Unit 1

September 10, 2015

Francesca Maria Marchetti
Departamento de Fisica Teorica de la Materia Condensada
Facultad de Ciencias
Office : C-V 606
Phone: 91 497 5590
E-mail: francesca.marchetti@uam.es
Web: http://www.uam.es/francesca.marchetti

Complementary material, notes, exercises and **solutions** will be posted on the web-page of this course:
http://www.uam.es/personal_pdi/ciencias/fmarchet/computacion1_15-16.html.
Please do check it regularly.

Note that these notes are in English but classes will be in Spanish; any doubt you have you can come to my office (writing to me in advance to pick up a day and time is desirable but not compulsory), as well as you are encouraged to ask as much as you like during classes.

1 Basic concepts

1.1 Matlab as a calculator

Matlab can perform all the calculations available in a common calculator.

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- ^ (power)

In addition, in Matlab there are several mathematical functions already built-in and ready to use. Among those:

- `sin`
- `cos`
- `log`

- `tan`
- `exp`
- `sqrt`
- ... try to find out what mathematical functions are available in Matlab.

There are useful commands to start with

- `help` (try `help help`)
- `more on`
- `format` (try `help format`)

1.1.1 Exercise:

1. perform basic calculations such as `12-3/2`, `(12-3)/2`, `exp(3+5)`, `exp(3)+5`, ..., and get familiar with the correct use of parenthesis;
2. compare `-5^2` with `(-5)^2`;
3. compare the different formats you can have for the answers of your calculations (type `help format` for informations); explain what you get;
4. evaluate `sin(5.2)/cos(5.2)`, compare with `tan(5.2)`
5. evaluate `asin(sin(pi/2))`; is the answer given in radians or degrees? How do you switch between one and the other?
6. evaluate `sqrt(2)` and `sqrt(-2)` and explain what you get.

Note that one can equivalently write either `0.00078` or `7.8*10^(-4)`; in addition in Matlab one can use the shortcut `7.8e-4`.

1.2 Variables

Matlab can store numbers inside variables; for example, consider

```
a=1;
b=4;
a+b
```

Note that the computer 'reads' from right to left, i.e., `a=1` is 'read' by the computer as the following: assign the value `1` to the variable `a`; you cannot write `1=a` (try it and see what you get). When you define variables, for examples by defining `b=4`, the computer stores the number `4` in a part of memory which is called `b`. Until you somehow 'clean' that part of the memory `b` will be `4`. During calculations, often it is useful to use commands such as `clear`, `clear all`, `who`, `clc` (use the `help` command to understand their meaning and use).

1.2.1 Exercise:

1. get familiar with the commands `clear`, `clear all`, `who` (i.e., type `help clear`, ...) and explain what they are useful for;
2. explain what the variable `ans` is;
3. repeat the exercises in Ex. 1.1.1 now using variables.

Note that there are predefined variables in Matlab. For example,

- `ans`
- `pi`
- `i` and `j`
- `eps`

What do they correspond to? The names of these pre-defined variable should be avoided. Otherwise, you can give your variables the name you like — use of meaningful variable names is advised.

1.2.2 Exercise:

1. Define a variable `rad2deg` which converts radians in degrees; evaluate `asin(sqrt(3)/2) * rad2deg` and check it gives 60°.

Variables can also store complex numbers, as in the following examples

```
z=10.2+i*2.3;
x=10.2+j*2.3;
y=complex(10.2,2.3);
```

Are the three variables `z`, `x`, and `y` the same complex number? With the use of the `help` command, discover the role of the following functions operating on complex numbers:

- `abs`
- `complex`
- `conj`
- `imag`
- `real`

2 Vectors and Matrices

Matlab can also store several numbers in a vector. For example you can define the **row vector**

```
a=[3 5 9 2 11 7];
```

and access the individual elements of the vector by using the **indices**: `a(1)` stores 3, `a(2)` is 5, `a(3)` is 9, `a(4)` is 2, `a(5)` is 11, `a(6)` is 7. Try, typing

```
a(1)
```

Note that **an index can only be a natural number (i.e., a counting number!!), 1, 2, 3, etc. etc.** Try typing either `a(0)` or `a(-1)` and see what you get. Similarly `a(7)` will give you error; why?

You can also define **column vectors**

```
at=[3; 5; 9; 2; 11; 7];
```

check that the indexing will be exactly the same as above for the row vectors, i.e. say `at(3)` is 9. You can go from column to row vectors and the other way around via the transposition $'$; for example `at'` is `a` and `a'` is `at`. Check it on your own.

Matrices are defined similarly to vectors — after all a raw vector is a $n \times 1$ matrix and a column vector is a $1 \times n$ matrix. This is done by using a comma separated list of column vectors, or a semicolon separated list of row vectors. For example:

```
M=[1.3, 5.2, 9.3; 4.2, 4.5, 6.1]
```

defines the matrix

$$M = \begin{pmatrix} 1.3 & 5.2 & 9.3 \\ 4.2 & 4.5 & 6.1 \end{pmatrix} .$$

In order to access the single elements of a matrix you have two options:

1. either by using `(i,j)` indices, where `i` is the index in the row, and `j` in the column; i.e., in the previous example
$$\begin{pmatrix} M(1,1) & M(1,2) & M(1,3) \\ M(2,1) & M(2,2) & M(2,3) \end{pmatrix} ;$$
2. or by using a single index. In this case, the order is column major, meaning you first go through all elements of the first column, then the second column, etc... — for example `M(1,2)` is 2 and coincides with `M(3)`; i.e.,
$$\begin{pmatrix} M(1) & M(3) & M(5) \\ M(2) & M(4) & M(6) \end{pmatrix} ;$$
Try it yourself.

In reality every variable in Matlab is a matrix: `a=0.2` is a 1×1 matrix, `a=[3 5 9 2 11 7];` is a 1×6 matrix, its transposed `at=a'` is a 6×1 matrix, `M=[1, 2, 3; 4, 5, 6]` is a 2×3 matrix and so on.

Try the following commands on the previously defined matrices and explain their use:

- `size(M)`
- `diag(M)`
- `zeros(3)`
- `eye(5)`
- `ones(2)`
- `find(M>2.2)`

2.0.3 Exercise:

Define the matrix `M=[1.1, 12.4, 3.6; 7.5, 2.4, 7.8; 10.3, 33.5, 18.4]` and:

1. consider the new vector `index=find(M>5.0)`;
2. what is in practice this vector `index`?
3. what do you get if you evaluate `N=M(index)`? Which kind of vector/matrix is now `N`?

2.0.4 Exercise:

Define the matrix `M=[4, 12, 23; 2, 1, 3; 4, 2, 1]` and:

1. evaluate its `(2,3)`-element in two different ways;
2. What do you get if you consider `a=M(:)`? Is now `a` a scalar or a vector? If it is a vector, which kind of vector? What do you get if you evaluate `a(3)`?
3. What do you get if you consider `b=M(2,:)`? Is now `b` a scalar or a vector? If it is a vector, which kind of vector? Can you evaluate `b(3)` and `b(4)`?
4. What do you get if you consider `c=M(1,1:2)`? Is now `c` a scalar or a vector? If it is a vector, which kind of vector?
5. try `help colon`;
6. try `help length`.

2.1 Vector and matrix operations

There are several mathematical operations you can perform with vectors and matrices:

- `+`
- `-`
- `*`

- `.*`
- `./`
- ^
- .^

Matlab distinguishes between mathematical and numerical vector products; for example considering the following two vectors

`a`= [1.5 6.2 4.4] `b`= [2.1 9.4 7.1]

- `*` is a **mathematical vector multiplication** and the result is either a scalar or a matrix. For example `a*b'` gives as a result a scalar corresponding to $a(1) * b(1) + a(2) * b(2) + a(3) * b(3)$; in fact mathematically one multiplies matrices with the "row times column" rule:

$$\begin{pmatrix} a(1) & a(2) & a(3) \end{pmatrix} \begin{pmatrix} b(1) \\ b(2) \\ b(3) \end{pmatrix} = a(1) * b(1) + a(2) * b(2) + a(3) * b(3) \ .$$

  The operation `a*b'` is equivalent to `dot(a,b)`. Instead `a'*b` is a matrix which elements are $a(i) * b(j)$, i.e., following the same "row times column" mathematical rule:

$$\begin{pmatrix} a(1) \\ a(2) \\ a(3) \end{pmatrix} \begin{pmatrix} b(1) & b(2) & b(3) \end{pmatrix} = \begin{pmatrix} a(1)b(1) & a(1)b(2) & a(1)b(3) \\ a(2)b(1) & a(2)b(2) & a(2)b(3) \\ a(3)b(1) & a(3)b(2) & a(3)b(3) \end{pmatrix} \ .$$

  Finally, `a*b` and `a'*b'` are not valid operations, as you can check.

- `.*` is instead a **component-wise multiplication** (there is no corresponding mathematical operation) and the result is a vector. For example `a.*b` is a row-vector which components are $a(i) * b(i)$, i.e., in the example above:

$$\texttt{a.} * \texttt{b} \quad \text{gives} \quad \begin{pmatrix} a(1)b(1) & a(2)b(2) & a(3)b(3) \end{pmatrix} \ ;$$

  similarly `a'.*b'` is the same vector but now column:

$$\texttt{a}'\texttt{.} * \texttt{b}' \quad \text{gives} \quad \begin{pmatrix} a(1)b(1) \\ a(2)b(2) \\ a(3)b(3) \end{pmatrix} \ ;$$

  The two operations `a'.*b` and `a.*b'` are instead not allowed.

The best way to understand how the above operations work is by trying via several examples.

2.1.1 Exercise:

Consider the following matrices

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 5 & 9 & 12 \end{pmatrix} \qquad B = \begin{pmatrix} 4 & 2 & -7 \\ 9 & 2 & 0 \end{pmatrix} \qquad C = \begin{pmatrix} 2 & 5 \\ -3 & 2 \\ 5 & -9 \end{pmatrix}.$$

1. Consider the following operations `A.*B`, `A.*C`, `A.*C'`, `B.*C`, `B.*C'`, `A*C`, `C*A`, `A*B`, `A*B'`, and determine which of these operations is valid and explain the result;
2. explain what is the difference between the operation `*` and `.*`. When can you use one and when the other?
3. evaluate `A.^2` and explain the result; why you cannot consider `A^2`?

2.1.2 Exercise:

Consider the following vectors

`a`= [1.5 6.2 4.4] `b`= [2.1 9.4 7.1]

1. explain the operations `norm`, `dot`, and `cross`;
2. find an equivalent definition of `norm(a)` using the function `sqrt` and the operation `*` ;
3. how else can you define `dot(a,b)`?
4. find the components of the vector `cross(a,b)`.

2.2 Commands which define vectors: : and `linspace`

Matlab can generate vectors containing equally spaced values in two different ways:

- `x1=(0:2:10)`: in this case you are asking the computer to define a vector `x1` which goes between 0 and 10 in steps of 2;
- `x2=linspace(0,10,6)`: now you are asking to define a vector with 6 elements equally spaced (N.B. $10/(6-1) = 2$) which goes from 0 to 10.

You can check the two vectors `x1` and `x2` coincide.

2.2.1 Exercise:

Define a vector which goes from 0 to 10 in steps of 0.5 in two different and equivalent ways, using the commands `:` and `linspace` defined above.

In general, if in an interval $[a, b]$, you want to generate a grid of equally spaced N numbers x_i, as in Fig. 1, you can use equivalently the following equivalent

definitions:

definition 1	$\texttt{x} = \texttt{linspace}(\texttt{a}, \texttt{b}, \texttt{N})$
definition 2	$\texttt{y} = [\texttt{a} : (\texttt{b} - \texttt{a})/(\texttt{N} - 1) : \texttt{b}]$

where we indicate the components of each vector as $x(i) = x_i$ and $y(i) = y_i$. The vectors `x` and `y` are exactly the same; try it with different values of starting-point `a`, end-point `b`, and number of points on the grid `N`. How do you access the single components of the vectors `x` and `y`? With an index vector $i = 1, 2, \ldots, N$, so that `x(i)` and `y(i)` will give you the component:

$$x(i) = y(i) = a + (i-1)\frac{b-a}{N-1} \, .$$

Here, as it is shown in Fig. 1, the value $\delta x = \frac{b-a}{N-1}$ is the value of the interval step on the grid.

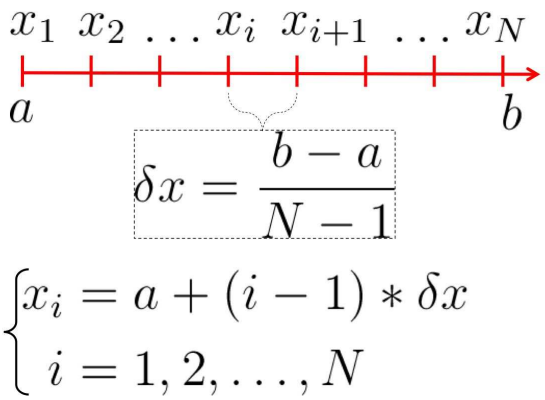

Figure 1: Meaning of the equivalent commands `x=linspace(a,b,N)` and `[a:(b-a)/(N-1):b]` used to define vectors of N equally spaced numbers. Here the components of the vector `x` are indicated as $x(i) = x_i$.

Note that often is useful to generate an index vector to address certain values in a matrix or in a vector. For example, consider the following vector `y=(0:2:100)`, and now define the index `i=(1:10:51)`; what do you get by evaluating `y(i)`? Explain the result.

Finally note that instead than `linspace` you can use `logspace`. Type `help logspace` to understand the role of this command. The syntax is `x = logspace(log10(a), log10(b), N)`. Show that the two definitions

```
x = logspace(log10(a), log10(b), N)
y = 10.^(linspace(log10(a),log10(b),N))
```

are equivalent.

3 Scripts

Scripts are collections of Matlab commands stored in plain text files. When you run a Matlab script, the commands in the script file are automatically executed

as if you had typed them in from the keyboard. Script files must end with the extension ".m" (for example "myScript.m"), and often these files are referred to as m-files. A script can be thought of as a keyboard macro: when you type the name of the script, all of the commands contained in it are executed just as if you had typed these commands into the command window. Thus, all variables created in the script are added to the workspace for the current session. Furthermore, if any of the variables in the script file have the same name as the ones in your current workspace, the values of those variables in the workspace are changed by the actions in the script. This can be used to your advantage. It can also cause unwanted side effects.

Script files are usually created with a plain text editor. You can also use the Matlab `diary` command to record commands as you type.

Example of script:

```
%------------------------------%
% Ex.1 on matrices and vectors %
%------------------------------%
% everything written after the % is a
% comment and is not executed
clear all
A=[2,1,3; 5,4,3]
%element (1,3)
A(1,3)
%compare with
A(5)
%vector a2 equal to the second row of the matrix A;
a2=[A(2,(1:1:3))]
%equivalently
a2=A(2,:)
sort(a2)
b2=a2( find(a2>3.5) )
```

To execute the script you can also press Control-Enter. Also you can separate various independent parts of a script which you want to run independently with `%%`. The script page splits and you run the part of the script which is highlighted. From now onwards, you are asked to write scripts rather than typing on the command window. Later on in the course you will be explained the use and advantage of external functions.

3.1 Additional exercises on vectors and matrices

Solve the following exercises by writing scripts

3.1.1 Exercise:

The number e can be equivalently defined as

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = \sum_{n=0}^{\infty} \frac{1}{n!} \, .$$

Find an estimate of e by using both the definitions given above and compare them with the built-in value of Matlab (or `exp(1)`) — for summing the vector components you can use the command `sum` or you can find an equivalent way of doing it by using the multiplication operation of appropriate vectors.

3.1.2 Exercise:

Define the matrix `A=[1,2,3; 5,4,3]` and:

1. evaluate its `(1,3)`-element;
2. define a vector $a2$ equal to the second row of the matrix A;
3. sort $a2$ in ascending order;
4. define a new vector $b2$, which contains only the elements of $a2$ bigger than 3.5.

Hints:

- Use the command `sort(a2)`;
- Use the command `find(a2>3.5)`;

3.1.3 Exercise:

Define two row vectors a and b of 4 elements each: a has the first even numbers (2,4,6,8) and b the first odd numbers in reverse order (7,5,3,1) — use a different definition than the trivial one!

1. Find two equivalent ways to define the vector dot product between the two vectors ($\sum_{i=1}^{4} a(i)b(i)$);
2. Find two equivalent ways to define the modulus of each vector;
3. Evaluate the angle between a and b in radians and degrees;
4. Describe which kind of matrix/vector one gets by considering `a*b'`, `a'*b`, `a.*b`, `(b.*a)'`.

Hints: Use the commands `dot(a,b)` and `norm(a)`.

3.1.4 Exercise:

Generate a vector c going from 99 down to 0 in steps of 3, extract every 10th element of that vector and generate an index vector which gets the first and last element of c.

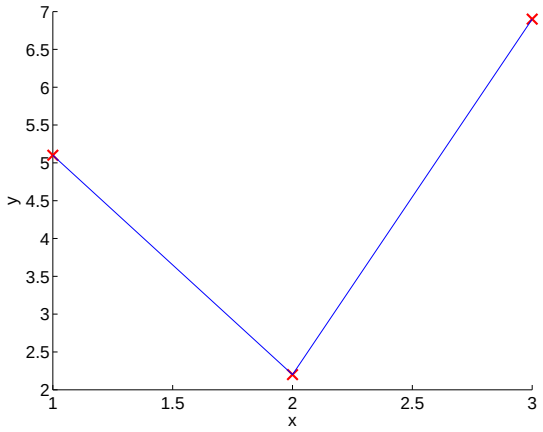

Figure 2: Plot of the two vectors `x=[1 2 3]` and `y=[5.1 2.2 6.9]`, obtained with the command `plot(x,y)`.

Hints Use the command `length(c)`.

4 Plotting

Plotting with Matlab is very easy and quick. For functions of a single variable (say, $y = y(x)$) the command is `plot(x,y)`, by having previously defined both vectors x and y, with the same size, i.e., the same number of components, `x=[x1 x2 ... xN]` and `y=[y1 y2 ... yN]`. Then, the command `plot(x,y)` creates a sequence of points in the x-y plane corresponding to the values $(x1, y1), \ldots (xN, yN)$ and joints them with lines — see the examples shown in Fig. 2. The visual result is therefore the graph of the segmented line representing the function $y = y(x)$ in the interval $x \in [x1, xN]$; the more points N you put, the less segmented the line will look like. You can also choose to represent the line with symbols of different shape and colors rather that with a segmented line. Use the command `help plot` in order to learn more about plotting functions and its various options.

For example, if you want to plot the function $y = \cos(x)$ in the interval $x \in [0, 2\pi]$, you can write:

```
a=0; b=2*pi; N=100;
x=linspace(a,b,N);
y=cos(x);
plot(x,y)
```

You can plot two functions on the same graph, use different colors and line types, add legends, axis labels, a title to the plot,

```
a=0; b=2*pi; N=100;
x=linspace(a,b,N);
y1=cos(x);
y2=sin(x);
hold on
plot(x,y1,'b--')
plot(x,y2,'r-','LineWidth',2)
title('Sine and Cosine functions')
legend('cos','sin')
xlabel('x','fontsize',16)
ylabel('y(x)','fontsize',16)
hold off
```

You can generate different figures in different windows

```
figure(1)
hold on
plot(x,y1,'r-','LineWidth',2)
title('Cosine function')
xlabel('x','fontsize',16)
ylabel('y(x)','fontsize',16)
hold off
figure(2)
hold on
plot(x,y2,'r-','LineWidth',2)
title('Sine function')
xlabel('x','fontsize',16)
ylabel('y(x)','fontsize',16)
hold off
```

Note that while `figure(i)` opens a new window for a plot, the command `close(i)` will close such a window. The command `close all` closes all previously opened windows.

The axis range can be fixed with `axis` — e.g., in the previous examples, try `axis([0 pi -1 1])` for `figure(1)` and `axis([0 pi 0 1])` for `figure(2)`.

4.0.5 Exercise:

Write a script which plots the function $f(x) = x\exp(-x)$ in the interval $[a, b] = [0, 10]$ with $N = 200$ equally spaced points both with a solid continuous line and with symbols. Add the axis labels. Where are the parts of the curve where the symbols are less dense and where are more dense? Why?

Error bars can be added in the plot by making use of the command `errorbar(x,y,error)`. Here, `x` and `y` are, as above, the vectors (of the same length) we want to plot as pair of variables, i.e., as symbols associated to the coordinates `(x(i), y(i))`, while the vector `error` (which also has the same length as `x` and `y`) contains the error `error(i)` associated to each pair `(x(i),`

`y(i))` and will be plotted as a bar of length `2*error(i)` around each symbol. Try the following example (the function `randn()` is used to return a matrix of random numbers and will be introduced later in unit 4):

```
x = linspace(0,10,10);
y = x.*exp(-x);
e=0.1*ones(1,length(x));
x1=linspace(0,10,100);
y1=x1.*exp(-x1);

figure(1)
hold on
errorbar(x,y,e,'or')
plot(x1,y1)
hold off

e = 0.05*randn(1,length(x));
figure(2)
hold on
errorbar(x,y,e,'or')
plot(x1,y1)
hold off
```

In addition, `subplot` generates multiple plots in one window. This work with the syntax `subplot(M, N, i)`, i.e., it generates an $M \times N$ matrix of subplots and plot the i-th one (ordered by fixing the raw first).

4.0.6 Exercise:

Write a script which plots $x(t) = vt + A\cos(\omega t)$ and $y(t) = A\sin(\omega t)$ as a function of $t \in [0, 10\pi]$ (fixing the frequency to $\omega = 1$) on the same subplot. Generate 4 sublots for different values of the coefficients A (amplitude) and v: in two subplots fix A and change v and in the other two subplots fix v and change A. Generate a second figure with 4 subplots where you plot the trajectories $y(x)$ using the same parameters values of the previous subplots. Comment about the role of the parameters A and v.

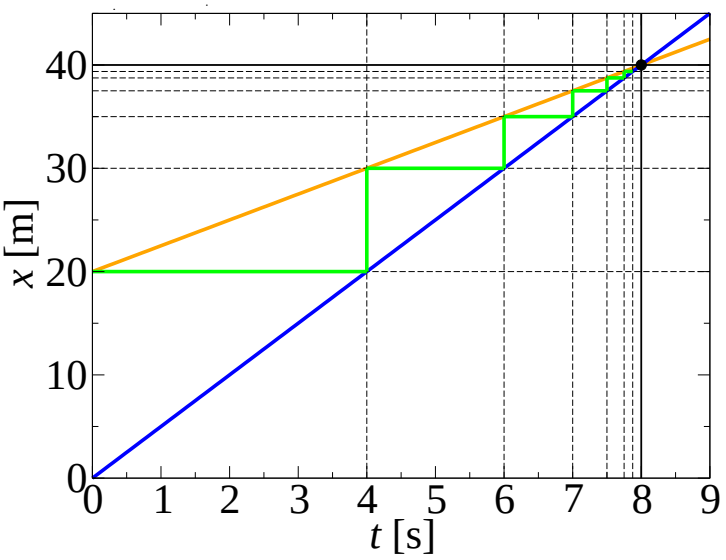

Figure 3: Graphical representation of Zeno's paradox of "Achilles and the Tortoise". Achilles trajectory is plotted in blue and the tortoise one is plotted in orange.

4.0.7 Exercise: "Achilles and the Tortoise" (Zeno's paradox)

"In a race, the quickest runner can never overtake the slowest, since the pursuer must first reach the point whence the pursued started, so that the slower must always hold a lead."

Zeno's paradox about "Achilles and the Tortoise" can be formulated in the following way: Achilles and the Tortoise compete in a race. The Tortoise (which position as a function of time is plotted as an orange line in Fig. 3) starts (at a lower speed) 20 m ahead of Achilles (who will run at a higher speed). Achilles reaches the 20 m distance after 4 s, but at that time the Tortoise is at a $30(= 20+10)$ m distance; Achilles reaches the 30 m distance after $6(= 4+2)$ s, when the Tortoise reaches the $35(= 20+10+5)$ m distance. Will Achilles and the Tortoise ever meet? According how the problem is formulated the answer is paradoxically no.

1. Formulate mathematically the problem above, complete the (geometrical) series and find an estimate of time and distance at which Achilles finally meets the Tortoise;
2. evaluate the velocity at which Achilles and Tortoise respectively run (not very realistic for the Tortoise!);
3. create a plot similar to Fig. 3 and find graphically the point at which the two trajectories meet and compare it with your numerical estimate.

Moral: paradoxes might be solved by introducing the concept of limit.

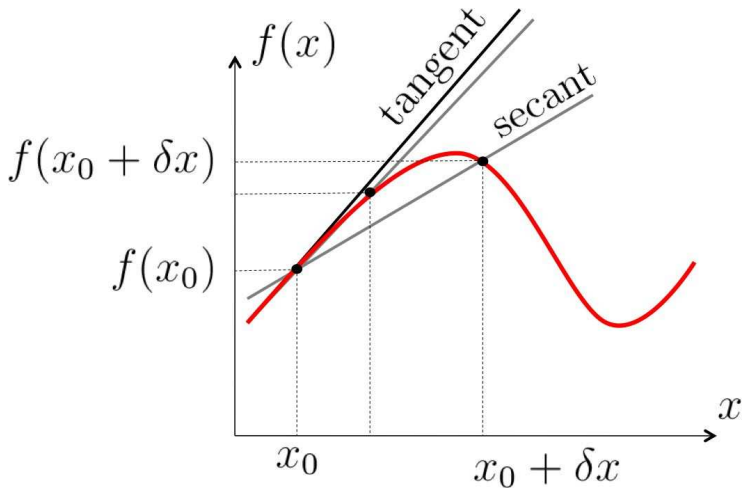

Figure 4: Schematic figure showing the meaning of first derivative of a function $f(x)$ at a given point x_0 and its first approximated value.

4.0.8 Exercise:

It is in general convenient to plot power-laws ($f(x) = x^n$) in logarithmic scale, as in the following example

```
x=logspace(log10(1), log10(1000), 100)
plot(log10(x), log10(x.^3), 'o')
loglog(x,x.^3,'+')
```

1. What is the difference between `plot()` and `loglog()`?
2. Can you determine the power-law exponent from the plots in logarithmic scale?
3. What happens if you use `linspace()` rather than `logspace()`?

5 Numerical differentiation

The derivative of a single variable function, $f(x)$, at a given point x_0 is defined as the following limit

$$\left.\frac{df(x)}{dx}\right|_{x=x_0} \equiv f'(x_0) \equiv \lim_{\delta x \to 0} \frac{f(x_0 + \delta x) - f(x_0)}{\delta x} , \tag{1}$$

i.e., is the slope of the tangent of $f(x)$ in x_0. If we want evaluate the derivative of a function at a given point we can thus, as a first approximation, use the finite difference approximations, i.e., compute the slope of a nearby secant line passing through the points $(x_0, f(x_0))$ and $(x_0 + \delta x, f(x_0 + \delta x))$ (see Fig. 4):

$$f'(x_0) \simeq \pm \frac{f(x_0 \pm \delta x) - f(x_0)}{\delta x} + O(\delta x) . \tag{2}$$

The slope of the secant line differs from the slope of the tangent line at x_0 by an amount that is approximately proportional to δx: this is the meaning of the symbol $O(\delta x)$, i.e., by using Eq. (2) in order to estimate numerically the derivative of the function $f(x)$ in x_0, we are making an error of the order of the increment δx — for smaller and smaller increments this error will be smaller and smaller and the slope of the secant line approaches the slope of the tangent line, which is the exact derivative.

How the definition (2) is implemented in practice? Let us consider the following example, where we evaluate the derivative of the function $f(x) = x^2 + 5x$ at $x_0 = 2$:

```
% Example
% derivative of f(x)=x^2+5x at x0=2
x0=2;
deltax=0.1;
f2=(x0+deltax)^2+5*(x0+deltax);
f1=(x0)^2+5*(x0);
% approximated numerical value
fpx0=(f2-f1)/deltax
% compare with exact value
fpx0exact=2*x0+5
```

5.0.9 Exercise:

Plot the value of the numerical derivative of the function $f(x) = x^2 + 5x$ at $x_0 = 2$ for different values of the increment δx; establish how fast the numerical value converges to the exact one, $f'(2) = 9$ and comment the result you get.

As well as evaluating the numerical derivative of a function $f(x)$ in one particular point, we can also evaluate it in a given interval and plot it. Note that, if a function is defined on the interval $[a, b]$ on a grid of N points, as in Fig. 1, then the numerical derivative will be defined on a grid with $N - 1$ points corresponding to either the interval $[a, b - \frac{b-a}{N-1}]$ or $[a + \frac{b-a}{N-1}, b]$. Consider the following example:

```
% Example
% derivative of f(x)=x^2+5x defined on the interval [a,b]
a=0; b=10; N=10;
x=linspace(a ,b, N);
f=x.^2+5*x;
il=(1:1:N-1); ir=(2:1:N);
deltax=x(ir)-x(il);
deltaf=f(ir)-f(il);
fp=deltaf./deltax;
plot(x(il),fp,'ro', x(ir),fp,'gx', x,2*x+5,'b')
legend('approx 1', 'approx 2', 'exact','Location','NorthWest')
xlabel('x'), ylabel('f')
```

5.0.10 Exercise:

Understand the example above and repeat it for the function $f(x) = e^x$ in the interval $[0, 10]$ and compare the numerical approximate derivative with the exact result.

6 Higher order approximations

One can do better than using the approximation (2) and instead consider

$$f'(x_0) = \frac{f(x_0 + \delta x) - f(x_0 - \delta x)}{2\delta x} + O(\delta x^2) \,, \tag{3}$$

where it can be shown that now the error made is smaller than in the previous approximation in (2) — note that, for $\delta x < 1$ then $\delta x^2 < \delta x$!

In fact in your analysis course you will study the Taylor expansion

$$f(x_0 + \delta x) \simeq f(x_0) + f'(x_0)\delta x + \frac{f''(x_0)}{2}(\delta x)^2 + O(\delta x^3)$$
$$f(x_0 - \delta x) \simeq f(x_0) - f'(x_0)\delta x + \frac{f''(x_0)}{2}(\delta x)^2 + O(\delta x^3) \,.$$

From these expressions, by taking the difference, and dividing by δx, you can obtain the expression (3) and the demonstration that, by evaluating the approximated first derivative $f'(x_0)$ the error is of order $O(\delta x^2)$.

This higher order approximation is implemented as the following

```
% Example on higher order approximation for
% the derivative of f(x)=x^2+5x at x0=2
clear all
close all
clc
x0=2;
deltax=0.1;
f2=(x0+deltax)^2+5*(x0+deltax);
f1=(x0)^2+5*(x0);
f3=(x0-deltax)^2+5*(x0-deltax);

% approximated numerical value
fpx0=(f2-f1)/deltax
% higher order:  comment:  why now is already exact?
fp2x0=(f2-f3)/(2*deltax)
% compare with exact value
fpx0exact=2*x0+5
```

6.0.11 Exercise:

Comment why in the case of the function $f(x) = x^2 + 5x$, the higher order approximation (3) to the derivative, already gives the exact result.

6.0.12 Exercise:

Plot the value of the numerical derivative of the function $f(x) = x^3 + 5x^2$ at $x_0 = 2$ for different values of the increment δx; establish how fast the numerical value converges to the exact one $f'(2) = 32$ by using both the approximations of Eqs. (2) and (3), and comment the result you get.

Similarly to what seen before in the linear approximation, if we want to plot the derivative in the higher order approximation (3) in a given interval we can do the following:

```
% Example:  derivative of f(x)=x^2+5x
% defined on the interval [a,b]
clear all
a=0; b=10; N=10;
x=linspace(a ,b, N);
f=x.^2+5*x;
il=(1:1:N-1); ir=(2:1:N);
deltax=x(ir)-x(il);
deltaf=f(ir)-f(il);
fp=deltaf./deltax;
plot(x(il),fp,'ro', x(ir),fp,'gx', x(il)+deltax/2,fp,'ks',
x,2*x+5,'b')
legend('approx 1', 'approx 2', 'approx 3',
'exact','Location','NorthWest')
xlabel('x'), ylabel('f')
```

Note that in the case of the derivative of functions which are polynomial of degrees 2 or less (such as $f(x) = x^2 + 5x$), the definition (3) will coincide with the correct answer. This will be clear once you will understand the Taylor expansion.

6.0.13 Exercise:

Evaluate the derivative of $f(x) = x^2+5x^5$ in the interval $[a, b] = [0, 10]$, with a grid of $N = 20$ points equally spaced. Use both definitions (2) and (3), plot the derivative obtained both ways and compare the results.
Questions:

1. If the function $f(x)$ is defined on a grid of N points, i.e., is a vector with N elements, how many elements will contain its derivative?
2. Describe the difference between the definitions (2) and (3);
3. How does the command `diff( )` work?

Hints to solve Exercise 6.0.13

- Remember the command `linspace(a ,b, N)` to define the vector x; if instead you use `x=a:?:b` what do you have to choose for `?` in order to have the same vector as before? (see Fig. 1);

- In order to define the derivative, you can define index vectors
  `ileft=(1:1:N-1)`
  `iright=(2:1:N)` for example to address `x(ileft)` and `x(iright)` and define the derivative.

7 Numerical integration

We can consider different numerical approximations of the definite integral

$$I = \int_a^b dx f(x) \, . \tag{4}$$

By using the same notation used in Fig. 1 (which is realised in Matlab by the command `x=linspace(a,b,N)`), if $x_i = a + (i-1)\delta x$, with $i = 1, 2, \ldots, N$ and $a + (N-1)\delta x = b$ (i.e., $\delta x = (b-a)/(N-1)$), then one can use the following approximated values to the integral (4) (see Fig. 5 for the graphical representation of each of these integrals):

$$I_l = \sum_{i=1}^{N-1} f(x_i)\delta x \tag{5}$$

$$I_r = \sum_{i=2}^{N} f(x_i)\delta x \tag{6}$$

$$I_c = \sum_{i=1}^{N-1} f\left(\frac{x_i + x_{i+1}}{2}\right)\delta x \tag{7}$$

$$I_{tr} = \sum_{i=1}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2}\delta x \tag{8}$$

7.0.14 Exercise:

Evaluate the numerical integral of $f(x) = e^x$ between $[a, b] = [0, 1]$, for $N = 100$, using all four definitions given above. Compare the four results with the exact one (N.B. use `format long` is order to see the difference).
Questions:

1. what is the order of the error for I_l, I_r, I_c, and I_{tr}? $O(\delta x)$? $O(\delta x^2)$?
2. Which result is the closest to the exact one?
3. Why?
4. What happens if you choose N larger or smaller?

Hints to solve Exercise 7.0.14

- Define two vector indices `i1=(1:1:N-1)` and `i2=(2:1:N)`;
- Use `sum` to sum the elements of the vectors $f(x_i)\delta x$, ...

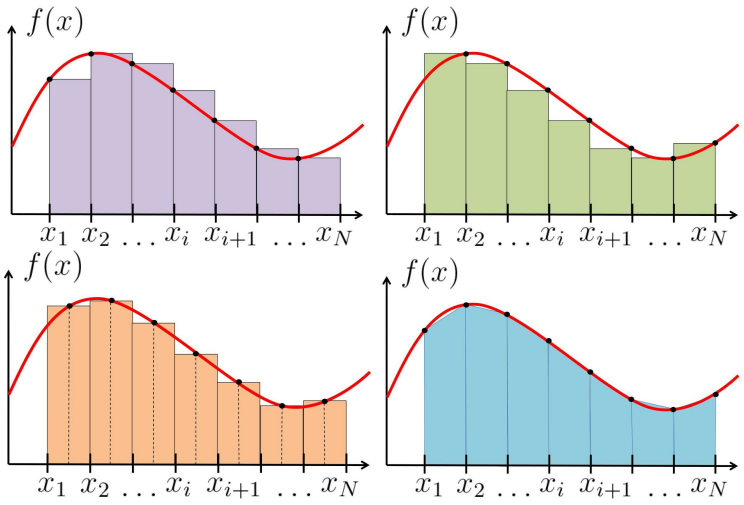

Figure 5: Graphical representations of the various approximations I_l, I_r, I_c, and I_{tr} of the definite integral $\int_{x_1}^{x_N} dx f(x)$.

Matlab has a built-in function for numerical integration `trapz(x,y)`. Repeat the previous exercise using this function and comment about which of the four numerical integration methods I_l, I_r, I_c, or I_{tr} this is equivalent to.

A useful Matlab built-in function for evaluating **indefinite** integrals is `cumsum(x)`. Given a vector `x` with `N= length(x)` components, `y = cumsum(x)` generates a new vector `y` with the same length `N`, whose elements are given by:

$$y(i) = \sum_{k=1}^{i} x(k) \quad \text{where } i = 1, 2, \ldots, N \, .$$

This function allows thus to evaluate indefinite integrals (or primitive functions) such as

$$F(x) - F(a) = \int_a^x ds g(s) \, ,$$

where $\dfrac{dF(x)}{dx} = g(x)$.

7.0.15 Exercise:

Consider the following function

$$f(x) = \int_{\pi/2}^{x} ds \cos(s) \, ,$$

evaluate it numerically, plot it in the interval $x \in [\pi/2, 5\pi]$, together with the analytical expression for $f(x)$.

Hints to solve Exercise 7.0.15

- Define `x` on a grid in the interval $[\pi/2, 5\pi]$;

- note that $f(\pi/2) = 0$;
- choose one formula for integration, for example I_{tr}, and combine it together with the function `cumsum` in order to evaluate $f(x)$.

7.0.16 Exercise:

Consider the following function

$$f(x) = \int_0^x ds e^{-s^2} \cos(3s) \ ,$$

evaluate it and plot it in the interval $x \in [0, 2\pi]$.

7.0.17 Exercise:

Consider the following function

$$f(x) = \int_0^x ds \frac{\sin(s)}{s} \ ,$$

evaluate it and plot it in the following interval $x \in [0, 6\pi]$. By choosing larger and larger intervals, can you guess the value of the integral $\int_0^\infty ds \frac{\sin(s)}{s} (= \frac{\pi}{2})$?

8 Applications: position, velocity and acceleration

8.0.18 Exercise

Let's suppose that of the motion of a particle in a one-dimensional line, we know its velocity as a function of time

$$v(t) = v_0 + V_a \cos(\omega t) e^{-t\gamma} \ , \tag{9}$$

where the parameters are: $v_0 = 2.3$ m/s, $V_a = 10$ m/s, $\omega = 2$ s^{-1}, and $\gamma = 0.2$ s^{-1} — careful about the units.

1. Plot the particle velocity $v(t)$ in the interval $t \in [t_{min}, t_{max}]$, where $t_{min} = 0$ s and $t_{max} = 10$ s N.B. label the plot axis with the correct units.
2. Numerically evaluate the particle acceleration $a(t)$ for the same interval of time and plot it (axis labels and units); compare the numerical result you get with the analytical one.
3. Numerically evaluate the particle position $x(t)$ in the same interval of time, by integrating (9) and knowing that $x(t_{min}) = x0 = 1$ m. Plot the position $x(t)$ for $t \in [t_{min}, t_{max}]$.

Hints to solve Exercise 8.0.18

- Position $x(t)$, velocity $v(t)$ and acceleration $a(t)$ of a particle that moves on a line (one dimension) are related by

$$v(t) = \frac{dx(t)}{dt} \qquad a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2} \; . \tag{10}$$

- Thus, if the velocity is known in a certain interval of time and one wants to evaluate the position at a given time in that interval, one has to evaluate the following integral:

$$x(t) = x(t_0) + \int_{t_0}^{t} ds v(s) \; , \tag{11}$$

where $x(t_0)$ is the position of the particle at the time $t = t_0$.

8.0.19 Exercise

The equation of motion for the one-dimensional harmonic oscillator can be derived from the Newton's law $F = ma = m\frac{d^2x}{dt^2}$ and the Hooke's law, $F = -kx$, describing the motion of an object of mass m which, displaced from its equilibrium position by a distance x, experiences a restoring force proportional to the displacement:

$$m\frac{d^2x(t)}{dt^2} + kx(t) = 0 \; . \tag{12}$$

This differential equation, with initial conditions $x(t = 0) = x_0$ and $v(t = 0) = v_0$, where $v(t) = \frac{dx(t)}{dt}$ is the velocity, admits the exact solution

$$x(t) = \sqrt{x_0^2 + \frac{v_0^2}{\omega_0^2}} \cos(\omega_0 t + \phi) \; , \tag{13}$$

where $\omega_0 = \sqrt{\frac{k}{m}}$ and $\phi = \arccos \frac{x_0}{\sqrt{x_0^2 + v_0^2/\omega_0^2}}$.

1. Plot the position $x(t)$ in the interval of time $t \in [0, 4\pi/\omega_0]$ (label the plot axis with the correct units) with the initial conditions $x_0 = 3.2$ m and $v_0 = -2$ m/s, and for the following values of the system parameters: mass $m = 2$ kg and spring constant $k = 4$ kg/s^2;
2. evaluate numerically the velocity $v(t)$ and plot the numerical result together with the analytical one;
3. evaluate numerically the acceleration $a(t)$ and plot the numerical result together with the analytical one.