# Course Notes — part II

September 28, 2017

## 1 System of linear equations

**Summary of definitions** A system of linear equations is a set of $M$ equations involving $N$ variables, $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ (unknowns). In general a system of linear equations can be written as a matrix equation of the form:

$$A\mathbf{x} = \mathbf{b} \,, \tag{1}$$

where $A_{11}, A_{12}, \ldots, A_{MN}$ are the coefficients of the system ($A$ is a $M \times N$ matrix) and $\mathbf{b} = (b_1, b_2, \ldots, b_M)^T$ are the constant terms:

$$\begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ A_{M1} & A_{M2} & \ldots & A_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_M \end{pmatrix} \,. \tag{2}$$

Clearly, Eqs. (1) and (2) can be equivalently written in terms in its components:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1N}x_N &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2N}x_N &= b_2 \\ &\ldots \\ A_{M1}x_1 + A_{M2}x_2 + \cdots + A_{MN}x_N &= b_M \,. \end{aligned}$$

A system can either have infinitely many solutions, or one unique solution or else no solution. If the solution to the system exists, then it is given by

$$\mathbf{x} = A^{-1}\mathbf{b} \,, \tag{3}$$

where $A^{-1}$ is the inverse matrix of $A$ (note that if $A$ is a $M \times N$ matrix, then $A^{-1}$ is a $N \times M$ matrix), i.e., is such that $A^{-1}A = \mathbb{I}$ (where $\mathbb{I}$ is here the $N \times N$ identity matrix) and $AA^{-1} = \mathbb{I}$ (while $\mathbb{I}$ is here the $M \times M$ identity matrix). Note also that the elements of $A^{-1}$, $(A^{-1})_{ij}$, are **not** the inverse of each element of $A$, $A^{-1}_{ij}$. Show that explicitly!

For example let's consider the simplest case of $M = N = 2$, a system of 2 equations with two unknowns:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 &= b_1 \\ A_{21}x_1 + A_{22}x_2 &= b_2 \,. \end{aligned}$$

Each equation is a line in the plane $(x_1, x_2)$, therefore is clear that the solution will be the intersection of these two lines: this can either be a line (infinitely

many solutions), a point (one unique solution) or else the two lines never meet (no solution).

Matlab has built-in commands to solve systems of linear equation and calculate inverse matrices

```
x=mldivide(A,b)
x=A\b
x=inv(A)*b
```

### 1.0.1 Exercise:

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} ,$$

find the solution using the command `mldivide(A,b)` (or equivalently `A\b`) and compare it with the exact solution you find analytically as well as with `inv(A)*b`. Plot the two lines $A_{11}x_1 + A_{12}x_2 = b_1$ and $A_{21}x_1 + A_{22}x_2 = b_2$ in the plane $(x_1, x_2)$ and check they intersect at the point previously found.

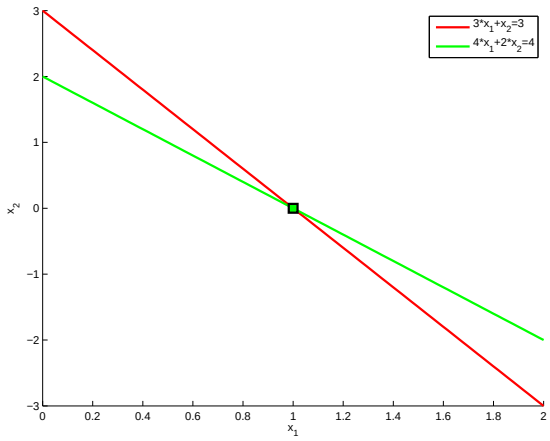



Figure 1: Plotting the two lines $A_{11}x_1 + A_{12}x_2 = b_1$ and $A_{21}x_1 + A_{22}x_2 = b_2$ in the plane $(x_1, x_2)$ of the Ex. 1.0.1.

Similarly in the case of $N$ variables, each equation of the system describes a hyper-plane in the $N$-dimensional space $(x_1, \ldots, x_N)$ and the solution to the system is the intersection of these hyper-planes.

**1.0.2 Exercise:**

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & 5 & 4 \\ 7 & 9 & 3 \\ 4 & 5 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \\ 9 \end{pmatrix} ,$$

find the solution using the command `mldivide(A,b)` and compare it with `inv(A)*b` — setting `format long`.

**1.0.3 Exercise:**

Find the solution of the system of linear equations $\mathbf{x}^T A = \mathbf{b}^T$, where $A$ and $\mathbf{b}$ are given in the Ex. 1.0.1. Compare the solution with the one obtained by plotting the lines defined by each one of the two equations of the system. Check that `(b/A)'=A'\b'`, i.e. $(\mathbf{b}^T A^{-1})^T = (A^{-1})^T \mathbf{b}$.

In general, for $M$ **linearly independent** equations[1] (i.e. none of the equations can be derived from the others) we can have the following situations

1. **Undetermined system:** If $M < N$ there are less equations than unknowns and the system has infinitely many solutions. For example when $N = 2$ and $M = 1$, the solutions lie on the line defined by the single equation in two variables. In this case the dimension of the solution is equal to $2 - 1 = 1$ — and in general by $N - M$;

2. There is a **unique solution** when $N = M$ and when the square matrix $A$ has an inverse $A^{-1}$; the solution is given by $\mathbf{x} = A^{-1}\mathbf{b}$;

3. **Overdetermined system:** There are **no solutions** when instead $M > N$, i.e. there are more equations than unknowns. For example, if $N = 2$ and $M = 3$, then each equation of the system describes different lines and they can meet in three different points.

**1.0.4 Exercise:**

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & -2 \\ 2 & -4 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 5 \\ 6 \end{pmatrix} .$$

Discuss why there are no solutions to this system of linear equations. What does it mean that $\det(A) = 0$?

[1] One can make sure this is the case by checking that $\det(A)$ is finite. For a singular matrix $A$, i.e. $\det(A) = 0$, at least two equations of (1) are linearly dependent.

**1.0.5 Exercise:**

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & 2 \\ 3 & -4 \\ 3 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 5 \\ 6 \\ 8 \end{pmatrix} .$$

Explain why there is no solution to such a system of linear equations by plotting the three lines it describes. What happens if you evaluate `A\b`? What is Matlab finding? What does it minimise? (Hint: type `help mldivide` and `doc mldivide`).

Solving systems of linear equations can also be useful for non-linear functions, as long as these functions have the same structure, as in the following exercise.

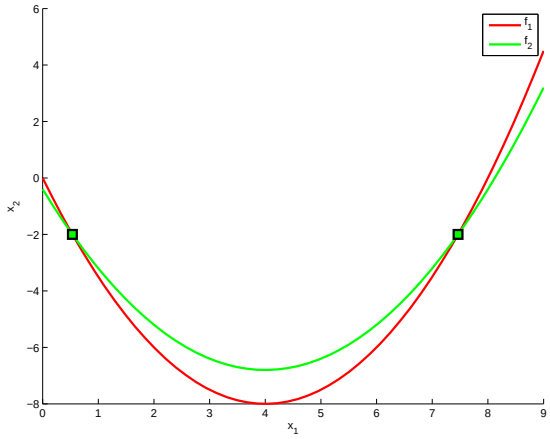



Figure 2: Plotting the two functions $f_1(x) = \left[3 - \left(3 + 8x - x^2\right)\right]/2$ and $f_2(x) = \left[4 - 2\left(3 + 8x - x^2\right)\right]/5$ of the Ex. 1.0.6.

**1.0.6 Exercise:**

Consider the following non-linear system of two equations in two unknowns:

$$2y + \left(3 + 8x - x^2\right) = 3$$
$$5y + 2\left(3 + 8x - x^2\right) = 4 \, .$$

1. Plot the two functions $y = f_1(x) = \left[3 - \left(3 + 8x - x^2\right)\right]/2$ and $y = f_2(x) = \left[4 - 2\left(3 + 8x - x^2\right)\right]/5$ and establish graphically the values of the two intersection points $(x_1, y_0)$ and $(x_2, y_0)$;
2. introduce a new variable, $z = 3 + 8x - x^2$ and solve the system of two linear equations for $y_0$ and $z_0$:

$$2y + z = 3$$
$$5y + 2z = 4 \, .$$

3. Now solve $z_0 = 3 + 8x_{1,2} - x_{1,2}^2$ and verify that the solutions you found above, $(x_1, y_0)$ and $(x_2, y_0)$ coincide with the intersection points found in the first point 1. Plot the two points on the graph as in Fig 2.

# 2 Application: Electrical circuits

Currents $I_i$ and potential differences $V_j$ in electrical circuits are ruled by the Kirchhoff's circuit laws. The first (current) law states that, at any node in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing away from that node:

$$\sum_{i \in \text{into the node}} I_i = \sum_{k \in \text{out from the node}} I_k \, .$$

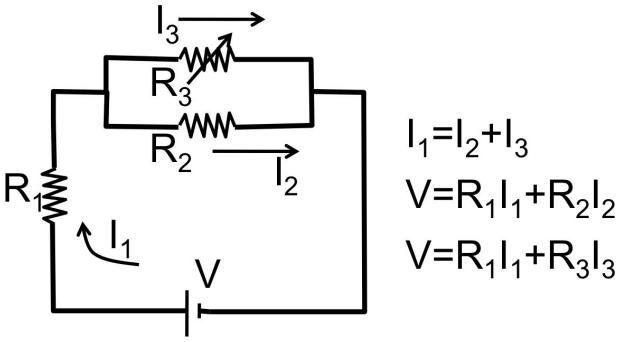
This law reflects the principle of conservation of electric charge. On the contrary, Kirchhoff's second law applies to closed loops in an electrical circuit and states that the sum of the electrical potential differences (i.e., the voltages) around any closed network is zero,

$$\sum_{j \in \text{network}} V_j = 0 \, .$$

This law follows from the principle of conservation of energy.

**2.0.7 Exercise:**

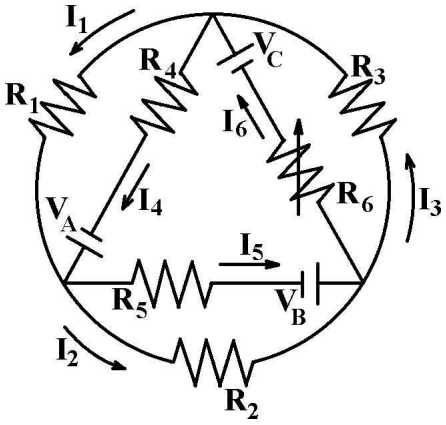
Applying the two Kirchhoff's circuit laws, find the three currents, $I_1$, $I_2$, and $I_3$, of the following electric circuit:




by knowing the values of the voltage $V = 10$ V, and the resistances $R_1 = 2\ \Omega$, $R_2 = 4\ \Omega$, and $R_3 = 3\ \Omega$. By assuming that $R_3$ is a variable electric resistance, plot the three currents as function of $R_3 \in [0, 100]\ \Omega$.

**2.0.8 Exercise:**

Applying the two Kirchhoff's circuit laws, find the six currents, $I_1$, ..., $I_6$ of the following electric circuit:




Plot all currents knowing the values of the voltages $V_A = 3$ V, $V_B = 2$ V, and $V_C = 3$ V, and the resistances $R_1 = 2\ \Omega$, $R_2 = 1\ \Omega$, $R_3 = 2\ \Omega$, $R_4 = 2\ \Omega$, $R_3 = 1\ \Omega$, as a function of the variable resistance $R_6 \in [0, 3]\ \Omega$.

# 3 Zeros of a function: bisection method

**Summary of definitions** The bisection method is the easiest algorithm one can think of that finds the root of a function, $f(x) = 0$, in a given interval $[a, b]$. The interval has to be chosen so that one knows for sure in advance that one of the zeros lies inside it. The algorithm repeatedly divides the interval in half and selects the subinterval in which the root must lie. Despite its simplicity this method is quite slow.

The structure of the algorithm (see Fig. 3) is the following one:

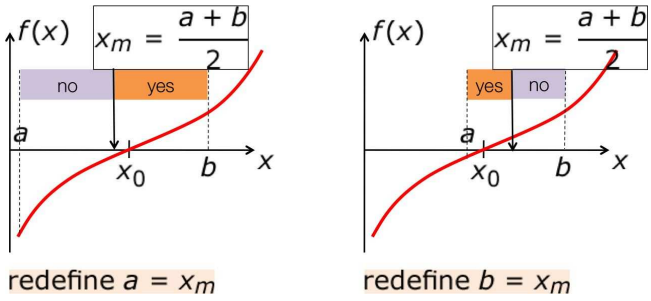



Figure 3: Schematic representation of the way the bisection method works.

```
a=...; b=...; fa=...; fb=...;
while b-a>small number
            xm=(a+b)/2; fm=...;
            if (fa*fm)>0
               a=xm
            else
               b=xm
            end
end
```

For the *small number* you can choose whatever number you want the accuracy of your answer to be in the . Type `help eps` and `doc eps`.

**3.0.9 Exercise:**

Find the two zeros of the function $U(x) = -3x^2 + x^4 + x^5/10$ in the two intervals $[1, 2]$ and $[-3, -1]$ by writing a bisection algorithm. Compare the results you find in this way with the ones you obtain graphically by plotting the function (see Fig. 2). Finally find the zeros of $U(x)$ by using the built-in Matlab routine `fzero(f,[1,2])` — N.B. you need to first write an anonymous function, i.e. `f = @(x)-3*x^2+x^4+x^5/10`.

**3.0.10 Exercise:**

Use the bisection method in order to find the zero of the function $f(x) = x^3 - 7x^2 + 14x - 6$ in within the interval $[0, 1]$, with an accuracy of $10^{-2}$.

# 4 Roots of a function: Newton-Raphson method

A better (i.e., converging faster) algorithm to find the zeros (or roots) of a function than the bisection method is the Newton-Raphson method. In the root-finding process, this method uses not only the actual values of the function but also its first derivatives. It is based on the fact that, if we Taylor expand the function $f(x)$ up to the first order term at the point $x_0$, which is close to

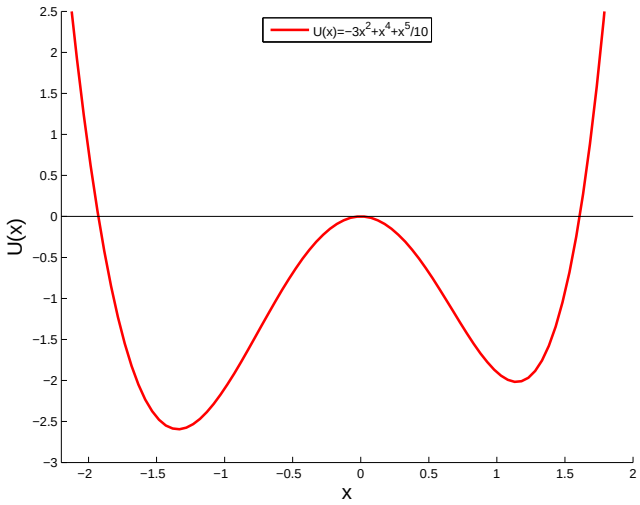



Figure 4: Plot of the function $U(x) = -3x^2 + x^4 + x^5/10$.

the root of $f(x)$,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + O((x - x_0)^2) \,, \tag{4}$$

as we are looking for the solution to $f(x_1) = 0$, then, ignoring higher order terms, we get that

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \,. \tag{5}$$

The point $x_1$ found as described above is not the real root of $f(x)$ because we have truncated the Taylor expansion, but if we apply iteratively this formula, expanding now around $x_1$ and so on and so fort, we can converge quickly to the real zero of the function. This method can converge very quickly, and faster than the bi-section method, if the initial guess of the root is quite close to the one we are looking for. In addition, the derivative of $f(x)$ should neither be zero nor infinite in the region of interest.

The implementation of the method is quite easy. Let's suppose we want to find numerically the root of the function $f(x) = \exp(x) - x - 4$. By plotting $f(x)$ (see Fig. 3), we can see the root is around $x \simeq 1.7$. We start giving a first guess of the root, say $x_0 = 2.5$; evaluating the tangent line to $f(x)$ at $x_0 = 2.5$,

$$f(x) = (e^{2.5} - 1)(x - 2.5) + e^{2.5} - 2.5 - 4 + O((x - 2.5)^2) \,, \tag{6}$$

we can find an approximation of the root of $f(x)$ better than $x_0 = 2.5$ by finding the root $x_1$ of $f(x)$ approximated as in the above expression around $x_0 = 2.5$, which is $x_1 \simeq 1.99$ (see Fig. 3). If we now implement to the next step the above procedure, by defining as new initial guess $x_0 = x_1 \simeq 1.99$, then the new approximated value of the zero will give $x_1 \simeq 1.78$, which is already very close (after two steps only!) to the real zero of the function $f(x) = \exp(x) - x - 4$, which is $x \simeq 1.75$ (check this yourself).

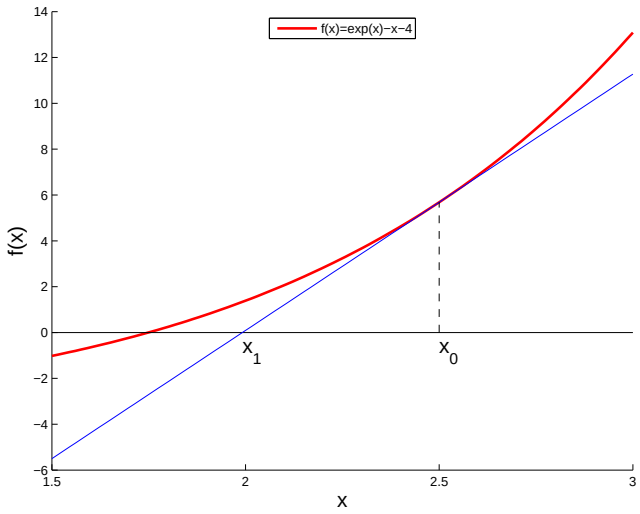



Figure 5: Function $f(x) = \exp(x) - x - 4$ and first step of the Newton-Raphson method if we use as initial guess $x_0 = 2.5$.

**4.0.11 Exercise**

Write a routine which reproduces Fig. 3 for $f(x) = \exp(x) - x - 4$, plot $f_1(x)$ for $x_0 = 2.5$ and find $x_1$. (Optional: Export the figure into an .eps file).

**4.0.12 Exercise**

Find the zero of $f(x) = \exp(x) - x - 4$ by using the built-in Matlab routine `fzero(f,2)` — N.B. you need to first write an anonymous function, i.e. `f = @(x)exp(x)-x-4`.

**4.0.13 Exercise**

Find the zero of the function $f(x) = \exp(x) - x - 4$ by writing a Newton-Raphson algorithm. Use as initial guess $x_0 = 2.5$. Set `format long` and compare the result obtained in this exercise with the one obtained in the previous exercise.

**Hints to solve Exercise 4.0.13**

- Given the initial guess $x_0$, evaluate the derivative of $f(x)$ in $x_0$, $f'(x_0) \simeq [f(x_0) - f(x_0 - \delta x)]/\delta x$, where $\delta x$ is a small number;
- once you know the derivative, use Eq. (5) to find the next guess $x_1$;
- build a `while` loop which runs until the condition $f(x_1) >$ eps is satisfied.

#### 4.0.14 Exercise

Solve the previous exercise by writing a bisection algorithm. Set `format long` and compare the result obtained in this exercise with the ones obtained in the previous two exercises.

#### 4.0.15 Exercise

Count the number of times the loop `while` is called in the Newton-Raphson algorithm you developed in Ex. 4.0.13 and how many times instead is called in the bisection algorithm you developed in Ex. 4.0.14. For the particular case of the function $f(x) = \exp(x) - x - 4$, which algorithm is more efficient and why?

#### 4.0.16 Exercise

Write a routine using the Newton-Raphson algorithm which evaluates the minimum of the function $f(x) = -3x^2 + x^4 + x^5/10$ (shown, together with its derivative in Fig. 4) close to $x_0 = 1$. Compare the result you get this way with the one you obtain by writing a bisection method. Indicate which method is more efficient and why.

**Hints to solve Exercise 4.0.16**

- Remember that at the maximum or at a minimum of a function, $f'(x_0) = 0$ — and in this specific case, $f'(x) = -6x + 4x^3 + x^4/2$;
- apply the Newton-Raphson algorithm to find the root of $f'(x)$ close to $x_0 = 1$ — i.e., Eq. (5) now reads $x_1 = x_0 - f'(x_0)/f''(x_0)$, where $f''(x) = -6 + 12x^2 + 2x^3$.

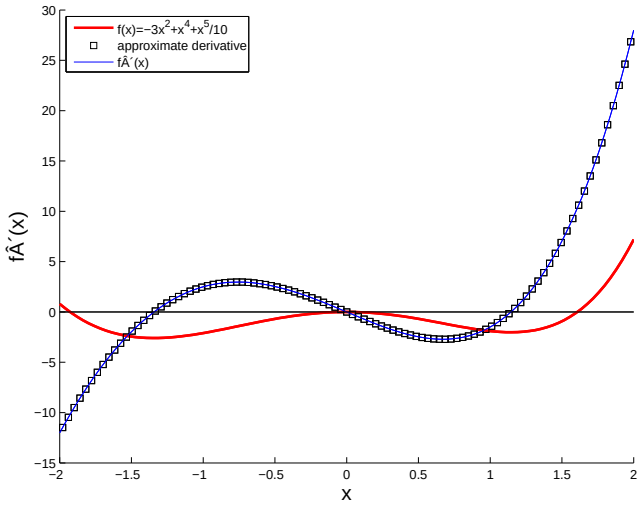
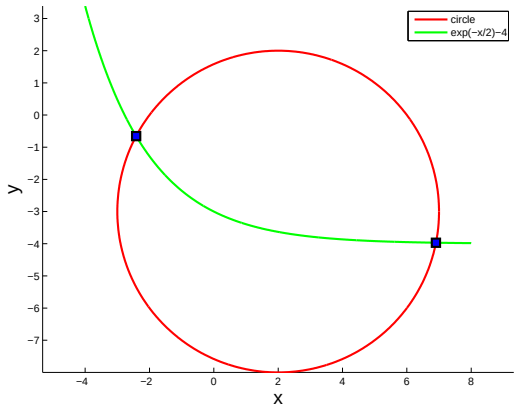



Figure 6:

**4.0.17 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 $x_2 \simeq 6.91$).




# 5 Application: energy conservation

In mechanics, the conservation of energy means that the sum of the kinetic energy $T$ and the potential energy $U$ is a constant, $E$. In case of a particle moving in one dimension, the conservation of energy reads:

$$E = T + U(x) = \frac{1}{2} m \left(\frac{dx}{dt}\right)^2 + U(x) \,, \tag{7}$$

where $v = dx/dt$ is the velocity of the particle and $m$ its mass. As a consequence, the velocity of the particle can be evaluated as a function of the position only,

$$\frac{1}{2} m \left(\frac{dx}{dt}\right)^2 = E - U(x) \,,$$

which means that the motion is restricted only to the region where $U(x) \leq E$. At the points $x_i$ for which $E = U(x_i)$, the particle has zero velocity — points of motion inversion. Finally, the force in terms of the potential energy reads:

$$F(x) = -\frac{dU(x)}{dx} \,. \tag{8}$$

**5.0.18 Exercise**

Consider the case of the energy potential $U(x)/E_c = -3x^2 + x^4 + x^5/10$ plotted in Fig. 5. Evaluate the points of motion inversion $x_1$ and $x_2$ for the total energy $E/E_c = -1$.

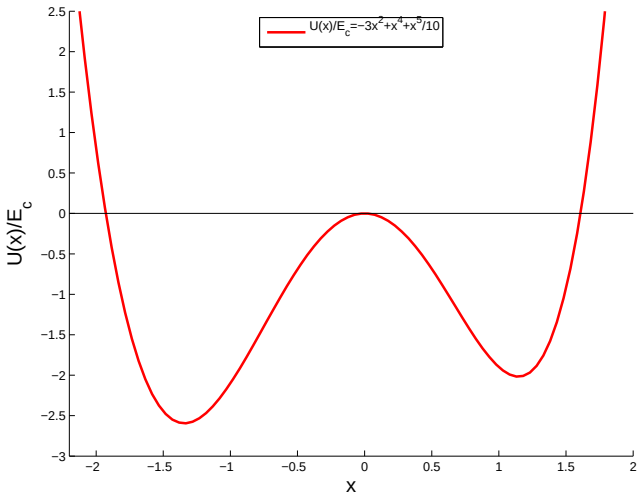



Figure 7:

# 6 Example of external function routine for the bisection method

We remember that in order to create an external function script we have to use the following syntax,

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

and that 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!

The following example (see Fig. 8) creates a function script zero_bisection.m that evaluates the zero of a function in a given interval `(a, b)` and with a given accuracy `err`. Thus `a, b, err` are input values, while `x_0, fx_0, iter` are output values: `x_0` is the approximated value of the zero, `fx_0` the value of the function at this point, `iter` the number of iterations the programme needed to find the zero with the accuracy `err` initially required. This external function can be used in the following script

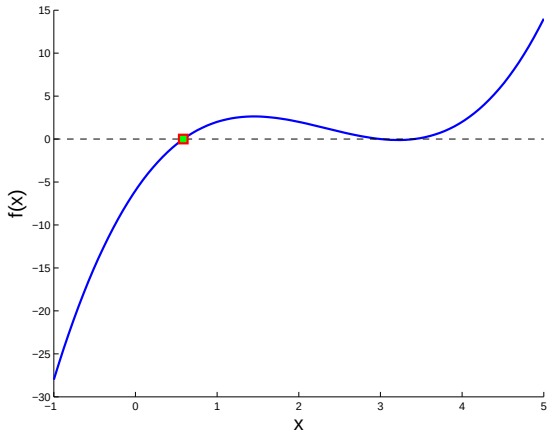



Figure 8: Result of the script that uses the function script zero_bisection.m, finding the zero of the function $f(x) = x^3 - 7x^2 + 14x - 6$.

```
clear all
close all
clc
% example on how to use the external function zero_bisection.m
fun=@(x)(x.^3-7*x.^2+14*x-6);

a=-1; b=1.5; err=2*eps;
[ x_0, fx_0, iter ] = zero_bisection( fun, a, b, err );
x_0
fx_0
iter

a=-1; b=5; N=100;
x=linspace(a, b, N);
hold on
plot(x,fun(x),'Linewidth', 2)
plot(x,zeros(length(x),1),'k--')
plot(x_0,fx_0,'sr')
hold off
```

```
function [ x_0, fx_0, iter ] = zero_bisection( fun, a, b, err )
% bisection method in order to find the root of the function f
% INPUT
% fun function
% a
% b interval [a, b]
% OUTPUT
% x_0 root
% fx_0 value of the function at x_0
% iter number of iterations used

fa=fun(a);
fb=fun(b);
i=0
while b-a>2*err
      %define the middle point
      x=(a+b)/2;
      fx=fun(x);
      if (fa*fx)>0
           %eliminate left half interval
           a=x;
      else
           %eliminate right half interval
           b=x;
      end
      i=i+1;
end
x_0=x;
fx_0=fun(x_0);
iter=i;
end
```

#### 6.0.19 Exercise

By writing external function routines for both the bisection method as well as the Newton-Raphson method, find the zero of the function $f(x) = \exp(x) - x - 4$ with both methods.

#### 6.0.20 Exercise

By writing an external function routine, 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 $x_2 \simeq 6.91$).

# 7 Ordinary differential equations

An ordinary differential equation of order $n$ is an equation involving a single independent variable $x$, a function $f(x)$ and the derivative of such a function up to the $n^{\text{th}}$–derivative,

$$g\left(\frac{d^n f(x)}{dx^n}, \frac{d^{n-1} f(x)}{dx^{n-1}}, \dots \frac{df(x)}{dx}, f(x), x\right) = 0 \,, \tag{9}$$

where $g$ is some arbitrary function. The order of the equation is set by the highest derivative, $\frac{d^n f(x)}{dx^n}$. In addition, the differential equation (9) is called ordinary because there is only one independent variable, $x$. The form of (9) is implicit, but in some cases it can be rewritten in an explicit form:

$$\frac{d^n f(x)}{dx^n} = h\left(\frac{d^{n-1} f(x)}{dx^{n-1}}, \dots \frac{df(x)}{dx}, f(x), x\right) \,. \tag{10}$$

The example we are going to use the most through this unit is the one of the Newton's second law of motion for a particle of mass $m$ moving under the influence of a force $F$. The force in general can be a function of the position $x(t)$ of the particle, but it can also be a function of it's velocity $v(t) = dx(t)/dt$ (like in the case when a particles suffers friction) and time $t$ (like for other non-conservative forces, such as time–dependent driving forces):

$$m\frac{d^2 x(t)}{dt^2} = F\left(x(t), \frac{dx(t)}{dt}, t\right) \,. \tag{11}$$

This is a second order explicit ordinary differential equation. Most of equations in physics are differential equations telling us how certain quantities change as functions of others; in the case of the Newton equation (11), it tells us how the position $x(t)$ of a particle of mass $m$ changes with time $t$.

## 7.1 Reduction of order

Any ordinary differential equation of order $n$ can be rewritten as a system of $n$ ordinary differential equations of order 1. For example, in the case of Newton's second law of motion (11), we can introduce the velocity field $v(t)$,

$$\begin{cases} v(t) = \dfrac{dx(t)}{dt} \\ m\dfrac{dv(t)}{dt} = F(x(t), v(t), t) \,, \end{cases} \tag{12}$$

and thus, rather than a second order explicit ordinary differential equation for the position $x(t)$ (11), we have now two explicit first order differential equations for both the position $x(t)$ and the velocity $v(t)$. Therefore we will first introduce numerical techniques for solving $1^{\text{st}}$–order ordinary differential equations and later on we will see how to apply such techniques in order to solve the system (12).

# 8 First order ordinary differential equations: Euler method

The simplest method for numerical integration of a first order ordinary differential equation is given by the Euler method. As we will see later, as well as very simple, this method is also not very 'accurate' and we will consider better methods later on. Let us suppose we want to solve the following $1^{\text{st}}$–order ordinary differential equation, with a given initial condition for $t = t_1$:

$$\frac{dx(t)}{dt} = f\left(x(t), t\right) \qquad\qquad x(t_1) = x_1 \ . \tag{13}$$

Basically we know the value of the function at a given point (i.e., $x_1 = x(t_1)$) and the slope of the tangent to $x(t)$ at the same point (i.e., $\left.\frac{dx(t)}{dt}\right|_{t=t_1} = f(x_1, t_1)$) and from this information we want to reconstruct the entire solution $x(t)$. Starting from the initial condition, $x(t_1) = x_1$, we can evaluate the solution at a point $t_2$ close to $t_1$ by approximating the derivative to the first order,

$$\frac{x_2 - x_1}{t_2 - t_1} \simeq f(x_1, t_1) \qquad\Rightarrow\qquad x_2 \simeq x_1 + \delta t f(x_1, t_1) \ ,$$

where $x_2 = x(t_2)$ and $\delta t = t_2 - t_1$. The smaller $\delta t$, the more accurate the approximation we are doing is. How to find $x_2$ is also schematically plotted in Fig. 1.

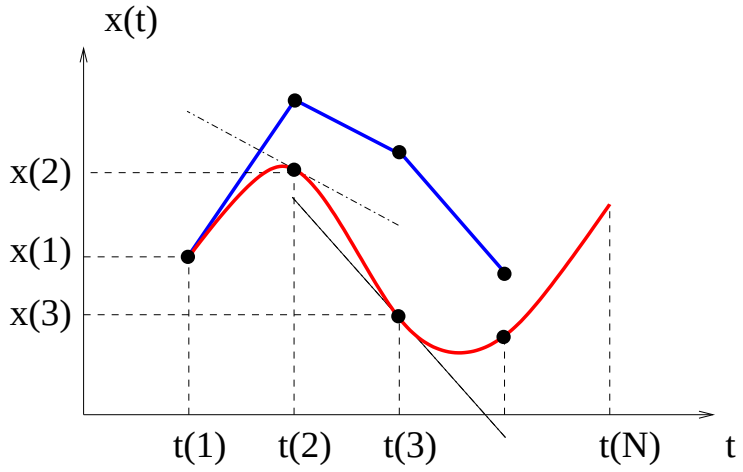



Figure 9: Schematic representation of the Euler method. The exact solution to a given differential equation satisfying the initial condition $x(t_1) = x_1$ is plotted in red and in blue is instead plotted the approximated solution obtained by using the Euler method.

Generalising this expression to the case in which we consider a grid of $N$ points in the interval $[t_1, t_N]$ in which we want to evaluate the solution to the first order differential equation (13), we can write that (see Fig. 1):

$$\frac{x_{i+1} - x_i}{t_{i+1} - t_i} \simeq f(x_i, t_i) \qquad\Rightarrow\qquad x_{i+1} \simeq x_i + \delta t f(x_i, t_i) \ , \tag{14}$$

where $x_i = x(t_i)$, $x_{i+1} = x(t_{i+1})$, and $\delta t = t_{i+1} - t_i$. A schematic representation of the Euler method is shown in Fig. 1. The structure of the Euler algorithm can be written as follows

```
t1= ...; tN= ...; N=...;
t=linspace(t1, tN, N);
dt=t(2)-t(1);
% initial condition:
x_E(1)=...
% anonymous function f(x,t):
f=@(x,t)(...);
% loop for the Euler method:
for i=1:N-1
            x_E(i+1)=x_E(i) + dt*f(x_E(i), t(i));
end
```

Alternatively you can evaluate the values of the function `f(i)=...` inside the loop. Once you have become familiar with the method, you are invited to write an external script function routine for the Euler method, as in the following example:

```
%-----------------------------------------------%
% Euler_method                                  %
%-----------------------------------------------%
% INPUTs (to be provided):
% t1 = starting point
% x1 = initial condition
% tN = end point
% N = number of steps
% xp = anonymous function containing the first derivative
% OUTPUTs:
% xt_E = vector solution x(t)
% t = grid of point t where x(t) has been evaluated

function [xt_E, t] = Euler_method(t1, x1, tN, N, xp)
xt_E(1)=x1;
t=linspace(t1,tN,N);
dt=t(2)-t(1);
for i=1:N-1
    xt_E(i+1)=xt_E(i) + dt*xp(xt_E(i),t(i));
end
end
```

### 8.0.1 Exercise

Consider the following first order differential equation

$$\frac{dx}{dt} = -2\left(t - \frac{5}{2}\right) ,$$

with the initial condition $x(1) = -9/4$. Solve the equation exactly and compare the exact solution with the approximated one obtained with the Euler method. The approximated solution evaluated on a grid of $N = 5$ points within the interval $[1, 4]$ is shown in Fig. 2.

**Questions:**

1. if $x(t)$ is the position of a particle (of mass $m = 1$kg) in meters, $t$ the time in seconds, evaluate the acceleration of the particle and describe its motion — N.B. a freely falling body on the moon has an acceleration of 1.6m/s$^2$.

After you have completed the exercise, solve it again by writing an external function routine for the part containing the numerical integration via the Euler method.

**Hints to solve Exercise 8.0.1**

- In order to find the exact solution you have to evaluate

$$\int_{t_1=1}^{t} dt' \frac{dx(t')}{dt'} = x(t) - x(t_1) = \int_1^t dt' \left[-2\left(t' - \frac{5}{2}\right)\right] = -\left(t - \frac{5}{2}\right)^2 + \left(1 - \frac{5}{2}\right)^2 .$$

  Thus the exact solution is $x(t) = -\left(t - \frac{5}{2}\right)^2$.

### 8.0.2 Exercise

Consider the following first order differential equation

$$\frac{dx}{dt} = -3x , \tag{15}$$

with the initial condition $x(0) = 1$. Solve the equation exactly and compare the exact solution with the approximated one obtained with the Euler method. Describe the motion in time $t$ of a particle with position $x(t)$ — N.B. a force proportional to the velocity of a particle and with opposite direction,

$$F = m\frac{d^2x}{dt^2} = -|\alpha|\frac{dx}{dt} ,$$

describes the friction on the particle. The solution you get should look like the one plotted in Fig. 3.
After you have completed the exercise, solve it again by using the very same external function routine for the Euler method that you have written for the previous exercise 8.0.1.

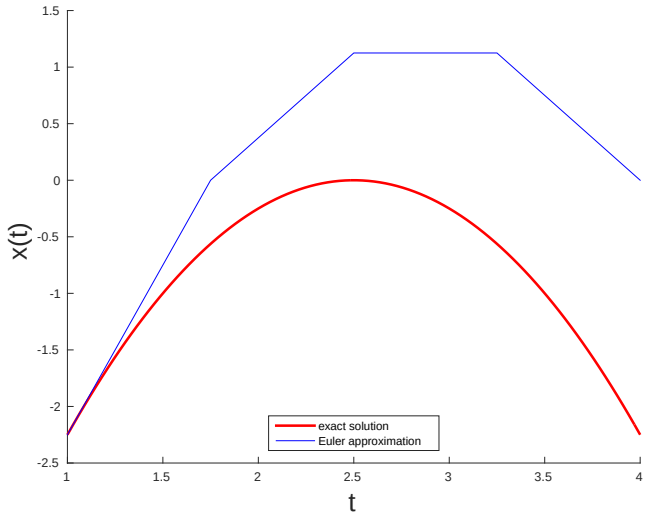



Figure 10: Solutions of the of the differential equation $dx/dt = -2(t-5/2)$ with initial condition $x(1) = -9/4$: exact solution (red) and approximated solution (blue) obtained with the Euler method on a grid of $N = 5$ points in the interval $[1, 4]$.

**Hints to solve Exercise 8.0.2**

- An exact solution of (15) can be found by evaluating

$$\int_{x(0)}^{x} \frac{dx'}{x'} = \ln\left(\frac{x(t)}{x(0)}\right) = -3\int_{0}^{t} dt' = -3t \; .$$

**8.0.3 Exercise**

Solve numerically the following first order differential equation

$$\frac{dx(t)}{dt} = -(3t^2 - 2t + 5)[x(t) - 1] \qquad x(0) = x0 \; ,$$

for three different initial conditions, $x(0) = 0$, $x(0) = 1$, and $x(0) = 2$ and compare the numerical results with the exact solution.

# 9 Second order ordinary differential equations: Euler method

Let us consider the particular case of the differential equation for an harmonic oscillator, like the one describing the motion of a particle of mass $m$, which, displaced from its equilibrium position, experiences a restoring force proportional to the displacement $x$, $F = -\kappa x$ (Hooke's law):

$$m\frac{d^2x}{dt^2} + \kappa x = 0 \; , \tag{16}$$

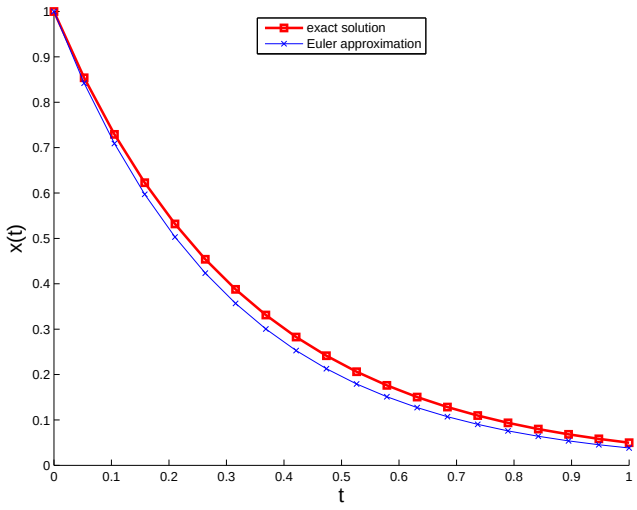



Figure 11: Solutions of the of the differential equation $dx/dt = -3x$ with initial condition $x(0) = 1$: exact solution (red squares) and approximated solution (blue crosses) obtained with the Euler method on a grid of $N = 20$ points in the interval $[0, 1]$.

As already explained previously in Sec. 7.1, we can rewrite this second order ordinary differential equation as a system of two coupled first order differential equations:

$$\begin{cases} \dfrac{dx}{dt} = g_1(x, v, t) = v \\ \dfrac{dv}{dt} = g_2(x, v, t) = -\dfrac{\kappa}{m} x \ . \end{cases} \tag{17}$$

In this way, we can solve the initial problem by making use of the Euler method, as in the following exercise.

### 9.0.4 Exercise: the harmonic oscillator

Consider the second order differential equation describing an harmonic oscillator:

$$m\frac{d^2x}{dt^2} = -\kappa x \ .$$

This equation can be rewritten in terms of the dimensionless time $\tilde{t} = \omega_0 t$, where $\omega_0 = \sqrt{\kappa/m}$, as

$$\frac{d^2x}{d\tilde{t}^2} = -x \ . \tag{18}$$

Solve this equation with the following initial conditions

$$x(0) = 1 \qquad\qquad v(0) = \left.\frac{dx}{d\tilde{t}}\right|_{\tilde{t}=0} = 0 \ ,$$

applying the Euler method.
**Questions:**

1. How the numerical result compares with the exact one in the interval $\tilde{t} \in [0, 4\pi]$?
2. roughly how many points $N$ do you need to consider in the interval $\tilde{t} \in [0, 4\pi]$ so that to have a good numerical approximation to the exact solution?
3. in a separate figure, plot the velocity $v(\tilde{t})$ as a function of the position $x(\tilde{t})$ (phase space). Why the exact solution gives a closed trajectory? Why the numerical solution is not a closed trajectory?

**Hints to solve Exercise 9.0.4**

- The exact solution of the harmonic oscillator in dimensionless units is given by

$$x(t) = \sqrt{x^2(0) + v^2(0)}\sin(\tilde{t} + \delta) \qquad \delta = \arctan\left(\frac{x(0)}{v(0)}\right) \ . \tag{19}$$

  Note that if $v(0) = 0$, then $\delta = \pi/2$;

- the $2^{\text{nd}}$–order ordinary differential equation (18) can be written as a system of two $1^{\text{st}}$–order ordinary differential equations:

$$\begin{cases} \dfrac{dx}{d\tilde{t}} = v \\ \dfrac{dv}{d\tilde{t}} = -x \ . \end{cases}$$

  In this case the Euler method reads

$$\begin{cases} x_{i+1} = x_i + \delta t v_i \\ v_{i+1} = v_i + \delta t(-x_i) \ . \end{cases}$$

## 9.1 Error in the Euler method

At each step in the Euler method (14) we are neglecting terms of the order of $\delta t^2$, therefore we say the 'local' error is of the order $(\delta t)^2$. However, each step is iterated $N-1$ times, where $N$ is the number of points on the grid. Therefore the magnitude of the 'global' Euler error is given by

$$\text{error} \sim (\delta t)^2 \times (N-1) \; . \tag{20}$$

As $N$ is of the order of $1/\delta t$ then the Euler method global error is of the order of $\delta t$. For this reason, the Euler method is a **first order** method. As the following exercise shows, the convergence to the exact solution is quite slow.

### 9.1.1 Exercise about the error in the Euler method

Evaluate the error done with the Euler method in the previous three exercises by using the estimate (20) and compare this error with the one you obtained evaluating the maximum distance of the approximated solution from the exact one. Choosing one of the three previous exercises, evaluate the behaviour of the error (20) with increasing the number of points $N$ on the grid. How fast is the convergence to the exact solution reached with the Euler method?

# 10 First order differential equations: modified Euler method

In the previous classes, we have seen how to solve a $1^{\text{st}}$–order differential equation with an assigned initial condition,

$$\frac{dx(t)}{dt} = f\left(x(t), t\right) \qquad\qquad x(t_1) = x_1 \; . \tag{21}$$

via the Euler method. In particular, if we consider a grid of $N$ points in the interval $[t_1, t_N]$ where we want to evaluate the solution, the solution at the time $t_{i+1}$, $x_{i+1} = x(t_{i+1})$, can be evaluated approximatively (to the first order) starting from the solution at the time $t_i$, $x_i = x(t_i)$ and adding to it the derivative of such a solution in $t_i$, $f(x_i, t_i)$, times the increment in time $\delta t = t_{i+1} - t_i$:

$$x_{i+1} \simeq x_i + \delta t f(x_i, t_i) \; . \tag{22}$$

We have seen that the Euler method is a first order method, which means the error made is of the order of $\delta t$ — note that $\delta t \sim (\delta t)^2 \times (N-1)$, where $N$ is the number of times the method is iterated (the error accumulates at each iteration step). For this reason this method converges slowly, and is, in some cases, unstable.

A better accuracy could be obtained by considering a higher order approximation to the derivative, i.e., instead of using the slope $f(x_i, t_i)$, we use the average slope $[f(x_i, t_i) + f(x_{i+1}, t_{i+1})]/2$:

$$x_{i+1} \simeq x_i + \delta t \frac{f(x_i, t_i) + f(x_{i+1}, t_{i+1})}{2} \; . \tag{23}$$

However, the problem is that on the right-hand side of this formula it appears $x_{i+1}$ that we don't know yet, because at each step we only know $x_i$ and $t_i$. To solve this problem, the modified Euler method first calculates the intermediate value $\tilde{x}_{i+1}$ by means of the Euler method, and then uses this value for the final approximation $x_{i+1}$ at the next integration point, in other words:

$$\tilde{x}_{i+1} = x_i + \delta t f(x_i, t_i) \tag{24}$$

$$x_{i+1} = x_i + \delta t \frac{f(x_i, t_i) + f(\tilde{x}_{i+1}, t_{i+1})}{2} \ . \tag{25}$$

# 11 First order differential equations: Runge-Kutta method

Another second order method is given by the 2$^{\text{nd}}$–order Runge–Kutta method. The idea this method is based on is similar to the one of the modified Euler method explained before. In fact, an equivalent way of using a better approximation than (22) consists in replacing the derivative evaluated at the left–point of the interval $[t_i, t_{i+1}]$, $f(x_i, t_i)$, with the derivative evaluated at the middle–point, $f(x_{i+1/2}, t_{i+1/2})$:

$$x_{i+1} \simeq x_i + \delta t f(x_{i+1/2}, t_{i+1/2}) \ . \tag{26}$$

The notation in this expression means $t_{i+1/2} \equiv t_i + \delta t/2$ and $x_{i+1/2} \equiv x(t_{i+1/2})$. The problem with the expression (26) is that the algorithm cannot be applied in the form (26) since it requires the knowledge of the derivative evaluated at the middle–point, $f(x_{i+1/2}, t_{i+1/2})$, which we don't know because at each step we only know $x_i$ and $t_i$. We can however approximate $x_{i+1/2}$ by again using (22), i.e., the Euler method:

$$x_{i+1/2} \simeq x_i + \frac{\delta t}{2} f(x_i, t_i) \ .$$

Therefore the 2$^{\text{nd}}$–order Runge–Kutta algorithm reads as:

$$x_{i+1} \simeq x_i + \delta t f(x_i + \frac{\delta t}{2} f(x_i, t_i), t_i + \delta t/2) \ . \tag{27}$$

We will see later in Sec. 9.1 why this method is second order rather than first.

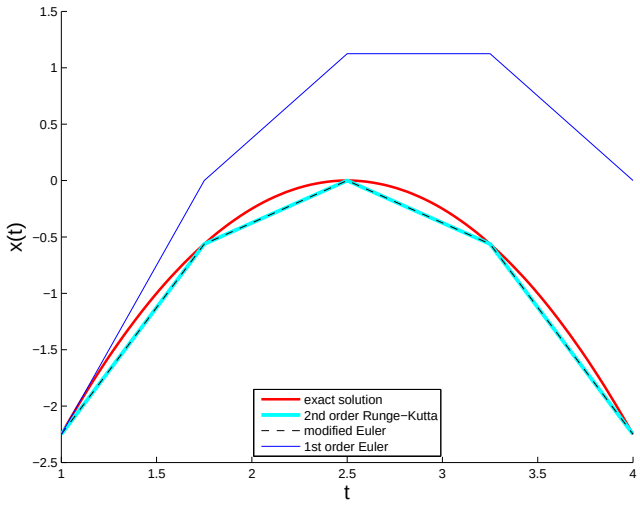



Figure 12: Solutions of the of the differential equation $dx/dt = -2(t - 5/2)$ with initial condition $x(t = 1) = -9/4$. Exact solution (red) and approximated solutions: in blue the one obtained with the $1^{\text{st}}$–order Euler method, in cyan the one obtained with a $2^{\text{nd}}$–order Runge-Kutta method, and in black dashed line the one obtained with the modified Euler method (it gives exactly the same result as the $2^{\text{nd}}$–order Runge-Kutta method) on a grid of $N = 5$ points in the interval $[1, 4]$.

# 12 Exercises

### 12.0.2 Exercise

Consider the following first order differential equation

$$\frac{dx}{dt} = -2\left(t - \frac{5}{2}\right) ,$$

with the initial condition $x(t = 1) = -9/4$. Evaluate the approximated solutions obtained with the Euler method, the modified Euler method and with the $2^{\text{nd}}$–order Runge–Kutta method and compare them with the exact solution $x(t) = -(t - 5/2)^2$. One example is shown in Fig. 1 for a grid of $N = 5$ points within the interval $[1, 4]$.

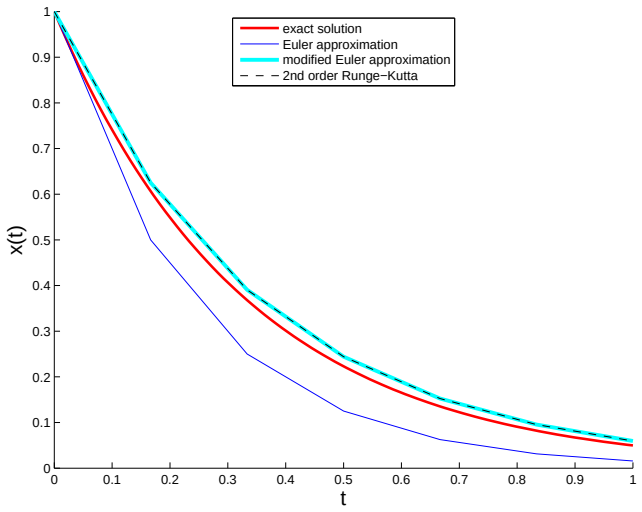



Figure 13: Solutions of the of the differential equation $dx/dt = -3x$ with initial condition $x(t = 0) = 1$. Exact solution (red) and approximated solutions: in blue the one obtained with the $1^{\text{st}}$–order Euler method, in cyan the one obtained with a $2^{\text{nd}}$–order Runge-Kutta method, and in black dashed line the one obtained with the modified Euler method on a grid of $N = 7$ points in the interval $[0, 1]$.

### 12.0.3 Exercise

Consider the following first order differential equation

$$\frac{dx}{dt} = -3x \ , \tag{28}$$

with the initial condition $x(t = 0) = 1$. Compare the exact solution to this equation with the approximated ones obtained with the Euler method, the modified Euler method and the second order Runge-Kutta method — see Fig. 2.
After you have completed this and the previous exercise, solve them again by building two external function routines for the modified Euler method and the second order Runge-Kutta method, respectively.

**12.0.4 Exercise:**

Solve the following differential equation

$$\frac{dx}{dt} = 5x - x^2 \qquad\qquad x(t=0) = 1 \ , \tag{29}$$

with the three approximated methods we have so far introduced (Euler, modified Euler, and second order Runge-Kutta) and compare the approximated solutions you get with the exact one:

$$x(t) = \frac{5e^{5t}}{4 + e^{5t}} \ . \tag{30}$$

**12.0.5 Exercise**

Solve numerically the following first order differential equation with the second order Runge-Kutta method

$$\frac{dx(t)}{dt} = -(3t^2 - 2t + 5)[x(t) - 1] \qquad\qquad x(0) = x0 \ ,$$

for three different initial conditions, $x(0) = 0$, $x(0) = 1$, and $x(0) = 2$ and compare the numerical results you obtain with the exact solution.

## 12.1 Error in the Runge–Kutta method

It easy to understand why, by taking the middle point, the approximation we do is now second order rather than first. Let us consider for simplicity the case in which $f(x,t) = f(t)$ therefore the solution of the differential equation can now be obtained by direct integration,

$$x(t) - x(t_0) = \int_{t_0}^{t} dt' f(t') \ ,$$

and the solution at the point $t_{n+1}$ can be exactly derived by the one at $t_n$ by

$$x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} dt' f(t') \ . \tag{31}$$

If, in the interval $[t_n, t_{n+1}]$ we approximate the function $f(t')$ by it's middle point $f(t') \simeq f(t_{n+1/2}) + (t' - t_{n+1/2}) f'(t_{n+1/2}) + O(\delta t^2)$, we can notice that substituting back into the expression (31) the term

$$\int_{t_n}^{t_{n+1}} dt' (t' - t_{n+1/2}) = 0 \ ,$$

is zero. Therefore automatically by considering the middle point, we are doing an error in the solution $x_{n+1}$ which is of the order of $\delta t^2$.

# 13 Application: Harmonic oscillator, friction, and external drive

### 13.0.1 Exercise: the harmonic oscillator

Consider the equation of motion for an undamped spring–mass,

$$\frac{d^2x}{d\tilde{t}^2} = -x \ , \tag{32}$$

and solve this equation with the following initial conditions

$$x(0) = 1 \qquad\qquad v(0) = \left.\frac{dx}{d\tilde{t}}\right|_{\tilde{t}=0} = 0 \ ,$$

applying the Runge–Kutta method.

**Questions:**

1. Compare the Runge–Kutta numerical solution with the solution obtained with the Euler algorithm and with the exact one, $x(t) = \cos(\tilde{t})$;
2. for a given number of points $N$ in the interval $\tilde{t} \in [0, 4\pi]$ which approximations is closer to the exact one?
3. evaluate the errors for the Euler method, `max(abs(xRK-xexact))`, and the one for the Runge–Kutta method, `max(abs(xE-xexact))` with increasing the number of points $N$ and observes which method gives the fastest convergence — you need to plot the error versus $N$ in a log-log scale to see which curve has the steepest slope.

**Hints to solve Exercise 13.0.1**

- You can write a 2$^{\text{nd}}$–order differential equation,

$$\frac{d^2x}{dt^2} = g(x, \frac{dx}{dt}) \ ,$$

  as a system of two 1$^{\text{st}}$–order differential equations:

$$\begin{cases} \dfrac{dx}{dt} = v \\ \dfrac{dv}{dt} = g(x, v) \ ; \end{cases}$$

- for such equations the 2$^{\text{nd}}$–order Runge–Kutta method writes

$$\begin{cases} x_{i+1} \simeq x_i + \delta t v_{i+1/2} \\ v_{i+1} \simeq v_i + \delta t g(x_{i+1/2}, v_{i+1/2}) \ ; \end{cases}$$

- the middle–point values, $v_{i+1/2}$ and $x_{i+1/2}$ can be approximated as:

$$\begin{cases} v_{i+1/2} \simeq v_i + \dfrac{\delta t}{2} g(x_i, v_i) \\ x_{i+1/2} \simeq x_i + \dfrac{\delta t}{2} v_i \ . \end{cases}$$

The damped harmonic oscillator, such as a mass $m$, connected to a spring and submerged in a fluid, experiences a frictional force, which can be modeled as a force proportional, and opposite in direction, to the oscillator velocity:

$$\frac{d^2x(t)}{dt^2} + 2\zeta\omega_0 \frac{dx(t)}{dt} + \omega_0^2 x(t) = 0 \qquad v(0) = v_0 \qquad x(0) = x_0 \ , \tag{33}$$

where $\omega_0 = \sqrt{\dfrac{k}{m}}$ and $\zeta = \dfrac{c}{2\sqrt{mk}}$.

**13.0.2 Exercise**

Solve numerically the damped harmonic oscillator (33) in the interval $t \in [t_0, t_1] = [0, 18]$ s for the underdamped case where $m = 1.4$ kg, $k = 6.5$ N/m, $c = 0.8$ kg/s and with initial conditions $x_0 = 2.8$ m and $v_0 = 0$ and compare the numerical result with the exact one

$$x(t) = \frac{x_0}{\sqrt{1-\zeta^2}} e^{-\gamma t} \cos(\omega_0 \sqrt{1-\zeta^2} - \varphi) \ ,$$

where $\gamma = \dfrac{c}{2m}$ and $\varphi = \arccos(\sqrt{1-\zeta^2})$.

In presence of an external drive $F(t)$, the equation of motion of a damped harmonic oscillator reads as:

$$\frac{d^2x(t)}{dt^2} + 2\zeta\omega_0 \frac{dx(t)}{dt} + \omega_0^2 x(t) = \frac{F(t)}{m} \qquad v(0) = v_0 \qquad x(0) = x_0 \ . \tag{34}$$

**13.0.3 Exercise**

Consider the case of an external sinusoidal drive, $F(t) = F_0 \sin(\omega t)$, and solve numerically the driven damped harmonic oscillator (34) in the time interval $t \in [t_0, t_1] = [0, 80]$ s for $\omega_0 = 1$ s$^{-1}$, $2\zeta\omega_0 = 0.2$ s$^{-1}$, $F_0/m = 0.1$ m s$^{-2}$, and $\omega = 1.2$ s$^{-1}$, and with initial conditions $x_0 = 0.2$ m and $v_0 = 0.8$ m s$^{-1}$.

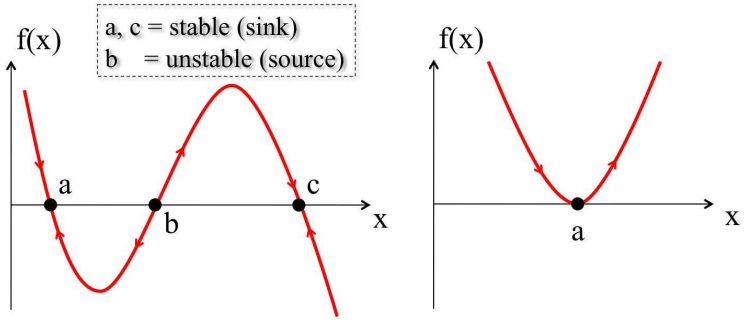



Figure 14: Schematic representation of a phase line diagram and critical points of a differential equation with a form $\frac{dx(t)}{dt} = f(x)$. Left panel: The critical points $a$ and $c$ are stable, while the critical point $b$ is unstable. Right panel: phase line diagram of exercise 15.0.5.

# 14 Application: Planar Pendulum

**14.0.4 Exercise:**

Solve numerically the planar pendulum

$$\frac{d^2\theta}{d\tilde{t}^2} = -\sin\theta \,. \tag{35}$$

with initial condition $\theta_0 = 30°$ and $v_0 = 0$ in the interval $\tilde{t} \in [0, 12\pi]$ by making use of both the Euler method and the $2^{\text{nd}}$–order Runge–Kutta algorithm.

**Questions:**

1. Plot the numerical solution. Determine for which value of the integration steps $N$ the solution starts not to visibly change any longer;
2. for this value of $N$, compare by plotting the numerical solution with the exact solution $\theta(\tilde{t}) = \theta_0 \cos(\tilde{t})$ valid for small angles only. Comment the result you get;
3. change the initial condition $\theta_0$: for which value of $\theta_0$ the two curves are approximatively equal? Why?
4. plot the solution in the phase space $\theta$, $d\theta/d\tilde{t}$.

# 15 Critical points and phase lines

The qualitative behaviour of an ordinary differential equation of the form

$$\frac{dx(t)}{dt} = f(x) \qquad\qquad x(0) = x_0 \,, \tag{36}$$

can be deduced by the phase line diagram (see the schematic Fig. (14)), where one can identify the critical points, i.e. the points where $f(x) = 0$, and their stability. In the particular example of the left panel of Fig. 14, the critical points

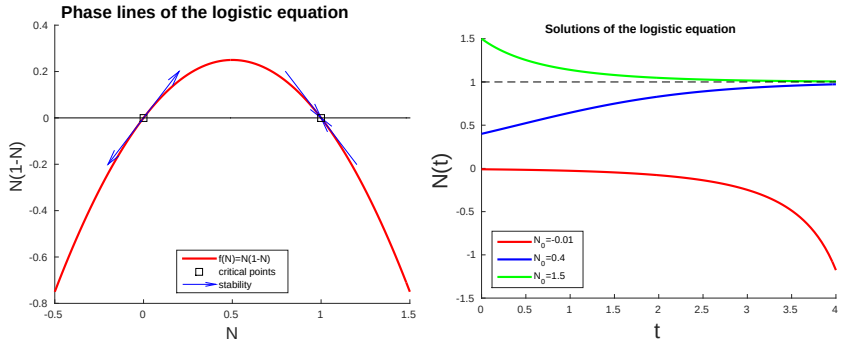



Figure 15: Left: Critical points and phase lines of the logistic equation (38); solutions of the equation for different initial conditions.

$a$ and $c$ are stable and thus act as sinks: all the solutions with initial conditions $x_0 < b$ tend asymptotically to $a$, while the solutions with $x_0 > b$ tend to $c$:

$$\lim_{t\to+\infty} x(t) = \begin{cases} a^- & x_0 < a \\ a^+ & a < x_0 < b \end{cases} \qquad \lim_{t\to+\infty} x(t) = \begin{cases} c^- & b < x_0 < c \\ c^+ & x_0 > c \, . \end{cases}$$

**15.0.5 Exercise:**

Solve both numerically and analytically the differential equation (see right panel of Fig. 14):

$$\frac{dx(t)}{dt} = x^2 \qquad\qquad x(0) = x_0 \, ,$$

with initial conditions $x_0 < 0$ and $x_0 > 0$ and compare the solutions you get.

**15.0.6 Exercise:**

Find the phase lines, critical point and stability of the differential equation

$$\frac{dx(t)}{dt} = 1 - x(t) \, .$$

Solve it exactly and numerically for $x(0) = x_0 > 1$ and $x_0 < 1$.

**15.0.7 Exercise:**

Find the phase lines, critical point and stability of the differential equation

$$\frac{dx(t)}{dt} = \log[x(t)] + x(t)^{1/3} - 0.2x(t)^2 \, .$$

Solve it numerically for different initial conditions $x_0$.

# 16 Application: Logistic equation

Equations of the form

$$\frac{dN(t)}{dt} = \kappa N(t) \qquad\qquad N(0) = N_0 \ , \tag{37}$$

describe either the exponential growth or the exponential decay of a population at a rate proportional to the size of the population, $N(t)$, with a rate constant $\kappa$. If $\kappa > 0$, $N(t)$ grows exponentially, $N(t) = N(0)e^{\kappa t}$, while if $\kappa < 0$, the population decays exponentially, $N(t) = N(0)e^{-|\kappa| t}$.

A more accurate model assumes that the relative growth start decreasing when $N(t)$ approaches a fraction of the carrying capacity $N_c$ of the environment — population growth in a constrained environment. The corresponding equation is called logistic differential equation, and, for $N_c = 1$ and $\kappa = 1$, reads as:

$$\frac{dN(t)}{dt} = N(t)\left[1 - N(t)\right] \qquad\qquad N(0) = N_0 \ . \tag{38}$$

As explained in the previous section, there is no need to explicitly solve this equation to understand the behaviour of the population $N(t)$ with the time $t$ for different initial conditions $N_0$, rather one has to understand the phase lines and the stability of the two critical points $N = 0$ and $N = 1$ — see left panel of Fig. 15. Nevertheless, Eq. (38) can be solved exactly by variable separation, giving:

$$N(t) = \frac{N_0}{N_0 + (1 - N_0)e^{-t}} \ . \tag{39}$$

Different solutions can be seen in the right panel of Fig. 15.

**16.0.8 Exercise:**

Numerically solve the logistic equation (38) for different values of the initial population $N_0$ and compare the numerical result you get with the exact one of Eq. (39). Linearise the problem around the critical points $N = 0$ (unstable) and $N = 1$ (stable), and compare the solutions you get in terms of an exponential function with the general solution. When can you use the linearised ones?

**16.0.9 Exercise:**

Consider the following modified logistic equation

$$\frac{dN(t)}{dt} = aN^2(t)\left[1 - bN(t)\right] \qquad\qquad N(0) = N_0 \ .$$

Where $a = 2$ and $b = 50$. Find the critical points and their stability. Study the numerical solutions for different initial populations $N_0$, as well as close to the stable critical point, comparing your results with the ones of the usual logistic equation.

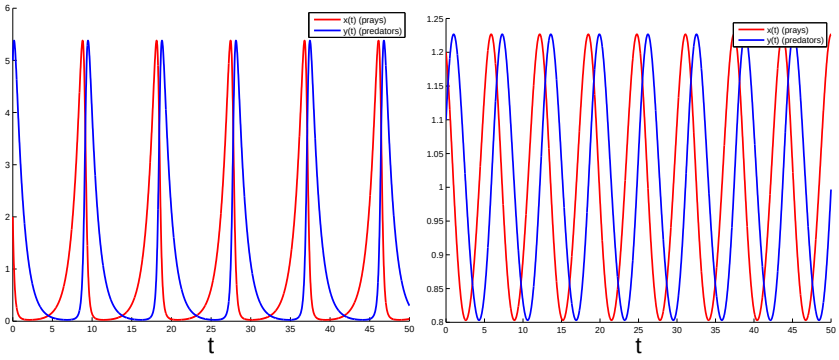



Figure 16: Solutions of the prays and predators problem (41) with parameters $a = b = c = d = 1$ and initial conditions: $x_0 = 1.2$ and $y_0 = 1.1$ (left panel); $x_0 = 2$ and $y_0 = 5$ (right panel).

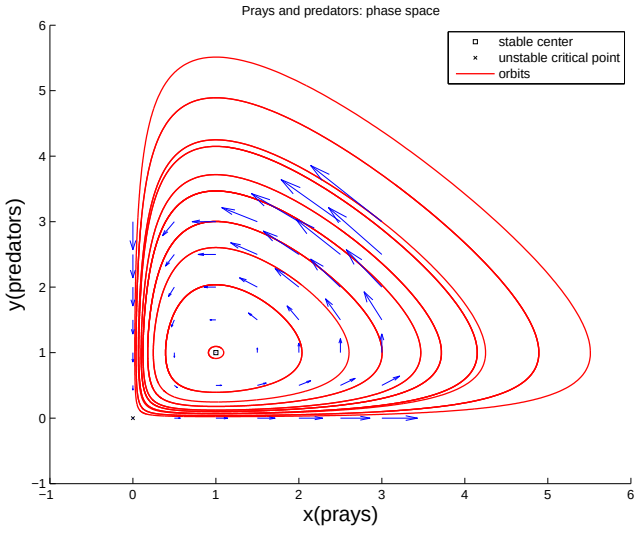



Figure 17: Orbits in the phase space of the pray and predator problems for the following parameters $a = b = c = d = 1$ and different initial conditions $x_0$ and $y_0$.

# 17 Application: Prays and predators

Volterra-Lotka equations are differential equations that can be used to model predator-prey interactions. The original system discovered by both Volterra and Lotka independently consisted of two entities. Vito Volterra developed these equations in order to model a situation where one type of fish is the prey for another type of fish. The model was simplified by the following assumptions:

1. The prey population increases exponentially in the absence of predators.
2. The predator population decreases exponentially in the absence of prey.
3. The prey population decreases relative to the frequency with which predators meet prey as a result of predation.
4. The predator population increases relative to the frequency with which predators meet prey as a result of predation.

Using these assumptions, the Volterra-Lotka equations for the two-dimensional predator-prey system with exponential growth is defined by the following system of differential equations:

$$\frac{dx(t)}{dt} = ax - bxy \qquad x(0) = x_0 \tag{40}$$

$$\frac{dy(t)}{dt} = dxy - cy \qquad y(0) = y_0 \ . \tag{41}$$

The critical points, $\frac{dx(t)}{dt} = 0 = \frac{dy(t)}{dt}$ are given by $(x, y) = (0, 0)$ and $(x, y) = (\frac{a}{b}, \frac{c}{d})$. By evaluating the Jacobian

$$J(x, y) = \begin{pmatrix} a - by & -bx \\ dy & dx - c \end{pmatrix} \tag{42}$$

it can shown that the first is unstable, while the second is a stable center — see Fig. 16.

**17.0.10 Exercise:**

Numerically solve the prays and predators equations (41) for $a = 4$, $b = 2$, $c = 3$, and $d = 3$ and for different initial conditions $x_0$ and $y_0$ far and close to the critical points. Plot the corresponding orbits as in Fig. 16 — plot also the flow lines. Linearise the equations close to the stable critical point, evaluate analytically the solutions and plot them together with the numerical ones for the full problem.

**17.0.11 Exercise:**

Numerically solve the following equations describing two competing populations

$$\frac{dx(t)}{dt} = 60x - 4x^2 - 3xy$$
$$\frac{dy(t)}{dt} = 42y - 2y^2 - 3xy$$

for different initial conditions $x_0$ and $y_0$. Evaluate the critical points and their stability. Linearise close to one of the stable critical points, evaluate analytically the solutions and plot them together with the numerical ones for the full problem.

# 18 Statistical analysis: basic notions

The statistical analysis deals with the organisation and interpretation of set of data, which are the result of either an experiment, or, for example, a survey.

## 18.1 Descriptive statistics: Mean, Variance, Standard deviation

Given a somehow large set of data $\{x\} = \{x_1, x_2, \ldots, x_N\}$ (which can be the results of either measurements, or a survey, etc. etc.), the descriptive statistics quantify in few parameters some of the main features of such a collection of data. In addition, these parameters may enable us to make comparison with a different set of data describing, say, a similar experiment taken at different conditions and allowing to draw some important conclusions about the physics behind the experiment. Given $\{x\} = \{x_1, x_2, \ldots, x_N\}$, the statistical parameters that can be defined are — in Matlab, we can store $\{x\}$ in a vector (e.g., a row vector), therefore from now onwards we will write $\mathtt{x} = [\mathtt{x_1\ x_2\ \ldots\ x_N}]$):

**Mean** — the arithmetic mean is the sum of the data divided by the total data number:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \ . \tag{43}$$

**Standard deviation** — the standard deviation is a measure of the data variability and shows how much dispersed the data are from their mean $\overline{x}$: A smaller value of the standard deviation $\sigma$ means the data are more closely distributed around the mean than if the standard deviation is larger, meaning the data are more spread out from the mean:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (x_i - \overline{x})^2} \tag{44}$$

**18.1.1 Example:**

Let us consider the two sets of data $x = [2\ 9\ 3\ 12\ 9]$ and $y = [5\ 6\ 7\ 8\ 9]$; though it is clear they have the same average ($\overline{x} = 7 = \overline{y}$), the data in $x$ are more spread out ($\sigma_x = 4.3$) then the ones in $y$, which standard deviation is given by $\sigma_y = 1.6$ (check it).

**Variance** — squared value of the standard deviation:

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{45}$$

**Median** — the median $x_{1/2}$ is the data that separates the higher half of the data set, from the lower half, when the data are ordered from the lowest value to the highest one. In the cases considered in the example 18.1.1, $x_{1/2} = 9$ and $y_{1/2} = 7$. **Note** that if one has an even number of data, then the median is defined as the mean of the two middle values. Therefore in general:

$$x_{1/2} = \begin{cases} x_{\frac{N-1}{2}+1} & \text{if } N \text{ is odd} \\ \dfrac{x_{\frac{N}{2}} + x_{\frac{N}{2}+1}}{2} & \text{if } N \text{ is even .} \end{cases}$$

N.B. the command `floor(a)` rounds the elements of `a` to the nearest integer — towards minus infinity, i.e., `floor(4.2) = floor(4.8) = 4` and `floor(-4.8) = floor(-4.2) = -5`.

**18.1.2 Example:**

The following set of data $z = [13\ 0\ 7\ 10\ 8\ 7\ 2\ 9\ 20\ 11]$ has 10 elements, therefore the median is given by (`sort(z)` $= [0\ 2\ 7\ 7\ 8\ 9\ 10\ 11\ 13\ 20]$) $z_{1/2} = (z_5 + z_6)/2 = (8+9)/2 = 8.5$ (check it).

**18.1.3 Exercise:**

Consider the following data representing the monthly average rainfall (in millimeters) in respectively Spain, Madrid, and London, where the data are ordered from January to December:

$$r_{sp} = [50\ 48\ 55\ 44\ 47\ 13\ 8\ 18\ 39\ 78\ 60\ 55]$$
$$r_{ma} = [62\ 42\ 70\ 35\ 35\ 4\ 1\ 3\ 32\ 94\ 71\ 56]$$
$$r_{lo} = [58\ 39\ 51\ 49\ 54\ 53\ 48\ 56\ 58\ 62\ 61\ 63]\ .$$

This means that for example in January in Spain, on average, 50 mm of water have fallen ($r_{sp}(1) = 50$), 62 mm in Madrid, and 58 mm in London, and so on.

- Evaluate mean, standard deviation and median of the monthly average rainfall in Spain, Madrid and London; compare your results with the ones obtained by making use of the built-in functions `mean(), std()`, and `median()`;
- what can you deduce from the values you have obtained of mean and standard deviation? And what can you say about the generic behaviour of rainfall in the three different locations?

**Hints to solve Exercise 18.1.3**

- Remember to use the commands `sum(), length()`, and `sort()`.

**Data binning** — It might be sometimes useful to group together the data that belong to the same interval of values, i.e., to bin the data, and to separately consider their mean and standard deviation in each bin. For example in the case of the exercise 18.1.3, it might be useful to bin the data in four different intervals, depending on the season rather than on the specific month:

$$\text{spring} = \text{March, April, May}$$
$$\text{summer} = \text{June, July, August}$$
$$\text{autumn} = \text{September, October, November}$$
$$\text{winter} = \text{December, January, February}\ ,$$

so that one can compare behaviours of rainfall in a specific season.

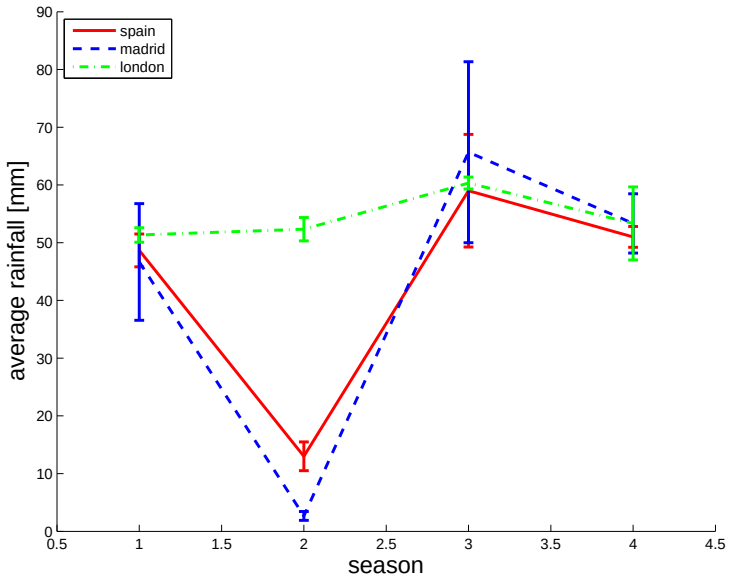



Figure 18: The figure shows the averaged monthly rainfall in millimeters in respectively Spain, Madrid , and London, where the data have been binned into the four seasons (1=spring, 2=summer, 3=autumn, 4=winter): Data are mean values and error bars the associated standard deviations.

### 18.1.4 Exercise:

Consider the same data of exercise 18.1.3:

$$r_{sp} = [50\ 48\ 55\ 44\ 47\ 13\ 8\ 18\ 39\ 78\ 60\ 55]$$
$$r_{ma} = [62\ 42\ 70\ 35\ 35\ 4\ 1\ 3\ 32\ 94\ 71\ 56]$$
$$r_{lo} = [58\ 39\ 51\ 49\ 54\ 53\ 48\ 56\ 58\ 62\ 61\ 63]\ .$$

- Bin the data about the rainfall in Spain, Madrid, and London, into the four seasons;
- plot the mean in each bin as a function of the season;
- use the standard deviation in each bin as error bars, so that your plot should look like the one in Fig.1;
- what can you conclude from this analysis?

**Hints to solve Exercise 18.1.4**

- Use the command `errorbar()` to plot the means with their errors (`doc errorbar`).

## 18.2 Histograms and probability distributions

We have seen that descriptive statistics offers an important summary of a large set of data, allowing the comparison of different sets. However, describing a large set of data with few parameters it is only an approximation and might lead to the risk of distorting or loosing important details of the original data. Probability distributions, $p(x)$, will give us a more complete information, telling us the probability to find the value $x$ within a given interval $[x_{min}, x_{max}]$ when we perform an experiment.

### 18.2.1 Histograms

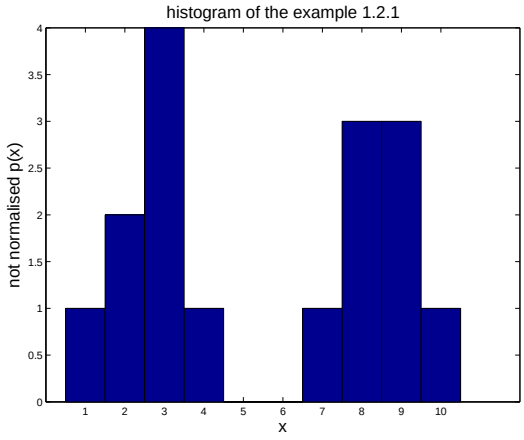
A previous step in order to determine the probability distribution, is the one of constructing an histograms with the data.

### 18.2.2 Example:

Let us consider the following data comprised of natural numbers between $[1, 10]$:

$$\texttt{data} = [\texttt{9 2 8 7 1 3 4 8 3 2 3 10 9 3 8 9}] \, .$$

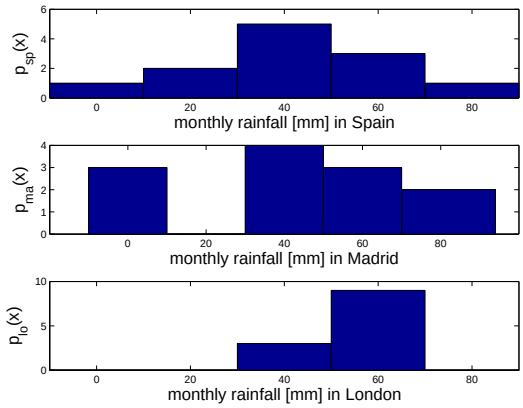
For example, we can automatically bin the elements of the vector `data` into 10 containers centered on the values stored in $\texttt{x} = [\texttt{1 2 3 4 5 6 7 8 9 10}]$, with the command `hist(data,x)` (see the figure below and try to reproduce it).




If we define $[\texttt{p x}] = \texttt{hist(data,x)}$, then we can check that the vector `p` contains `length(x)` elements specifying how many times the number $\texttt{x(1)} = 1$ is repeated in `data` (i.e., `p(1)=1`), how many time the number $\texttt{x(2)} = 2$ is repeated in `data` (i.e., `p(2)=2`), and so on (`p= [1 2 4 1 0 0 1 3 3 1]`).

### 18.2.3 Exercise:

Consider the data of exercise 18.1.3 and plot an histogram for each set of data for the intervals of rainfall millimeters `x=(0:20:80)`.




### 18.2.4 Probability distributions

The probability distribution can be obtained by normalising the histogram. For a discrete variable $x$, then normalising means dividing by the total number of data. For a continuous variable $x$, defined in an interval $[x_{min}, x_{max}]$, its normalised probability distribution satisfies the condition

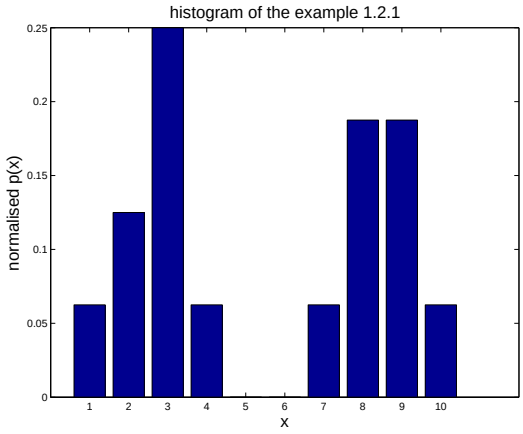
$$\int_{x_{min}}^{x_{max}} dxp(x) = 1 \, . \tag{46}$$

**18.2.5 Exercise:**

Considering the same data of the example 18.2.2

$$\texttt{data} = [9\ 2\ 8\ 7\ 1\ 3\ 4\ 8\ 3\ 2\ 3\ 10\ 9\ 3\ 8\ 9]\ .$$

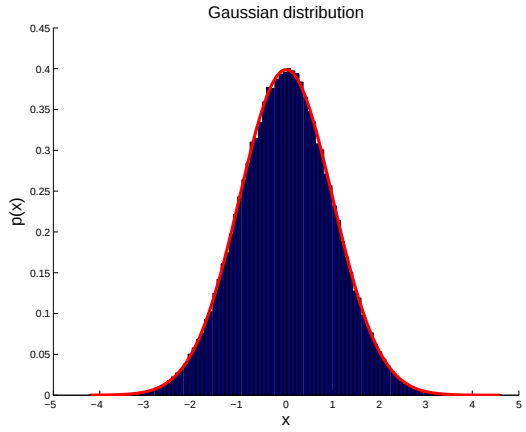
normalise the histogram with a binning vector $\texttt{x} = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$.



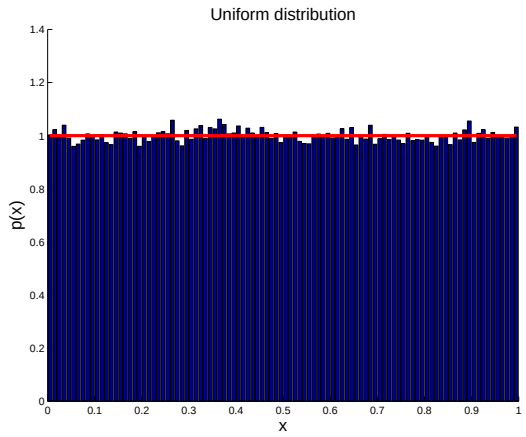

Compare your result with the one obtained with the command `normalize()`.

### 18.2.6 Exercise:

1. Use the function `randn()` to generate $N$ Gaussian-distributed random numbers centered on $x = 0$ with a standard deviation equal to 1;
2. Use the function `hist()` in order to produce an histogram with the data obtained above; try with different values of $N$;
3. Normalise the histogram in order to obtain the probability distribution $p(x)$;
4. Compare what you obtain with the analytical expression $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

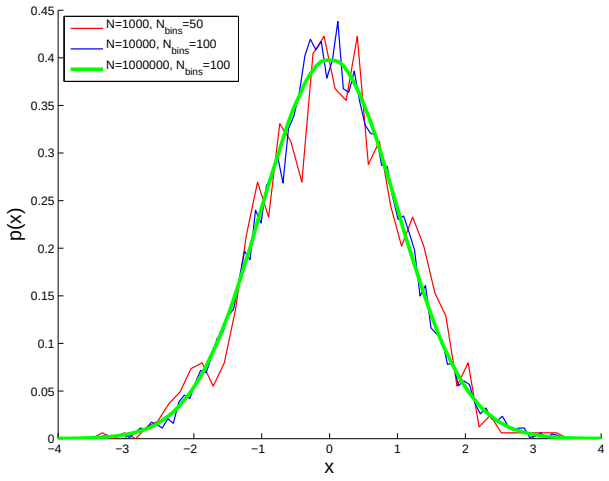



- Use the function `rand()` to generate $N$ random numbers uniformly-distributed on the interval $[0, 1]$ and repeat the same steps 1–4 above.



**18.2.7 Exercise:**

Plot the (normalised) Gaussian distribution obtained in the exercise 18.2.6 with the command `randn()` with different numbers of point $N$ and different numbers of bins $N_bins$ and compare them.




# 19 Data Fitting: Introduction

There are cases where we want to compare the data $\mathbf{y} = (y_1, y_2, y_3, \ldots, y_N)$ we have obtained as the result of an experiment to a theoretical model and therefore to fit the model to the data. Let us consider for examples the coldest and hottest average temperatures per month in Madrid (see Fig. 1):

$$\begin{aligned} T_{min} &= (2.6, 3.7, 5.6, 7.2, 10.7, 15.1, 18.4, 18.2, 15.0, 10.2, 6.0, 3.8)^\circ C \\ T_{max} &= (9.7, 12.0, 15.7, 17.5, 21.4, 26.9, 31.2, 30.7, 26.0, 19.0, 13.4, 10.1)^\circ C \ , \end{aligned} \tag{47}$$

where the first data refer to January, the second to February and so on (in each case we have $N = 12$ data). And let us suppose that a theoretical model foresee that both minimum and maximum temperatures vary like a 3$^{\text{rd}}$ order polynomial, i.e., like

$$f(x, \mathbf{a}) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 = \sum_{j=1}^{Np=4} a_j x^{j-1} \ , \tag{48}$$

where $N_p = 4$ is the number of free parameters of our model that we have to determine via fitting. The question is what is the best choice for $\mathbf{a} = (a_1, a_2, a_3, a_4)$, i.e., how do we obtain the values of the parameters $a_1$, $a_2$, $a_3$, and $a_4$ that are best fitting of respectively either $T_{min}$ or $T_{max}$? (see Fig. 1).

## 19.1 Least-square fitting

One of the different possible methods to optimise a fit is the least-square fitting. The best fitting parameters obtained according to the least-square method are

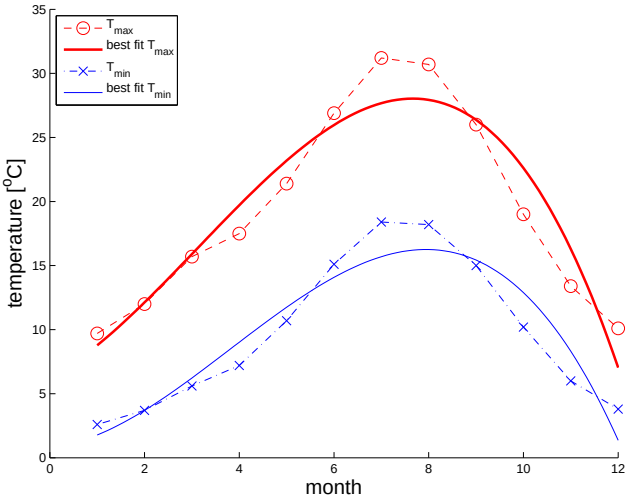



Figure 19: The figure shows the averaged monthly maximum (dashed line) and minimum (dot-dashed line) temperatures in Madrid, where $\mathbf{x} = (x_1, x_2, \ldots, x_{N=12}) = (1, 2, \ldots, 12)$ are the months starting from January. Continue lines represent the best fitting $3^{\text{rd}}$ order polynomial curves obtained with a linear least-square fitting.

the ones that minimise the sum of the squared distances between the data $\mathbf{y} = (y_1, y_2, y_3, \ldots, y_N)$ and the fitting function obtained from a certain model, $f(x, \mathbf{a})$, where $\mathbf{a} = (a_1, a_2, a_3, \ldots, a_{N_p})$. Therefore one has to minimise the following quantity:

$$\mathcal{F}(\mathbf{a}) = \sum_{i=1}^{N} [y_i - f(x_i, \mathbf{a})]^2 \ . \tag{49}$$

Here, we indicate with $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_N)$ the values of the parameter $x_i$ which corresponds to the data $y_i$, i.e., $y_i = y(x_i)$ — in the previous example, $\mathbf{x} = (1, 2, 3, \ldots, 12)$ are the months of the year starting from January. Let us for the moment consider the general case where $f(x, \mathbf{a})$ is a generic function of the variable $x$ and of the $N_p$ parameters $\mathbf{a}$ — later, in Sec. 19.2, we will see the specific case of the linear least-square fitting. So let us keep in mind that $N_p$ is the number of parameters of our model (in our notation $\mathbf{a}$ is a vector with $N_p$ components), while $N$ is the number of data points (in our notation both $\mathbf{y}$ and $\mathbf{x}$ are vectors with $N$ components). Clearly the more parameters one uses, the better the fit will be, however the less parameters one uses, the better will be the model.

The distances between the fitting curve and the data are plotted in Fig. 2 for $T_{max}$ given in Eq. (47) .

In order to minimise $\mathcal{F}(\mathbf{a})$, one has to solve the following $N_p$ coupled equa-

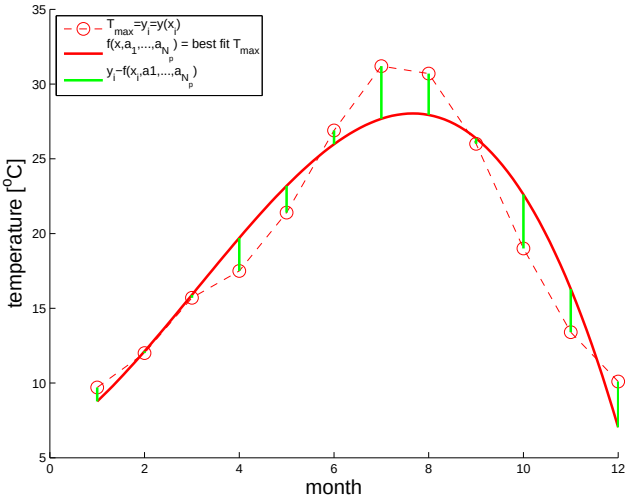



Figure 20: The figure shows the averaged monthly maximum (dashed line) temperature in Madrid and the best fitting 3rd order polynomial curve (solid line) obtained with a linear least-square fitting like in Fig. 1. In green are plotted the distances between the data points $y_i$ and the fitting function $f(x_i, \mathbf{a})$, for each value of $\mathbf{x} = (x_1, x_2, \ldots, x_{N=12}) = (1, 2, \ldots, 12)$.

tions:

$$\begin{aligned}
\frac{\partial \mathcal{F}(\mathbf{a})}{\partial a_1} &= 0 = -2 \sum_{i=1}^{N} [y_i - f(x_i, \mathbf{a})] \frac{\partial f(x_i, \mathbf{a})}{\partial a_1} \\
\frac{\partial \mathcal{F}(\mathbf{a})}{\partial a_2} &= 0 = -2 \sum_{i=1}^{N} [y_i - f(x_i, \mathbf{a})] \frac{\partial f(x_i, \mathbf{a})}{\partial a_2} \\
&\ldots\ldots \\
\frac{\partial \mathcal{F}(\mathbf{a})}{\partial a_{N_p}} &= 0 = -2 \sum_{i=1}^{N} [y_i - f(x_i, \mathbf{a})] \frac{\partial f(x_i, \mathbf{a})}{\partial a_{N_p}} \ .
\end{aligned} \tag{50}$$

In general, this is a system of $N_p$ non-linear equations for the $N_P$ parameters $\mathbf{a}$.

## 19.2 Linear least-square fitting

In this particular case the fitting function of our model $f(x, \mathbf{a})$ contains a linear combination of the $N_p$ fitting parameters $\mathbf{a}$:

$$f(x, \mathbf{a}) = \sum_{j=1}^{N_p} a_j \varphi_j(x) \ , \tag{51}$$

where the $N_p$ functions $\varphi_j(x)$ can be arbitrary functions of $x$. Now, we can find the solution $\mathbf{a}$ of the least-square minimisation by solving a linear system

of equations for the vector $\mathbf{a} = (a_1, a_2, \ldots, a_{N_p})$. In fact, the system (50) now becomes:

$$\begin{aligned} \sum_{i=1}^{N} \left[ y_i - f(x_i, \mathbf{a}) \right] \varphi_1(x_i) &= 0 \\ \ldots\ldots & \\ \sum_{i=1}^{N} \left[ y_i - f(x_i, \mathbf{a}) \right] \varphi_{N_p}(x_i) &= 0 \ , \end{aligned} \tag{52}$$

which can also be written as

$$\sum_{i=1}^{N} \left[ y_i - \sum_{k=1}^{N_p} a_k \varphi_k(x_i) \right] \varphi_j(x_i) = 0 \qquad \forall j = (1, 2, \ldots, N_p) \ . \tag{53}$$

Defining the $N \times N_p$ matrix $A_{ij} = \varphi_j(x_i)$, i.e.,

$$A = \begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \ldots & \varphi_{N_p}(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \ldots & \varphi_{N_p}(x_2) \\ \ldots & \ldots & \ldots & \ldots \\ \varphi_1(x_N) & \varphi_2(x_N) & \ldots & \varphi_{N_p}(x_N) \end{pmatrix} \ , \tag{54}$$

then Eq. (53) reads as:

$$\sum_{i=1}^{N} \left[ y_i - \sum_{k=1}^{N_p} a_k A_{ik} \right] A_{ij} = 0 \qquad \forall j = (1, 2, \ldots, N_p) \ ,$$

or equivalently as

$$\sum_{i=1}^{N} \sum_{k=1}^{N_p} \left( A^T \right)_{ji} A_{ik} a_k = \sum_{i=1}^{N} \left( A^T \right)_{ji} y_i \ ,$$

or in other words $A^T A \mathbf{a} = A^T \mathbf{x}$. This system of linear equations admits as a solution:

$$\mathbf{a} = \left( A^T A \right)^{-1} A^T \mathbf{y} \ . \tag{55}$$

### 19.2.1 Example:

As a specific case let us consider the fitting to a 3rd order polynomial function $f(x, \mathbf{a}) = \sum_{j=1}^{Np=4} a_j x^{j-1}$ like in Eq.(48). Let us indicate generically the data with $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ and the corresponding $x$-values with $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where $N \gg N_p = 4$. In this case the matrix $A$ can be written as:

$$A_{ij} = \varphi_j(x_i) = x_i^{j-1} \qquad A = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & x_N^3 \end{pmatrix} , \qquad (56)$$

where $i = (1, 2, \ldots, N)$ and $j = (1, 2, 3, N_p = 4)$. Therefore, a script defining the matrix $A$ is very easy to write:

```
for i=1:N
      for j=1:Np
            A(i,j)=x(i)^(j-1)
      end
end
```

In the specific case considered in Eqs. (47), one has

$$A = \begin{pmatrix} 1^0 = 1 & 1^1 & 1^2 & 1^3 \\ 2^0 = 1 & 2^1 & 2^2 & 2^3 \\ \ldots & \ldots & \ldots & \ldots \\ 12^0 = 1 & 12^1 & 12^2 & 12^3 \end{pmatrix} ; . \qquad (57)$$

**N.B.** In the notation of Eq. (55), $\mathbf{y}$ is a **column** vector, therefore if you have defined it as a row vector, you will have to consider its transpose ($\mathtt{y}'$). Also note that in general $A$ is not a squared matrix, but $A^T A$ in Eq. (55) certainly is, so we can consider its inverse.

### 19.2.2 Exercise:

Consider the data given in Eqs. (47) and fit both data with a 3rd order polynomial function $f(x, \mathbf{a}) = \sum_{j=1}^{Np=4} a_j x^{j-1}$ using the linear least-square fitting.

- Determine the best fitting parameters $\mathbf{a} = (a_1, a_2, a_3, a_4)$ (note that in this case $\mathbf{x} = (1, 2, 3, \ldots, 12)$;
- reproduce both Figs. 1 and 2;
- evaluate the value of $\mathcal{F}(\mathbf{a})$ that you obtain with the best set of parameters $\mathbf{a}$ evaluated above.

**19.2.3 Exercise:**

Repeat the same exercise 19.2.2 but with a function $f(x, \mathbf{a})$ that gives a better fitting, i.e., a $4^{\text{th}}$ order polynomial function. Evaluate $\mathcal{F}(\mathbf{a})$ for both the exercise 19.2.2 and this one. Then try with a sinusoidal function, $f(x, a_1, a_2, a_3) = a_1 \sin(a_2 x + a_3)$ — start from the definition (49) and try to minimise this function for different values of the parameters $a_1$, $a_2$, and $a_3$.

**Hints to solve Exercise 19.2.3**

- When you try with a sinusoidal function, remember you have to minimise Eq. (49).

**19.2.4 Exercise:**

Consider the probability distribution function that you can obtained numerically by making use of the command `randn()` and `hist()` — do not forget to normalise your distribution! Fit the data you get with a Gaussian probability distribution

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\overline{x})^2}{2\sigma^2}} , \tag{58}$$

with zero mean, $\overline{x} = 0$.

- If you use a fitting function $p(x) = a_1 e^{-a_2 x^2}$, what values of $\sigma$ do you get separately from $a_1$ and from $a_2$? Do these values agree and what value for $\sigma$ do you expect?
- Try different fittings by varying the values of $N$ (number of the data point) and the number of bins used to evaluate the normalised histogram.

**Hints to solve Exercise 19.2.4**

- you can perform a linear least-square fit on $\ln p(x) = \ln a_1 - a_2 x^2 = \tilde{a}_1 - a_2 x^2$.