writeVideo(writerObj, getframe)
end

close(writerObj)
```

Best practice

- Provide a complete description of your graphics, with a title, plots and axes labels, ...
- Use markers and line properties so that your plots can be also differentiated when printed in black and white
- Always save the figures *before* clearing them
- Always clear your figures when switching exercise

Exercises

Exercises: Visualisation

Exercise

1. Plot in one figure the functions

   $f(x) = e^{x/10}\sin(2\pi x)$ and $g(x) = log(3 + x)\cos(4\pi x)$

   on the interval $[0, 1]$. The plot be such that:

   a) f is plotted in red colour and dashed lines
   b) g is in blue and it is alternating dots and dashes
   c) It should contain a title "Cute functions"
   d) The x axis ranges from 0 to 1 and is labelled "Time"
   e) The y axis ranges from -2 to 2 and is labelled "Money"
   f) Specify a legend: f relates to "Marc" and g to "John"

2. Save the plot as `"my_first_functions.fig"` and close the figure (using the functions we learned).

Exercises: Visualisation

Exercise

3. Given the serie $\sum_{k=1}^{\infty} \frac{1}{2^k} = 1$
   a) Plot the partial sums $A_n = \sum_{k=1}^{n} \frac{1}{2^k}$ with respect to n and an horizontal line at the hight of the limit 1
   b) Plot the error $e_n = |1 - \sum_{k=1}^{n} \frac{1}{2^k}|$ with respect to n by choosing the more appropriate norm

4. Plot the first 4 (ν=0,1,2,3) Bessel-functions of 1st and 2nd typology. Hint: use the `besselj`, `bessely` functions
   a) Plot all the functions of the first type in a same figure, and all the functions of the second type in an other one
   b) The $\nu = 1, 2$ functions must be put in a 1×2 array of plots, with corresponding descriptions and labels
   c) Change the x-axis to be $[0.2, 20]$ and the y-axis $[-1, 1]$

Exercise

5. Check `help plot`. Create then the plots of:

   a) $f = x^2 - 0.5$ on $[-1, 1]$ with red dashed lines

   b) $f = sin(2 * pi * x)$, where the function values have to be displayed as little black circles at the points $x = n/10$.

   c) the functions $f = sin(s * x)$, $s = 1, 2, 3, 4$ with different colors and a legend

Exercises: Visualisation

Exercise

6. a) Plot the first eigenmodes of a quadratic cymbal with the length of its side 1
   b) Create a video of an overlay of two eigenmodes. Hint: The eigen-oscillations of the trumpet are given through the formula:

$$sin(m \cdot \pi \cdot x) \;\; sin(n \cdot \pi \cdot y) \;\; sin(c\sqrt{m^2 + n^2}t + \varphi)$$