Course Notes — part I

September 28, 2017

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`

I remind you here the timing and day of classes, as well as the location:

09:30 – 12:30	Friday 29 Sept	Ciencias - CIE7 (01.17.LD.101) - modulo 17, 101
09:30 – 12:30	Friday 6 Oct	
09:30 – 12:30	Friday 20 Oct	
09:30 – 12:30	Friday 3 Nov	
09:30 – 12:30	Friday 17 Nov	
09:30 – 12:30	Friday 24 Nov	
09:30 – 12:30	Friday 1 Dec	
09:30 – 12:30	Friday 15 Dec	
09:30 – 12:30	Friday 19 Jan	EXAM

At the end of each class I will give a set of problems to carry on at home and to submit before the following class (use file name `your_name_problem-set-number.m`) at the following e-mail address `francesca.marchetti@uam.es` — the solutions to these problems (as well as the interaction in class) will allow me to evaluate you at the end of the course.

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

$$\texttt{a}= [1.5\ 6.2\ 4.4] \qquad\qquad \texttt{b}= [2.1\ 9.4\ 7.1]$$

- `*` 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:

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

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

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

  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.

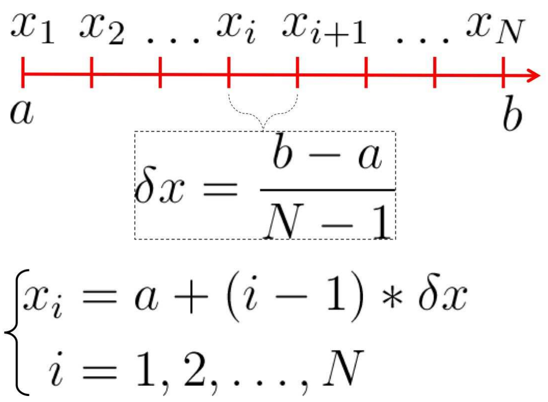
In general, if in an interval $[a, b]$, you want to generate a grid of equally spaced N numbers x_i, as in Fig. 1, you can use equivalently the following equivalent definitions:

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

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 $i = 1, 2, \dots, N$, 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.



x_1 x_2 ... x_i x_{i+1} ... x_N
a
b
$\delta x = \frac{b-a}{N-1}$
$\begin{cases} x_i = a + (i-1) * \delta x \\ i = 1, 2, \dots, N \end{cases}$

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:

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)`;

3.1.2 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)`.

3.1.3 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.

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 $(x1, y1), \dots (xN, yN)$ and joints them with lines — see the examples

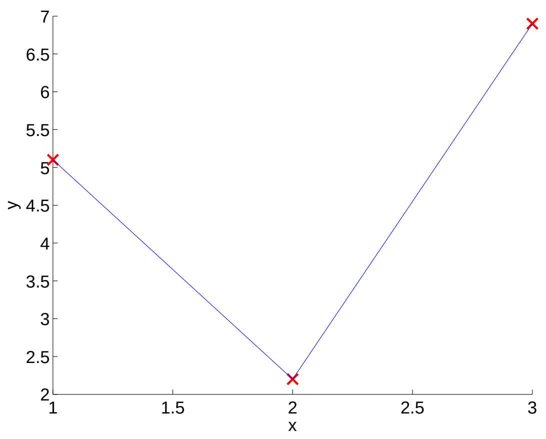


Figure 2: Plot of the two vectors `x=[1 2 3]` and `y=[5.1 2.2 6.9]`, obtained with the command `plot(x,y)`.

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)`.

4.0.4 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).

4.0.5 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.

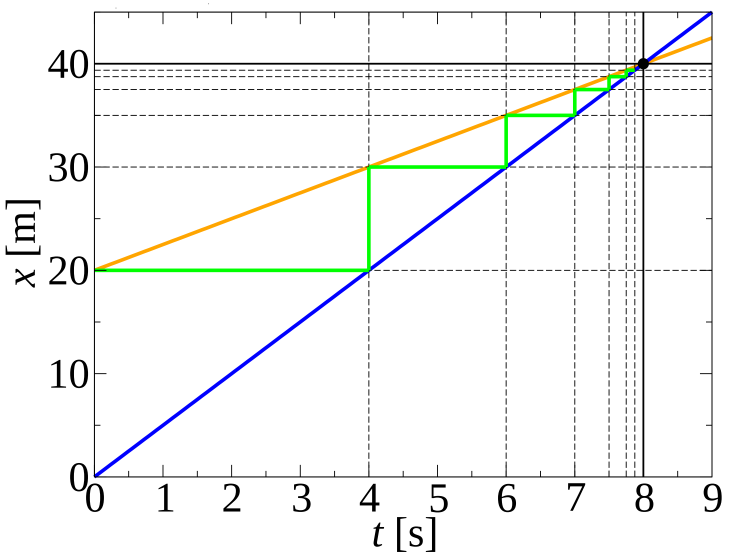


Figure 3: Graphical representation of Zeno's paradox of "Achilles and the Tortoise". Achilles trajectory is plotted in blue and the tortoise one is plotted in orange.

4.0.6 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.

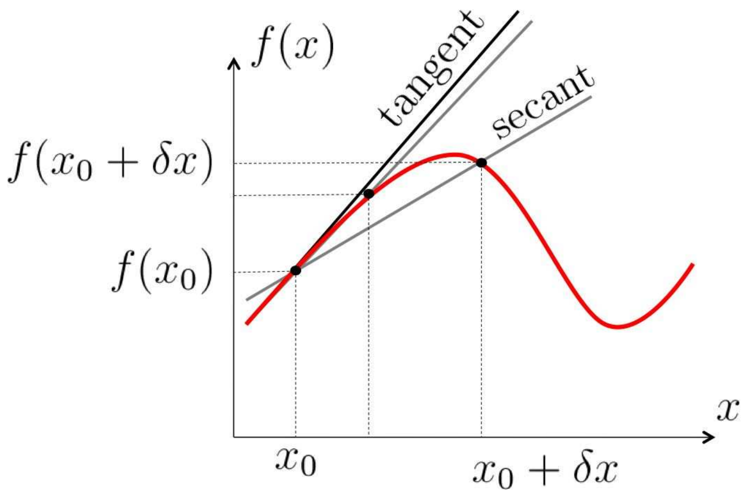


Figure 4: Schematic figure showing the meaning of first derivative of a function $f(x)$ at a given point x_0 and its first approximated value.

4.0.7 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
```

5.0.8 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')
```

5.0.9 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

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

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
```

6.0.10 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.

6.0.11 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.

6.0.12 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 6.0.12

- 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 Application: Taylor expansion

7.0.13 Exercise:

Taylor expand analytically the function $f(x) = e^x$ around $x_0 = 0$ up to the first, second, and third order; plot the original function and the various Taylor polynomial you have obtained, $P_n(x)$, in the interval $x \in [-2, 2]$ and discuss the result you get. Repeat the exercise by expanding around the point $x_0 = 1$.
Do the same exercise for the following functions and expansion points (and plot the results in the interval indicated):

$$\begin{array}{lll} f(x) = \sin(x) & x_0 = 0 & x \in [0, 2\pi] \\ f(x) = \sin(x) & x_0 = \pi/2 & x \in [0, 2\pi] \\ f(x) = \dfrac{1}{1-x} & x_0 = 0 & x \in [-1, 1) \\ f(x) = \log(1+x) & x_0 = 0 & x \in (-1, 1] \end{array}$$

N.B. you can get acquainted with the build-in function `taylor()`, but, for your own benefit, first always do the expansions analytically.

7.0.14 Exercise:

A one-dimensional energy potential landscape is given by $U(x) = x^4 - 2x^2 - 1$. Find the two minima of the potential and do a Taylor expansion around one of them to second order. Show the original function and the approximated one.

7.0.15 Exercise: Lennard-Jones potential

The Lennard-Jones potential

$$V(r) = \frac{a}{r^{12}} - \frac{b}{r^6} , \tag{4}$$

is a model potential describing the interaction between a pair of neutral atoms or molecules. Assuming that the typical bond energy $V_0 \sim 150\ k_B T \simeq 6 \times 10^{-19}$ J corresponds to the depth of the potential and that the typical bond length $r_0 \sim 0.15$ nm corresponds to its equilibrium position:

1. Derive the expressions and values of the parameters a and b.
2. For these values of a and b, plot the dimensionless expression of the potential
$$\frac{r_0^{12}}{a} V(\tilde{r} r_0) ,$$
in the interval $\tilde{r} \in [0.8, 2]$.
3. Apply a Taylor expansion around the equilibrium position and determine the effective spring constant $k = \left.\frac{d^2 V(r)}{dr^2}\right|_{r=r_0}$.

8 Numerical integration

We can consider different numerical approximations of the definite integral

$$I = \int_a^b dx f(x) . \tag{5}$$

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 $i = 1, 2, \dots, N$ 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 (5) (see Fig. 5 for the graphical representation of each of these integrals):

$$I_l = \sum_{i=1}^{N-1} f(x_i) \delta x \tag{6}$$

$$I_r = \sum_{i=2}^{N} f(x_i) \delta x \tag{7}$$

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

$$I_{tr} = \sum_{i=1}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2} \delta x \tag{9}$$

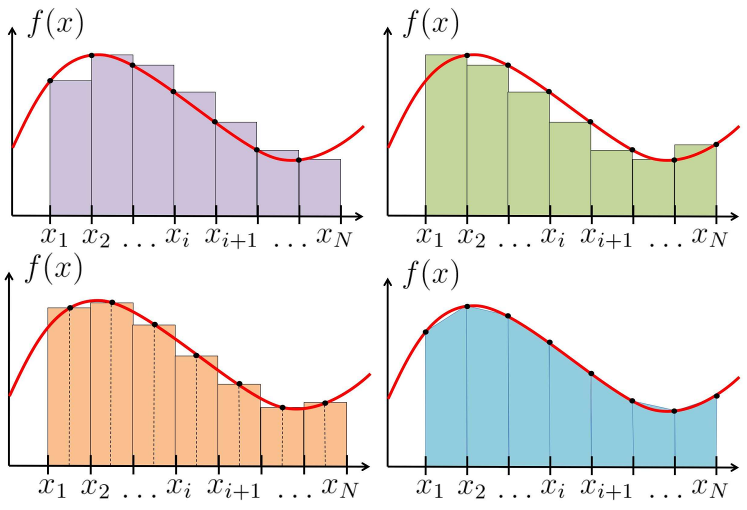


Figure 5: Graphical representations of the various approximations I_l, I_r, I_c, and I_{tr} of the definite integral $\int_{x_1}^{x_N} dx f(x)$.

8.0.16 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 8.0.16

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

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:

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

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

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

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

8.0.17 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 8.0.17

- 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 $f(x)$.

8.0.18 Exercise:

Consider the following function

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

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

8.0.19 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 $\int_0^\infty ds \frac{\sin(s)}{s} (= \frac{\pi}{2})$?

9 Application: position, velocity and acceleration

9.0.20 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

$$v(t) = v_0 + V_a \cos(\omega t) e^{-t\gamma} \;, \tag{10}$$

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 (10) 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 9.0.20

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

$$v(t) = \frac{dx(t)}{dt} \qquad a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2} \;. \tag{11}$$

- 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:

$$x(t) = x(t_0) + \int_{t_0}^{t} ds v(s) \;, \tag{12}$$

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

9.0.21 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:

$$m\frac{d^2x(t)}{dt^2} + kx(t) = 0 \; . \tag{13}$$

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

$$x(t) = \sqrt{x_0^2 + \frac{v_0^2}{\omega_0^2}} \cos(\omega_0 t + \phi) \; , \tag{14}$$

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.

10 External function routines

We have already seen that Matlab has built-in functions, such as

- `sin()`
- `asin()`
- `sinh()`
- `acos()`
- `cos()`
- `cosh()`
- `tan()`
- `atan()`
- `tanh()`

- `exp()`
- `log()`
- `log10()`
- `sqrt()`
- `abs()`
- `sign()`
- `factorial()`
- ...

which are ready to use. As explained in the following, there are also other ways to create your-own function in Matlab.

11 Anonymous functions

An anonymous function can be defined at any point in a script and it is a way of defining a mathematical function rather than a block of operations. For example

```
f=@(x)(x.^3-exp(-0.5*x).*x)
x=linspace(-2,2,100);
plot(x, f(x))
```

Note that you can have functions with more than one argument, such in the following example

```
g=@(x,a)(x.^3-exp(-0.5*x).*x+a*x.^3)
x=linspace(-3,3,100);
plot(x, g(x,0.1))
```

Do not get confused between **indices** and **arguments of a function**, i.e., put great attention about what you put in parenthesis after a vector or a function. If we define

```
x=linspace(0,1,10);
f=2*x-x.^2
```

then f is a vector whose values `f(1)`, `f(2)`, ... can be accessed only for **integer values of the indices**, i.e. $f(i)$, where the index `i` can only be a natural number `i=1, 2, ..., 10`. Clearly, if you type `f(0.5)`, it will give you an error message:

```
??? Attempted to access f(0.5); index must be a positive
integer or logical
```

In this case, if we want to plot the function `f`, we use the command `plot(x,f)`. If we instead define an anonymous function via the command `f=@(x)(...)`, i.e.,

```
x=linspace(0,1,10);
f=@(x)(2*x-x.^2)
```

then `f(x)` is an anonymous function that can be used to evaluate `f` at any given **real** value of `x`, exactly as you would do with Matlab built-in functions. Here, if we want to plot the function, we will use the command `plot(x,f(x))`.

11.0.22 Exercise:

Write a script which plots $x(t) = vt + A\cos(\omega t)$ and $y(t) = A\sin(\omega t)$ as a function of $t \in [0, 10\pi]$ by making use of anonymous functions. Generate 4 subplots 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. N.B. Remember you already solved this problem in Unit 1 so already have the script for it, to which you only have to make the necessary (small) changes.

12 Script functions

Script functions require that you write a separate file.m with the name of the file being the same as the one of the function. The important advantage of script functions is that they can contain a block of certain operations that you plan to do repeatedly in another script, or later in time in a different script — i.e., you can create your own script functions for differentiating at a given order, integrate at a given order, or any other operation that we will see in the following units, such as evaluating zeros of functions with a certain precision and with a given method, solving differential equations, and so on.

Let's consider the following example which evaluates the first derivative of a given function. We will see other examples of script functions all along the course.

```
clear all
close all
clc
% script to evaluate the first derivative of a function
% in a given interval
% and compare the numerical result with the exact one
a=0; b=1; N=20;
fun=@(x)(x.^2+5*x.^5);
df_exact=@(x)(2*x+25*x.^4)

% call the external script function diff_first.m
% a, b, N, fun are INPUTs given above
% x, fp, deltax, il, ir are OUTPUTs
[x, fp, deltax, il, ir]=diff_first(a, b, N, fun);

hold on
plot(x,fun(x),'b-','Linewidth', 2)
plot(x(il),fp,'r-s')
plot(x(ir),fp,'g-s')
plot(x(il)+deltax/2,fp,'ks')
plot(x,df_exact(x),'k--')
legend('function', 'approx 1', 'approx 2', 'approx 3',
'exact','Location','NorthWest')
hold off
```

In the same directory where you run this script, you have to create the following file named diff_first.m:

```
%---------------------------------------------%
% diff_first():  evaluates the derivative of f %
%---------------------------------------------%
% INPUTs:  to be provided
% a --- scalar:  lower bound of the interval
% b --- scalar:  upper bound of the interval
% N --- number of point in the grid
% fun --- anonymous function

function [x, fp, deltax, il, ir] = diff_first(a, b, N, fun)
x=linspace(a ,b, N);
f=fun(x);
il=(1:1:N-1); ir=(2:1:N);
deltax=x(ir)-x(il);
deltaf=f(ir)-f(il);
fp=deltaf./deltax;
end
```

Thus, summarising, to create a function,

$$[\mathtt{a},\mathtt{b},\mathtt{c}] = \mathtt{your\_function}(\mathtt{x},\mathtt{y},\mathtt{z},\mathtt{t})$$

you need to write a separate script file and name it your_function.m. In this file, `x, y, z, t` are INPUT variables, while `a, b, c` are OUTPUT variables, i.e., what the function returns. To use this function in a script, you then call the function by `[a, b, c] = your_function(x,y,z,t)`. Note that the names of the variables inside the function do not need to be the same as the names of the variables you pass to the function! Also note that before you can be able to use an external function, you must set the path to tell Matlab where that function can be found (we will see this in class).

13 Some old exercises redone with the use of external function routines

13.0.23 Exercise:

By making use of an external function routine, 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 with first and second order approximation schemes; establish how fast the numerical value converges to the exact one, $f'(2) = 9$ and comment the result you get.

13.0.24 Exercise:

By writing an external function routine for the integral (in different approximations schemes), consider the following function

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

evaluate it and plot it in the interval $x \in [0, 2\pi]$ — remember the use of the command `cumsum()`.

13.0.25 Exercise:

By writing an external function routine for the integral (choose one approximation scheme, for example the trapezoidal one), 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 $\int_0^\infty ds \frac{\sin(s)}{s} (= \frac{\pi}{2})$?

13.0.26 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:

$$m\frac{d^2x(t)}{dt^2} + kx(t) = 0 \; . \tag{15}$$

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

$$x(t) = \sqrt{x_0^2 + \frac{v_0^2}{\omega_0^2}} \cos(\omega_0 t + \phi) \; , \tag{16}$$

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

1. Create an external function `[x] = harm_oscill(k, m, x0, v0, t)` that evaluates the position of the harmonic oscillator on the time interval specified by the input time vector `t`.
2. 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;
3. by writing an external function routine for differentiating, evaluate numerically the velocity $v(t)$ and plot the numerical result together with the analytical one;
4. by using the same external function routine generated previously, evaluate numerically the acceleration $a(t)$ and plot the numerical result together with the analytical one.

14 Loops and conditions: Loop `for`

The loop `for` allows to repeat certain commands a specified number of times and it is useful when one wants to repeat a certain action in a predetermined way. All loop structures are started with a keyword, in this case `for`, and end with the word `end`. After the `for` command a loop vector is given and Matlab will loop through for each value of the vector. For example

```
for j=1:10
    j
end
```

In this simple example the loop goes around 10 times, each time changing the value of the variable `j` and printing it. Compare the example above with the following one and explain the difference:

```
for j=1:10
    x(j)=j
end x
```

Note that in certain simple cases Matlab provides short-cuts to the loop `for`, i.e., Matlab does the loop on its own automatically; for example, imagine to have defined a certain vector `x = linspace(0,10,21)` and you want to store in another vector `f` of the same length the values of $\sqrt{x}$, then you just have to type `f = sqrt(x)`. This means that for this simple case this operation is equivalent to

```
for i=1:21
    f2(i)=sqrt(x(i));
end
```

Try it and compare the two vectors `f` and `f2`. Clearly there are cases where Matlab does not provide this short-cut and you have no other ways than to use the loop `for`.

14.0.27 Exercise:

Define the following vector `y`

```
x=linspace(0,1,5);
for i=1:5
    y(i)=x(i)^i;
end
```

Now define an identical vector `z` without using the loop `for`.

14.0.28 Exercise:

Define the following matrix

$$M = \begin{pmatrix} 1 & 9 & 2 & 23 \\ 2 & 8 & 13 & 2 \\ -2 & 9 & 7 & 1 \\ 45 & 2 & 82 & -11 \end{pmatrix}$$

and, by making use of a loop `for`, redefine a new 4×4 matrix, `M2`, where, starting from the second row its elements are obtained by subtracting the previous row (and the first row is left unchanged), i.e.:

$$M2 = \begin{pmatrix} 1 & 9 & 2 & 23 \\ 1 & -1 & 11 & -21 \\ -4 & 1 & -6 & -1 \\ 47 & -7 & 75 & -12 \end{pmatrix}$$

14.0.29 Exercise:

After having defined the vector `x=linspace(0,2,20)`, use the loop `for` in order to find an equivalent way of defining the vector `f=x.*x`.

14.0.30 Exercise:

Write a script that, by making use of the loop `for`

1. generates five different figures: `sin(1*x)` in `figure(1)`, `sin(2*x)` in `figure(2)`, and so on;
2. repeat the exercise of "Achilles and the Tortoise" of *Unit 1 — part 2* (Ex. 2.0.4); in particular use the loop `for` in order to find an estimate of time and distance at which Achilles finally meets the Tortoise; also use the loop `for` in order to generate some of the various vertical and horizontal segments of Fig.1 of that exercise.

14.0.31 Exercise:

Write a script that evaluates the factorial `n!` of a given natural number `n` and compare the results with the built-in function `factorial(n)` — remember that the factorial is defined as $n! = n(n-1)(n-2)\dots 2*1$; Hint: store the result in a variable `f` that needs to be initialised to `f=1` prior to the loop.

14.0.32 Exercise:

Given two matrices `A` and `B`, we have seen in the previous unit the two possible multiplication operations we can use:

$\mathtt{C} = \mathtt{A} * \mathtt{B}$	mathematical multiplication (row × column)
$\mathtt{D} = \mathtt{A}\,.* \,\mathtt{B}'$	component-wise multiplication

In particular, define the following two matrices

$$A = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix};.$$

1. Use the loop `for` rather than explicitly the `*` operation in order to evaluate the matrix `C = A * B` and compare the two results — remember the formula for the mathematical multiplication of matrices $C_{ij} = \sum_k A_{ik}B_{kj}$;
2. use the loop `for` rather than the `.*` operation in order to evaluate the matrix `D = A .* B'` and compare the two results — remember that now you are multiplying component-wise, thus `A` and `B'` have to have the same dimension, in this case 3×2.

14.0.33 Exercise:

By making use of the loop `for N=10:100`, evaluate the numerical integral of $f(x) = e^x$ between $[a, b] = [0, 1]$ for different values of the number of points on the grid N. Observe how the result converges to the exact one by plotting the numerical integral first as a function of N and then as a function of $1/N$.

15 Loops and conditions: Command `if`

When one needs a statement to be executed only in limited circumstances, the command to use is `if`. This command executes a statement if the stated condition is met — as usual, you can familiarise yourself with this command by typing `help if` and `doc if`. For example you can compare two numbers with the following script:

```
x=input('type x');
y=input('type y');

if x>y
    disp('x greater than y')
elseif x<y
    disp('x smaller than y')
else
    disp('x equal to y')
end
```

The conditions are comparisons which include:

- $<$ (less than)
- $>$ (greater than)
- $<=$ (less or equal than)
- $>=$ (greater or equal than)
- $==$ (equal)
- $\sim=$ (not equal).

There are two possible formulations of the `if` command:

```
% case 1:  if-then-formulation
if condition
    command
end
```

```
% case 2:  if-then-else-formulation
if condition
    command
else
    some other command
end
```

15.0.34 Exercise: the half-wave rectifier

Consider the function $f(x) = \sin(x)$ in the interval $x \in [-2\pi, 2\pi]$ and redefine a new function $g(x)$ which is zero in the interval where $f(x)$ is negative. Plot both functions.

16 Loops and conditions: Loop `while`

Another command that executes loops is the `while ...end` construct. This command will repeat the command which is contained in between `while ...end` an indefinite number of times until some logical condition is satisfied. You can find more information about it by typing either `help while` or `doc while`. For example we can define a vector n with ten components from 1 to 10 by using the command `while`:

```
i=1;
while i<=10,
      n(i)=i;
      i=i+1;
end
n
```

In many cases using the loop `while` can make for a more efficient algorithm than using the loop `for`, even though the particular example above is not the case. We will see many examples later on, when numerically solving differential equations.

As for the case of the condition `if`, also for the loop `while` you can use the following logical conditions:

- $<$ (less than)
- $>$ (greater than)
- $<=$ (less or equal than)
- $>=$ (greater or equal than)
- $==$ (equal)
- $\sim=$ (not equal).

Let's consider the following simple example

```
f=64;
n=0;
while f>1
      f=f/2
      n=n+1;
end
n
```

This is a way of determining how often a number can be divided by 2, and you can check that `2^n` gives you the initial number `64` — of course there is a better and faster way to do the same, which is evaluating `log2(64)`!

16.0.35 Exercise:

Use the loop `while` to generate a routine which is able to establish whether a natural number n is a prime number (i.e., a natural number which can only be divided by 1 and itself) or not. Rewrite the same routine with the loop `for`.

Hints to solve Exercise 16.0.35

- You can use the command `rem(n,div)` (or equivalently the command `mod(n,div)`) to evaluate the rest of a division; if an integer number `div` exists such that `mod(n,div)=0`, then clearly it means that `n` cannot be a prime number;
- remember the use of `if` (see previous section);
- you can terminate a loop like `while` and `for` with the command `break` (e.g., once the conditions you are looking for are met, without the need to leave the computer run till the natural end the loop; in this way you shorten the evaluation time);
- once you have established the conditions you are looking for, you can print "n is a prime number" (or "n is not a prime number") with the command `disp('n is a prime number')`.

17 Application: Vector rotations

Let us consider a two-dimensional vector $\mathbf{r}_1 = (x1, y1)$, whose components in polar coordinates can be written as $x_1 = r\cos\phi_1$ and $y_1 = r\sin\phi_1$. And now let us suppose we want to rotate $\mathbf{r}_1$ anti-clockwise by an angle θ (see Fig. 6) so that to get a new vector $\mathbf{r}_2 = (x2, y2) = (r\cos\phi_2, r\sin\phi_2)$, where $\phi_2 = \phi_1 + \theta$. Thus the new component of the rotated vector can be written as

$$\begin{aligned} x_2 &= r\cos(\phi_1+\theta) = r(\cos\phi_1\cos\theta - \sin\phi_1\sin\theta) = x_1\cos\theta - y_1\sin\theta \\ y_2 &= r\sin(\phi_1+\theta) = r(\sin\phi_1\cos\theta + \cos\phi_1\sin\theta) = y_1\cos\theta + x_1\sin\theta \ . \end{aligned}$$

Thus, the rotation can be written as the following matrix operation acting on the vector $\mathbf{r}_1$:

$$\mathbf{r}_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \mathbf{r}_1 \ . \tag{17}$$

You can easily show that both vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ have the same norm, i.e., length: $|\mathbf{r}_1| = \mathbf{r}_2 = r$.

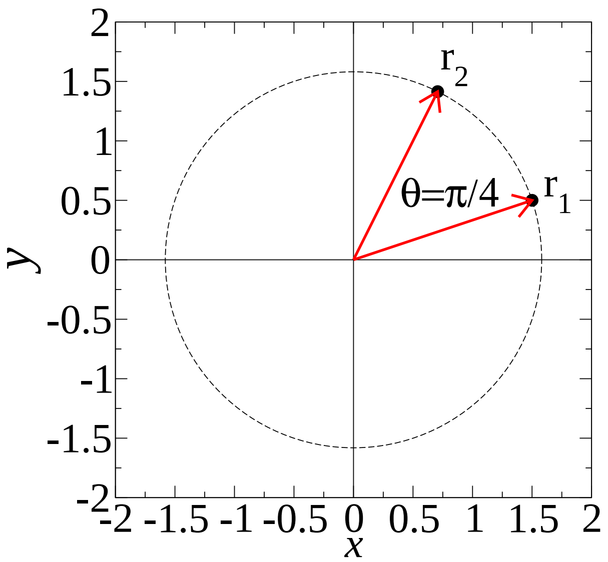


Figure 6: Rotating anti-clockwise the vector $\mathbf{r}_1 = (x_1, y_1) = (1.5, 0.5)$ by an angle $\theta = \pi/4$.

17.0.36 Exercise: Rotation

Consider a vector in two-dimensions $\mathbf{r}_1 = (x_1, y_1) = (1.5, 0.5)$ as in Fig. 6.

1. Rotate $\mathbf{r}_1$ anti-clockwise around the z-axis by an angle $\theta = \pi/4$, and find the coordinates of the rotated vector $\mathbf{r}_2 = (x_2, y_2) = A\mathbf{r}_1$:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = A \begin{pmatrix} x_1 \\ y_2 \end{pmatrix} \qquad A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} .$$

2. What do you have to change if you would like to rotate $\mathbf{r}_1$ clockwise?

3. Which matrix is the inverse of the rotation matrix A, i.e., A^{-1}? (the command in Matlab is `inv(A)`).

4. Which operation are you doing by considering either `A * inv(A)` or `inv(A) * A` and what do you get?

17.0.37 Exercise: Rotation and shifting of parametric curves

Let's consider the following ellipse

$$x = a\sin(\theta) \qquad\qquad y = b\cos(\theta)\ ;$$

where $\theta \in [0, 2\pi]$ and $a = 2$ and $b = 1$.

1. Plot the ellipse (use the command `axis('equal')` to have the same axis ranges for both x and y).
2. Plot now the ellipse rotated clockwise by an angle $\theta = \pi/8$ and shifted by $(x_0, y_0) = (2, 3)$, i.e., now centered in $(x_0, y_0) = (2, 3)$.

18 Application: numerical series and asymptotic limits

18.0.38 Exercise:

The number e can be equivalently defined as

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

1. 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.
2. Consider the finite sum

$$e(N) = \sum_{n=0}^{N}\frac{1}{n!}\ ,$$

and plot it in two subplots both as a function of N and N^{-1}, showing that it asymptotically converges to e.

18.0.39 Exercise:

Let us consider the geometric series

$$a_{exact} = \sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \qquad \text{for } |x| < 1 \,. \tag{18}$$

1. Choose a numerical value for x (say $x = 1/2$) and plot the sum of the first N terms of the geometric series, i.e., it's approximated numerical value
$$a(N) = \sum_{n=0}^{N} x^n \,, \tag{19}$$
as a function of N and check that it converges to $1/(1-x)$. Compare the values you get for a fixed N with the exact analytical expression $a(N) = \frac{1-x^{N+1}}{1-x}$.
2. Plot now $a(N)$ as a function of N^{-1}.

18.1 Application: Binding polynomials and polymerization

(From K. Dill and S. Bromberg, *Molecular Driving Forces*, Garland Science, 2011.)
Many cellular processes involve polymerization, where a bunch of monomers bind together to form a polymer. Here we consider a simple model of polymerization where each monomer is added to the growing chain with the same equilibrium binding constant K. This situation is described by the following set of chemical equations:

$$X_1 + X_{n-1} \overset{K}{\rightleftharpoons} X_n \qquad n = 2, 3, 4, \ldots \,. \tag{20}$$

The symbol X_n denotes a polymer n monomers in size. In equilibrium, there will be polymers of all different sizes. We use this model to compute the average polymer size, and how it depends on the concentration of monomers.

18.1.1 Exercise:

1. By applying recursively the formula $X_n = KX_1X_{n-1} = KxX_{n-1}$, evaluate analytically an expression for the probability that a polymer is n monomers in length

$$P(n) = \frac{X_n}{\sum_{n=1}^{\infty} X_n} , \tag{21}$$

in terms of K and $x = [X_1]$, the concentration of free monomers.

2. Compare the analytical expression $P(n)$ found above with an approximate numerical one $P_{num}(n)$, by plotting both $P(n)$ and $P_{num}(n)$ as a function of n for a fixed value of Kx (figure(1)), as well as by plotting both $P(n)$ and $P_{num}(n)$ as a function of Kx for a fixed value of n (figure(2)). Comment the results you get.

3. Use the analytical result for $P(n)$ to get an analytical expression for the average polymer size $\langle n \rangle = \sum_{n=1}^{\infty} nP(n)$ and compare it with an approximated numerical one $\langle n \rangle_{num} = \sum_{n=1}^{N} nP(n)$ by plotting both $\langle n \rangle$ and $\langle n \rangle_{num}$ as a function of Kx.

4. Show that $\langle n \rangle$ diverges as Kx approaches 1 from below — Note that, as $Kx \to 1$ the average polymer size goes to infinity. In fact, $Kx = X_n/X_{n-1}$, and thus it if it equals 1, the equilibrium concentration of longer polymers is not smaller than that of shorter ones, which would imply infinite polymers.

19 Some additional exercise

19.0.2 Exercise

Find the crossing points between a circle centered in $(x_0, y_0) = (2, -3)$ and radius $R = 5$ and the exponential function $f(x) = e^{-x/2} - 4$ (answer: $x_1 \simeq -2.42$ and $x_2 \simeq 6.91$). Plot both functions and mark the crossing points you have found numerically.

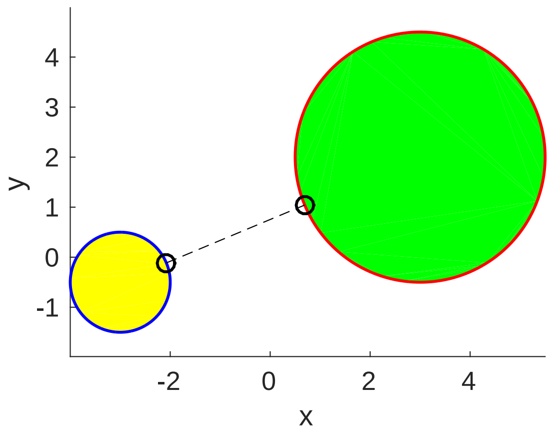


Figure 7: Minimal distance between two circles.

19.0.3 Exercise: Maxwell-Boltzmann distribution

The Maxwell-Boltzmann distribution describes the velocity (v) distribution of a classical gas of atoms with mass m at thermal equilibrium at a given temperature T:

$$f(v) = 4\pi \left(\frac{m}{2\pi k_B T}\right)^{3/2} v^2 e^{-\frac{mv^2}{2k_BT}} \ . \tag{22}$$

In SI units the Boltzmann constant is $k_B = 1.38 \times 10^{-23}$ J/K and let's consider the specific case of Argon atoms, whose mass is $m = 67 \times 10^{-27}$ kg.

1. Numerically evaluate the maximum of this distribution `fmax` for a temperature $T = 300$ K and its corresponding velocity `vmax`; use the loop `for` and compare your result with the built-in operation `max`, taking into account the syntax `[fmax,imax]=max(f)`, where `imax` is the index of the maximum value of the vector `f`;

2. Assuming the system is kept at a pressure low enough to always remain in its gaseous phase in the temperature range $T \in [1, 1000]$ K, plot in two separate subplots `fmax` and `vmax` as a function of temperature.

19.0.4 Exercise: Minimal distance between two circles

Consider the following two circles,

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \ , \tag{23}$$

one with center $(x_{01}, y_{01}) = (-3, -0.5)$ and radius $r_1 = 1$ and the other with $(x_{02}, y_{02}) = (3, 2)$ and $r_2 = 2.5$. Numerically evaluate the minimal distance between the two circles, compare the result you get with the exact one, $\sqrt{(x_{02} - x_{01})^2 + (y_{02} - y_{01})^2} - (r_1 + r_2)$, and represent graphically your result by plotting the two circles and the segment representing the minimal distance (see Fig. 7).

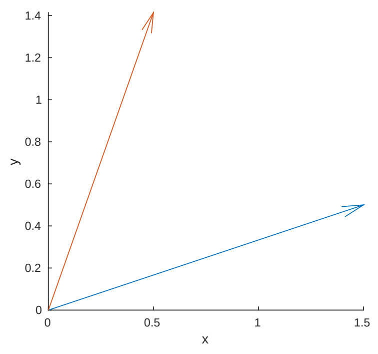


Figure 8: Rotating anti-clockwise the vector $\mathbf{r}_1 = (x_1, y_1) = (1.5, 0.5)$ by an angle $\theta = \pi/4$: representation with the command `quiver(x,y,vx,vy)`.

20 Advanced plotting: scalar fields

In order to plot functions of two variables, such as $f(x,y) = x^2 + y^2$, one needs to generate a two-dimensional grid of points with the commands

```
xmesh=linspace[a,b,N];
ymesh=linspace[c,d,M];
[x,y]=meshgrid(xmesh,ymesh);
f=x.^2+y.^2;
```

One can then choose one of the following options for the plotting of the surface: `contour(x,y,f)`, `mesh(x,y,f)`, `surf(x,y,f)`, `surfc(x,y,f)`, `surfl(x,y,f)`. By using the `help` and by trying the various options, find out what these commands corresponds to.

20.0.5 Exercise:

Write a scripts which plots the following function of two variables $f(x,y) = (x^2 + y^2) - (x^2 + y^2)^2/2$ in the interval $x, y \in [-1, 1]$ using the commands `meshgrid` and `mesh(x,y,f)`. Evaluate the maximum value of the function $f(x,y)$, by making use of a loop `for` and the condition `if`. Compare the result you find this way with the one found with the command `max`, and check they match with the exact answer — N.B. For a matrix A, `max(A)` returns a row vector containing the maximum element from each column. When you have doubts about a command, type `help command`, e.g. in this case `help max`.

21 Advanced plotting: vector fields

Matlab can plot vectors with the command `quiver(x,y,vx,vy)`, which plots velocity vectors as arrows with components `(vx,vy)` at the points `(x,y)`. Here, `x`, `y`, `vx`, and `vy` must be matrices all of the same size — `x` and `y` can also be vectors to specify a uniform grid. N.B. Quiver automatically scales the arrows to fit within the grid. Use the version `quiver(x,y,vx,vy,S)` with `S=0` to plot the arrows without the automatic scaling. Consider the following two examples.

In the first one we reproduce the exercise of Fig. 6; the result is plotted in Fig. 8.

```
r1=[1.5; 0.5];
theta=pi/4;
A=[cos(theta) -sin(theta); sin(theta) cos(theta)];
r2=A*r1;

S=0;
hold on
quiver(0,0,r1(1),r1(2),S)
quiver(0,0,r1(2),r2(2),S)
xlabel('x')
ylabel('y')
axis image
hold off
```

In the second example we plot a two-dimensional random vector field on a grid:

```
N=10; range=2;

xmin=-range; xmax=range;
ymin=-range; ymax=range;

xmesh=linspace(xmin,xmax,N);
ymesh=linspace(ymin,ymax,N);
[x,y]=meshgrid(xmesh,ymesh);

vx=rand(N,N)-0.5;
vy=rand(N,N)-0.5;

quiver(x,y,vx,vy)
axis image
```

21.0.6 Exercise:

Consider the energy potential $U(x, y) = -(x^2 + y^2) + (x^2 + y^2)^2/2$, plot the potential using the commands `meshgrid` and `mesh(x,y,f)`, evaluate the force field $(F_x, F_y) = -(\partial_x U, \partial_y U)$ and plot it with the command `quiver`. Describe the motion of a particle under such a vector force field. Use first your routine to evaluate the partial derivatives $\partial_x U$ and $\partial_y U$, and then use the build-in Matlab routine `[Fx, Fy] = gradient(U)` to calculate the gradient of a given scalar field `U`.

22 Application: Charge distribution

The electric (Coulomb) potential of a point charge Q at a distance $|\mathbf{r} - \mathbf{r}_0|$ from the charge position $\mathbf{r}_0$ is given, in SI units, by

$$U(r) = \frac{Q}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}_0|} \ , \tag{24}$$

where ϵ_0 is the vacuum permittivity — in the CGS units, one can fix $1/(4\pi\epsilon_0) = 1$.

22.0.7 Exercise:

Plot the electric potential for an electric charge $Q = -1$ positioned at $\mathbf{r}_0 = (1, 1.5)$ in a two-dimensional (2D) plane $\mathbf{r} = (x, y)$ — generate a 2D mesh covering an interval around $\mathbf{r}_0$ and use the command `meshgrid()`; plot the potential using `contour()`, `mesh()`, `surf()`, either in multiple figures (using `figure()`) or in one figure (using `subplot()`).

22.0.8 Exercise:

Plot the potential for a system of two point charges in a 2D plane, one with charge $Q = -1$ positioned at $\mathbf{r}_0 = (1, 1.5)$ and another of charge $Q = +1$ positioned at $\mathbf{r}_0 = (-1, -1.5)$ — the electric potential for a system of point charges is the sum of the individual potentials.

The electric field of a single point charge located at the origin is given by

$$\mathbf{E} = \frac{Q}{4\pi\epsilon_0} \frac{\hat{\mathbf{r}}}{r^2} \ . \tag{25}$$

22.0.9 Exercise:

Imposing $1/(4\pi\epsilon_0) = 1$, write a script which plots the electric field $\mathbf{E}$ of a point charge $Q = 1$ located at $(x_0, y_0) = (1, 0)$ and plot $\mathbf{E}$ in the interval $x \in [0, 2]$ and $y \in [-1, 1]$ on a grid of 10 points for each axis.

The electric field due to a collection of charges Q_1, Q_2, ..., is obtained using the principle of superposition, $\mathbf{E} = \mathbf{E}_1 + \mathbf{E}_2 + \dots$, thus at a given position $\mathbf{r}$,

$$\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \sum_i Q_i \frac{\mathbf{r} - \mathbf{r}_{i0}}{|\mathbf{r} - \mathbf{r}_{i0}|^3} \ . \tag{26}$$

22.0.10 Exercise:

Write a script which plots the electric field $\mathbf{E}$ of two point charges $Q_1 = 1$ located at $\mathbf{r}_{10} = (1, 0)$ and $Q_2 = -1$ located at $\mathbf{r}_{20} = (-1, 0)$ (dipole).