

Physics Laboratory with Arduino and smartphones

Giovanni Organtini

August 9, 2020

Contents

Preface	5
1 Physics and Nature	7
Physics and other sciences	7
Measurements and the laws of physics	10
The process of measurement	11
2 Establishing a system of units	17
Measuring light intensity	17
Units definition	18
Systems of units	22
3 Taking data	31
Instruments	31
Smartphones	33
Arduino	34
Open source make it better	35
Measuring light intensity with a smartphone	36
Measuring light intensity with Arduino	38
Understanding Arduino programming	41
Python data collection	42
4 Uncertainties	47
Data analysis	47
Data analysis with Python	52

5 Establishing physics laws	59
Light absorption	59
Taking the average with Arduino	60
A first look at data	63
Plotting graphs and interpolating data	67
An approximated model	70
Non-polynomial models	71
The exponential model	74
6 Free fall and accelerations	79
Setting up the experiment	79
Measuring times	80
Photogates	81
Measuring time with Arduino	82
An acoustic stopwatch	84
Uncertainty propagation	88
Measuring accelerations	89
MEMS accelerometers	92
Annotating graphs	95
Instruments characteristics	96
7 Understanding data distributions	103
On the values returned by instruments	103
Probability	104
Bayes's Theorem and Physics	115
Statistical distributions of data	121
Uniform distribution	122
Expected value, variance and moments	125
Combining errors, revisited	128
The binomial distribution	131
The shape of the binomial distribution	134

Random walk	138
The Poisson distribution	145
The shape of the Poisson distribution	149
8 Counting experiments	159
Experiments with binomial and Poisson statistics	159
Operations on lists	160
The Chauvenet's criterion	161
Simulating advanced experiments	162
Using Arduino pins	165
The PHYPHOX editor	170
Readily available particle detectors	172
Images manipulation with Python	176
9 The normal distribution	185
A distribution depending on the distance	185
The central limit theorem	189
Experimental proof of the central limit theorem	190
The Markov's and Chebyschev's Inequalities	194
Testing the Chebyschev's Inequality	197
The Law of large numbers	198
The uncertainty on the average	199
10 Kinematics	205
Designing the experiment	205
Measuring time and distances with Arduino	206
Ultrasonic sensors	207
Arduino data acquisition	210
Taking data	214
Data analysis	215
Evaluating the goodness of a fit	219
Data processing	223

The χ^2 -distribution	226
The least squares method	228
Discarding bad data	232
Measuring the gravity acceleration	233
11 Oscillations	239
An experiment to study elasticity	239
A study of spring dynamics with smartphones	242
Obtaining parameters from data	246
Extracting and manipulating data	249
Optimisation methods	254
A harmonic oscillator with Arduino	256
Newton's Laws	263
A widely applicable model	264
12 Maximum likelihood	273
Bayes' Theorem application to measurements.	273
An experimental proof	275
Parameter estimation	278
13 Physics in non-inertial systems	283
Dynamics in non-inertial systems	283
Free fall	285
Custom experiments with PHYPHOX	286
Centripetal acceleration	287
Coriolis' acceleration	291
Euler's acceleration	292
14 Dynamics of rigid bodies	301
A cylinder rolling along an incline	301
Using a smartphone's gyroscope remotely	306
Arduino Gyroscopes and I2C communications	307

The Arduino Wire library	312
Using an SD card	315
Using SD cards to store data	316
The native SPI protocol	319
15 Wave mechanics	325
Making waves	325
Command line options	328
Properties of a wave	331
The Student's t-distribution	334
Interference	336
Finding the distribution of a random variable	338
Beats	339
Taking audio data with Arduino	341
Dimensional analysis	342
Temperature measurements with Arduino	346
The 1-wire protocol	348
Establishing a correlation	354

Preface

I was dreaming to write this textbook since years. Finally, in late 2019, before the COVID-19 pandemic exploded, I met a far-sighted publisher who agreed to print it. I wrote three chapters of this book when Italy was put in lockdown and courses in our University were moved to remote, with obvious detriment of laboratory courses. I was then asked to provide some help to make them possible and I came with few ideas that were taken skeptically at the beginning, but, indeed, resulted in a complete success. Students were in fact learning even more with respect to what they did in traditional courses. The reason, in my opinion, is simple: making experiments by yourself (I mean, assembling all what is needed to perform the experiment, organise data taking, analyse results writing your own code) teaches much much more than just pushing buttons to start an automatic device made ready by a teacher. Moreover, classical physics laboratory courses suffer from the wrong idea that the purpose of the experiments to be proposed to students is to closely follow the lectured material and imitate original equipments, thus making experiments look quite old-fashioned.

On the contrary, the purpose of laboratory courses should be to teach students how to design and make an experiment, interpret data and derive models. These skills must be developed using modern tools, both in the hardware and software domain, such that students are prepared to the practice in modern physics laboratories.

The idea behind this textbook is that the laboratory practice is learnt by doing and is much less formal than theoretical physics courses. Often, in these courses the emphasis is on statistics and the courses are full of mathematics that seldom catches the attention of students devoted to experimental physics. The main goal of the proposed experiment is to prove that a physics law is correct, but given that most of the physics laws can be formulated within a framework in which lot of approximations are implicitly done, this goal is rarely achieved. As a result, students get frustrated and lose confidence on their experimental abilities. Finally, in many cases operations are tedious (repeat many times the same measurement, compute complicated derivatives and solve difficult equations or systems of equations, etc.), though not very useful for the development of the

competencies. None of us computes averages or uncertainties on measurements using the techniques adopted in university courses. We rather use automatic computation, having learnt a programming language. That's why computer programming is taught in physics courses. However, too often, computers are not used in laboratory courses to derive results in the mistaken belief that doing math manually is the only way to learn and that writing computer programs is a loss of time.

My textbook is instead organised such that the knowledge is acquired progressively, increasing from chapter to chapter, sometimes overriding (in fact, updating) the knowledge acquired up to a given point. Different topics (experiment design, data acquisition, statistical data analysis and their interpretation) are not discussed in dedicated chapters, but are mixed with an introduction to a modern programming language as Python and complemented by a relatively detailed description of Arduino programming.

Sections are color coded such that students (and teachers) less interested in programming can easily skip them without detrimental or, at most, reading them superficially. In particular, specific experimental setup are illustrated under green-coloured section, while programming (both Arduino and Python) are treated under orange-coloured ones. There is no need to make the experiments exactly as proposed: the descriptions are intended to be suggestions from which the reader can draw inspiration for his/her own design. Also, the suggested experiments are just examples from which any teacher can start to propose his/her own. The organisation of the volume is such that a teacher is also free to organise his/her own course in the way he/she prefer. Even the order in which topics are presented is not to be strictly followed, even if chapters are necessarily organised such that the initial ones presents very basic data analysis that is progressively refined.

Main ideas and important results are highlighted as side notes. All the content of the side notes is then summarised at the end of each chapter.

A final remark is important: my personal idea is that laboratory courses must not necessarily follow theoretical courses. In fact, physics is done the opposite way: investigate new and unknown phenomena experimentally, formulate models and test them. As a consequence, it is perfectly viable (indeed, in my opinion, sometimes it is preferable) to perform experiments on topics not yet mastered by students. However, most physics courses are organised differently and we agreed that it was a bit audacious to propose a radically different textbook. For this reason the experiments proposed have to do with mechanics, traditionally the first topic taught in physics courses. There is no need to perform exactly the same experiments. Many variants of them can be imagined and possible alternatives are proposed on the book website.

Chapter 1

Physics and Nature

This chapter is devoted to tentatively define the subject of your studies: physics. We see that trying to define precisely what physics is about is, at least, very difficult, if not impossible. However, it emerges clearly that physics needs measurements: the subject of this book.

1. Physics and other sciences

Physics is often identified as a science, namely, the one devoted to the study of natural phenomena. However, biology, too, is considered a science and it is certainly concerned about something belonging to the set of natural phenomena. On the other hand, many physicists (i.e. people with a degree in physics) work in fields that has anything to do with Nature, like finance, and completely new branches of physics are being born, like *econophysics*.

Even trying to define science is not simple at all. There is plenty of philosophers who have tried their hand at finding a precise definition of it and they still do not agree with each other. Popper's (1902–1994) falsificationism [**Popper, 1959**] is probably the most widely known work about epistemology, i.e. the theory of knowledge, but it is not the only one. Thomas Kuhn (1922–1996) developed a model [**Kuhn, 1962**] of scientific knowledge involving the so called scientific revolutions, introducing the *paradigm shift*, consisting in a radical change in the interpretation of basic concepts, as a driver in the scientific progress. In "Against Method" [**Feyerabend, 1975**], Feyerabend (1924–1994) exposed his radical idea that there are no recognised rules in the scientific community. Lakatos (1922–1974), on the contrary, was firmly convinced that scientific progress can be modelled as a sequence of *research programmes* [**Lakatos, 1968**]: progressive actions driven by a *positive heuristics*, i.e. a self-corroborating approach to solving problems.

Given the above mentioned ambiguities, anyone can try to figure out what is science and what is physics, depending on his/her own attitudes. Our opinion (in fact, it is more than an opinion, but this word has been chosen on purpose, see below) is that, indeed, the scientific community does not follow formal rules,

It is not simple to define physics. It is usually defined as the science of natural phenomena, but physicists deal with artificial ones, too. We suggest you to read some literature about epistemology. In particular, reading the works of Popper, Kuhn, Feyerabend and Lakatos is recommended.

as suggested by Feyerabend, however there are unspoken rules, often not very well defined, that makes scientific results to be accepted as modelled by Lakatos. Rarely, paradigm shifts happen that, however, must be corroborated by some positive heuristic to be successful, still like in the Lakatos model.

Physics is a science based on experimental data, i.e. the science of what can be measured. As long as something can be measured, the phenomena to which it gives rise can be investigated with the methods of physics. Measurements play a major role in our science: this is why it is so important that you learn what measuring means and how to perform and interpret measurements. Even if you are going to become a theorist.

Before addressing the problem of measurement, we must still spend few words about physics in general and its relationships with other sciences. From the above discussion it should be quite clear that physics is quite different from mathematics, considered itself a ~~science~~ ~~Mathematics~~ is not at all an experimental science. Mathematicians ~~choose freely a set of axioms, true by definition,~~ then derives consequences ~~strictly adhering to it (sometimes arbitrarily defined)~~ set of rules. As a result, many ~~mathematicians~~ ~~exist~~ at the same time: anyone knows that one can define ~~alternatively~~ ~~very different~~ called non-Euclidean geometries, for example. Physicists are not as free as mathematicians: they cannot establish the rules to which physics must adhere. They can only observe Universe and try to describe it in the proper way, i.e. in a way such that they are able to predict what will happen in the future or what happened in the past, irrespective of how far past and future are, depending on how good we are in describing the systems under investigation.

In ~~shorts~~ we ~~cannot tell~~ the Universe how to behave or, said in other words, ~~K~~ ~~the Universe does not obey~~ the laws of physics. It's the laws of physics that obey the behaviour ~~of the~~ Universe.

~~laws obey the way in~~
This is why we used the word "opinion", above. A well known quote in Italy is "mathematics is not an opinion", ascribed to Senator Filippo Mariotti (1833–1911), whose meaning is that a statement cannot be discussed or subject to interpretations. According to the above considerations, mathematics is indeed an opinion (in the sense that mathematicians are free to have "opinions" about what has to be considered as true). A more correct quote would be "Physics is not an opinion" since, whatever the beliefs of a physicist, he/she never can model phenomena in such a way the results of the model do not agree with observations.

Studying ~~Physics~~ ~~you will~~ learn that many of your common sense beliefs are wrong, ~~in the sense that they~~ cannot be used to build models able to predict (or retrodict) phenomena. There's nothing you can do about it. You must believe (working) ~~physics~~ ~~modeling~~ the truth, whether you like it or not. It is worth noting that ~~this often happens~~ in classical physics, too, even if the majority of may evolve with time.

There is no absolute
truth in this science.

people consider it a peculiarity of modern physics.

The evolution of cosmology

It was once believed that our planet defined the center of the Universe and that everything else (the Sun, the Moon, the stars and the planets) revolved around it. Even if it nowadays may appear odd, there are good reasons to believe that. Looking at celestial bodies we are induced to conclude that they move around us in a more or less circular path. After centuries of observations, Ptolemy (ca 100–178) recorded a set of rules in his work *Almagest* [**Ptolemy, ca.100**]. Those rules can be taken as a primitive form of physics laws.

Looking carefully at the night sky, one can realise that in fact not all the celestial bodies move following regular circular paths. Some of them appear to invert their motion and exhibit the so-called apparent retrograde motion. To explain such motions, Ptolemy and his followers introduced the epicycles: circular path around a point that, in turn, follow a circular path around Earth.

It took more than 1 000 years to realise, with Nicolaus Copernicus (1473–1543), that in fact Earth was not at rest in the centre of the Universe [**Copernicus, 1543**]. In his cosmology, Copernicus showed that all the planets, including Earth, rotated around the Sun, that was taken as fixed, still at the centre of the Universe. The idea must have seemed odd to his contemporaries, but today the fact that Earth and other planets orbit around the Sun is widely accepted.

Later, it was discovered that even the Sun moves around the centre of the galaxy and many galaxies move around in the Universe that surely does not have a center near us, if there is a center.

Till the work [**Newton, 1687**] of Sir Isaac Newton (1642–1726), no one realised that it took a force to hold together the Solar System and the initial theory [**Kepler, 1619**] formulated by Johannes Kepler (1571–1630), according to which planets moved as observed because they were playing a music for the pleasure of God, was wiped out.

Albert Einstein (1879–1955) finally found that Newtonian's gravitation is only a first order approximation of his general relativity theory [**Einstein, 1916**].

From this long story made short, one can see that from time to time a *theory* has been superseded by another *theory*. As a consequence, the newest theory has to be considered true, while the superseded ones must be taken as false. However, even Ptolemy's theory is not false in the sense that it correctly describes planet motions, at least if the precision with which their position is measured is not too high. Only the interpretation of his data can be considered wrong. It is worth noting that, even when the interpretation of a set of data is *wrong*, as in the case of Newton's gravitational theory as compared with general relativity, it can be taken as correct for many practical purposes. If we must compute the motion of freely falling bodies on Earth or even the trajectory of an interplanetary spacecraft, we do not use Einstein's general relativity: Newtonian's mechanics is precise enough to provide the correct results. Hence, even if superseded

by a new theory, an old one continue to remain valid and *true* within the domains in which they were formulated.

Another important difference between mathematics and physics is that the second is inevitably less rigorous than the other. Sometimes, physics concepts or quantities are not as well defined as mathematical concepts. Indeed, there is no need in physics to be as rigorous as in mathematics. Many physicists still discuss about what is the *true* or *correct* definition of heat. Indeed, there is no need to struggle to find it: as long as we can measure it and we can use heat to compute other physics quantities, heat is well enough defined. In order for a physics concept to be well defined, it is enough that we are able to describe, as precisely as possible, a procedure to measure it.

2. Measurements and the laws of physics

A measurement consists in a set of operations leading to a measure. Measures, *per se*, are not so important if they do not lead to a model that is usually expressed in mathematical form as an equation, often called a **physics law**. Physics laws are even if such a term can be misleading according to the above. equations expressing the relation observed on two or more physics quantities, i.e. things that can be measured.

Equations represents a physics law only if it expresses the relationship between at least two **physics quantities**, i.e. something that can be measured. There are no physics laws about love, for example, since there is no way to measure love. One can say that love does not exist for physicists, in the sense that physicists cannot make statements about love with the same “authority” with which they make similar statements about heat. A physicist’s statement about love is just an *opinion*.

An **equation** representing a physics law has little to do with an equation in mathematics, but the rules to manipulate it. Consider an equation like

matters. Two

mathematically equivalent expressions

$$a = \frac{F}{m}. \quad (1.1)$$

may not have the same meaning when interpreted as in mathematics, such an equation means that a number represented by a is always equal to the ratio of two numbers F and m and can be interpreted as physics written equivalently as laws.

$$F = ma \quad (1.2)$$

or

$$m = \frac{F}{a}. \quad (1.3)$$

Moreover, as long as $m \neq 0$, eq. (1.1) always makes sense and even in the limit $m \rightarrow 0$ the value of a is still a number and the equation holds.

For a physicist, equation (1.1), has a completely different meaning. It says that the physics quantity on the left (the acceleration a of a body) depends on those on the right side of the equation (the intensity of the force F acting on the body whose mass is m). Even if equation (1.2) can be found in many physics textbooks, it does not take the same meaning and, in some sense, is not as correct. Read as a physics law, eq. (1.2) states that a force depends on the product of a mass and an acceleration, that makes no sense. That equation, as well as eq. (1.3), holds just for the numbers representing the outcome of the measurement of the acceleration of a body of mass m subject to a force F .

There are other differences, too. All the equations hold if we substitute the numbers obtained during a measurement, but only approximately. A physics law like (1.1) does not guarantee that measuring the acceleration a of a body with $m = 2$ subject to a force $F = 3$, will be exactly $\frac{3}{2} = 1.5$. First of all, we must specify the units of a , F and m , otherwise the values do not make sense. Moreover, any measurement is affected by some uncertainty, hence the result of a measurement of a will probably lead to a value different from 1.5 (irrespective of the units), even if we expect to obtain a number close to it. How close depends on the uncertainty of the measurement, which we must learn how to estimate. In the end, the meaning of the symbol $=$ is somewhat different with respect to what it means in mathematics. The equal symbol states that we *believe* that those quantities must depend on each other in the given way, but only if we neglect many effects that, in practice, cannot be eliminated, such that the numerical results provided by the measurements only holds approximately and $=$ becomes, in fact, \simeq .

Another difference is that those equations hold only within the framework in which they have been experimentally established, i.e. *Newtonian mechanics*. For bodies moving at high speed (where high means close to the speed of light), equation (1.1) does not hold anymore.

3. The process of measurement

A physics quantity is anything that can be measured, i.e. something to which we can assign a value, usually (but not always) expressed as a number.

For example, the height of a person is a physics quantity since we can tell how tall is a person (e.g. 1.82 m). The color of his/her eyes can be regarded as a physics quantity too, since we can attach a characteristic to it, as brown or blue. It turns out that, in fact, all of the characteristics expressing the *value* of a physics quantity can be expressed as numbers. For example, one can define a table in which each color correspond to an integer. For this reason, we are

going to treat all the quantities of potential interest for a physicist as numbers.

Even if we can assign a value to any characteristic of anything, we are not going to define all of them as physics quantities. We define as such only those characteristics that are of potential interest in the framework of the topic we are investigating. Collecting the value of all the potentially interesting characteristics of a system under investigation is said *characterisation*.

Measuring a physics quantity means to perform a set of activities aiming at attaching a value to it, such that its value depends as less as possible on the procedure, on who performs the measurement and on any other condition that can alter the result. A good measurement should give the most objective result as possible. The activities leading to the measurement of one or more physics quantities are said to be an **experiment**. *whose purpose is always to obtain (at least) a number to be published for the community, always,*

In what follows we are going to formally develop a theory of measurement by doing them. Instead of giving definitions and to apply them in the process of measuring something, we will try to characterise a process or a system and find that things are not as simple as they might seem. We will fall into many pitfalls from which we will have to come out by revising our naive beliefs about the measurement process and we will come to define, through experience, the correct way to conduct it.

At present physicists perform measurements collecting data using some automatic instruments often driven by computers. Data analysis, moreover, is as well done electronically, either using spreadsheets or *ad hoc* designed computer programs. Together with learning how to perform measurements and interpret collected data, we should then learn how to perform measurements using modern data acquisition systems and how to collect and analyse them using computers. The ability of **coding** is as important as the ability of solving equations. Sometimes it is even more important.

The details of data acquisition can change a lot from system to system. In order to provide general enough information we are going to use simple and cheap tools like smartphones and Arduino boards (in fact, smartphones are not as cheap, however anyone buy them for other purposes and, regarded as instruments, they can be considered almost as for free). Of course, the considerations arising from the analysis of the proposed experiments, will be completely general and apply to any data acquisition system (abbreviated DAQ), being it automatic or manual:
able to use the best available technologies when needed.

Summary

Physics is an experimental science. Physics theories are grounded on experimental data. There are no physics theories that are not based on them. As such we can define physics as the science of things that can be measured.

Solid experimental observations have to be considered to be the *truth*, irrespective of the fact that they may appear to us as unreasonable. However, their interpretation is subject to change with time. Still, they remain true in the framework in which they were obtained.

A theory is a set of experimentally grounded mathematical rules allowing to make predictions about the results of a measurement. If the rules are not able to make predictions, their set is not a theory. Two theories making exactly the same predictions

are the same theory formulated in a different formalism.

Taking measurements is the job of experimental physicists. Physics laws are the mathematical formulation of observations. In particular they are expressed as equations relating two or more physics quantities, i.e. things that can be measured.

A physics law can be treated as a mathematical equation as long as it is used to compute and to make predictions. However, they are quite different from an equation. They state what we believe depends on what and, as a matter of fact, they must be considered as exact only in principle.

An experiment consists in taking measurements in conditions under control. The outcome of an experiment is always, at least, a number.

Bibliography

- [1] Popper, Karl R. *The Logic of Scientific Discovery*. London: Hutchinson, 1959.
- [2] Kuhn, Thomas S. *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press, 1962.
- [3] Feyerabend, Paul. *Against Method*. London: New Left Books (1975)
- [4] Lakatos, Imre. *Falsification and the Methodology of Scientific Research Programmes*. Cambridge University Press, 1968.
- [5] Claudius Ptolemy. *Almagest* (100–178).
- [6] Nicolaus Copernicus. *De revolutionibus orbium coelestium*, Nuremberg (1543).
- [7] Sir Isaac Newton. *Philosophiaæ Naturalis Principia Mathematica*.
- [8] Johannes Kepler. *Harmonices Mundi*, Linz (1619).
- [9] Albert Einstein. The foundation of the General Theory of Relativity. *Annalen Phys.* 49 (1916) 7, 769–822.

Chapter 2

Establishing a system of units

When we try to make a measurement, i.e., to attach a number to a physics quantity, we need a **unit**. A unit is a standard to which the quantity compare. We always use units without much formalities. For example, the height of a person is often given in centimetres or in feet, depending on the Country; the age is given in years (but for very young babies for which it is customary to give it in months); food is often sold in units of weight, etc.. In this chapter we analyse the problem of defining a unit properly and we learn the basics of the systems of units used in physics.

1. Measuring light intensity

Most textbooks start with the definition of simple and well known units such as length. In this book we make a different choice: we want to start with a more from naive measurement. Analysing units people are familiar with might bore and, more importantly, can induce to neglect some fundamental steps in understanding the processes that lead to their definition. This is why we chose to introduce the problem of measuring something with an unusual quantity yet light intensity.

It is well known that looking at some light standing on the opposite side of a (partially) transparent screen, makes that light to appear as less intense. This simple observation describes a physical phenomenon, the absorption of light in qualitative terms. It implicitly defines a physics quantity (the intensity of the light) and subtends that there is a relationship between the intensity of the light and the characteristics of the screen. The aim of the physicist is to precisely define what is intended for light intensity, i.e. how to measure it, establish on which of the characteristics of the screen it depends (thickness, shape, material, weight, etc.) and how. Eventually, the origin of the observed phenomena may be ascribed to other known phenomena, leading to a unified explanation of them, and theorists analyse them to make mathematical models.

The need for measurements comes from the need to express such a qualitative observation in a more formal and precise way, such that one can make predictions about what will happen with a different screen. It is not enough, for a physicist, to affirm that a thicker screen will lead to a more intense absorption of the light.

We must know how much light is absorbed by a screen with given characteristics. It is not just a matter of practical things (for example, we may be requested to design a filter that limit the intensity of the light shining on a plant to a level the plant can tolerate). Attaching numbers (and units) to quantities is needed to understand the processes in which that quantity is involved. This, ultimately, is the physicist's job.

2. Units definition

The simplest way to define light intensity is to find a standard sample (briefly a *standard*) to be used as a **unit**, whose comparison with the system under investigation allows us to assign a number to it. The standard must be manifestly chosen among those who exhibits the same characteristic and are then **homogeneous** with the system to be characterised.

Units must be stable. To define a unit for light intensity we need a source of light whose intensity is reproducible and easy as stable as possible. Sunlight cannot be a good choice, since it changes to us with time, season, weather, etc.. We might use artificial light like the one produced by a candle or a lamp powered by some electricity. The light of a candle fluctuates a lot, then it is a bit impractical. A good choice could be a smartphone's flashlight. We hardly see any difference in its intensity, at least if the smartphone is charged enough. There can be differences from smartphone to smartphone, however we can overcome this difficulty establishing a specific model as the reference one.

Direct measurements are made by comparing the quantity to be measured with a standard unit.

In order to perform a direct measurement of the light intensity of another light source we need a way to compare it with the unit. Suppose that the intensity of the light to be measured is lower than the one of the reference smartphone. In this case it is enough to choose a set of standard *absorbers*, equal to each other, to be used as screens for the light of the reference smartphone. One can observe the light to be measured and compare it with the light of the reference smartphone transmitted by n such screens.

The procedure is as follows: Screens can be cut out from a clear pocket for A4 sheets of paper. As long as the screens can be considered equal to each other, one can identify the measurement is part of the number n of screens needed to observe the light of the reference smartphone as identical to the one emitted by the source under investigation and say that physical quantity its light intensity is $I = n$.

However, we cannot require everyone to own the same smartphone as us. Different units can be while, e.g., Amelia claims that the intensity of the light is $I_A = n$, Ben may correctly report that the intensity of the same light is $I_B = m \neq n$. Manifestly the thing. the light intensity is the same, then so must be their measurements, i.e.

$$I_a = I_b. \quad (2.1)$$

~~If the above were mathematical equations, the only possible solution is $n = m$. However, these are not mere mathematical equations. They represent the relationships between physics quantities: the quantities are the same, but the way they express them are not.~~ A possible workaround is to write an equation like

$$n u_1 = m u_2 \quad (2.2)$$

that allows $n \neq m$ provided that there is a precise relationship between u_1 and u_2 . The equation written above make it clear that attaching just a number to a measurement is not enough. We need to specify a unit, i.e. the standard with which the measured quantity had been compared to. In the example above, Amelia used the u_1 unit, while Ben the u_2 unit. Those names assigned to units are impractical. A better choice is to use specific words invented for each of them. Often, the names of the units comes from the names of famous physicists of the past, as joule, newton, ohm, etc. Moreover, units can be indicated by using a symbol, for the sake of brevity (J , N , Ω). Adopting the same habit, let's call **floyd** the unit represented by u_1 (in honour of the Pink Floyd rock band), to which we assign the symbol **fl**, and **stone** (symbol **st**) the one represented by u_2 (from the Rolling Stones). Equation

$$n \text{ fl} = m \text{ st} \quad (2.3)$$

is perfectly valid in both physics and mathematics. Given a measurement x in floyds, one can transform it in stones using a conversion factor obtained treating units as algebraic symbols. Following the usual rules one can write

$$x \text{ fl} = x \frac{m}{n} \text{ st}. \quad (2.4)$$

The ratio $\frac{m}{n}$ is the wanted conversion factor which allows to transform a light intensity x expressed in fl in units of st. It is enough to multiply x by the conversion factor. For simplicity, we can construct tables of conversion factors where, for each unit u_1 , we provide the corresponding value in another unit u_2 when the value in u_1 is one, e.g.

$$1 \text{ m} = 100 \text{ cm}. \quad (2.5)$$

So, to transform a height of 1.82 m into centimetres, we can just multiply the number by 100, to obtain

$$1.82 \text{ m} = 182 \text{ cm}. \quad (2.6)$$

In fact one can work as if we were substituting the symbol m with $\frac{100}{1}$ cm. Each time you are in doubt, just write the equation stating the equivalence and treat units as algebraic symbols.

Exercise 1

A slim, elegant TV is on sale in an online shop for 449 USD. The data sheet says it has a screen size of 58" and its aspect ratio is 16 : 9. Before buying it, you must check if there is enough space on the wall of your room. Moreover, you want to compare the price of this TV with another on sale at a physical shop whose price is quoted in euro. Compute the width of the TV and its price in euros.

The size of a TV screen is given in **inches** (symbol "), a unit commonly used in countries formerly under the influence of the British Empire, and represents the length of its diagonal. One inch correspond to 2.54 cm, i.e.

$$1" = 2.54 \text{ cm}. \quad (2.7)$$

In this case, it is very easy to find the size of the TV in cm as $d = 58 \times 2.54 = 147.32$ cm. A bit less straightforward is to express the price in another currency. The exchange rate is usually given as EUR/USD ratio. Assuming that, at the time of the purchase, the rate is EUR/USD= 1.12, treating the units as symbols, we can invert both members to obtain

$$\frac{\text{USD}}{\text{EUR}} = \frac{1}{1.12} \quad (2.8)$$

such that

$$449 \text{ USD} = 449 \frac{1}{1.12} \text{ EUR} \simeq 400.89 \text{ EUR}. \quad (2.9)$$

Exercise 2

Cosmetics are often sold in bottles, whose content is expressed in **fluid ounces** (fl. oz.). A *cologne* bottle contains 1.7 fl.oz. and costs 156.25 \$. How much does the cologne costs per ml, in euros?

Units used in the Anglo-Saxon world are usually expressed as fractions. For fluids,

$$\frac{1}{8} \text{ fl.oz.} = 3.70 \text{ ml}. \quad (2.10)$$

We can get the conversion factor from this equivalence as

$$1 \text{ fl.oz.} = 8 \times 3.70 \text{ ml} = 29.6 \text{ ml}, \quad (2.11)$$

such that the bottle's content is $1.7 \times 29.6 \text{ ml} = 50.32 \text{ ml}$. It then costs

$$\frac{156.25}{50.32} \frac{\text{USD}}{\text{ml}} \simeq 3.11 \frac{\text{USD}}{\text{ml}} = 3.11 \frac{\text{USD}}{\text{ml}} \times 1.12 \frac{\text{EUR}}{\text{USD}} = 3.48 \frac{\text{EUR}}{\text{ml}} \quad (2.12)$$

For historical reasons, there is one notable exception to the above rule. Temperatures can be measured in units such that conversion factors are not as simple to obtain.

Temperatures are measured in degrees Celsius ($^{\circ}\text{C}$) in most countries, but in few Anglo-Saxons ones, where they are measured in degrees Fahrenheit ($^{\circ}\text{F}$). The following equation holds:

$$T(^{\circ}\text{F}) = \frac{9}{5}T(^{\circ}\text{C}) + 32, \quad (2.13)$$

such that $0^{\circ}\text{C} = 32^{\circ}\text{F}$ and $100^{\circ}\text{C} = 212^{\circ}\text{F}$ (correspondingly, $0^{\circ}\text{F} \simeq -18^{\circ}\text{C}$ and $100^{\circ}\text{F} \simeq 38^{\circ}\text{C}$). The reason for this bizarre conversion is historical (see box).

Temperature scales

The German physicist Gabriel Fahrenheit proposed a temperature scale in which the coldest temperature he could achieve in his laboratory was defined as the zero of the scale, while the temperature of a human body was taken to be 100 degrees. It turns out that, indeed, this temperature corresponds to about 38°C , so either Fahrenheit had a fever or he was wrong in taking his measurements. Someone argue that he used the blood of his horse to define such a temperature, that is a bit hotter with respect to the human one. According to Encyclopedia Britannica, the reason for the discrepancy has to be ascribed to a redefinition of the scale, after Fahrenheit's death, made for convenience (such that the conversion between Celsius and Fahrenheit was easier).

Anders Celsius proposal was in favour of a more objective scale [**Celsius, 1742**], in which $T = 0^{\circ}$ was taken as the temperature of the boiling water, while $T = 100^{\circ}$ was the temperature of . Note that this scale use the opposite convention with respect to the Fahrenheit one: the higher the temperature, the colder the system.

Jean-Pierre Cristin, then, proposed [**Cristin, 1743**] to invert the Celsius scale and called it the Centigrade scale. Nowadays Celsius and Centigrade are, in practice, synonyms.

Converting between units is made extremely simple using the Google search engine (<https://www.google.com/>). Just typing the requested conversion in the search input box will result in the proper value, e.g., typing 3.2 miles to km shows something like

Length

3.2	=	5.1499
Mile		Kilometre

Formula for an approximate result, multiply the length value by 1.609

Some computer's operating system includes such a feature, too. Note that if tools exist to having an easy-to-use tool to make conversion does not mean that you can ignore how to make them. This is a general rule: throughout this book, you need to make use of various automatic tools. However, we always give details about how these tools are implemented. Physicists need to know how their tools work without their help.

3. Systems of units

From the example above, it is clear that defining a unit is not as straightforward as it may appear at first sight. The definition of the units must be stable and independent on as many manufacturing details as possible, however it is always subject to improvements and, if needed, it can be redefined. Units must be agreed by anybody working in the field and their definition has been assigned to an international body called the *Bureau International des Poids et Mesures* (BIPM) based in Paris, following the first adoption of the **metric system** in 1799, after the French Revolution. The metric system was first introduced in order to define units based on natural dimensions. For example, the unit of length, the meter (symbol: m), was initially defined as one ten-millionth of the distance between the equator to the North Pole, measured along a great circle.

Since then, researchers tried to define fundamental units such that they are defined in terms of universal constants. The meter, for example, is now defined as "the length of the path travelled by light in vacuum during a time interval with duration of $1/299792\,458$ of a second", as stated on the BIPM website at bipm.org.

The **International System** of units, abbreviated in SI (from its french name *Système International*), comprises seven fundamental units, namely, the second (s) for time, the meter (m) for length, the kilogram (kg) for mass, the ampère (A) for electrical current, the kelvin (K) for the temperature, the mole (mol) for the quantity of matter and the candela (cd) for the luminous intensity. It is worth noting that the symbol for second is just s, not sec, as often found, and that the unit for temperature is the kelvin, not the "degree kelvin".

temperature, quantity
of matter and light
intensity.

multiple	symbol	prefix name	submultiple	symbol	prefix name
10^{15}	P	peta	10^{-1}	d	deci
10^{12}	T	tera	10^{-2}	c	centi
10^9	G	giga	10^{-3}	m	milli
10^6	M	mega	10^{-6}	μ	micro
10^3	k	kilo	10^{-9}	n	nano
10^2	h	hecto	10^{-12}	p	pico
10^1	da	deca	10^{-15}	f	femto

Table 2.1: Most common multiples and submultiples prefixes and names.

Fundamental units are units arbitrarily chosen as such, whose definition does not rely on other units. We stress that the choice of what unit is fundamental is completely arbitrary and, in fact, other systems of units exist in which units that are fundamental in the SI are not such in those systems. For example, in the **natural units system**, the speed of light c is taken as a fundamental unit and its value is chosen to be $c = 1$. In such a system, lengths are measured as the time needed for light to cover the corresponding distance and are given in units of time. The light year, for example, used in astronomy, is defined as the distance travelled by light in one year. Following this system of units $1 \text{ m} = 1/299\,792\,458 \text{ s} = 3.335640952 \times 10^{-9} \text{ m}$. Following the Einstein relation $E_0 = mc^2$, masses can be measured in units of energy in this system. Since $c = 1$, $E_0 = m$. Using natural units it is easy to remember the masses of electrons and protons that are, respectively, 0.5 MeV and 1 GeV (see Table 2.1 for prefixes). To obtain the masses in SI units it is enough to divide these numbers by c^2 and convert eV to J (the unit of energy in SI), knowing that $1 \text{ eV} = 1.6 \times 10^{-19} \text{ J}$.

The imperial units are popular in countries like USA, United Kingdom and countries formerly belonging to the British Empire. In this system lengths are measured in feet (ft), corresponding to 12 inches (in) and to 0.3048 m.

Duodecimal and other numeral systems

Few systems of units are based on the duodecimal system, i.e. a base-12 system, contrary to SI and other systems, based on base-10 numeral system.

The most common way to express numbers use positional notation, in which each digit in the number has a *weight* depending on its position. The base of a positional system is the number of digits defined in that system and each digit has a value corresponding to the value represented by the digit itself times an increasing power of the base from right to left. The decimal point in a number separates negative powers from positive powers. For example, the number 8237.43 in

Imp
pop
Ang
cour

a decimal system, i.e. a system in base-10 corresponds to

$$8 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 + 4 \times 10^{-1} + 3 \times 10^{-2}. \quad (2.14)$$

The same number can be expressed in other positional system. For example, in computer science hexadecimal (base-16) and binary (base-2) are popular. The latter because it is easy to build devices that, having two states, can represent any of the two digits (0 and 1) existing in this base and it is then possible to represent, e.g., numbers in memories made of capacitors (charged or empty), transistors (in conduction or in interdiction state), switches (open or close), etc.. Hexadecimal is used because it allows a short notation that can be easily turned into binary and viceversa, thanks to the fact that 16 is a power of 2.

In base-16, the number above reads 202D.6E. In fact, the digits of the hexadecimal system are the same used in the decimal one, to which we add the first six letters of the alphabet, such that A=10, B=11, etc.. The value of the above numeral (i.e. the symbol used to write a number) is

$$2 \times 16^3 + 0 \times 16^2 + 2 \times 16^1 + 13 \times 16^0 + 6 \times 16^{-1} + 14 \times 16^{-2} \simeq 8432.42$$

It is worth noting that a rational number in a positional system can be irrational in a different system and the number of digits after the decimal point of a number can be different in a different system. The same number in base-2 can be obtained in the same way. Converting it from base-16 is easy, as each digit in the hexadecimal system corresponds to four digits in base-2. The first digit, 2, in binary is written as 10 ($1 \times 2^1 + 0 \times 2^0$), equivalent to 0010 (the trailing zeros are called non-significant), D=13=1101, 6=0110 and E=14=1110, such that

$$202D.6E = 0010\ 0000\ 0010\ 1101.\ 0110\ 1110.$$

Historically, in the past people used numerals expressed in different notations. A rather common base is 12 on which many units of length are based. This system is called the duodecimal system. We still use it to measure time, indeed. The reason being that the number of divisors of 12 (2, 3, 4, 6) is higher than that of 10 (2 and 5) and it is simpler to express numbers as fractions of the units. This is the reason for which in USA, where a system derived from the imperial one is used, most road signs report distances as fractions of a mile (a unit of length corresponding to 1.609 km).



Measurements can be expressed in terms of the units defined in the chosen system or in one of its multiples and submultiples, represented by a prefix added to the unit symbol. Table 2.1 shows the most common multiples and submultiples, the corresponding symbols and how it is pronounced. For example, a centimeter, whose symbol is cm, corresponds to 10^{-2} m, while 1 MW (megawatt) corresponds to 10^6 W and 1 m in natural units is about 3.3 ns.

Measurements can also be expressed as a combination of units. For example, speed tells how fast you travel as it is defined as the length of the trajectory per unit time, i.e.

$$v = \frac{\Delta x}{\Delta t} \quad (2.15)$$

(the Δ is used when measurements are obtained as differences: $\Delta x = x_2 - x_1$ and $\Delta t = t_2 - t_1$). As a result, measuring lengths in m and times in s, the units for speed are m/s (meters per second). The unit of speed is said to be a derived unit.

Derived
obtained
product
base uni

Summary

When quoting the value of a physics quantity, its unit must be quoted, too.

A unit is a standard, adopted worldwide, to compare with. It must exhibit the same characteristic as the subject of the measurement, e.g., a unit length must have a length.

Measures in one unit can be converted in another unit by conversion. Conversion factors can be found treating units as algebraic symbols in equations.

Despite the existence of tools providing answers to more or less complex problems, it is of capital importance that you learn how to solve these problems without using them. Only when you master the solution you can start using them regularly.

A system of units comprises a set of units arbitrarily chosen as base or fundamental units.

The Bureau International des Poids et Mesures (BIPM – <https://www.bipm.org/>) is the organism responsible for the definition and maintenance of the International System of units (SI).

Fundamental units in SI are the second (s) for time, the meter (m) for length, the kilogram (kg) for mass, the ampère (A) for electrical current, the kelvin (K) for temperature, the mole (mol) for the quantity of matter and the candela (cd) for luminous intensity. Visit the website for definitions.

Units that are fundamental in a system may not be such in another system. For example, lengths are measured in units of time in the natural system of units in which the speed of light $c = 1$.

Units whose definition is based on combinations of fundamental units are said to be derived. For example, speed is measured in units of length per unit of time. In SI its units are m/s.

Bibliography

- [1] Celsius, Anders (1742) "Observationer om twänne beständiga grader på en thermometer" (Observations about two stable degrees on a thermometer), Proc. of the Royal Swedish Academy of Sciences, 3, pagg. 171–180.
- [2] Fournet, M.J., "Sur l'invention du thermomètre centigrade à mercure", Not. lue à la Société d'Agriculture de Lyon, 4 juillet 1845, pagg. 1–17.

Chapter 3

Taking data

It is now time to start making some serious measurement. We avoid boring you with trivial measurements nobody is interested in. Apart from very basic ones, most modern apparatus to take measurements are based on electromagnetic phenomena and usually display the results on digital screens. Advanced experiments profit from the ability of computers to interface to external devices to control data acquisition, filter data and provide intermediate results before storing raw data on files for further analysis. You are still not used to make measurements and then you need to face with simple enough problems, however, we mimic more complex experiments using instruments working similarly to professional ones.

1. Instruments

Instruments are the tools to make measurements. The simplest instrument is a tool exhibiting the same characteristic of the subject of the measurement with which it is directly compared. A ruler is an example of such an instrument. It has a length and the length of anything else can be obtained comparing directly its length with that of the ruler. In fact, rulers ~~last graduated instruments~~, i.e. instruments on which a scale is impressed such that we ~~can~~ directly compare the length of something with a fraction of its full length.

To make measurements we need instruments.

Sometimes, graduated instruments are equipped with a Vernier scale: a device that facilitates the interpolation of fractions of the interval between two adjacent marks on the instrument scale. Technology made the usage of Vernier scales almost obsolete and this device can only be found on rather old tools (still, they are widely used).

Instruments often respond to certain stimuli by producing signals whose measurement depends on the intensity of the stimulus. In that case we measure a different quantity with respect to the one we are interested in, that, however, is proportional to the latter. The graduated instrument in Fig. 3.1 can be used to measure the mass of some flour. Since the mass is proportional to its volume, a measurement of volume V is turned into a measurement of mass m just by



Figure 3.1: A *pint* is an instrument to directly measure volumes containing 0.473 l of a liquid. The jug is a graduated instrument, when reading the red scale, since it reports the fractions of its volume. It works as a calibrated instrument if we read the blue scale.

multiplying the volume by its density, defined as

$$\rho = \frac{m}{V}. \quad (3.1)$$

When the instrument measures the quantity a to provide the value of quantity b as a function $b = f(\vec{p}, a)$ of a , the instrument is said to be calibrated. The process of finding the parameters \vec{p} needed to transform b to a is called calibration.

For example, temperatures are often measured with a device called platinum resistance thermometers (PRT). As the name suggests, they are made of platinum, whose electrical resistance depends on the temperature. Assuming we have an instrument able to provide the measurement of the resistance in the appropriate unit (ohm, symbol Ω), the following relation between temperature and resistance holds for the very common PT100 sensor:

$$R(T) \simeq R(0) (1 + AT + BT^2), \quad (3.2)$$

where $R(T)$ is the resistance at temperature T , given in degrees Celsius, and A and B are constants, whose values, for resistance given in ohm, are $A = 3.9083 \times 10^{-3}$ and $B = -5.775 \times 10^{-7}$. Measuring $R(T)$ one can obtain T as

$$T = \frac{1}{2B^2} \left(\sqrt{\frac{A^2 R(0) + 4B^2 (R(T) - R(0))}{R(0)}} - A \right). \quad (3.3)$$

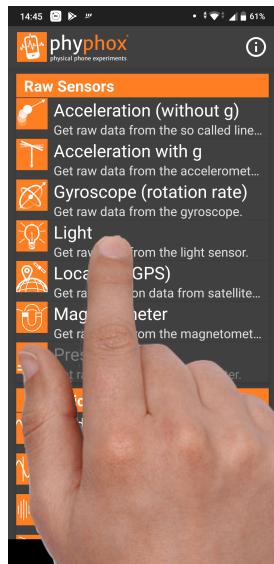
Here, the right member of the equation is $f(\vec{p}, T)$, while $\vec{p} = (A, B, R(0))$.

In the following we use **smartphones** and **Arduino** boards as instruments. Smartphones and Smartphones are capable to make many types of measurement, thanks to the Arduino boards are various sensors often on board. Arduino is a programmable microprocessor to very effective which we can attach various sensors. instruments to make physics experiments.

2. Smartphones

All smartphones are equipped with a microphone to place phone calls, a camera to take pictures and an accelerometer to get its orientation. Many of them include a light sensor to determine the environmental illumination (to adjust the display luminosity) and a magnetometer (to work as a compass in navigation systems). In the latter case they also have a GPS sensor to get its position on Earth. Few smartphones are equipped with a gyroscope, used to play games and for augmented and virtual reality applications. Some have also a pressure sensor.

Most of these sensors can be exploited to make physics experiments.



For them we need data coming directly from the sensors and to this aim we can use an App called PHYPHOX (acronym for *Physics Phone Xperiment*), developed by a team of physicists from the RWTH Aachen University, lead by Sebastian Staacks. The App is available for free and can be downloaded and installed for the two major operating systems: Android and iOS. Once installed, PHYPHOX looks for sensors in your phone and configure it such that you can start taking measurements with them. Missing sensors are shown as shaded in the list of available sensors (see picture on the left).

Choosing among available sensors starts a dedicated application that collects data from it upon request. Once data are collected, they can be exported in the form of a table via various method, including e-mailing it to some address or saving it directly on the cloud, like in Google Drive or Dropbox.

There exist other Apps dedicated to the same task. As we write the only other App we feel like recommending is PHYSICS TOOLBOX. In the following, all the experiments are proposed using PHYPHOX, however they can be easily done using the latter, too.



Figure 3.2: The Arduino UNO board. The processor is the big black box on its surface. I/O ports are aligned on the top and bottom rows of connectors. On the left side, the USB connector and the external power connector can be seen.

3. Arduino

The term “Arduino” is often used to identify an electronic board with a programmable microcontroller on board and a series of input and output interfaces of various kinds (digital, analogue and serial) allowing communication with external devices. In fact, it is much more than that. It is what computer scientists call an ecosystem of which the board is just a component. The ecosystem is made up of numerous external devices such as sensors (devices sensitive to some physical or chemical phenomenon) and **actuators** (devices intended to perform actions such as activating an engine, turning on an LED, producing sound with a speaker or displaying something on an LCD screen). A software platform (IDE: *Integrated Development Environment*) needed for its programming is part of the ecosystem. It can be downloaded for free from the Arduino very simple website and exists for the most popular operating systems. Its installation is C-lang simple and guided. There is also a completely online version of the development environment that does not require installation, but requires an Internet connection.

The whole community of users who share their experiences with others is part of the ecosystem, which has made Arduino a success on a planetary scale.

Its Italian origin is revealed by the fact that most Arduino-based projects have an Italian-sounding name.

All the examples in the Arduino UNO (Fig. 3.2) is the most common and looks like a module with an approximate rectangular shape of about $69 \times 53 \text{ mm}^2$. Pins, i.e. connectors that allow to interface the microprocessor with the outside world, are aligned in two rows on the long sides of the board. A USB connector allows communications with a computer. If not powered through the USB connection, Arduino can be connected to any voltage generator between 9 and 12 V.

used.

Its form factor had been carefully designed such that it is small enough to fit it in many environments, while, at the same time, not too small as to make it difficult to manipulate.

The genesis of Arduino

The name Arduino comes from the name of the bar where Massimo Banzi and David Cuartielles, the two main inventors of the project, were used to sit when they were working at the Interaction Design Institute at Ivrea.

In turn, Arduino was the name of a nobleman of Ivrea, who later became King of Italy in 1002. At that time Italy was a constituent of the Holy Roman Empire.

Ivrea, a UNESCO World Heritage Site, is renowned for being one of the most innovative Italian districts, distinguished by attention to both performance and design. Here Adriano Olivetti developed the first personal computer, called Programma 101.

A plethora of different boards are sold, with various characteristics in terms of computing power, memory, speed, etc.. Since they all share the same programming language and the same architecture, in the following we refer all the projects to the most common and less expensive board: Arduino UNO. Every flavour, however, is equivalent to it for the purposes of this book.

4. Open source make it better

Both Arduino and PHYPHOX are open source projects. In the case of Arduino, the most common even the hardware is released under an open source license. Check out the open source licenses documentation section of each product on the Arduino website to obtain there the General Public License or GPL (<https://www.gnu.org/>) and the Creative Commons or CC (<https://creativecommons.org/>).

The open source movement originated in the software realm. The source code of open source software, also called *free software*, can be inspected, modified and redistributed. This practice has been extended to hardware, where the construction plans of a device can be, as well, inspected, modified and redistributed. A frequent misconception about open source is that products released under this license are free, in the sense that users must not pay to use them (whatever "to use" means). The best definition of what it means to be free software is given on the GNU project website:

"Free software" means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, "free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech", not as in "free beer".

Open source is a great opportunity for learning. Copying source code is encouraged through this course, provided you understand what you are doing and you perform measurements learn from it. Using the techniques developed in this book you are going to use Arduino and learn much more than just how to make good physics experiments. Our goal is to develop your skills in computing and programming, critical thinking, design and communication.

Learning about the Arduino programming language you are going to learn some C and C++ languages. We use Python for offline data analysis, as its popularity is already high and still increasing. However, we do not enter into too many details. The idea here is that you must learn about "programming", in whatever language. You are not supposed to just learn a programming language. Being a good programmer is important. The language in which you are proficient is much less important. Speaking is not just a matter of pronouncing words. In order to speak you must be able to think and translate concepts into the right words in the right order, such that the meaning of your thoughts is reflected in the commonly agreed meaning of the pronounced words. As in spoken languages, once you know how to speak, it is not so difficult to learn a different language.

5. Measuring light intensity with a smartphone

In order to perform our first measurement, we start PHYPHOX and click on the LIGHT icon. The display shows the tab GRAPH of the "Light" experiment.

ch experiment has icon to start data acquisition (a triangle-shaped arrow) and one to discard data (a bin).

SI, light intensity is measured in “candela” (cd). The flux of light emitted by a source in a solid angle is measured in “lumen” (lm). The flux per unit area is called illuminance and is measured in “lux” (lx).

The top, orange, row is common to all experiments. It includes a triangle-shaped arrow, a bin-like icon and three dots aligned in a vertical line. The arrow is used to start data acquisition. Clicking on the bin discards all the data collected so far. A menu with different options is shown when touching the dots.

The display is divided in tabs. The default one “GRAPH” is shown underlined. It is intended to show a graph of the data as a function of time. The graph is displayed in the rectangle below, whose title is ILLUMINANCE, as a function of time. The “SIMPLE” tab shows the same data in numerical format.

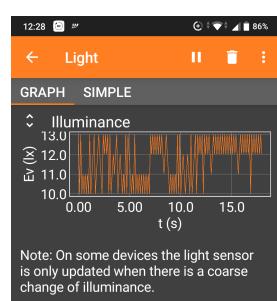
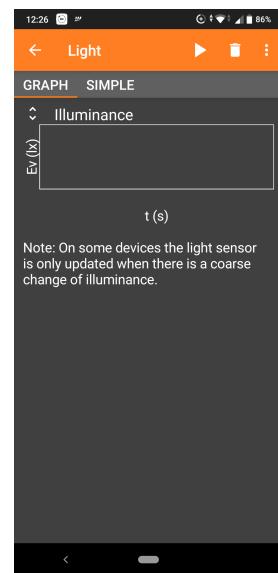
In this experiment, times are measured in seconds, while **illuminance** is given in lux (symbol lx). Illuminance is defined as the luminous flux per unit area. In turn, the luminous flux is defined as the amount of light per solid angle radiated from a source. The luminous flux is measured in lumen (lm) and 1 lm correspond to the light emitted by a source whose intensity is 1 cd (one of the base units of the SI) under a solid angle of 1 steradian. What is important to understand, here, is that illuminance is a measure of the light effectively impinging on the sensor: the larger the sensor, the higher the illuminance for the same source. As a result, we cannot compare illuminance measured with a detector with illuminance measured with another, unless we know the area of both.

Touching the triangle-shaped arrow on the top right corner, data acquisition begins and data is shown in the graph as a function of time.

Clicking on SIMPLE shows the current reading as a number. If the light sensor is updated frequently enough, you can see the number changing (and, correspondingly, the same behaviour can be seen on the graph of the other view).

Data acquisition can be paused touching the “pause” symbol . Data collected so far are still in the memory of the smartphone and can be saved as a two-columns table containing time and illuminance.

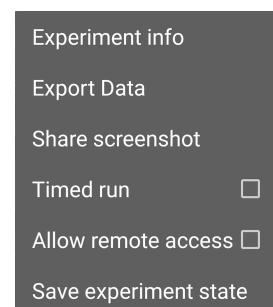
Data can be saved in a variety of formats. Click on the vertically-aligned dots, such that a menu appears and click on EXPORT the DATA”. The menu contains few other lines. “EXPERIMENT INFO” gives access to a short text explaining how the experiment works. “SHARE” allows to share the three self-explanatory, while “TIMED RUN” allows the user to program the start and duration of the data acquisition phase. Activating the “ALLOW REMOTE



Start the experiment touching the triangle-shaped arrow and pause it clicking on . Then export data to your preferred service.

ACCESS” the display can be replicated on a web page, pointing your favourite web browser to the address specified at the bottom of the display when the experiment begins. Note that the configuration of your network may make this option ineffective. The current configuration of the experiment can be saved for later use by “SAVE EXPERIMENT STATE”.

Data can be exported in a variety of formats, as tables. Upon clicking “EXPORT DATA”, a choice by the user is required about the format of the output. Data can be exported as an Excel file or as a CSV (comma separated value) text file. In the latter case the separator between fields can be a comma, a tabulator or a semi-colon. The user can also choose if he/she wants the separator between integer and fractional part of numbers to be a point or a comma. Once the choice is made, the user has the opportunity to choose where data must be exported. Common options are sending them to an e-mail address or save them on the cloud, e.g. on a Google Drive or on Dropbox. The first line always contains a legend describing the data.



Time (s)	Illuminance (lx)
0	12
0.10876	13
0.218672	12
0.329632	15
0.438514	14
0.548012	15
0.659466	14
0.768024	15
0.878746	14

Usually the sampling rate of a smartphone is relatively high. A quick look to the data gives a rough idea of the sampling rate of the smartphone that, in the example, was about 10 Hz (the measurements follow each other at intervals of about 0.1 s from each other).

Though Excel is very convenient in many cases, we are going to use the CSV format with the comma as a separator and the decimal point as the separator between the integer and the fractional part of numbers. This way we can play with Python to analyse them and learn about the latter.

6. Measuring light intensity with Arduino

Analog sensors have three leads: two analog and one digital. Analog sensors are devices producing a voltage proportional to the quantity needed to connect which they are sensitive. They must be powered connecting it to a power source power supply, while the third lead is the output providing a voltage usually between 3.3 V and 5 V. As a response, they provide an electrical signal within 0 and V_0 , where V_0 is the voltage at which they are converted into supplied.

voltage between 0 V and V_0 , where V_0 is the voltage at which the sensor is powered. All analog sensors have three leads: two for power supply and one for the

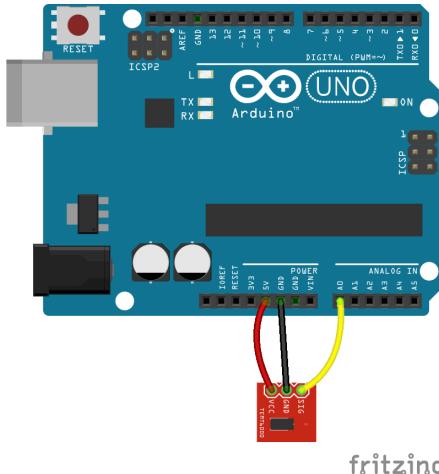


Figure 3.3: Connecting an analog sensor requires three wires: two for the power supply and one for the output signal.

outcome. The power supply connectors are usually marked as GND (short for *ground*, a lead whose voltage is taken to be 0 V) and VCC (Voltage Continuous Current).

Among the Arduino pins, two serve for providing power to external sensors. They are marked 3.3V and 5V, respectively.

Sensors are connected to Arduino by means of dedicated cables, called jumpers. Jumpers ends with Dupont connectors that can be either be male or female.

A set of six Arduino pins, marked from A0 to A5, are internally connected to an ADC: an Analog to Digital Converter. This device digitises voltages up to 5 V and convert them into integer numbers that can be read by the processor onboard. The output of an analog converter, usually marked as SIG or OUT, must be connected to one of these ports. For this project any analog light sensor is fine.

Fig. 3.3 shows a schematic of the connections needed. Schemas like this can be easily made using FRITZING, an open source software tool available under <https://fritzing.org>. The project is open source, so you are free to download the source code and *compile* it yourself, i.e. generate the executable application. You can also download the executable application from the website, ready to be used. In this case you are asked to pay a small fee.

When connecting a sensor to Arduino, the color of the wires is obviously irrelevant. However, following conventions helps to maintain projects, document and debug them. Take the habit of always using red and black wires for power, reserving the black color for the GND pin. The signal output can be connected to Arduino with a wire of a different color.

Once the system is ready for taking measurements, connect Arduino to a com-

The GND and VCC leads of the sensor must be connected using jumpers to, respectively, the pins marked GND and 5V or 3.3V on Arduino, depending on the sensor.

Schematics can be done using Fritzing, an open source tool.

Always use red jumpers for VCC and black ones for GND. This way, debugging your project will be easier.

puter via a USB cable and open its programming tool (IDE). Write the following program (called “sketch” in Arduino jargon) in the editor window.

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println(analogRead(A0));  
}
```

After ~~saving the program~~, ~~compile~~ it by clicking on the icon with a checkmark on the top left side of the window. This translates the program, written in the ~~Arduino programming language~~, into machine code, consisting of binary coded, low level instructions for the processor.

~~deploying it to the Arduino memory. The transfer happens via the USB cable.~~ Machine code must be transferred to the Arduino memory. This is done via the USB cable upon clicking on the button with a small arrow, close to the one to compile the program. LED's on Arduino start blinking and, as soon as the code has been loaded into the Arduino memory, it starts to execute it.

The program ~~configures~~ the communication channel via the `Serial.begin(9600)` statement, then repeatedly execute `analogRead(A0)`, needed to obtain the voltage on the A0 port, and transfer its value to the computer by `Serial.println()`.

~~Arduino and the computer. Data can be inspected using the serial monitor.~~ A tool called “serial monitor” can be activated clicking on the button with a magnifying lens on the top right corner of the IDE or via its menu, to quickly inspect the output, consisting in an infinite series of integer numbers scrolling in the window.

In order to collect data onto a file on the computer disk we can use the following Python code.

```
import serial  
import time  
  
usb = serial.Serial('/dev/cu.usbmodem14101')  
f = open('illuminance.csv', 'w')  
while True:  
    arduinoReading = usb.readline().rstrip()  
    print(arduinoReading.decode())  
    f.write('{}\n'.format(time.time(),  
                          arduinoReading.decode()))
```

When running ~~the Python~~ code, the serial monitor must be off. In short, this program ~~opens the communication channel with Arduino using `serial.Serial`~~

~~Python applications does not need to be compiled before execution. Indeed, each statement is translated in machine language at run time.~~

(), then write the data on a file called `illuminance.csv`. Data are read using `usb.readline().rstrip()`.

The USB port is identified by the string `/dev/cu.usbmodem14101`, typical on macOS X operating system. On Linux, USB ports are usually called `/dev/ttyUSB0`, `/dev/ttyUSB1`, or so, while on Microsoft Windows they are called `COM1`, `COM2`, etc..

After collecting data for 20–30 seconds, one can stop the program pressing together the `Ctrl` and `C` keys.

7. Understanding Arduino programming

Any Arduino sketch must contain at least two “functions”: blocks of code enclosed in braces. Their names are `setup()` and `loop()`. The user is responsible to define them. This is done writing their “type” in front of their name, followed by the list of actions to be taken by the function enclosed in braces.

There are no strict rules about how to write statements in the Arduino language, as in C or C++. Nevertheless it is a good habit to be polite when writing code to make its maintenance easier. Strictly following conventions helps in making you recognised as a good programmer. Usually, no more than one statement is present on each line of code with very few exceptions. A newline is started after opening a brace, while closing ones stay alone in a line. Use consistent indentation rules: add few blanks on the left each time a brace opens and align the closing one to the beginning of the line in which it opens. Many text editor automatically indent lines for you, but the programmer can always override the editor behaviour. If lines are very long, it is customary to break them into more lines.

When a new program is loaded into the Arduino memory, the execution of the program starts with executing once the statements listed in `setup()`, then `loop()` is executed repeatedly: it restarts upon exiting. The sketch begins again when the `RESET` button on Arduino is pressed or when it is powered up.

`Serial.begin(9600)` in `setup()` is needed to configure the communication channel between Arduino and the computer to which it is connected via the USB cable. The number 9600 in parenthesis represents the communication speed measured in “baud”: a unit for transmission speed. Data travels over the USB cable as a series of bits 0 and 1. Each bit is represented by a voltage level and is transmitted over the line in series, i.e. one bit after the other. The speed of serial communication is given by the maximum number of voltage changes over the line per unit time. 9600 baud is in practice equivalent to 9600 bits per second and is a relatively low speed. Modern computers can afford speeds up to more than 115200 baud, that is the current maximum speed affordable by

Arduino.

The Arduino collects data from the ADC's with `analogRead(A0)`, where A0 is the `Analog port identifier` to be read. Executing `analogRead()` returns an integer number between 0 and 1023. Such a number n is proportional to the voltage V_{pin} present on the given pin (A0), i.e.

integer between 0 and

$$2^{10} - 1 = 1023.$$

Digitised data can be

$$n = \alpha V_{pin}, \quad (3.4)$$

where $\alpha = \frac{1023}{5V}$. You can imagine some statements as a sort of mathematical function `analogRead()`; in this case `analogRead(A0)` represents $f(x)$, and the data returned is y . Putting this statement inside the parenthesis of `Serial.println()` make its returned value to be transferred over the serial line to the computer; a newline character is added at the end of transmission.

When a computer is connected to the other end of the USB cable, it can extract characters from the line after the other. They can be used to be shown on the display, as the `Serial Monitor` does. The newline character makes the computer to advance by line, such that next data appears on the line below the previous one. Using line. Data can

be actually printed
`Serial.print(analogRead(A0));`
using a computer.

without the `ln` after `print`, data are shown on the same line and it is impossible to separate the different readings.

8. Python data collection

The `import` keyword requires Python to load the corresponding module: a collection of software intended to simplify certain operations (also known as library). The `serial` module is needed to communicate with serial ports (the USB port is a serial port; in fact USB stands for Universal Serial Bus). The `time` module helps in manipulating times.

Contrary to C or C++, Python requires statements to be written one per line, even if a single statement can be broken on more than one line to improve readability. Indenting is not an option: it is mandatory under certain conditions, as seen below for the `while` statement.

To establish communication with the port the statement `serial.Serial` (always used). The statement requires an argument representing the port identifier. The identifier differs from computer to computer and from port to port, however it is usually in the form `/dev/cu.usbmodemxxx` on macOS X, `/dev/ttUSBxxx` on Linux and `COMxxx` on Microsoft Windows. Here `xxx` is a number. Strings are written in single ' or double " quotes. `serial.Serial()` returns an object to which we give the name `usb` via the `=` operator.

The `File` object of `file` represents a file. It is returned by the `open()` function, that requires two string arguments: the name of the file to be opened and its mode of operation. A file can be opened in write ('w'), read ('r') or append ('a') mode. When a file is opened in write mode, if it does not exist it is created, while if it exists the next writing operation will overwrite its content. In read mode data can only be read from the file, while in append mode, if the file exists, new data will be appended to the end of the existing ones.

`while` is used to build an iterating structure. With it we can repeat the same set of instructions as long as a condition, placed after `while` is true. In the example, the condition is, by definition, always true (True). The colon punctuation sign marks the start of the instructions to be repeated, that must be written indented, i.e. with leading whitespaces corresponding to a tabulator sign (tab). All the following indented lines contains statements that will be repeated forever.

The first operation done in the while loop is extracting data from the serial line, interpret these data as characters building a string and assigning it to a variable whose name is `arduinoReading`.

Python is an “Object Oriented Programming” (OOP) language. In OOP, “objects” belonging to a “class” have the ability to perform actions by means of “methods”. Objects of the same class share the same methods, then exhibit the same behaviour. In our application `usb` is an object of the class `serial`. The method `readline()` of this class returns a sequence of bytes, i.e. a series of 8 bits, extracted from a serial line. You can imagine the USB port as a stockroom where data enters from outside one by one keeping their order. They can then be extracted from the stockroom by the processor that works as what computer scientists call a FIFO (first input, first output). With `usb.readline()` we extract all the characters up to the EOL (end-of-line) character, currently in the FIFO represented by the `usb` object, in the order in which they entered it. This operation returns an ordered set of bytes called an array. An array can be regarded as a list whose elements belongs to the same type. In the following we often refer to homogeneous lists as arrays, while the term list is used for more general collections. The last byte, interpreted as an 8-bit integer, corresponds to 10 (0xA in hexadecimal) and corresponds to a newline character. If printed, this character makes the current line to scroll up, such that next characters will be printed on the next line. `rstrip()` is a method of the `String` class. It returns the object to which it is applied, but the last character. `arduinoReading`, this way, is a `String` object containing all the characters read from the USB port, but the last newline one.

Iterations are realised using the `while` statement. It repeats the following indented lines as long as the associated logical expression is true.

In OOP a program is made of objects belonging to classes, interacting via their methods. Methods allow accessing and altering the state of an object. The state of any object of the `serial` class, for example, comprises data in the channel, that can be retrieved by the `readline()` method.

ta of any type can
be printed using
`int()`. The same
data can be
interpreted in a variety
ways. For example,
array of bytes can
be interpreted as a
character string.

With `print()` we print the string to the display associated to the running application. The method `decode()` forces the system to interpret the list of bytes as a character string.

Finally, we write both the current time and the content of `arduinoReading`

to the file, previously opened, represented by `f`. The `write()` method of the file object takes, as argument, the string to be written. Here, the string is represented by '`{}`, `{}`'. Applying the method `format()` to the string object, each pair of braces is substituted by the actual values of the arguments of `format()`. In the example, `time.time()` returns the current time expressed as the number of seconds elapsed since the "epoch". The latter corresponds, by convention, to January 1, 1970 at 00:00:00 UTC (Universal Time Coordinated). The second argument of the `format()` method is the string content, just read from the USB channel. As a result, time is substituted to the first pair of braces, while the number provided by Arduino as a result of the measurement of the light intensity is substituted to the second pair. In the end, a pair of numbers, separated by a comma, is written on the output file, as follows

```
1581615606.776129, 36  
1581615606.7800908, 37  
1581615606.784203, 37  
1581615606.788279, 36  
1581615606.7923782, 36  
1581615606.79651, 36  
1581615606.800616, 37
```

The first number in the first line corresponds to the time of the measurement. It happened about 1581615607 of seconds after the epoch. Using any epoch converter online you can easily see that such a measurement has been done on February, 16, 2020, at 10:35:13 UTC (or GMT, standing for Greenwich Mean Time, that coincides with UTC). The number after the comma represents the measured intensity of the light in arbitrary units, i.e. as an integer proportional to the actual value.

Since there are no other lines in the program, those after the `while` statement are repeated indefinitely, until the program is forced to stop.

Summary

Graduated instruments have a scale on which marks are engraved, allowing a direct comparison of a quantity with the instrument itself. If the quantity to be measured is not homogeneous with the instrument characteristics giving the response, the instrument is said to be calibrated.

Automatic data acquisition allows us to take several measurement in short times. Smartphones and Arduino boards are useful tools.

Both PHYPHOX and Arduino are open source projects.

phypox

Smartphones' sensors can be exploited using PHYPHOX: a free App available for both Android and iOS operating systems that gives access to all the sensors found in a smartphone. Any smartphone have at least a microphone, a camera and an accelerometer. Many have a GPS sensor, a gyroscope, a magnetometer and/or a pressure sensor.

Data from PHYPHOX can be exported in various formats, including Excel tables.

Light intensity can be directly measured with a smartphone, thanks to the light sensor. The values are given in lux, a measurement of the light intensity per unit area. PHYPHOX performs as many measurements as possible and report them as a function of time.

Arduino

Arduino is a versatile programmable board with

a microprocessor connected to several I/O ports. It can be connected to sensors and actuators and communicates with a computer via an USB cable.

Light sensors exist for Arduino as analog sensors. Once powered, analog sensors provide a voltage on the signal lead proportional to the quantity to which they are sensitive. The voltage can be digitised thanks to Arduino's analog pins and transferred to a computer to store its value on a file for further analysis. In this case, light intensity is given in arbitrary units.

When writing code, always adhere to a formatting convention consistently and never deviate from it, even if the programming language allows it.

An Arduino program consists in executing once the statements in the `setup()` function, then repeating indefinitely those listed in `loop()`. A program is started each time Arduino is switched on, is loaded with a new program, is reset or when tools like the serial monitor and the serial plotter are started.

The `analogRead()` statement returns an integer number n between 0 and 1 023 proportional to the voltage V present on the pin given as an argument, such that $V = n \frac{5}{1023}$.

Data can be sent over a serial line using `Serial.println()`, after it has been properly configured with `Serial.begin()`.

Python

Data extracted from the USB channel can be inspected using the serial monitor and collected using a Python script. The latter open the communication channel, then starts reading characters in a FIFO (First Input First Output) structure prior to store them on a file previously opened.

Python is an Object Oriented Programming language. A program is made of objects belonging to classes, interacting between them. Methods are used to access or alter objects' state. The latter is a collection of data characterising the ob-

ject. Methods are similar to functions in procedural programming and applies to objects whose name is specified before a dot, like in `serial.readline()`. Methods can be concatenated like `serial.readline().rstrip()`. In this case the `rstrip()` method is applied to the object returned by the `readline()` method applied to the `serial` object.

On computers, time is measured in units of seconds elapsed since the epoch, corresponding to the conventional date January, 1, 1970 at 00:00:00 UTC.

Chapter 4

Uncertainties

In this chapter we make a first analysis of data collected using both PHYPHOX and Arduino. We see that, despite the subject of the experiment is the same, so we expect the same values for all the physics quantities we can collect, the numbers we get from the instruments are different and are differently distributed. We learn that each experiment results in numbers with an associated uncertainty, that depends on the experiment setup.

1. Data analysis

The first characteristic we notice is that collected data are not constant, as collected data are we may naively expect. They may not be constant for a variety of reasons observed to fluctuate. Indeed, the light we are measuring can be intrinsically variable. However, during the process of measuring it involves a set of processes, each of which is subject to measurement. to be influenced by many unwanted effects, and that may lead to fluctuations. Fluctuations can be Even if we believe that the light is in fact constant, it must travel to the sensor, subject to tiny and in this travel it may partly scatter with dust. Even microscopic variations and random effects of pressure, temperature and speed of the air in front of the sensor may change influencing the result the way in which it reacts to light. Also, the voltage used to feed the sensor for the measurement. can be not so stable, causing fluctuations in its response.

We can try to reduce the size of the effects of all these sources of fluctuations. Despite all our efforts however, it turns out that it is impossible to remove them completely. we can never get rid of fluctuations. If they

In any case, the reading of any instrument will always be a number with a limited number of digits, so, in the end, our reading precision will be limited. At least, then, measurements are affected by a reading uncertainty corresponding to the smallest division of the instrument scale. Even if you are tempted to do that, you are not usually authorised to interpolate between two adjacent marks on a scale, since marks are imprinted by the manufacturer according to the instrument resolution, corresponding to the smallest unit the instrument can appreciate.

Even in principle, a measurement cannot be taken with infinite precision. In classical mechanics, physics quantities are considered as continuous and real.

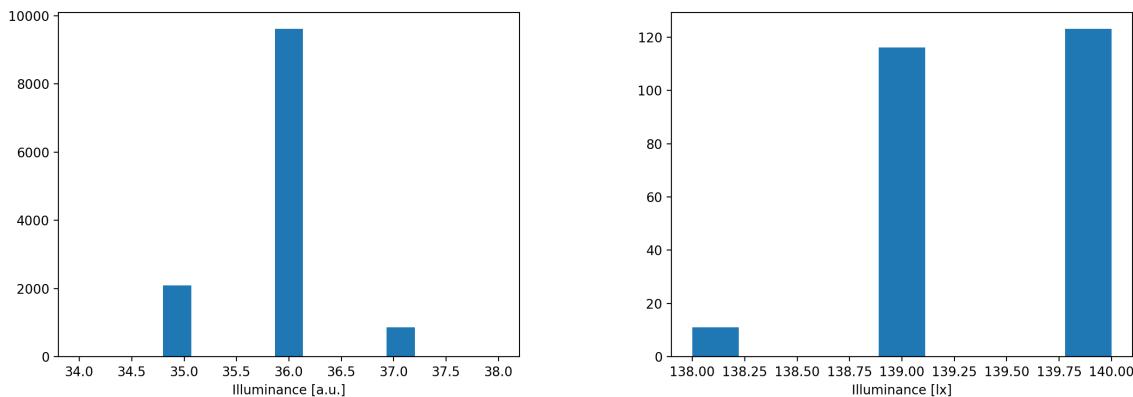


Figure 4.1: Histograms of data collected using Arduino (left) and PHYPHOX (right).

In fact, this is a prejudice. We believe that length, in fact, is a continuous function and that, in principle, it is always possible to divide something with a given length in two parts indefinitely, such that it makes sense to write

$$\ell = \lim_{n \rightarrow \infty} \frac{L}{2^n}, \quad (4.1)$$

where L is a distance. Though we assume that, we must know that such an assumption is valid only approximately, in the sense that the smallest length we could imagine to measure is so small that, in practice, it can be considered as null, but in fact it is not. However, that can only be true in classical mechanics. Quantum mechanics taught us that lengths cannot be as small as we want. The microscopic world (and not only it) is discrete. According to quantum mechanics, lengths can only be measured with an **uncertainty**

$$\Delta x \geq \frac{\hbar}{2\Delta p}, \quad (4.2)$$

where $\hbar = \frac{h}{2\pi}$ is a constant. $h \simeq 6.63 \times 10^{-34} \text{ m}^2 \text{kg s}^{-1}$ is called the Planck's constant, after Max Planck (1858–1947), and Δp is the uncertainty with which we can measure momentum.

In this book, uncertainty, then, is not just accidental and due to our limited technology. It is assumed that quantities are continuous quantities as in classical mechanics: continuous functions of real variables. In what follows we make such an assumption, for simplicity. However, we should never forget that every result is only a low energy approximation of the reality. It may not be rigorously true. In summary, experimental data are always affected by an uncertainty. As a consequence, we need to find a way to express the result of a measurement that is not a useless, sometimes very long list of numbers.

According to quantum mechanics, there is an intrinsic limit to the resolution we can achieve.

rogram is a plot
the number of
ts occurring for
ch result of the
ement or group
of them.

It can be useful to look at how data distribute, making a “histogram”. In a histogram we plot the number of events as a function of the result of the measurement. For example, in the Arduino data we have 2, 2 099, 9 609, 859 and 5 occurrences, respectively, of 34, 35, 36, 37 and 38. Its graphical representation is given on the left of Fig. 4.1. The right plot shows the histogram of data taken with the smartphone.

Instead of plotting the absolute number of events, in a histogram we can show the frequencies, defined as

$$f_i = \frac{n_i}{N}, \quad (4.3)$$

where n_i is the number of events belonging to class (or bin) i and N is the total number of events $N = \sum_i n_i$. From the latter equation it follows that

$$\sum f_i = \frac{1}{N} \sum_i n_i = 1. \quad (4.4)$$

Clearly, data tend to concentrate in one place that, for Arduino, is close to 36. Data distribute around An estimate of the position at which data are thickened is given by the mean, few values. A good defined as

$$\langle x \rangle = \frac{1}{N} \sum_{i=1}^N x_i, \quad (4.5)$$

where x_i are the single measurements and N their number. The mean is often of data. called the average. The two terms are used as synonyms, even if they are not rigorously as such.

For our data the mean is 35.90186098298076 Does it makes sense to specify all these digits? Indeed, what we learn from our first experiment, is that the value of the illuminance is known with some uncertainty that, in this case, is of the order of ± 1 , since data fluctuate mostly between 35 and 37. A better way to express our knowledge is to write the illuminance \mathcal{L} as

$$\mathcal{L}_A = 36 \pm 1 \text{ a.u.} \quad (4.6)$$

(the subscript A stands for Arduino) Looking at PHYPHOX data, it seems more difficult to identify the center and the width of the distribution. Using the same definition as above we find

$$\mathcal{L}_p = 139 \text{ lx.} \quad (4.7)$$

that, in fact, can be taken as an index of position of the distribution, even if it is unbalanced towards right. It is possible to define other indices like the median, Besides the position of data, we need to quote their dispersion.

defined as the value such that exactly half of the data is on its left, and as many on its right. The mode is the most frequent value in a distribution. In the following we take the mean as the index of position of our data.

What about dispersion? Can we take it as ± 1 , too? Well, the PHYPHOX distribution seems a bit *wider* than that of Arduino, not only because of the scale factor (35 vs 139), but also because the average position appears more uncertain in the first case. We should find a better way to express the width of the distribution or, better, its index of dispersion.

A possible, natural definition is that the index of dispersion could be the average distance between individual measurements and their average, i.e.

$$\Delta = \frac{1}{N} \sum_{i=1}^N |x_i - \langle x \rangle|. \quad (4.8)$$

When the absolute value of any quantity is needed in a formula, it is very impractical because it forces us to evaluate the sign for each single term in the sum. However, the distance squared is always positive and we can define square of z and get its square root.

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \langle x \rangle)^2 \quad (4.9)$$

as the square of the index of dispersion. We call this number the variance of the distribution. Still, this definition have a problem. If we take just one measurement, the variance is zero. This leads to the absurd situation in which it appears to be better to take just one measurement to obtain an infinitely precise measurement. To force us to take at least two measurements, we define the variance as

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \langle x \rangle)^2, \quad (4.10)$$

A modified variance such that for $N = 1$, $\sigma^2 = \infty$. On the other hand, the distance $\sigma = \sqrt{\sigma^2}$ is still a measurement of the width of the distribution. It is just a bit wider than the first one, however the difference tends to zero as N tends to infinity. σ is called the standard deviation. Computing it for our data, and taking it as the uncertainty of the sample. We take its square root as the uncertainty and we call it the standard deviation.

$$\begin{aligned} \mathcal{L}_A &= 35.9 \pm 0.5 \text{ a.u.} \\ \mathcal{L}_p &= 139.4 \pm 0.6 \text{ lx.} \end{aligned} \quad (4.11)$$

Stating that, e.g., $\mathcal{L}_A = 35.9 \pm 0.5 \text{ a.u.}$, means that, according to our measurements, the illuminance *true* value is close to 35.9, however we do not ex-

Eq. (4.9) defines the variance of the population.

actly know it, but with an uncertainty of 0.5, i.e., it is comprised between $35.9 - 0.5 = 35.4$ and $35.9 + 0.5 = 36.4$. In turn, this does not mean that all the measurements lie within the interval [35.4, 36.4]. As a matter of fact, data appear also outside this interval.

However, if we take another set of measurements, we should find that about 70 % of them lie within the given interval in both cases. Statistics tell us that this fraction is expected to be close to 68 %, when $N \rightarrow \infty$. Correspondingly, 95 % of data lie within $\pm 2\sigma$ from the mean and 99.7 % within $\pm 3\sigma$.

If we take the standard deviation as a measure of the dispersion of the data, we give it a statistical meaning. Manifestly, the more the data, the better the estimation of the uncertainty. When quoting a result as above, we intend that repeating the measurements will lead to a result within $\pm \sigma$ around the mean with a probability of 68 %. We are implying that fluctuations observed in the measurements are of statistical nature and must then be understood using statistics.

Note that, when quoting the results, we truncate the standard deviation to one significant figure and quote the mean using a number of digits that matches those of the uncertainty. It makes no sense to quote an uncertainty of 0.4773289079373387 . . .

If we cannot know a number precisely enough to distinguish it between 35.9 and 36.3, because of the uncertainty of the order of 0.4, further we cannot distinguish them if they differ on the second significant figure. We then keep the first two digits and approximate them to the closest value written with one digit. The closest number to 0.477 . . . is 0.5. In general, we approximate the first non null digit to the next integer if the following digit is between 5 and 9, while we keep it as such if the following digit is between 0 and 4.

It is worth noting that, if $\sigma = 0$, it does not mean that the value of the physics quantity is known with infinite precision. It only means that the reading uncertainty is larger than statistical fluctuations. Then, the uncertainty must be quoted after it. When quoting the reading uncertainty, however, we provide a different information with respect to the one provided with the standard deviation. The interval defined by the standard deviation has an amplitude of about $\frac{2}{3}$ of the whole interval in which all the measurements fall. If we want to quote an uncertainty having more or less the same meaning when the reading uncertainty dominates, we need to divide the width of the resolution by three. For example, if we measure the width of an A4 sheet of paper to be 210 mm, we in fact intend that repeating the measurement, for sure we will find a number between 209.5 mm and 210.5 mm. Dividing the width of the interval (1 mm) by three we can quote the result as

$$w = 210.0 \pm 0.3 \text{ mm}. \quad (4.12)$$

This way about 70 % of the measurements fall within $\pm 1-\sigma$ interval, even if

Using the standard deviation as a measure of the uncertainty gives a probabilistic interpretation of it.

Num
writ
num
figu
one
the
writ
acc

A null
deviations
the reso
instrume
the read
dominan
case, w
resoluti
three a
uncerta
this nu
interpre
to that

their distribution is completely different. In fact, we are forced to believe that the probability to find any value between 209.5 mm and 210.5 mm is the same and the distribution is said to be uniform.

2. Data analysis with Python

To make the above analysis we can use Python. There are a variety of useful modules for data analysis, namely, `pandas`, a data structures and data analysis tool, `numpy` a package for scientific computing and `matplotlib` for plotting. The following script has been used for the analysis above.
and `matplotlib` is

```
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

if len(sys.argv) <= 1:
    print('Usage: analyse.py [filename]')
else:
    filename = sys.argv[1]
    unit = 'a.u.'
    if len(sys.argv) > 2:
        unit = sys.argv[2]
    f = pd.read_csv(filename)
    data = f.T.values.tolist()
    h, bins, rect = plt.hist(x=data[1], bins='auto')
    print('===== histogram content ')
    print(h)
    print('===== ')
    plt.xlabel('Illuminance [{}]'.format(unit))
    mean = np.mean(data[1])
    median = np.median(data[1])
    stdev = np.std(data[1])
    print('Average = ' + str(mean))
    print('Average = ' + str(median))
    print('StDev = ' + str(stdev))
    plt.show()
```

When **Importing modules** we can give them a nickname (an alias) using `import ... as nickname` or an alias.

```
import pandas as pd
```

Aliases are useful to shorten names. We can even import only part of a module, called a submodule, using a syntax like

```
import matplotlib.pyplot as plt
```

where we ask the script to load the submodule `pyplot` of module `matplotlib`, identified as `plt` in the rest of the script.

Module `sys` provides access to system-specific variables and functions. We use it at the beginning of the script to check if the script is called with arguments. An array of strings called `sys.argv` is defined in the module, containing the name of the actual script in the first component `sys.argv[0]`, and any other parameter on the subsequent ones.

When writing `sys.argv` we intend the attribute `argv` of an object of class `sys`. In some cases there is no need to explicitly instantiate an object of a class, like in `unit = sys.argv[2]` where `unit` is an object of class `String`. The name of the class refers to a default object created *ad hoc*.

Running the script without any parameter on the command line results then in a `sys.argv` array with just one component. We check the length of the array with the function `len()`. If it is less or equal to one, the script prints an error message and exit.

The `if` statement work as follows: it checks the value of a Boolean expression; if it turns to be true it executes the following indented lines, otherwise, it optionally executes a different set of lines, if the `else` clause is present.

Boolean expressions (after the name of George Boole – 1815–1864) are logical statements whose value can be either true or false, such as “it rains”. Statements to which we cannot assign a truth value are not Boolean expression (e.g. “Hey teacher, leave those kids alone!”).

The lines to be executed when the expression is true are written indented after a colon. Those to be executed alternatively, if any, are written, still indented, after the `else:` clause, that can be omitted. In the example we are analysing, if there is at least one parameter after the script name, the length of `sys.argv` is 2 and the string `filename` is given the name of the file, contained in `sys.argv[1]`. For example, if the script, whose name is `analyse.py` is called as

```
analyse.py lightIntensity.csv
```

where the parameter `lightIntensity.csv` is the name of the file containing the data, the length of `sys.argv` is equal to 2, `sys.argv[0]` contains `analyse.py`, while `sys.argv[1]` contains `lightIntensity.csv`.

Once opened in read mode, data can be read using native Python `read()` function. Here we show a different method exploiting the power of the pandas

(alias pd) module. A statement like

```
f = pd.read_csv(filename)
```

open and read the content of a CSV file whose name is `filename` and put its whole content in a data structure, returned by the module and assigned to the object `f`. All in just one line of code!

Data is arranged in a structure called `DataFrame`, from which we want to extract the second column as a list. `DataFrame.T` is the transposed data frame, i.e., assigning `filename` content is

```
"Time (s)", "Illuminance (lx)"  
0.00E0, 1.39E2  
1.10E-1, 1.40E2  
2.21E-1, 1.38E2  
3.33E-1, 1.39E2  
4.43E-1, 1.38E2  
...
```

the resulting data frame can be imagined like a matrix with two rows like

```
"Time (s)" 0.00E0 1.10E-1 2.21E-1 3.33E-1...  
"Illuminance (lx)" 1.39E2 1.40E2 1.38E2 1.39E2...
```

It is worth noting that numbers are written in scientific notation as `mEn` meaning $m \times 10^n$; such that `2.21E-1` corresponds to $2.2 \times 10^{-1} = 0.22$ and `1.40E2` to 1.40×10^2 in scientific notation.

In fact, the data frame contains more information than just numbers. We can access the values with `values` and transform them into a `list` with `tolist()`. This way, data is a list of two lists: the first, whose index is zero, contains times, while the second, with index 1, contains the illuminance. Note that `f.T` still contains the information about the caption "Time (s)" and "Illuminance (lx)", while `f.T.values` contains just the numerical values.

The `histogram` method of `matplotlib.pyplot` (alias `plt`) takes the data passed passing and computes the number of events per bin. Each bin is a class of data that, in the example, are computed automatically by the method, based on the data found. It returns an array containing the number of events per bin, the edges of each corresponding bin and a data structure (`rect`) representing the graphical counterpart of the histogram, usually represented as a series of rectangles are graphically represented as in

We assign a label to the horizontal axis of the histogram using the method `xlabel()` to which we pass a string formatted such that it contains the proper unit. The latter is passed as the second parameter of the script. It defaults to

a.u., but if the script is called with a second parameter, the latter is used to assign the unit object, from which we build the label.

The mean, the median and the standard deviations are computed on the `second` row of the transposed data frame exploiting `mean()`, `median()` and `stddev()` compute the methods defined in `numpy` (for which we chose the alias `np`). With `plt.show` the median and `()` we ask Python to show a graphical representation of the histogram~~that~~ standard deviation. appears as in Fig. 4.1.

Summary

Data collected by an instrument fluctuate because of random effects. The result is that a measurement consists of a distribution rather than a number, unless the resolution of the instrument is poor.

A histogram is a plot showing how experimental data distribute.

Statistics

The value of the measurement is taken to be the mean of the data, defined as

$$\langle x \rangle = \frac{1}{N} \sum_{i=1}^N x_i .$$

The width of the distribution is taken as its standard deviation defined as

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \langle x \rangle)^2} .$$

Its square σ^2 is called the variance. Measurements are quoted as $\langle x \rangle \pm \sigma$ followed by its units, as in

$$\mathcal{L} = 35.9 \pm 0.5 \text{ a.u.} .$$

When the reading uncertainty δ dominates and $\sigma = 0$, the uncertainty is taken as $\simeq \frac{\delta}{3}$.

There is 68 % of probability to find a measurement within an interval of $\pm 1\sigma$ around the mean. The percentage is 95 % within $\pm 2\sigma$ and 99.7 % within $\pm 3\sigma$.

We quote uncertainties with at most one significant digit and write the mean accordingly.

Python

`pandas`, `numpy` and `matplotlib` are useful modules for scientific applications.

We can choose among two or more options checking the truth value of a Boolean expression using the `if` statement.

Data in a CSV file can be read as a data frame via `pandas`. From a data frame we can extract columns to be used to compute the mean and the standard deviation thanks to `numpy` functions and build histograms with `matplotlib.pyplot.hist()`.

Chapter 5

Establishing physics laws

This chapter is devoted to illustrate the way in which physicists establish phenomenological physics laws, i.e. purely experimentally based relations between physics quantity. In doing that we do not make any attempt to understand the underlying physics, apart for few very basic properties.

1. Light absorption

In this chapter we want to study light absorption by a transparent material. It is well known that light can be partly absorbed by a transparent filter, depending on the thickness and the nature of the material of which is made of. Here we want to quantitatively study this phenomenon. As already said, making an experiment means to find numerical values and not just observing qualitatively something. In this case we need to find the precise relationship existing between the intensity of the transmitted light and the properties of the filter, such as its thickness.

To setup such an experiment we need an intense enough and stable source of light, an instrument to measure its intensity and a series of transparent filters. As usual, there can be plenty of professional instrumentation in your university or college, however, we try to perform the experiment using readily available material, such that you can do that at home, too.

Irrespective of the place where you setup your experiment, it is of capital importance that the setup is clean, stable and safe. Take your time to fix everything in the proper way such that you do not risk to introduce biases in your experiment due to bad alignments, parts moving, etc.. Usually, a good criterion to understand if an experiment has been well designed is aesthetic. A nice setup is certainly better than an ugly one.

The light source can be either a tabletop lamp or the flash light of a smartphone. The instrument to measure the light intensity can be either another smartphone equipped with PHYPHOX or an Arduino with a light sensor. Filters can be realised cutting out 5-cm wide strips from an A4-crystal-clear plastic punched pocket wallet, stapled together on one of the short sides.

Experiments must look good to be effective.
Always pay attention to the realisation of the setup.

Many experiments can be realised using readily available materials.

The experiment consists in measuring the light intensity after it traversed n strips, with n variable. Using PHYPHOX we put the smartphone under the light, then cover the light sensor with a strip and starts data acquisition. Once enough data have been collected, we put another strip on top of the latter and restart data acquisition and so on, for a number of strips.

We need to measure the light intensity that emerges from a series of n filters, as a function of n .

Take many measurements of the light intensity in timed runs of few tens of seconds, avoiding influencing the measurement with your fingers. For each set of measurement, compute the mean and the standard deviation. Record the values in a text file as a function of n .

To locate the light sensor on your mobile device, just start data acquisition, then move a finger along the surface of the phone until you see a drop in the intensity. When you start data acquisition, your hand must pass in front of the phone and can partially shield the light coming on the sensor. To avoid this effect it can be useful to delay data acquisition for about two seconds with a TIMED RUN. Taking data for 30–40 s is more than enough.

Save data collected for each number n of strips in different files, called after n (e.g. `lightIntensity-3.csv`, `lightIntensity-7.csv` and so on). Using the techniques described above build a table with n , and the corresponding mean and standard deviation of the corresponding intensity of the light.

2. Taking the average with Arduino

Arduino, contrary to PHYPHOX, is programmable and with software we can do many things, including computing the mean and the standard deviation directly during data acquisition (online).

Let's consider the following `loop()` function in the Arduino sketch.

```
void loop() {
    float S = 0.;
    float S2 = 0.;
    for (int i = 0; i < 1000; i++) {
        int k = analogRead(A0);
        int k2 = k*k;
        S += k;
        S2 += k2;
    }
    Serial.print(S/1000);
    Serial.print(", ");
    Serial.println(sqrt(S2/1000-S*S*1e-6));
    while (1) {
        // do nothing
    }
}
```

As soon as the function begins, two variables, `S` and `S2` are declared as `float` and assigned the value 0. A variable, in programming, is a container for

Data are stored in memory as sequences of bits and their interpretation depends on their type. The same sequence, in fact, can be interpreted differently. Consider a 32-bits long variable whose content is the following:

how to interpret them
01000101 11010010 01100000 00000000

where we separated groups of eight bits (a **byte**) for better readability. Its hexadecimal representation is 0x45D26000 (the prefix 0x identifies numbers in base-16). If interpreted as an integer number, its value corresponds to

$$n = \sum_{i=0}^{32} b_i \times 2^i \quad (5.1)$$

where $b_i = 0, 1$ is the i -th bit and bits are counted from right to left. The calculation leads to $n = 1171\,415\,040$ (work it out by yourself). If interpreted as a string it represents a three-characters string. Each character in a string is in fact represented as a sequence of eight bits. The correspondence between a given character and the integer number represented by each sequence has been agreed internationally as the ASCII table (ASCII stands for American Standard Code for Information Interchange). Searching the Internet for that table we find that the sequence 01000101, corresponding to 69, represents the character E. The next one, 210 if interpreted as an integer, represents the character Ò in the UTF-8 encoding. The third integer 01100000 corresponds to the ASCII code 96 representing the single quote `, while the latter is the so-called NULL character whose ASCII code is zero.

In the ASCII table, numbers from 0 to 127 are fixed and internationally valid characters. Characters whose ASCII code is 128 or more depend on the encoding. ISO Latin-1 is one of the possible encodings and is frequently used in western countries. Changing the encoding, the same integer may represent a different character.

The NULL character marks the end of a string. A string made of three characters needs four bytes to be represented: the three characters plus the NULL one.

If interpreted as a floating point number, the 32-bits represent the number 6 732. Non integer numbers are represented following the IEEE-754 standard. In this format, numbers are represented in a sort of scientific notation in base 2, as $m \times 2^e$, where m is written in the *normal* form and e in the excess-127 notation. To understand the form imagine to write 6 732 as above with $1 < m < 2$, i.e., as $1.6435546875 \times 2^{12}$ corresponding to the binary code $1.1010010011 \times 2^{1100}$. Since, following this convention, the first digit is always 1, it is not worth representing it in the memory and the so-called mantissa in normal form is 1010010011. The excess-127 notation is obtained adding 127 to the number, such that the exponent e is represented as $127 + 12 = 139$. The 32-bits number

String
by a
who

is composed as follows: the first bit represents the sign (0 for + and 1 for -), the following eight bits represent the exponent e in excess-127 notation (1000101 in binary), while the remaining 24 bits represent the mantissa in the normal form.

Variables of type **float** represent floating point numbers and hence are stored in the memory following the latter convention.

= is the assignment operator and initialisation can be done at the same time. When declaring a variable specifying its name and type, we can assign it an initial value with the = operator. Each statement, as usual, ends with a semicolon.

Loops can be done The **for** loop is used to iterate over a given number of times. In this case the using the **for** integer variable **i** is initially set to 0 (the first expression in parenthesis, where **i** statement, used when is declared as **int**, too). Before entering the loop, Arduino evaluates the middle you want to count the expression in the **for**, $i < 1000$. If it is true, it executes the statements within number of iterations. the pair of braces, otherwise it abandons the loop.

This way we repeat 1000 times the following statements.

```
int k = analogRead(A0);
int k2 = k*k;
S += k;
S2 += k2;
```

After each execution+ the third expression of the **for**, $i++$, is evaluated. $++$ is a post-increment operator: it returns the value of **i**, then increases it by one.

In the loop we put the value returned by `analogRead(A0)` into an integer variable **k**, and store in **k2** its square (`*` being the multiplication operator). Then we iteratively sum **k**, the last reading, to **S** and **k2** to **S2**. The sum is made by the auto-increment operator `+=` that adds the operand at its right to the variable at its left.

When we exit from the loop, **S** contains the sum of 1000 measurements of the light intensity in arbitrary units, and **S2** the sum of their squares.

We then send over the serial line the value of $S/1000$, corresponding to the mean of the data, followed by

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2 - \left(\frac{1}{N} \sum_{i=1}^N x_i \right)^2} = \sqrt{\langle x^2 \rangle - \langle x \rangle^2}, \quad (5.2)$$

with $N = 1000$, separated by a comma from the mean. Remembering the

definition of the variance of the population we find that The variance of a population can be evaluated as

$$\sigma = \frac{1}{N} \sum_{i=1}^N (x_i - \langle x \rangle)^2 = \frac{1}{N} \sum_{i=1}^N (x_i^2 + \langle x \rangle^2 - 2x_i\langle x \rangle) - \langle x \rangle^2 \quad (5.3)$$

$$= \langle x^2 \rangle + \langle x \rangle^2 - 2\langle x \rangle^2 = \langle x^2 \rangle - \langle x \rangle^2.$$

For $N = 1000$ there is not much difference between the variance of the population and that of the sample, so we take the standard deviation as the square root of the expression above.

Someone finds the formula difficult to remember. In particular, it is difficult to remember whether σ^2 is equal to $\langle x^2 \rangle - \langle x \rangle^2$ or to $\langle x \rangle^2 - \langle x^2 \rangle$. Actually, it is quite easy to choose between the two options: just note that in the case of two equal and opposite measures $\langle x^2 \rangle$ is certainly positive, while $\langle x \rangle^2$ may be worth 0. In this case $\langle x \rangle^2 - \langle x^2 \rangle$ would lead to a negative value for σ^2 which is impossible.

As in Python, in the Arduino language, `while` realises an iterative structure. The expression in parenthesis is evaluated as a Boolean expression. On Arduino, as true and false are represented as integer numbers different and equal to zero, respectively. Since $1 \neq 0$, the expression is always true and the `while` loop never finishes. There are no statements in the loop. The double slash // represents a comment, intended to provide some information to the reader. Any character written after that sign is ignored by the Arduino compiler. In fact, the execution of the program stops here.

We can read the mean and the standard deviation from the serial monitor. To restart data acquisition after adding a strip on top of the sensor, we just need to press the RESET button on one of the Arduino's corners. This way we can just manually record mean and standard deviation for various n .

3. A first look at data

To have an idea about how illuminance I depends on the thickness of the filter, we can make a first plot of the values of I as a function of n , as shown in Fig. 5.1. The vertical bars on each point represent the amplitude of the $1-\sigma$ interval.

To make the plot we used the techniques illustrated in the previous chapters. The measurements have been done using Arduino. Comparing the illuminance measured for $n = 0$ with PHYPHOX, $\mathcal{L}_p(0) = 532.0$ lx, with the one obtained with Arduino in arbitrary units, $\mathcal{L}_A(0) = 31.06$, we can easily obtain a calibration factor. If we assume that

$$\mathcal{L}_p(x) = C\mathcal{L}_A(x), \quad (5.4)$$

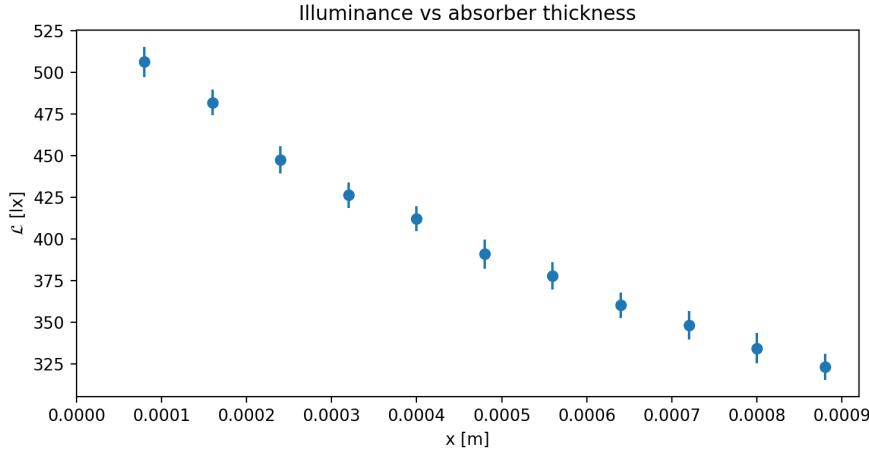


Figure 5.1: The intensity of the light measured as a function of the thickness of the filter. Uncertainties are shown as vertical bars on data.

the calibration factor C is

$$C = \frac{\mathcal{L}_p(x)}{\mathcal{L}_A(x)}. \quad (5.5)$$

Since $\mathcal{L}_A(x)$ is given in arbitrary units (then, dimensionless) and $\mathcal{L}_p(x)$ is in lux, the calibration factor is measured in lux, too, and a measurement made with Arduino in arbitrary units can be casted into a measurement in lux by multiplying it for

$$C = \frac{532.0}{31.06} \simeq 17.13. \quad (5.6)$$

Clearly, both $\mathcal{L}_p(0)$ and $\mathcal{L}_A(0)$ are affected by uncertainties, hence C , too, cannot be known with infinite precision. For a first interpretation of the data, where we want to study the relative change in illuminance, this problem can be neglected for the moment and we postpone its discussion.

The thickness of the filter is obtained multiplying the number of strips by the thickness of the plastic sheet $dx = 80 \pm 3 \mu\text{m}$, measured using a caliper. For the moment we ignore the effects caused by the possible inhomogeneities in strips' thickness.

At a very first glance, illuminance seems to decrease almost linearly. We might then try to model the data as

$$\mathcal{L} = \alpha x + \beta. \quad (5.7)$$

A **phenomenological** law is what we call a physics law, i.e. a relationship between two or more physics quantities. The law is valid as long as we cannot spot any relation between physics quantities guessed by experimental data.

significant deviation from it in our experiments. For the time being *significant* is only an adjective meaning “taking into account the experimental uncertainties”. We give it a more quantitative meaning in the following.

Since a physics law is an equality between physics quantities, the equation must hold for measurement results. Then, since the left side of the equation is measured in lux, so should be the right one. Moreover, the right hand side of the equation is made of two terms added together. Manifestly, each term must exhibit the same properties: if the sum has to be measured in lux, β must be expressed in this same unit, as the product αx . On the other hand, x is a length and is measured in m, and to make αx an illuminance, α must be measured in units of illuminance divided by units of length, e.g. lx/m.

The above considerations are part of what physicists call dimensional analysis. Dimensional analysis is very useful, not only to determine the units in which things are measured. It also helps in spotting mistakes in equations’ manipulations. Since every physics equation must be dimensionally consistent, such a consistency must be manifest at each stage of computation. If, at a certain point, you find that the equation you have written is not dimensionally consistent, for sure you made a mistake before. To make dimensional analysis easy, we use symbols enclosed in square brackets [. . .], where the symbols, represented by capital letters, indicate the *nature* of the quantity. For example, if we use I to indicate illuminance and L for length, we write

$$[I] = [\alpha] [L] + [\beta] , \quad (5.8)$$

such that

$$[\alpha] = \frac{[I]}{[L]} \quad \text{and} \quad [\beta] = [I] . \quad (5.9)$$

We can then assign the proper units substituting the chosen one for each quantity, e.g. lux for $[I]$ and m for $[L]$.

An essential requirement for any physics law is to be dimensionally consistent. Both sides of the equation that represents it must be measurable in the same units, as well as all the terms in a sum or a difference. The arguments of trigonometric functions, logarithms and exponentials must be dimensionless.

Dimensional analysis

It is worth noting that dimensional analysis is also great in helping memory or guessing the right formula for something. For example, in special relativity it is known that time appears dilated for systems moving with respect to an observer, i.e. $t \rightarrow t' = \gamma t$, where $\gamma > 1$ because $t' > t$. Since $[t] = [t']$, γ must be dimensionless and can always be written as a ratio between two numbers. It must also depend on the velocity \mathbf{v} of the moving system. The higher the speed, the bigger γ , such that \mathbf{v} must appear in the denominator. However, \mathbf{v} is a vector and γ is a scalar. The only way to obtain a scalar from just one vector is to multiply the vector by itself as in $v^2 = \mathbf{v} \cdot \mathbf{v}$. v^2 , on the

other hand, is not dimensionless, but $[v^2] = [L^2 T^{-2}]$, T being the symbol for time. In order to make a dimensionless combination we need to divide v^2 for the square of another velocity. Since the only other velocity involved is the one of light $c = 3 \times 10^8$ m/s and is universal, manifestly γ must depend on the ratio $\frac{v^2}{c^2}$. When $v = 0$, $\gamma = 1$ and we can then finally write

$$\gamma = \left(\frac{1}{1 - \frac{v^2}{c^2}} \right)^n. \quad (5.10)$$

It turns out that $n = \frac{1}{2}$. Besides dimensionless numerical factors, this is the only other information that dimensional analysis cannot provide.

Each time we have no reason to prefer a value among others, we average over them.

In order to determine the values of α and β we can keep any pair of data points and evaluate the slope as

$$\alpha_{ij} = \frac{I_i - I_j}{x_i - x_j}, \quad (5.11)$$

while the intercept can be obtained by imposing

$$I_i = \alpha_{ij} x_i + \beta. \quad (5.12)$$

How do we choose the pair to compute α_{ij} and which point (x_i, I_i) do we choose to find β ? In principle all the possible pairs should lead to the same value for α_{ij} , however, uncertainties lead to a different value for each pair. Choosing $i = 1$ and $j = N$, with $N = \max(n)$ could be a choice, giving us the largest leverage, but the extreme points can be problematic, leading to wrong values.

Combinatorics allows the computation of permutations of n elements $n!$, dispositions $\frac{n!}{(n-k)!}$ and combinations $\frac{n!}{k!(n-k)!}$ of n elements in groups of k .

Combinations are dispositions where the order of the elements does not matter.

Formulas are valid when elements do not repeat in the list.

As we learn in Chapter 4, averaging over many measurements gives better results. Then, averaging over all the possible α_{ij} guarantees the best possible result. The number of possible pairs is given by combinatorics, according to which the number of combinations of k elements in a set of n without repetitions is given by

$$C_{nk} = \binom{n}{k} = \frac{n!}{k!(n-k)!}, \quad (5.13)$$

where the symbol in parenthesis is called the binomial coefficient and ! denotes the factorial $n! = \prod_{i=1}^n i$. In this case $n = 11$ and $k = 2$, such that

$$C_{11,2} = \frac{11!}{2!9!} = 55. \quad (5.14)$$

Given the 55 possible slopes, we can average them and evaluate the corresponding standard deviation. The latter is $\sigma_\alpha \simeq 114\,632$ lx/m. Taking

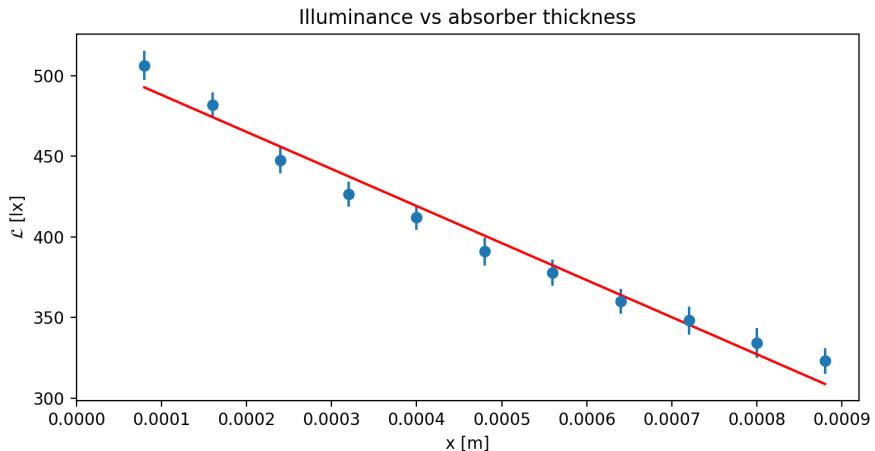


Figure 5.2: The intensity of the light measured as a function of the thickness of the filter with a linear model superimposed.

into account our rule according to which we only keep one significant digit, $\sigma_\alpha \simeq 100\,000 \text{ lx/m}$. It might appear as huge, but α is, on average, $\alpha \simeq -450\,473 \text{ lx/m}$, such that our result should be written as

$$\alpha = (-5 \pm 1) \times 10^5 \text{ lx/m}. \quad (5.15)$$

Written this way, the uncertainty does not seem so large, being $\frac{1}{5} = 0.2$, i.e. 20 %. The ratio between the uncertainty and the central value of a measure is called relative uncertainty and gives a more objective feeling about how large the uncertainty is.

We can do a similar thing for β . We can average over all the possible values of i , assuming $\alpha = -5 \times 10^5 \text{ lx/m}$. The result is

$$\beta = 520 \pm 10 \text{ lx}. \quad (5.16)$$

Fig 5.2 shows the resulting line superimposed to data. The agreement between data and the model is not bad, but certainly not very satisfactory.

4. Plotting graphs and interpolating data

Suppose that our CSV file contains three columns: the number of plastic strips covering the sensor, the measured illuminance in arbitrary units, and its uncertainty. We can read the file using Python and convert the number of strips into a thickness in units of length and the intensity in units of luminous flux per unit area, as follows.

```
f = pd.read_csv(sys.argv[1])
data = f.T.values.tolist()
```

If you are uncertain about how many digits you should quote in your result, write the standard deviation using the scientific notation, keeping just one digit in the mantissa, then write the central value using the same exponent, keeping only the digits before the decimal point.

```
x = [x*dx for x in data[0]]
y = [x*C for x in data[1]]
dy = [x*C for x in data[2]]
```

The first statement open the file whose name is passed as the first argument to the script and read it. It transpose the data, remove the headers, if any, and returns a list of three lists: `data[0]` containing the number of plastic strips, `data[1]` representing the illuminance in arbitrary units and `data[2]` that stores the uncertainties on the latter.

We generate three new lists called `x`, `y` and `dy` in the last three lines, using a technique called “list comprehension” consisting in an implicit iteration over all To iterate the elements of a list. On each line we iterate over the components of each ~~transf~~^{list} using `for x in data[0]` and store in the corresponding component of ~~use the~~^{new} `array` the result of the transformation applied to `x`, representing the actual values. In the case of `data[0]`, the number of strips, represented by `x`, is multiplied by `dx`, defined as

```
dx = 80e-6
```

`append()` add an element to the end of a list. A list with given length `N` can be created with `x = [0]*N`.

and representing the thickness of a single strip in m ($80 \mu\text{m}$). It is useful to see how the same effect can be obtained using a more *traditional* programming:

```
x = []
for i in range(len(data[0])):
    x.append(data[0][i]*dx)
```

The first statement creates an empty list whose name is `x`. Then, we iterate over `data[0]` components (remember that `data[0]` is itself a list) counting with `i`. `len()` returns the length of a list, while `range(n)` returns an integer range from 0 to $n-1$. Multiplying by `dx` the i -th component of the list `data[0]`, `data[0][i]`, we obtain the thickness in units of length and store them in subsequent components of `x` via `append()`.

The plot is made with the following code.

```
plt.errorbar(x, y, yerr=dy, fmt='o')
plt.title('Illuminance vs absorber thickness')
plt.xlabel('x [m]')
plt.ylabel('$\{\mathcal{L}\} [lx]')
plt.xticks(xticks)
plt.show()
```

The `errorbar()` method gets the two arrays `x` and `y` and plots the second as a function of the first. With `yerr=dy` we inform it that the uncertainties on `y` are stored in `dy`, while `fmt='o'` tells the method to represent data with circles.

The word “error” is used as a synonym for “uncertainty”. The name of the method is derived from the term by which measurement uncertainties are sometimes referred as “errors”. The latter is a bit misleading, because it suggests a mistake. Though the term “uncertainty” should be preferred, in what follows we sometimes use the term “error” as a synonym.

The name of the game

\LaTeX can be considered an evolution of \TeX , a document preparation system widely adopted especially in the scientific domain. \LaTeX is pronounced “latek” because \TeX was pronounced “tek”.

The reason for that is explained by \TeX ’s creator Donald Knuth (also known to be the author of the monumental “The Art of Computer Programming”):

English words like “technology” stem from a Greek root beginning with the letters $\tau\epsilon\chi\dots$; and this same Greek word means art as well as technology. Hence the name \TeX , which is an uppercase form of $\tau\epsilon\chi$.

The “La” prefix that turned \TeX into \LaTeX refers to the name of its author Leslie Lamport, who started writing *macros* to simplify the usage of the Knuth’s invention to write scientific documents.

The reference to art in Knuth’s words is not accidental. Throughout this textbook we refer to art few times. Science and coding, in fact, are arts, as long as art implies the “creation” of something new. The root for the word “artisan” is the same for the same reason. Moreover, art and technology could be considered as synonyms in ancient greek, given that the latter, too, implies creation.

We decorate the plot with a title and labels, one of which is rendered using \LaTeX (pronounced “latek”), a high-quality document preparation system that is worth learning for scientific typesetting. We also force the tick marks on the horizontal axis to appear at the coordinates given in the `xticks` list, defined as

```
xticks = np.arange(0, 0.50e-3, 1e-4)
```

\LaTeX is a document preparation system that is worth learning.

To compute the average slope and its error we define a function, i.e. a set of operations identified by a single name, whose behaviour depends on zero or more parameters, as in

```
m, sigma = averageSlope(x, y)
```

The function `averageSlope()` is defined as

```
from itertools import combinations as comb

def averageSlope(x, y):
```

Defin
way t
instru
name
depen
Funct
reada
chan
make
maint

```

ijpairs = list(it.comb(np.arange(len(y)), 2))
m = 0
m2 = 0
for ij in ijpairs:
    i = ij[0]
    j = ij[1]
    s = (y[i]-y[j])/(x[i]-x[j])
    m += s
    m2 += s*s
N = len(ijpairs)
m /= N
sigma = np.sqrt(m2/N-m*m)
return m, sigma

```

itertools is a useful module to iterate over elements in a sequence: an abstract type. Abstract types cannot be instantiated, i.e. variables of that type do not exist.

They are used to describe a common behaviour of different types.

```
from itertools import combinations as comb
```

Python methods and functions can return zero or more objects. The two variables `m` and `m2` are intended to represent, respectively, the mean of the slopes and the mean of their squares. The slope is computed in the `for` loop that, iterating over all the `ijpairs`, computes all the possible values of the slope `s` and sum them up in `m` and their squares in `m2`. The average of the slopes squared is used to compute the standard deviation.

Both the mean and the standard deviation are returned by this function.

5. An approximated model

It is not difficult to realise that the model above is not a good one. The reason is very simple, indeed. The illuminance \mathcal{L} is, by definition, positive, i.e. $\mathcal{L} > 0$. If $\mathcal{L}(x) = \alpha x + \beta$, there is always a value $x = x_0$ such that $\mathcal{L}(x > x_0) < 0$. On the other hand, a linear model seems to be quite good in intervals of thickness short enough. For example, for $x < 0.00015$ m, it is easy to find a line well describing the data.

Taylor expansion is a useful technique to approximate a function in a small interval.

The reason is that any function $f(x)$ infinitely differentiable at a point a can

be expanded in a Taylor's series as

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n, \quad (5.17)$$

$f^{(n)}(a)$ denoting the n -th derivative of $f(x)$ at $x = a$. For small enough $(x - a)$, i.e., in the vicinity of the point $x = a$, the higher order terms become smaller and smaller and the series can be truncated.

For small enough $(x - a)$, any physics model $\mathcal{L}(x) = f(x)$ can be approximated by a linear one

$$f(x) \simeq f(a) + f'(x)(x - a) \quad (5.18)$$

irrespective of how complicated $f(x)$ is. The approximated model looks better with increasing n . In general, any function (at least, those of interest to a physicist) can be approximated by a sufficiently high degree polynomial.

We use this feature when we study systems like springs and oscillators. Such systems are complex, indeed, but if the external forces to which they are subject is low, their state changes slightly and whatever the correct model is, it can always be approximated by linear or quadratic ones.

dating the Taylor series at the first order transforms any function into a line. Functions that exhibit a minimum or a maximum are better described by a second order truncation near the extreme points.

6. Non-polynomial models

Looking carefully at the plots suggests that the change $d\mathcal{L}$ of \mathcal{L} diminishes with x . Since also \mathcal{L} diminishes with x , we can write that

$$d\mathcal{L} = -\alpha \mathcal{L}. \quad (5.19)$$

Such an equation means that the change in illuminance is proportional to the illuminance itself. If the illuminance is high, the filter can absorb a lot of light and the luminous flux per unit area decreases of a relatively large quantity. On the other hand, if the light impinging on the filter is low, the filter cannot absorb more light than that it receives, and the change in the flux is lower. The change is negative, because the illuminance after the filter is lower than that before it, and if $\mathcal{L} = 0$, $d\mathcal{L} = 0$ as expected.

Often, in physics, we observe a change in a quantity that is proportional to the quantity itself.

This model seems already more physically plausible than the simple linear one of the previous section.

Moreover, it is clear that $d\mathcal{L}$ must be proportional to the thickness dx of the filter: the thicker the filter, the higher, in absolute value, the change in illuminance. This leads us to modify the model such that

$$d\mathcal{L} = -\beta \mathcal{L} dx, \quad (5.20)$$

with $\alpha = \beta dx$. Such an observation is suggested also by the fact that the left hand side of equation (5.19) contains an infinitesimal, while the right hand side is written as if it was a finite quantity. In order to make it explicit that even the right hand side of the equation is an infinitesimal quantity, we rewrite it as above.

Eq. (5.20) is what mathematicians call a separable differential equation whose solution is relatively simple. Dividing both members by \mathcal{L} and integrating we find functions and their derivatives.

$$\int_{\mathcal{L}(0)}^{\mathcal{L}(x)} \frac{d\mathcal{L}}{\mathcal{L}} = -\beta \int_0^x dx, \quad (5.21)$$

such that The solution of the above model is always

$$\text{an exponential function. } \log \frac{\mathcal{L}(x)}{\mathcal{L}(0)} = -\beta x, \quad (5.22)$$

that can be rewritten taking the exponential of both members as

$$\mathcal{L}(x) = \mathcal{L}(0) \exp(-\beta x). \quad (5.23)$$

The parameter β has to be determined from data and includes the modelling of various effects depending on the absorber nature. For example, a crystal clear filter absorbs less light than an embossed one.

The argument of the exponential must be dimensionless, then $[\beta] = [L^{-1}]$. Sometimes, to make this explicit, we prefer to define argument of functions

$$\text{like log or exp is dimensionless. We} \quad \lambda = \frac{1}{\beta} \quad (5.24)$$

such that $[\lambda] = [L]$ and λ , measured in m, is called mean free path. Finding the value for λ is easy if we use the linearised version of the law linking \mathcal{L} to x : their dimensions becomes apparent.

$$\log \frac{\mathcal{L}(x)}{\mathcal{L}(0)} = -\frac{1}{\lambda} x. \quad (5.25)$$

Linearisation consists in finding a transformation of an equation of the form $y = f(x)$, such that it appears as $w = az + b$, where $w = F(y)$ and $az + b = F(f(x))$. In the case of an equation such as (5.23), the transformation consists in taking the logarithm of both sides.

Fig. 5.3 (left) shows the linearised plot, i.e. a graph in which we show the logarithm of the ratio $\frac{w}{w(0)} = \frac{F(x)}{F(0)}$ versus the absorber thickness x . The average interpolating line, obtained as above, has been superimposed to the plot. The model appear quite good, indeed. Now data are found almost all along the line, but

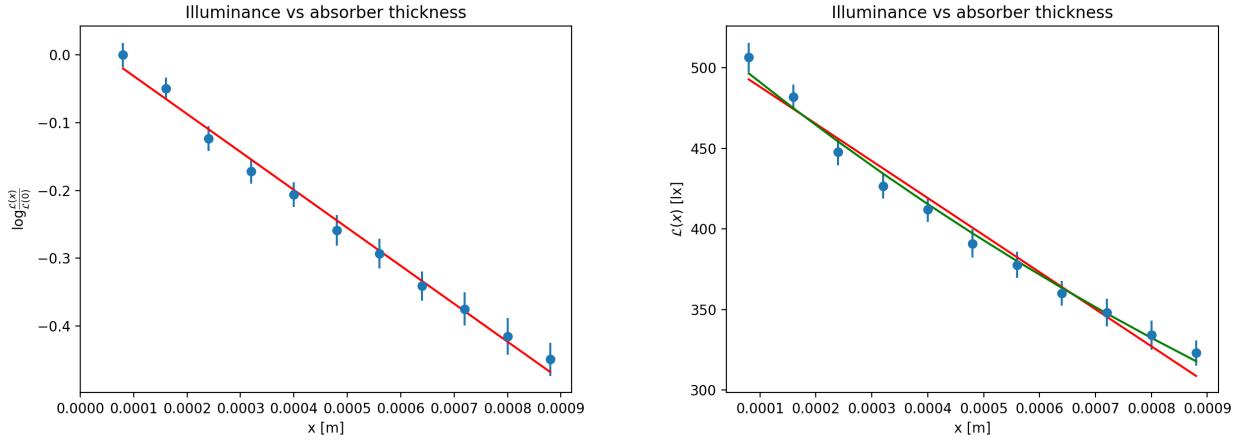


Figure 5.3: Left) Linearised data of the illuminance versus the filter thickness with the average line superimposed. Right) comparison of the two model: the linear model is in red, the exponential in green.

few of them, because of statistical fluctuations (so, this is expected). Indeed, knowing that a $1 - \sigma$ interval contains more or less 70 % of data, we expect about $\frac{1}{3}$ of the points not to lie on the interpolating curve. On the right side of the figure, the two models are compared. The linear model is represented in red, while the exponential one is in green. Manifestly, its agreement with data is better.

To obtain these plots, we follow the same procedure as above, finding that the slope of the interpolating line in the left one is

$$\frac{1}{\lambda} = -1100 \pm 200 \text{ m}^{-1} \quad (5.26)$$

while the intercept, that is dimensionless, is

$$q = 0.02 \pm 0.01. \quad (5.27)$$

From the equation we could predict $q = 0$. In this case $q \neq 0$, but it is *only* 2σ different from zero. Remember that there is a chance of 95 % that the measured value lies within an interval whose amplitude is 2σ . Data, then, are still compatible with our model, if we accept that a fluctuation happening with a probability of 5 % is not so unlikely.

It is worth noting that such a model is better than the previous one, not only because data follows it better, but mainly because it is grounded on reasonable considerations about the mechanisms involved in light absorption. Just finding the curve that best fit your data is not enough. You need to justify the assumptions with reasonable arguments that are not in contradiction with the rest of

To be
alternati
fit the
be gro
experi
knowle
reasona

your knowledge. If so, to be acceptable, the model must explain, possibly in an alternative way, already known facts.

Of course, intuition about how to describe data correctly, is supported by looking at how data distribute and that is why plotting is important.

The ~~Writing on the slope~~ tells us that the mean free path of the light inside the plastic of which the sleeves are made is about $0.001 \text{ m} = 1 \text{ mm}$. Using the form their interpretation

$$\text{easier: } \beta \simeq 1100 \text{ m}^{-1} \quad \text{is not as informative as} \quad \mathcal{L}(x) = \mathcal{L}(0) \exp\left(-\frac{x}{\lambda}\right) \quad (5.28)$$

~~for the model, we can see very clearly that, when $x \simeq \lambda$, the illuminance is reduced by a factor $\frac{1}{e} \simeq \frac{1}{3}$. After three mean free path, $\mathcal{L}(x) = \mathcal{L}(0)e^{-3} \simeq 0.05\mathcal{L}(0)$, i.e., the illuminance is reduced to 5 %.~~

7. The exponential model

The ~~behaviour seen for~~ light intensity is very common in physics. An exponential growth or decrease is observed any time some quantity y changes proportionally to itself and to another quantity x , as in

is either linear,
exponential or
periodic. Once you

$$dy = \alpha y dx, \quad (5.29)$$

~~dominate the math of these models, you can model almost any physics phenomena~~ where α is a constant having the dimensions of $[x^{-1}]$. In many cases you will be able to draw results on topics you still do not know well. Usually it is enough to know few basic laws and, even without a deep understanding of them, you can obtain important results. Examples of such a behaviour are the following.

When ~~Expenditure oscillates~~, its amplitude A decreases with time because of friction ~~Friction can subtract to the pendulum a fraction of its energy, leading the amplitude to reduce~~ to a fraction of its value: $dA \propto -A$. Moreover, the longer ~~the oscillation~~, the higher the decrease, so $dA = -\alpha A dt$ and $A(t)$ behave as above, decreasing exponentially.

In ~~electrical circuits~~ capacitors store energy by storing charges on conductors. ~~Changes moving through~~ the circuit generates a current, whose intensity depends on the resistance ~~emptying~~ of the circuit and the applied voltage. To predict their behaviour, knowing ~~the above~~ principles and two basic formula (the Ohm's Law and the definition of capacitance) is enough.

~~Electrical capacitors~~ In an RC circuit, in which a capacitor discharges injecting a current through a resistor R , the amount of electric charge dQ subtracted to the capacitor ~~accumulating charges~~ cannot be larger than the one stored in it Q and is larger when Q is larger, such that $dQ \propto -\alpha Q$. Again, the amount of charge subtracted to the capacitor is proportional to the elapsed time dt and $dQ = -\alpha Q dt$. As a result, the charge

in the capacitor decreases exponentially and, given that

$$\Delta V = \frac{Q}{C},$$

ΔV being the voltage across the capacitor and C its capacitance, ΔV decreases exponentially, too. By the Ohm's Law, $I = \frac{\Delta V}{R}$, the current I flowing in the circuit drops exponentially, too.

If we connect a voltage source V_0 to a capacitor through a resistor R , we have a similar behaviour. The current flowing into the circuit is again given by the Ohm's Law, where the capacitor acts as a voltage source, such that

$$V_0 - \Delta V = RI = R \frac{dQ}{dt}. \quad (5.31)$$

Substituting the expression of ΔV we obtain

$$dQ = \frac{1}{R} \left(V_0 - \frac{Q}{C} \right) dt. \quad (5.32)$$

We integrate this differential equation taking $y = \left(V_0 - \frac{Q}{C} \right)$, such that $dy = -\frac{dQ}{C}$, and substituting to rewrite it as

$$-C dy = \frac{1}{R} y dt \quad (5.33)$$

the equation can be rewritten in the usual form

$$\frac{dy}{y} = -\frac{1}{RC} dt \quad (5.34)$$

leading to an exponential decrease of y that, observing that $Q(0) = 0$ and remembering that $\frac{Q}{C} = \Delta V$, can be casted into a law for ΔV as

$$V_0 - \Delta V(t) = V_0 \exp \left(-\frac{t}{RC} \right). \quad (5.35)$$

Measuring ΔV , we then expect it to raise indefinitely tending to V_0 as $t \rightarrow \infty$:

$$\Delta V(t) = V_0 \left(1 - \exp \left(-\frac{t}{RC} \right) \right). \quad (5.36)$$

The same behaviour is exhibited by any body reaching thermal equilibrium, whose temperature changes exponentially towards the equilibrium one. Also, the radioactive decay rate follows an exponential model, as well as the absorption of subatomic particles in a beam impinging on a target. Thanks to our

Capacitors are characterised by their capacitance given by the ratio between the charge Q on their plates and the voltage across them.

Ohm's Law predicts that the current flowing in a circuit is proportional to the applied voltage.

Our very simple experiment allows us to develop a model to describe a variety of phenomena, from the simplest to the most intriguing ones, like the behaviour of subatomic particles.

understanding of this model, Viktor Hess (1883–1964) was able to discover cosmic rays in 1912 and we can treat cancer in radiotherapy facilities.

It is then very important that you realise that, starting from a model for which $dy = -\alpha y dy$, you can predict that $y(t)$ behaves exponentially and, conversely, that if you observe a quantity to change exponentially when another one increases linearly, an appropriate model can be formulated as above.

Summary

Dimensional analysis consists in identifying the nature of a physics quantity in an equation and imposing or verifying that both sides of the equation have the same dimensions. Two quantities of the same kind are said to be **commensurable** (i.e. can be measured with the same tool).

Graphs showing the width of uncertainty intervals are useful. The amplitude of the uncertainty interval is shown as an error bar extending from $\mu - \sigma$ to $\mu + \sigma$, where μ is the central value.

Almost any function $f(x)$ to which physicists can be interested in, can be rewritten as an infinite series of powers of x , i.e. like a polynomial. Often the series can be truncated at very small orders, if we can neglect higher order terms.

When a physics quantity changes proportionally to itself and to another quantity like $dy = \alpha y dx$, the resulting model is always exponential, i.e. $y(x) = y(0) \exp(\alpha x)$. This model describes a long list of physics phenomena and is very important you dominate it.

Linearisation consists in transforming an equation such that it appears as linear in the transformed independent variable. It usually helps in finding the parameters of a model from data.

Arduino

A variable, in a programming language, is a place where to store a value.

Data are represented in computer's memory as sequences of bits. To interpret them correctly we need to know their type.

A **for** loop is used to repeat one or more statements.

The content of a numeric variable can be altered using post-, pre- and auto-increment or decrement operators. Post- (e.g. `i++`) and pre- (e.g. `++i`) increment operators increase by one the content of a variable. The increment is done after or prior to use it in the current statement, depending on the position of the operator. The auto-increment operator increases the variable on the left by the amount on the right (e.g. `S += k`).

Iterative structures can be realised using **while**, too. The latter is preferred when we cannot predict the number of times the loop is run.

In programming languages we can define our own functions: collections of statements whose behaviour can depend on parameters.

Statistics

The variance of a population can be estimated as $\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$.

Chapter 6

Free fall and accelerations

In this chapter we make our first measurement whose result can be compared to another experiment. Comparing results with those of other experiments is of paramount importance in physics. As we believe that physics is universal, i.e. its laws are valid at any time and at any place, we expect that results taken by different people in different experiments should be consistent with each other. That does not mean that they must be the same in mathematical sense.

1. Setting up the experiment

Free fall consists in the motion of a body subject solely to gravity. In this experiment we want to study how objects fall and characterise their motion, i.e. determine their **state**. The latter consists in finding all the information needed to predict its state at time t , knowing the state at a different time t_0 .

Bodies subject only to gravity are said to fall freely.

To make the experiment ~~we must drop a body~~ and we should characterise its motion, measuring all the ~~possible~~ **physics** quantities affecting it. The shape and the length of the path followed ~~during the fall~~ for example, can be interesting quantities to measure. The ~~time needed to~~ follow the path, too. The shape of the object may be of ~~some interest~~ however, it is not a good idea to start experimenting with objects ~~whose shape is complicated enough to influence the motion~~. It is well known, ~~for example~~, that the path followed by a sheet of paper freely falling is rather complicated and it is then difficult to study it. We should get rid of such kind of effects. A possibility is to choose an object that has *no shape*, i.e. a point-like object. Manifestly, such an object does not exist. However, we can easily find objects whose shape and size are such that the details of them do not affect their free fall.

The mass of the object may have an influence, too. Light objects may ~~fall~~ differently from heavy ones. A simple qualitative observation shows that ~~heavy~~ **experiment**, many enough objects fall almost the same way.

aspects can influence the result. We need to keep them negligible or, at least, constant.

As a result, to make a meaningful experiment, we must choose a body ~~heavy~~ enough whose shape and size do not affect the motion. This way, we get rid of quantities from which the state of the object may depend and are difficult to

control. In every experiment, we should always try to make the results depend on one or at most very few variables.

With these choices, we describe, in fact, the motion of a pointlike particle of mass m , subject only to gravity. For simplicity, better to start it from a stationary position. It is easy to show that the trajectory described by such a particle can be represented by a vertical segment of length h .

In principle there are other characteristics identifying the state of the falling object: its temperature, its color, the material of which is made, etc.. However, many of these characteristics either do not affect the motion or are meaningless in the context. For example, a pointlike particle cannot have a color, since it has no surface. On the other hand, the color of the real object we are going to throw is not found to influence its motion. As a result, color is not part of the state of the object and it is useless to collect its value, as well as its temperature, its nature, etc..

In summary, an experiment consists in letting the object fall freely from a given height h and observing that the time needed to fall t depends on h , i.e. $t = f(h)$. Our goal is to find f .

2. Measuring times

A smartphone can be used as a stopwatch. To measure times we need a stopwatch. Every smartphone comes with a stopwatch application and we can use it. The application gives the time elapsed between two successive taps on the display.

A first attempt could be the following: keeping in one hand an object at a certain predefined height h , we tap on the smartphone while leaving the object falling and we tap again on it when we hear the object hitting the floor. Though it may work, there are a number of inconveniences in doing the experiment this way.

First of all, it is not so obvious that the time at which the object starts falling is exactly the same in which you tap on the display of the smartphone to start measuring time. Moreover, your reaction time when you hear the noise of the object hitting the floor is not negligible and it affects the measurement. The time needed to reach the floor, moreover, is indeed short.

As a result, the measurement is affected by large statistical fluctuations. Repeating the measurement several times make it clear.

Comparing the results of the same measurement made by different experimenters you can easily see that there are sometimes significant differences in the measurements. The difference may depend on the smartphones or on the person taking the measurement, whose reaction time could be more or less long.

We can easily get rid of the latter effect assigning the task to activate the stopwatch to some automatic device. Here we describe two possible solutions.

3. Photogates

A **photogate** consists of a light detector illuminated by some ~~photogate of light~~. Identifying precisely the time at which something breaks the ~~light beam passing~~ start between the source and the detector, provides a measurement ~~of the time~~ to a which the object coordinate along its trajectory coincides with ~~that of the gate~~ easy. It is easy to make a photogate with Arduino. We need a light intensity detector ~~to make a photogate with Arduino~~.

and a source of light. Monitoring continuously the illuminance, we can tell when it drops under a certain threshold, as in

```
#define THRESHOLD 15

void setup() {
    Serial.begin(9600);
}

void loop() {
    while (analogRead(A0) > THRESHOLD) {
        // do nothing
    }
    unsigned long t = micros();
    Serial.println(t);
}
```

In Arduino Here THRESHOLD is a **constant** whose value is 15. Entering the loop() nothing happens until analogRead(A0) returns a number lower than 15. When this happens, we assume that it is because something interrupted the light beam, and we measure the current time using micros(). The latter returns the time elapsed since the beginning of the execution of the sketch in microseconds. The number is represented as an **unsigned long**, i.e. a positive integer of 32 bits.

We can measure the time needed to pass by points *A* and *B* with two photogates in *A* and *B* measuring the difference in their concealment times. For example, connecting the output leads of two analog light sensors to A0 and A1 pins, respectively, the time needed to go from sensor 1 to sensor 2 can be measured with the following loop() function.

```
void loop() {
    while (analogRead(A0) > THRESHOLD) {
        // do nothing
    }
```

```

unsigned long t1 = micros();
while (analogRead(A1) > THRESHOLD) {
    // do nothing
}
unsigned long t2 = micros();
Serial.println(t2-t1);
}

```

Here we first wait until the first sensor detects the passage of something between itself and a light source, and put the time at which it happens in the variable called t1. Then, we wait for the same event happening on sensor 2. The time elapsed between the two events is computed as the difference t2–t1 and shown on the screen when the serial monitor is open.

4. Measuring time with Arduino

~~Microprocessors' operations happen at fixed rate regulated by a clock: a device operates at producing pulses at regular intervals. The Arduino UNO clock runs at 16 MHz, whose pace is another~~ measure time just counting the number of clock pulses.

~~a clock, Arduino's~~ Few functions are dedicated to time management in the Arduino programming language.
~~clock runs at 16 MHz.~~

delay() and **delayMicroseconds()** pauses the execution of the program for the given time.

The execution of a sketch can be paused using **delay()** or **delayMicroseconds()** functions. The first pauses the program for a number of milliseconds passed as an argument, e.g., **delay(1500)** suspends the execution of the sketch for 1.5 s. The latter works exactly the same way, but it suspends the execution for the number of microseconds passed as its argument: **delayMicroseconds(10)** waits for 10 μ s.

Using its clock, The time elapsed since the beginning of the sketch can be obtained in milliseconds using **millis()** and in microseconds with **micros()**. Both returns an unsigned long number. **long** is a type representing a 32-bits long integer.

the beginning of the computers' memory, negative numbers are represented in the so-called **two's complement**, exploiting the fact that numbers are always represented using a fixed number of digits. The idea is the following. Imagine a tailor's measuring tape arranged into a ring with its end corresponding to 0 coinciding with the mark 100 and suppose you want to write all the numbers with at most two digits. If you need just positive numbers, you can clearly represent those from 0 to 99. However, if you want to work with negative numbers, too, you can imagine the numbers to the right of 0 to be positive and those at its left (99, 98, 97, ...) to be negative, such that 99 corresponds to -1, 98 to -2 and so on.

To write a negative number this way is very simple: find the **complement** of each digit, i.e. the digit that summed to the previous gives the last digit in the complement of each digit, then add 1.

chosen base, then add 1 to the result. For example, -3 , with two digits, is -03 and can be written finding the complement of 3 in base 10, i.e. 6, and the one of 0, i.e. 9. The result is 96. Adding 1 to the latter gives 97 that in fact is the third number on the left with respect to 0. Similarly -23 can be represented as $76 + 1 = 77$.

The rule comes from the observation that with n digits you can represent numbers from 0 to $b^n - 1$, where b is the base of the numeral system, and that b^2 has $n + 1$ digits. Given a number m we can always find a number k such that

$$m + k = b^n \quad (6.1)$$

If we cannot represent more than n digits, the extra digit in b^2 drops and the above equation is equivalent to

$$m + k = 0 \quad (6.2)$$

being b^n always written as 1 followed by n zeros, whatever the base. Finding k is easy:

$$k = b^n - m = (b^n - 1) + 1 - m. \quad (6.3)$$

In the above example, $b = 10$ and $n = 2$, such that $b^n - 1 = 10^2 - 1 = 99$. Subtracting from it the number of which we want to know the complement ($m = 3$) leads to 96. Finally, adding 1 we obtain 97.

In two's complement the base of the numeral system is 2. Finding the complement of a number is super easy: just invert all the digits, then add 1 to the result. For example, in a 4-bits representation, 6 is represented as 0110. The complement of each digit gives 1001, while adding 1 gives 1010, corresponding to $8 + 2 = 10$ if interpreted as a positive number. Interpreted as a two's complement one, it has the property such that

$$0110 + 1010 = 10000 \quad (6.4)$$

where the leading 1 does not fit in the 4 bits and is ignored, making the sum of 0110 and 1010 equate to 0000, i.e. 0110 is the opposite of 1010.

Using this technique all negative numbers begin with 1 and there is only one representation for zero (0000). The highest positive number is 0111, corresponding to 7, while the lowest possible number is 1000 corresponding to -8 . In general, using the b 's complement technique with n digits, we can represent numbers ranging from $-\frac{b^n}{2}$ to $+\frac{b^n}{2} - 1$.

Variables declared as `long` can then contain values between $-\frac{2^{32}}{2} = -2\,147\,483\,648$ to $\frac{2^{32}}{2} = +2\,147\,483\,647$.

Whatever the base b of the numeral system, b^n is always written as 1, the first non zero digit, followed by n zeros.

Using the `unsigned` **qualifier**, we get rid of negative numbers and the variable can represent numbers from 0 to $2^{32} - 1 = 4\,294\,967\,295$. For the elapsed time this type is appropriate and allows us to represent it up to more than four billions microseconds, corresponding to almost 72 hours.

Using this number to represents milliseconds, the maximum representable time can be extended up to 50 days.

5. An acoustic stopwatch

PHYPHOX provides [several types of stopwatches](#). One of them is the **acoustic stopwatch** exploiting the microphone of the smartphone as a sound detector. The stopwatch starts when a sound of enough intensity is detected and stops when a second sound exceeding the given threshold is recorded after a minimum delay τ , returning the time $t > \tau$ between the two events.



We can use the acoustic stopwatch as follows. We tie an inflated balloon to a relatively heavy object, such as a small bag of bolts, and suspend the system to a height h using some support. For example, a sieve to which the mesh has been removed. The smartphone is placed on the floor, close to the vertical from the bolts (but far enough away not to risk the bolts falling on it). Popping the balloon with a needle starts the stopwatch that, however, often stops before the falling object reach the floor. The reason being that echos in the room will reach the microphone before the bolts finish falling off. In order to get rid of echos we must measure the echo time as above and use it as τ . For example, if the time we read on the smartphone is 0.086 s, we use $\tau = 0.1$ s as minimum the echos, setting delay and put the bolts at a height such that they take more than τ to fall. A proper rough estimation of the minimum height can be done multiplying τ by five. In our example, $h_{min} \approx 5 \times 0.1 = 0.5$ m.

Repeating the experiment with this setup gives the right falling time, usually with a resolution of 1 ms. As usual, we should repeat the measurement several times to estimate the uncertainty. However, in this case, we can avoid repeating the measurements in the same conditions, taking instead them at different heights h_i for which we obtain a measurement of t_i . The height can be measured with a self-retracting metal tape measure with an uncertainty of ± 0.3 mm.

Our data looks like in Fig. 6.1. Data seems to be aligned, but drawing the interpolating lines we clearly see that $t > 0$ when $h = 0$. There is no doubt that the time needed to fall from $h = 0$ must be null, hence this model is clearly wrong. The correct model can be recovered from the definitions of kinematic

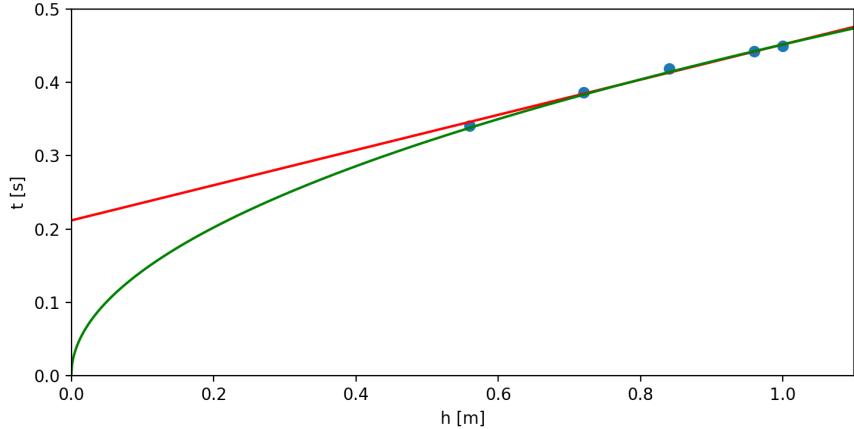


Figure 6.1: The time t needed for a body to fall from a height h . In red a linear interpolation is shown. The model for constant acceleration is in green.

variables. Velocity, for example, is defined as

$$v = \lim_{\Delta t \rightarrow 0} \frac{\Delta h}{\Delta t} = \frac{dh}{dt}, \quad (6.5)$$

then, $dt = \frac{dh}{v}$ and integrating both members,

$$t = \frac{h}{v} \quad (6.6)$$

if v is constant. In this case t grows linearly with h and manifestly this is not the case: any line passing by $O = (0, 0)$ cannot pass through (in the vicinity of) all the data points. It means that v is not constant, and the acceleration

$$a = \lim_{\Delta t \rightarrow 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt} \neq 0. \quad (6.7)$$

If a is constant, $v(t) = v(0) + at$. Since $dh = v dt$, integrating both members, we obtain

$$h = \int_0^t v(t) dt = v(0) \int_0^t dt + a \int_0^t t dt = v(0)t + \frac{1}{2}at^2. \quad (6.8)$$

In our case $v(0) = 0$, then

$$t = \sqrt{\frac{2h}{a}}. \quad (6.9)$$

It is easy to
 v is not con-
data. If $v \neq$
acceleration

The latter is shown in Fig 6.1 in green. In fact, the curve interpolate well the data are well described by the hypothesis $a = \text{const}$. There is no reason, then, to suppose $a \neq \text{const}$.

$$\sqrt{2\frac{h_0 + y}{a}} \simeq \sqrt{\frac{2h_0}{a}} + \frac{y}{2\sqrt{\frac{2h_0}{a}}}, \quad (6.10)$$

i.e. the falling time grows almost linearly with y . To estimate a we can linearise the relationship between t and h squaring the equation to obtain

$$t^2 = \frac{2h}{a}. \quad (6.11)$$

Linearisation is always a good technique to obtain the parameters of a curve. Slopes and intercepts are easy to be determined.

The slope of the graph of t^2 as a function of h is $m = \frac{2}{a}$ and can be obtained computing all the possible ratios $\frac{\Delta t^2}{\Delta h}$. The result is

$$m = 0.19 \pm 0.02 \frac{\text{s}^2}{\text{m}}. \quad (6.12)$$

From this number we can obtain the acceleration as

$$a = \frac{2}{m} = \frac{2}{0.19} \simeq 10.5 \frac{\text{m}}{\text{s}^2}. \quad (6.13)$$

The uncertainty σ_a what about its uncertainty? To estimate it, let's analyse what happens to a on x propagates when m fluctuate by σ_m .

to $f(x)$ such that the

uncertainty σ_f on the latter is given by

$f'(x)\sigma_x$

$$a' = \frac{2}{m'} = \frac{2}{m + \sigma_m} \simeq \frac{2}{m} - \frac{2}{m^2}\sigma_m. \quad (6.14)$$

The variance of a is the square of its deviation, i.e.

$$\sigma_a^2 = (a' - a)^2 = \frac{4}{m^4}\sigma_m^2, \quad (6.15)$$

then

$$\sigma_a = \frac{2\sigma_m}{m^2}. \quad (6.16)$$

We always write our result keeping one significant digit in the uncertainty and writing the central value accordingly.

In this expression we recognise the derivative of the expression for a . This is a general rule, in fact, as you may easily verify in numerous cases. The uncertainty on x propagates to a function $f(x)$ of it as $\sigma_f \simeq \frac{df}{dx}\sigma_x$. In our example, $\sigma_a = 0.2 \frac{\text{m}}{\text{s}^2}$ and our final result is

$$a = 10.5 \pm 0.2 \frac{\text{m}}{\text{s}^2}. \quad (6.17)$$

According to the *Bureau Gravimetrique International* (BGI, <http://bgi.obs-mip.fr/>) the gravity acceleration in Roma, where we made our measurements, is $g = 9.803 \frac{\text{m}}{\text{s}^2}$. Our data differ from this measurement by often available on Internet.

$$\Delta = \frac{10.5 - 9.803}{9.803} \simeq 0.07, \quad (6.18)$$

i.e. by 7 %. Not bad considering the quality of the tools chosen. Despite the apparent agreement, considering the uncertainties, data do not agree. In fact, the difference between the two numbers $10.5 - 9.803 \simeq 0.7 \frac{\text{m}}{\text{s}}$ is more than three standard deviations and there is little probability that this is due to some statistical fluctuation.

A discrepancy of three standard deviations is large, but not so large to be ascribed to new phenomena. A conventional threshold for claiming a discovery is five sigmas. Assuming, then, that the BGI measurement is known with much more precision than ours, we need to find the reason that leads to the **systematic** shift of our data.

The existence of some systematic effect is signalled by the fact that the intercept of the line interpolating data of t^2 vs h is not null, while it is expected to be so. In fact, it is $q = 0.013 \pm 0.002 \frac{\text{s}^2}{\text{m}}$. The consequence of $q \neq 0$ is that $t(0) = \sqrt{q} \simeq 0.1 \text{ s}$. A large effect, indeed, if compared to the 0.3–0.4 s of our data. Moreover, there are other effects we have not considered. The sound from the explosion of the balloon takes $t_s = \frac{h}{c}$, where $c \simeq 340 \text{ m/s}$ is the speed of sound, to reach the phone. As a consequence the effective travel time of the weight is longer than measured by t_s , even if $t_s \simeq 0.002 - 0.003$ is small compared to t_i . To correct for this effect we can find the slope of

Conventionally, a discovery is claimed when the discrepancy between a measurement and the theoretical prediction is larger than five standard deviations. This is a general rule that must be taken *cum grano salis*. It depends on how solid the theory is and how credible the measurements are.

$$t_{corr}^2 = \left(t + \frac{x}{c} \right)^2 \quad (6.19)$$

as a function of x , finding $m = 0.20 \pm 0.02 \frac{\text{s}^2}{\text{m}}$ and $q = 0.007 \pm 0.002 \frac{\text{s}^2}{\text{m}}$, still significantly different from zero. The corresponding value for a is

$$a = 10 \pm 1 \frac{\text{m}}{\text{s}^2}. \quad (6.20)$$

Now a is compatible with $g = 9.803 \frac{\text{m}}{\text{s}^2}$, because $a - g \simeq 0.2 \frac{\text{m}}{\text{s}^2}$, quite less than the uncertainty with which a is known. The fact that $q \neq 0$ can be ascribed to various causes. The smartphone takes a certain time to start and stop counting. If this time is not taken into account properly, it adds to the measured time and results in $q > 0$.

In these cases it is worth designing new experiments, in which systematic uncertainties are missing or, at least, different. New experiments may imply a

The signs in a be c and des expe just fully

completely different technology and methods, or can be made in different conditions. For example, repeating the above experiment for $h \gg 1$ m makes the possible offset t_s negligible.

6. Uncertainty propagation

Any uncertainty in the knowledge of the value of a quantity x affects, manifestly, each function $f(x)$ of it. The way in which the error σ_x propagate to $f(x)$ follows general rules.

In fact, if $f(x)$ is a function of x , any change from x to $x + \sigma_x$ results in a change of $f(x)$ to $f(x + \sigma_x)$. For σ_x small enough,

$$f(x + \sigma_x) \simeq f(x) + f'(x)\sigma_x. \quad (6.21)$$

The uncertainty on $f(x)$ is then

$$\sigma_f = |f(x + \sigma_x) - f(x)| \simeq |f'(x)\sigma_x|. \quad (6.22)$$

The absolute value operator is uncomfortable and we prefer to write the variance of $f(x)$,

$$\sigma_f^2 \simeq f'^2(x)\sigma_x^2. \quad (6.23)$$

If f is a function of more than one variable, e.g., $f = f(x, y, z)$, its uncertainty depends on the uncertainty on each quantity. Supposing that only x can fluctuate, we may write

$$\sigma_f^2 \simeq \left(\frac{\partial f}{\partial x} \right)^2 \sigma_x^2, \quad (6.24)$$

The variance where $\frac{\partial f}{\partial x}$ represents the partial derivative of f with respect to x , i.e., considering function of many y and z as constants. Of course, other variables behave the same and since variables is the sum of the effects sum up we can write partial variances. As a

result we say that uncertainties sum in quadrature.

$$\sigma_f^2 \simeq \sum_{i=1}^n \left(\frac{\partial f}{\partial x_i} \right)^2 \sigma_i^2 \quad (6.25)$$

For example, if the measurements of the three components of a vector are $x \pm \sigma_x$, $y \pm \sigma_y$ and $z \pm \sigma_z$, the magnitude of the vector is given by $v = \sqrt{x^2 + y^2 + z^2}$ and its uncertainty is computed as follows. First, compute the partial derivatives of v :

$$\frac{\partial v}{\partial x} = \frac{x}{v} \quad \frac{\partial v}{\partial y} = \frac{y}{v} \quad \frac{\partial v}{\partial z} = \frac{z}{v}. \quad (6.26)$$

Then multiply each derivative by the corresponding uncertainty, square the result and sum all the terms together to obtain

$$\sigma_f^2 = \left(\frac{x}{v}\sigma_x\right)^2 + \left(\frac{y}{v}\sigma_y\right)^2 + \left(\frac{z}{v}\sigma_z\right)^2. \quad (6.27)$$

Once we established that a given object falls with acceleration equal to \mathbf{g} , we can start experimenting with objects having different masses, to investigate if acceleration depends on the mass and, if so, how. We can also investigate if the acceleration depends on the shape or on the density of the falling object, keeping the mass fixed. In these cases we may have to take into account that the time needed to touch the floor may depend on the object orientation, so the experiment can be quite difficult, indeed. Eventually we find that, at least for large enough masses, objects fall with the same acceleration, irrespective of mass, shape, density, etc.. Very light objects (a small leaf) or objects not so light, but with large shapes (a cardboard), as well as low density ones (inflated balloons) fall differently. Further studies reveal that such a behaviour is due to the fact that these objects are not, in fact, freely falling: they are immersed in a fluid (the air) that produces forces on it opposing to gravity. Removing the fluid (e.g. repeating the experiments in vacuum) makes it clear that even those objects fall with the same acceleration of heavy ones. The latter are apparently not subjected to the effects of air: in fact they are, but the effects on them are negligible and cannot be spotted by our measurements, due to limited precision.

On YOUTUBE it is possible to look at interesting free falling experiments, like the one made in the NASA Space Power Facility in Ohio owning the world's biggest vacuum chamber or the one made by Commander David Scott during the Apollo 15 mission, who let a hammer and a feather fall on the moon.

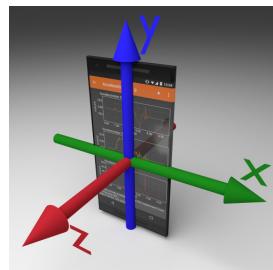
7. Measuring accelerations

It is instructive to work out few examples using functions of real data. We can [Collecting data from](#) for example, collect 30 s of accelerometer data from PHYPHOX. If the smart [the accelerometer of a](#) phone is at rest, the acceleration components must be those of the gravitational [smartphone using](#) acceleration, whose magnitude is 9.8 m/s^2 .

[PHYPHOX is a good exercise about uncertainty propagation.](#)

Smartphones use accelerometers to tell their orientation. They measure the three components of the acceleration in a reference frame centred on the phone, where the x axis is oriented along the width of the device, the y axis along its height and the z axis is perpendicular to the screen, as in the figure on the right.

Comparing the size of the x and y components, one can tell if the phone is kept in portrait or in landscape



Smartphones use accelerometers to tell their orientation.

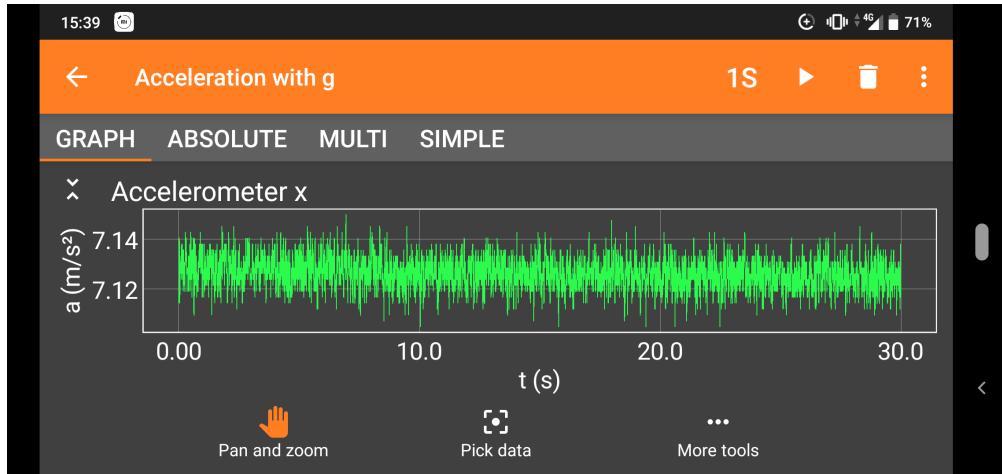


Figure 6.2: Data collected using the accelerometer of a smartphone, when it was kept almost in its landscape position.

mode. Raw data from the accelerometer are collected by PHYPHOX allowing us to use these data to do some physics.

To do the measurements we kept the smartphone in landscape mode and started a timed run with a delay of 1 s in order to avoid to register vibrations due to the tap on the start button. Fig. 6.2 shows a screenshot of the phone at the end of the data taking.

During the measurements, the smartphone was at rest, then the measured acceleration is $\mathbf{g} = (g_x, g_y, g_z) = \text{const.}$ Fig. 6.3 shows the histogram of the values collected for each component and for the vector magnitude.

We can notice that data distribute in such a way the histograms resemble Data distribute such a bell. The distribution is more or less symmetric around the mean value, that they appear to which data tend to accumulate. The width of the distribution represents denser around certain the uncertainty: the larger the latter, the wider the distribution. Computing values. averages and standard deviations of the components of \mathbf{g} results in the following:

$$\begin{aligned} g_x &= 7.126 \pm 0.006 \text{ ms}^{-1} \\ g_y &= 0.001 \pm 0.006 \text{ ms}^{-1} \\ g_z &= 6.829 \pm 0.007 \text{ ms}^{-1}. \end{aligned} \quad (6.28)$$

It is worth noting that, while $g_y \approx 0$, g_x and g_z are non null and positive. The reason for that is that during the measurements we kept the phone in landscape orientation leaning against our laptop display. The positive direction of the x axis is, respectively, to the right side of the phone and exiting from its display. The acceleration measured by the device has then the opposite direction with respect to the gravitational acceleration. Next section explains why.

The orange curve superimposed to the histograms is the so called **gaussian** well the shape of data distribution.

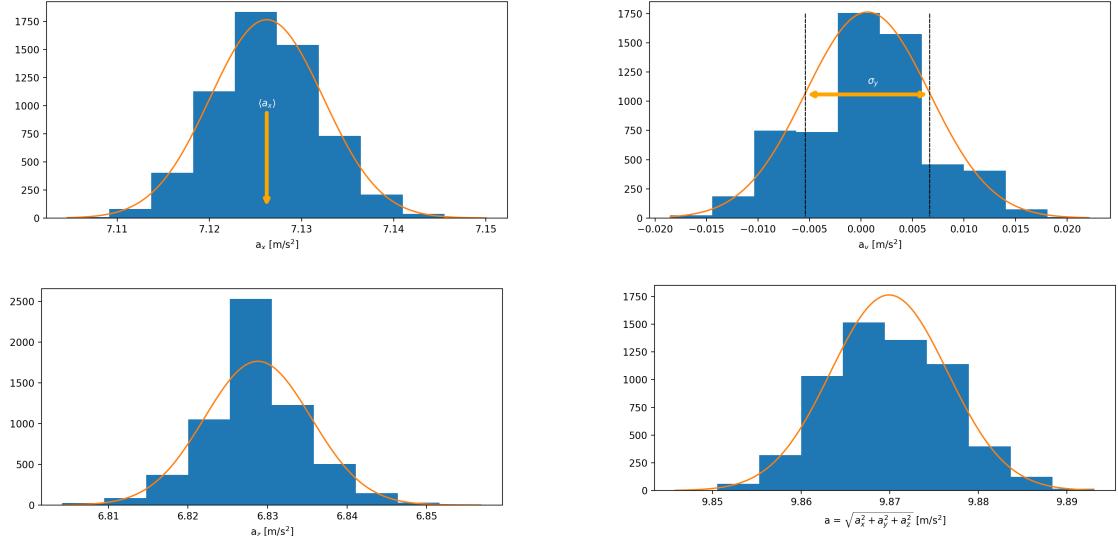


Figure 6.3: Histograms of the values of the three components of the acceleration vector and of its magnitude with the phone at rest. The mean value and the standard deviation are shown, respectively, on a_x and a_y .

curve, whose expression is

$$G(x) = \frac{C}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \quad (6.29)$$

A gaussian has its maximum at $x = \mu$ and its width is determined by the σ parameter. It can be shown that the curve width at half maximum (FWHM: Full Width at Half Maximum) is about $2\sqrt{2\ln 2}\sigma \simeq 2.4\sigma$. C is a **normalization** coefficient, representing the height of the curve at its maximum. In fact, for $x = \mu$, the exponential is 1 and $G(\mu) = \frac{C}{\sqrt{2\pi}\sigma}$, hence

$$C = \sqrt{2\pi}\sigma G(\mu). \quad (6.30)$$

The factor $\sqrt{2\pi}\sigma$ makes the area under the curve equal to

$$\int_{-\infty}^{+\infty} G(x)dx = C. \quad (6.31)$$

The magnitude of the average acceleration vector is then $\sqrt{g_x^2 + g_y^2 + g_z^2} \simeq 9.870 \text{ ms}^{-1}$. The predicted error on this values is given by the above formula (6.27) and is $\sigma_g = 0.006 \text{ ms}^{-1}$. From the data histogrammed in Fig. 6.3, we can easily see that the standard deviation is in fact $\sigma_v^{exp} = 0.006 \text{ ms}^{-1}$.

The above considerations are very general, indeed. Histograms of statistically independent measurements distribute often as gaussians, whose mean coincides

with the average of the measurements and whose width depends on their standard deviation. The uncertainty of data propagates in their functions such that their variance sum according to the uncertainty propagation formula (6.25).

8. MEMS accelerometers

A physicist ~~mis~~knowing your instruments is among the most important rules in experimental always know ~~how~~ physics. Never use an instrument without understanding its working principle. his/her instrument ~~Smartphones~~ Smartphones are no exceptions.

~~work~~ It is well known that being in a non-inertial, i.e. accelerated, reference frame, the Newton's second Law is not valid, unless we add **fictitious forces** to those acting as a result ~~of some interaction~~ ~~in some~~ ~~frame~~:

accelerometers exploits

$$\text{the second Newton's Law in non-inertial} \quad \mathbf{a} = \frac{\sum_i \mathbf{F}_i - \mathbf{F}_{fict}}{m}. \quad (6.32)$$

In a system where $\sum_i \mathbf{F}_i = 0$, then $\mathbf{F}_{fict} = -m\mathbf{a}$. For example, consider a smartphone at rest on the seat of a merry-go-round at rest, too. The smartphone is subject to the gravitational force \mathbf{F}_g , directed towards the earth center, and to the normal force \mathbf{F}_N , applied by the seat, directed in the opposite direction such that they cancel each other: $\mathbf{F}_g + \mathbf{F}_N = 0$. Its acceleration is then zero.

Letting the system rotate, the smartphone keeps at rest only until friction is large enough. At a certain point we see the smartphone moving radially and eventually fall. What happens is the following.

A smartphone on a ~~merry-go-round~~ When the merry-go-round starts rotating, it applies, thanks to static friction ~~the~~ acting on the phone, a centripetal force to the latter that, as a consequence, follows a circular path ~~moves along a circular path~~. At a certain point, static friction eventually vanish because of ~~the~~ and the phone is no more subject to any force. According to the Newton's **centripetal** second Law, it moves with constant velocity and, while the seat keeps rotating, acceleration provided ~~it slides along a line tangent to its original path until it reaches the border of~~ by static friction. ~~When~~ the seat and fall. The above description is fully coherent with Newtonian's it vanish, the mechanics, as it is done within an inertial reference frame.

smartphone starts moving according to the first Newton's Law. Seen from a reference frame attached to the rotating system, the smartphone motion can be described differently. Initially, the gravitational and normal forces cancel each other and the smartphone is at rest on the seat. At a certain point it starts sliding along a radius until it falls from the merry-go-round.

Manifestly, for the non-inertial observer, an acceleration had to develop and, according to Newton's Law, there must be a force to produce it. However, there is no such force, or, better, there are no interactions between the phone and other systems such that the phone is *attracted* towards the border of the carousel. In fact, if friction was acting at the beginning of the rotation while observing the system from outside, it must keep acting even for who is observing

the phone from inside the rotating frame. On the contrary, while there are no forces acting on the phone when it is sliding along the tangent, there must be no force for who is looking at it standing on the rotating frame.

The only way to make the two descriptions coherent is to add fictitious forces in non-inertial reference frames. Fictitious forces are just a mathematical trick to make Newton's laws valid in non-inertial frames. Since the systems experience an acceleration in those systems, we can pretend there are forces producing them, according to Newton's Law.

The term "fictitious" is somewhat misleading because, in fact, according to our experience, those forces are not at all fictitious. They are among the most real and frequently experienced forces. Even more than gravity. In fact, we do not feel gravity as a force, since our senses are immersed in the gravitational field, too. The feeling of falling does not coincide with the feeling of being drawn down. On the contrary, when we take a turn in the car, we clearly feel a force pushing us towards the outside of the road.

In fact, fictitious forces are as real as gravity, but they are not the result of an interaction. In other words, there is no source for the **centrifugal** force, as it is called, while there are sources of the gravitational, elastic, electrostatic and magnetic forces. We should make a difference between a force and an interaction that did not exist at Newton's times. Interactions produce forces, as well as forces develop in non-inertial frames. It is worth noting that the (local) equivalence between an accelerated frame and an inertial frame immersed in a gravitational field is the ground of one of the most advanced physics theories like **general relativity**.

According to the above considerations, fictitious forces can be written as

$$\mathbf{F}_{fict} = -m\mathbf{a}. \quad (6.33)$$

where \mathbf{a} is the acceleration of the reference frame. Note, too, that in this case the Newton's second law takes the form $\mathbf{F}_{fict} = m\mathbf{a}$ and not $\mathbf{a} = -\frac{\mathbf{F}_{fict}}{m}$. In the above formula, in fact \mathbf{F}_{fict} is defined as such and we state that the force is in fact equal to the product of m and \mathbf{a} .

Such a long preamble is to introduce the way in which accelerations are measured by an instrument called **accelerometer**. Suppose we attach a mass m to a spring and suspend this system, such that the spring stretches because the mass pulls it towards the floor with a force $F = mg$. If we move the suspension point abruptly upwards, the spring initially stretches even more, as if it is subject to a force whose intensity is $F_{fict} = ma$. Moreover, if we accelerate the suspension point downwards, the spring shortens as if a force of intensity $F_{fict} = ma$ acts on it. The force can easily be measured: a spring is, in fact, a dynamometer, and $\mathbf{F} = -k\Delta\mathbf{x}$, where k is the elastic constant of the spring and $\Delta\mathbf{x}$ the variation of the position of one of its ends with respect to the other. A measurement of the

When a physicist says that forces measured in non-inertial frames are fictitious, he/she intend that they are not the result of some interaction. There are no sources of fictitious forces, while there are sources for the gravitational, electrostatic and magnetic forces.

Measuring accelerations is equivalent to measure forces.

length Δx can then be turned into a measurement of the intensity $F = k\Delta x$ that, in turn, can be casted into an acceleration as $a = k \frac{\Delta x}{m}$.

A system like this measures the acceleration in the vertical direction. An horizontal spring measures the acceleration along the direction of its axis. With three orthogonal springs we can then measure the three components of the acceleration vector.

Manifestly, when this system is at rest, the springs are stretched in the vertical direction by $\Delta x = \frac{ma}{k}$ and, in this condition, we measure $\mathbf{a} = \mathbf{g}$. On the other hand, if the system is falling, it finds itself in a non-inertial reference frame moving with acceleration \mathbf{g} , hence the experienced acceleration is $\mathbf{a} - \mathbf{g} = 0$.

The way in which smartphones' accelerometers work is exactly this. Remember that the system described above is a **model**: it represents a mass attached to something exhibiting an elastic behaviour. The fact that the mass is represented as pointlike is an oversimplification of the mathematical model. It just means that the details of the shape of the mass are irrelevant for its dynamics. The spring is often represented as a zig-zag segment —~~—, inspired by the shape of a kind of spring. However, as shown on the side of the page, springs exist in a variety of shapes. Its shape is irrelevant for the dynamics. What is important is that its length changes proportionally to the applied force.

Smartphones' accelerometers are **microelectromechanical systems** (MEMS): microscopic mechanical devices obtained assembling tiny components whose size can be as short as few tens of micrometers. In these systems, three microscopic masses are suspended to three microscopic springs such that they can freely move along as many mutually orthogonal directions parallel to the sides of the phone.

The voltage across a The elongation Δx is measured exploiting the variation of some electrical property of the device as it changes its shape. Often, for example, the MEMS consists of two parallel conductor plates i of a pair of metallic plates, one of which is attached to a spring. Such a system proportional to θ of conductors works as a **capacitor**: a system to store electrical energy. As soon as the ratio between θ as the distance between the plates or the area of a surface exposed to the other distance d between change, an electrical current flows from or to the capacitor. Measuring such plates and the area S a current can be casted into a measurement of the acceleration. Because of in common between the above considerations, smartphones' accelerometers does not return 0 when them: $V \propto \frac{d}{S}$ they are at rest, but \mathbf{g} .

Fig. 6.4 shows a microphotograph of such a device (namely the LIS331DLH by ST-microelectronics).

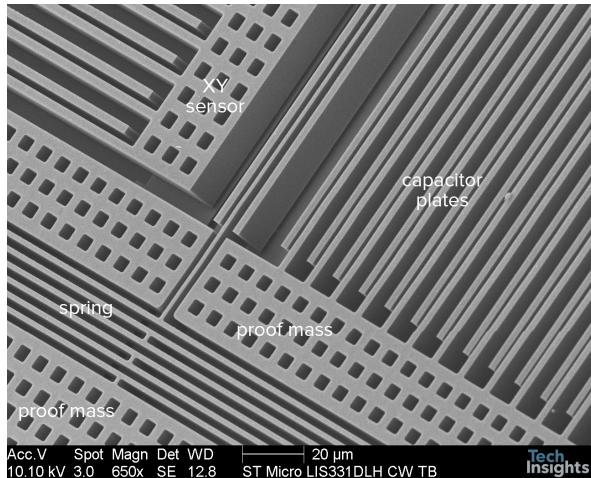


Figure 6.4: Microphotograph of a MEMS (left). The spring is made of long structures interconnected by tiny joints, resembling the springs of a bed (visible in the lower left corner of the figure). The mass resembles a brick and is connected to a set of parallel plates that moves together with the mass with respect to other plates fixed on the substrate. Courtesy of TechInsights Inc..

9. Annotating graphs

annotate()
rates graphs with
text and graphical
elements.

It can be useful, when presenting experimental data, to decorate graphs with graphical elements like arrows or text, as shown in the above examples.

This is done, in Python, using the `annotate()` method of `pyplot`. The `annotate()` method takes at least two parameters: a string and a pair of coordinates. A statement like

```
plt.annotate('$\langle a_x \rangle$',
            xy=(mu, sigma))
```

prints the string $\langle a_x \rangle$ at coordinates `mu` and `sigma`. Coordinates are referred to those used in the plot, by default. The behaviour is changed with the optional parameter `xycoords='...'`.

In the above example the text to be printed is given in `LATEX` and the string `Acute parenthesis` are `\langle a_x \rangle`. `\langle` and `\rangle` represented in `LATEX` as sent, respectively, the left and right acute parenthesis, while the characters inside `\langle` braces after the underscore are rendered as a subscript. The double backslash `\` is a meta-character representing an `escape` sequence. A character following an escape character is not interpreted as such, but as a non printable character. For example, to represent the newline character we use `\n`, where `n` is interpreted as a newline character, rather than a plain `n` character, being preceded by the backslash. Since the latter is a meta-character, in order to represent it, we need a meta-character, too. This is done writing `\\"`. A full example follows.

```

plt.annotate('$\sigma_y$', xy=(muy, C*0.65),
            color='white')
plt.annotate('', xy=(muy-sigmay, C*0.6),
             xytext=(muy+sigmay, C*0.6),
             arrowprops=dict(ec='orange',
                             arrowstyle='<|-|>', lw=4))

```

This code writes a white L^AT_EX symbol σ at coordinates represented by the variables `muy` and `C` multiplied by 0.65. Then, it makes a horizontal orange arrow with heads at both extremes (set by the `arrowstyle` option) and a line width of 4 points. `xytext` and `xy` are the starting and ending coordinates of the arrow, while `ec` stands for *edge color*, and is used to assign the color to the arrow. Note that the corresponding string in `annotate()` is empty, such that only the arrow is shown.

In such a command `arrowprops` is called a *dictionary* (hence, `dict(...)`), i.e. a list of key–value pairs.

10. Instruments characteristics

With the `Experiment` made with the smartphone at rest we clearly see that taking just one measurement is not enough and we must repeat the measurement several times, then compute the average to obtain a reasonably reliable value. We found that the gravitational acceleration is

difference in units of

uncertainty.

$$g = 9.870 \pm 0.006 \frac{\text{m}}{\text{s}^2}. \quad (6.34)$$

In order to compare this value to the one obtained from BGI we need to take the difference and evaluate their *distance* in units of uncertainty. For the time being we assume that the BGI value $g = 9.803 \frac{\text{m}}{\text{s}^2}$ is known with infinite precision (i.e. it has negligible uncertainty). We find

$$\Delta = \frac{9.870 - 9.803}{0.006} \simeq 11. \quad (6.35)$$

This is a large discrepancy, a fortiori if we consider that the uncertainty of 0.006 $\frac{\text{m}}{\text{s}^2}$ is now estimated as we see in the next chapter. The probability that two values differ by more than ten standard deviations just because of statistical fluctuations is in fact, with extremely small. Indeed, the probability for a gaussian distributed variable to lie between $-x$ and x is

particular value.

Modelling the distribution of $P(x|\mu, \sigma)$ as a gaussian, we can use its values as the observed frequency.

Putting $t^2 = \frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2$, the integral becomes

$$P(x) = \frac{1}{\sqrt{\pi}} \int_{-t}^{+t} \exp(-t^2) dt = \frac{2}{\sqrt{\pi}} \int_0^{+t} \exp(-t^2) dt, \quad (6.37)$$

where the latter equivalence comes from the fact that the function being integrated is symmetric around $t = 0$. $P(x)$ is then a universal function, since it does not depend on particular values of μ and σ and is called the **error function**

$$\text{erf}(t) = \frac{2}{\sqrt{\pi}} \int_0^{+t} \exp(-t^2) dt \quad (6.38)$$

whose values can be computed numerically. Clearly, substituting the expression of t in $\text{erf}(t)$, gives

$$\text{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right) = P(x), \quad (6.39)$$

hence $\text{erf}\left(2^{-\frac{1}{2}}\right) \simeq 0.68$ corresponds to the fractional area below a gaussian in a $\pm\sigma$ interval. Similarly $\text{erf}\left(\frac{2}{\sqrt{2}}\right) \simeq 0.95$ and $\text{erf}\left(\frac{3}{\sqrt{2}}\right) \simeq 0.997$.

The **complementary error function** $\text{erfc}(x)$ is defined as

$$\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp(-t^2) dt. \quad (6.40)$$

If x is distributed as a gaussian, $y = \frac{x-\mu}{\sigma}$ is said to be **normally distributed**, where a **normal distribution** is a gaussian with $\mu = 0$ and $\sigma = 1$, i.e.

$$P_N(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right). \quad (6.41)$$

Given the above definitions, the probability for a normally distributed variable x to be larger than $x_0 > 0$ is

$$\begin{aligned} P_N(x > x_0) &= \frac{1}{\sqrt{2\pi}} \int_{x_0}^{\infty} \exp\left(-\frac{x^2}{2}\right) dx = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x_0} \exp\left(-\frac{x^2}{2}\right) dx = \\ &= 1 - \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^0 \exp\left(-\frac{x^2}{2}\right) dx + \frac{1}{\sqrt{2\pi}} \int_0^{x_0} \exp\left(-\frac{x^2}{2}\right) dx \right) = \\ &= \frac{1}{2} - \frac{1}{2} \text{erf}\left(\frac{x_0}{\sqrt{2}}\right). \end{aligned} \quad (6.42)$$

Known values for $\text{erf}(x)$ are $\text{erf}(0) = 0$, $\text{erf}\left(\frac{1}{\sqrt{2}}\right) = 0.68$, $\text{erf}\left(\frac{2}{\sqrt{2}}\right) = 0.95$ and $\text{erf}\left(\frac{3}{\sqrt{2}}\right) = 0.997$ (you can recognise the values of the fractions of data found within one, two and three standard deviation intervals described in Section 4, Chapter 7).

We found a value that is 11 standard deviations greater than expected. Such an event can occur with a probability of

$$P_N(x > 11) = \frac{1}{2} - \frac{1}{2}\text{erf}\left(\frac{11}{\sqrt{2}}\right) \simeq 2 \times 10^{-28}, \quad (6.43)$$

if the discrepancy is ascribed only to random fluctuations. We believe that an event with such a probability is extremely unlikely to happen and we must then ascribe the discrepancy to something else.

A possible explanation can be that we are using a not very **accurate** tool to make this measurement. Indeed, smartphone MEMS are intended to be quite sensitive, i.e. they must respond to very tiny signals, but not to be accurate, i.e. to provide an exact value. A smartphone accelerometer is designed to provide a fast non-null response even for very small accelerations, since they are usually impressed by handling it. However, in order to tell the device if it is in a landscape or portrait orientation, the absolute value of the acceleration is irrelevant and they are not very carefully calibrated, to keep their cost affordable. Instruments can be classified according to different characteristics.

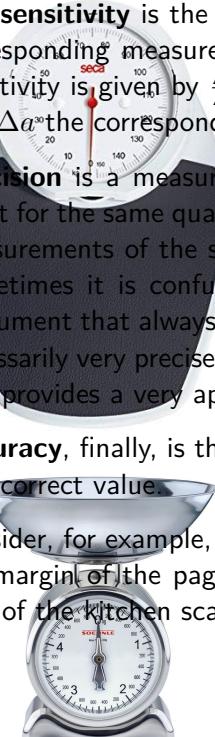
The **resolution** of an instrument is the smallest measurable unit.

- **sensitivity** is the ratio between the instrument smallest response and the corresponding measured quantity. In an accelerometer working as above, the sensitivity is given by $\frac{\Delta x}{\Delta a}$, where Δx is the variation of the length of its spring and Δa the corresponding acceleration.

Precision is a measure of the ability of the instrument to provide the same result for the same quantity. It can be defined as the closeness between different measurements of the same object. This characteristic is often misunderstood. Sometimes it is confused with sensitivity. More often it is believed that an instrument that always gives the same result in response to the same stimulus is necessarily very precise. In fact, a very precise instrument is often an instrument that provides a very approximate result.

Accuracy, finally, is the degree of conformity of the result of the measurement to a correct value.

Consider, for example, a bathroom scale and a kitchen scale as those shown on the margin of the page. The resolution of the bathroom scale is 1 kg, while that of the kitchen scale is 20 g. The sensitivity of the latter can be expressed



as

$$s_k = \frac{\Delta\theta}{\Delta m} = \frac{2\pi}{5 \text{ kg}} = 0.4 \text{ kg}^{-1}, \quad (6.44)$$

while that of the bathroom scale is

$$s_b = \frac{2\pi}{150 \text{ kg}} \simeq 0.013 \text{ kg}^{-1}. \quad (6.45)$$

The first is more sensitive because its response to the same stimulus is higher than that of the second one. Putting a weight of 3 kg on the kitchen scale, the needle rotates of $0.4 \times 3 = 1.2$ radians (69°), while the needle of the bathroom scale rotates of $0.013 \times 3 = 0.039$ radians (2.2°).

In digital scales, the sensitivity can be expressed as the inverse of the resolution. For example, if the scale have a resolution of 1 g, it responds with one digit per gram, then the sensitivity is $\frac{1}{1 \text{ g}} = 1 \text{ g}^{-1}$.

Both scales are quite precise. They always give the same response to the same stimulus. However, weighting several boxes of pasta with both scales will probably result in the same reading with the bathroom scale and in slightly different readings with the other one. Contrary to common beliefs, then, the bathroom scale can be considered more precise with respect to the kitchen one, as an instrument. Note, however, that the uncertainty of the result is expected to be better in the case of the kitchen scale, that in fact provides a more precise result. In fact, even if we always obtain the same result repeating the measurement in both cases, the closeness of the measurements among them can be represented by the corresponding resolution. When talking about instruments, being more precise does not always imply to be *better*, however, when talking about measurements properly quoted, the word “precision” takes its commonly agreed meaning.

The accuracy of the instruments depends on the quality of the calibration. The accuracy of an instrument may also depends on environmental conditions and it may change with time. An instrument may not be constant with time. One at your preferred deli counter, that is subject to periodical revisions by the metrological authorities.

The accelerometer of a smartphone is quite sensitive and has a relatively good precision, but is quite inaccurate. Though it is not a good idea to use it to measure absolute accelerations, we can be confident to use it to measure acceleration differences. In fact, the inaccuracy can be thought as an offset to the correct value that cancels out when two values are subtracted.

measurements we
should always check
the accuracy of the
instruments against
known values.

Summary

An experiment aims at measuring a physics quantity y as a function of another quantity x . It is possible that y depends on other physics quantities u, v, z , etc.. Typically we try to let only one of them to vary, either making the effects of a variable negligible or by keeping it constant. For example, the time needed to a body to fall may depend, in principle, on its height, its shape, its mass, etc.. To understand the laws of motion we study how bodies fall changing their height, keeping their shape and mass constant. Once we have fully understood the motion in this case we can study how shape and mass influence the results.

To measure times we need a stopwatch. Triggering it by hands introduces large fluctuations and biases. It is much better to use an automatic trigger.

Photogates are systems that monitor the intensity of a light source. If something pass between the light source and the sensor, the luminous flux is interrupted for a while. The abrupt change in intensity can be used to trigger a device.

Free fall happens at constant acceleration. We realise that from the fact that the falling time t as a function of the height h does not go to zero for $h \rightarrow 0$, while it does so if $a = \text{const.}$

Linearisation is a way to find the parameters of a curve that describes experimental data.

Instruments are characterised by their resolution (the smallest measurable unit), their sensitivity

(the ratio between the instrument response to the stimulus), their precision (the ability to give the same response to the same stimulus) and their accuracy (the closeness of the response to the correct value of the measured quantity).

Arduino

With Arduino a photogate can be realised using an analog light sensor illuminated by an external source such as a lamp, a laser or an LED.

Arduino provides several functions to manage time: `delay()` and `delayMicroseconds()` cause a pause in the execution of the program whose duration is expressed, respectively, in milliseconds and in microseconds, as an argument. `millis()` and `micros()` return the time elapsed since the beginning of the execution of the sketch in milliseconds and in microseconds, respectively.

Negative numbers are represented in the memory of a computer in two's complement. Variables intended to store non negative numbers can be declared as `unsigned`, extending the range of representable numbers by a factor two.

phyphox

PHYPHOX includes an acoustic stopwatch whose start and stop are triggered by sound. It starts when the phone detects a sound higher than the given threshold and stops when a second sound is detected after a user defined delay. During the delay, sounds do not stop the counter.

An experiment on free fall can be realised sus-

pending a weight to an inflated balloon. Popping the balloon trigger the start of the stopwatch that stops counting when the noise of the weight hitting the floor is detected. A proper delay must be set to avoid stopping measuring time because the echo of the balloon explosion is detected.

Smartphones use accelerometers to tell their orientation. Data from the accelerometer can be collected using PHYPHOX.

When at rest, an accelerometer measures the opposite of the gravitational acceleration.

Accelerometers exploits the second Newton's Law in non-inertial frames to measure their acceleration. They measure the force applied on a microscopic springs attached to microscopic masses.

Statistics

The uncertainty σ on a quantity x propagates to a function $f(x)$ as $\sigma_f = f'(x)\sigma$.

Uncertainties on more than one variable sum in quadrature, i.e. the variance of a function of more than one variable is the sum of the variances of the function taken as a function of only one variable at the time.

Experimental data often distribute as gaussians peaked at the mean value and with a width proportional to their standard deviation.

To compare two results we measure their difference in units of σ .

The error function $\text{erf}(x)$ is a universal function that can be used to evaluate the probability to observe a given value in an experiment. If the average value of the data is μ , the probability to observe a number higher than $\mu + x_0$ can be estimated as $\frac{1}{2} - \frac{1}{2}\text{erf}\left(\frac{x_0}{\sigma}\right)$, where σ is the standard deviation of the measurements.

Chapter 7

Understanding data distributions

In this chapter we examine the reasons for which data distribute as observed. One can formulate a rigorous, formal theory of data analysis based on statistics and probability. Here we concentrate on an intuitive explanations, leaving a more formal treatment to the following chapters and, for some of the more advanced topics, to specific courses.

1. On the values returned by instruments

Results of experiments often distribute over a range of values. The shape of the distribution can be different, depending on the experiment and on the instruments used to perform the measurements. Yet, data often appear to be distributed mostly as a bell. There must be a reason for that. Indeed, we can assume that data fluctuate around *true* values because of the occurrence of random effects that causes the instruments to respond differently with respect to the ideal case . For example, we believe that the measurement of the acceleration of a phone at rest should return a value for g , instead of a distribution. However, the measurement of the acceleration is affected by the fact that the phone has to be taken on some kind of support that for sure vibrates, even if subtly. Any attempt to isolate it from external vibrations will fail: one can reduce the amplitude of the vibrations, but it is impossible to get completely rid of them. Even if we succeeded, the thermal noise will always affect the accelerometer: its temperature depends, in fact, on the kinetic energy of the particles of which it is composed and it cannot be zero. Despite the fact that random fluctuations are random, we can take them into account in a relatively rigorous way thanks to statistics.

It must be noted that we implicitly assumed that there is, indeed, a *true* value for g . This is somewhat a prejudice, justified by the fact that data distribute closely around 9.8 m/s^2 , so we are induced to believe that there must be a constant value of g with arbitrary precision that is impossible to reach because of our limited technology. In fact, this is likely not to be true. If we had a *perfect* instrument we soon realise that g vary from point to point because its value is affected by all the masses in the vicinity of the point in which it is measured,

Due to uncertainties, the result of an experiment is often a set of data, distributed over a range of values.

From the shape of most common distributions we can infer that the result of measuring a physical quantity fluctuates around a value, which we define as a *true value*, due to stochastic effects not under control.

The concept of *true value* is not correct from an epistemological point of view.

even the one of the instrument used to obtain it. Moreover, since everything is moving in the Universe, g changes with time. Maybe on the tenth or the hundredth digit after the decimal point, but still we cannot define a constant true value of it. On the other hand there is no need for that. We need to know the value of g with enough precision to make predictions using a model that, in any case, will be approximated, neglecting some effect that happens in reality or oversimplifying something else. We agree that, by *true* value we intend the value that we would have measured if all the perturbing unwanted effects were removed. It is useless to know it, but it is useful to assume it exists.

2. Probability

Given that we are going to talk about random variables it is worth spending few words about the concept of **probability**. Even if anyone has an intuitive notion of what probability is, it took centuries before an internationally definition of it were finally accepted.

Laplace defined the probability P of the occurrence of a given event as the ratio between the number of cases in which the event can occur and the total number of possible events.

A first attempt to define probability was based on combinatorics and is due to Pierre-Simon Laplace (1749–1827) [Laplace, 1814]. In his essay, Laplace defined the probability P of the occurrence of a given **event** as the ratio between the number of cases in which the event can occur and the total number of possible events.

Here, an event is defined as something to which we can assign a truth value, i.e. something that can be true or false.

A very straightforward example of such a definition can be done using dices. The probability to obtain 3 in a launch is $\frac{1}{6} \simeq 0.17$ because there is one *favourable* case (the launch gave three as a result) out of six possible cases (the numbers from 1 to 6). Similarly, the probability to extract an ace from a deck of 54 cards is $\frac{4}{54} \simeq 0.074$ because there are four aces in a deck.

Such a definition is unsatisfactory because, in order to compute the probability of an event, we must assume that the probability of each event is the same, i.e. that each event is equiprobable. From a formal point of view, it sounds odd to define the concept of probability assuming that the probability of each event is the same. We need to know what means to be equiprobable to define it.

A second attempt to define probability is the so called frequentist approach, ~~that is not probability~~ to many authors (see the corresponding Wikipedia page for a comprehensive account of contributions). In this approach, the probability is ~~defined after observing~~ frequencies of event occurrence.

~~according to which the~~
~~The need for this new definition comes from the fact that it is often difficult,~~
~~frequency of events~~
~~if not impossible, to assign an *a priori* probability to some event based on~~
~~tends to their predicted~~
~~symmetry principles like in the case of the dices or the cards in a deck. Clearly,~~
~~combinatorial~~
~~probability for a large~~
~~number of trials.~~

the probability of obtaining 3 as a result of a launch is $\frac{1}{6}$ if we believe that the dice is symmetric. If it is not so (and most likely this is the case due to imperfections in its construction), we cannot assign the same probability to all the possible events. However, we can estimate it as the frequency f_3 , defined as

$$f_3 = \frac{n_3}{N}, \quad (7.1)$$

where n_3 is the number of times a launch gave 3 in a set of N . Clearly, this is only an approximation of the probability. In fact, if we make a new set of N launches, it is very unlikely to obtain again exactly the same number of 3, n_3 . However, we expect the new number n'_3 to be close to it, i.e. $n'_3 \simeq n_3$, if N is *large enough*. The higher N , the better the estimation. In principle one can say that the probability to obtain 3, $P(3)$, is

$$P(3) = \lim_{N \rightarrow \infty} \frac{n_3}{N}. \quad (7.2)$$

However, in this case, the limit operation cannot be taken as very well defined as in mathematics. This is an *experimental* limit and is manifestly impossible to execute an infinite number of trials. Even in this case, then, the definition of probability is not so obvious and depends on the events happened in the past, that must be assumed as representative of the whole set of possible events, i.e. they must have the same probability of similar events in the future.

In both cases the following rules hold. If an event is impossible, its probability is null (either $n_i = 0$ or the number of possible cases is zero). If an event is certain, its probability is 1 (either $n_i = N$ or there is only one favourable case in a set of only one possible case).

A more modern approach consists in the so called Bayesian probability, after According to Bayes, Thomas Bayes (1701–1761). In this approach the probability is taken as a subjective evaluation about the degree of belief that something is going to happen. It is often expressed in terms of a bet.

Suppose, for example, that you are going to bet an amount A that a given event E is going to happen. Your opponent bets B on the opposite event \bar{E} , such that the winner will get $S = A + B$. Manifestly, the amount you are willing to risk is proportional to the probability $P(E)$ you can estimate on the occurrence of that event. Formally speaking, we can say that

$$A \propto P(E), \quad (7.3)$$

even if we don't know $P(E)$. The amount A you are willing to bet is clearly

proportional to the total amount S you are going to get, if you win, i.e.

$$A \propto S. \quad (7.4)$$

Since A must be proportional to both $P(E)$ and S , we have

$$A \propto P(E)S \quad (7.5)$$

from which one can derive that

$$P(E) = \alpha \frac{A}{S} = \alpha \frac{A}{A + B}, \quad (7.6)$$

where α is a constant. It follows that

$$P(\bar{E}) = \alpha \frac{B}{S} = \alpha \frac{B}{A + B}. \quad (7.7)$$

From the last two equations we obtain $A = \frac{P(E)}{\alpha}S$, $B = \frac{P(\bar{E})}{\alpha}S$ and $S = A + B = \frac{P(E) + P(\bar{E})}{\alpha}S$. The actual value of α is practically irrelevant, then it is arbitrary with choosing $\alpha = 1$, $P(E) + P(\bar{E}) = 1$, i.e. the probability of obtaining any of the possible events is 1. Moreover,

$$P(E) = \frac{A}{A + B}. \quad (7.8)$$

The value of $P(E)$ determines our propensity to bet. As a concrete example, suppose that you bet on the fact that you will obtain 3 launching a dice. Is it worth betting 1 € on that? And what about 1000 € or more?

1 € could be worth, but it depends on how much you are going to win. If the amount you can win is 1 €, it is probably not worth. If it is 10 €, maybe yes. However, risking 1000 € to win 10000 maybe is not.

If you believe that $P(E) = 0$, then $A = 0$, i.e. you are not going to bet in any case, irrespective of S . On the other hand, if you believe $P(E) = 1$ you are going to bet up to $A = S$.

It is worth noting that $P(E)$ is subjective in the sense that it includes, often implicitly, your prior knowledge on the conditions that may influence the result. If we believe that a dice is perfectly realised, we can attach a probability of $\frac{1}{6}$ to the event $E = 3$, but if we found, in a set of launches, that 3 appears more often than expected, we may assign a higher probability to it. Though the higher observed frequency can be due to statistical fluctuations, it may also come from manufacturing imperfections of the instrument that favours the event $E = 3$. If, according to your previous experience, the dices appear to be honest, you can be reluctant to bet $A = 1$ € against $B = 1$ € on $E = 3$, while you may

be tempted to do that if $B = 9 \text{ €}$. The reason is that betting 1 € against the same amount on $E = 3$ corresponds to assign a probability of

$$P(3) = \frac{1}{1+1} = 0.5, \quad (7.9)$$

to $E = 3$, i.e. to overestimate the probability of observing the event with respect to the frequentist or combinatorial approach that anyone will judge as reasonable if the dices are close to be perfect. On the other hand, betting 1 € against 9 may be worth, since the assigned probability of the event is

$$P(3) = \frac{1}{1+9} = 0.1. \quad (7.10)$$

The *objective* probability is higher, so we can be confident that we have more chances to win with respect to those assumed subjectively, while the subjective probability depends on our knowledge about willing to participate in a lottery increases when the jackpot increases. The amount we are willing to loose increases with $S = A+B$ and with $P(E)$. Even if $P(E)$ does not change, the increase of S makes the bet more interesting.

On the other hand, should we bet $A = 1\,000 \text{ €}$ for $E = 3$ against $B = 10\,000 \text{ €}$ implies

$$P(3) = \frac{1\,000}{1\,000 + 10\,000} \simeq 0.09, \quad (7.11)$$

a much lower probability, indeed.

$P(E)$, the probability of the event E , is called the “marginal” or “prior” probability (see below for the explanation of the terminology). For a symmetric dice, it is reasonable to assume $P(3) = \frac{1}{6}$. As a general rule, we usually assume that there is no reason to believe that systems should not behave according to symmetry principles. As a consequence, we will estimate probabilities using either combinatorics or frequencies (in this case they are equivalent to the Bayesian one), unless we have reasons to act differently.

The subjective probability depends on our expectations. Our propensity in betting depends on the objective probability compared to the subjective one.

The subjective probability for an event to occur is called the marginal or prior probability.

Random number generators

In order to experimentally test our mathematically derived results we need some source of random events. In fact, it is hard, if not impossible, to find such a source.

Any device inevitably has imperfections that somewhat alter the theoretical distribution of purely random output of that device. Most of natural sources of random events are systems whose physics can be described by quantum mechanics. For example, nuclear decay is a random process

whose characteristics are well described by theoretical predictions, even if each single event happens at random times. The problem is that, in order to detect these events, we need detectors whose behaviour is subject to a number of conditions and have not a 100 % efficiency. As a result no perfect source of random events can be found neither natural nor artificial.

Computers can be used to simulate random events. There are, indeed, functions that return random numbers with the specified distribution and within a given interval. However, this source of random events is not at all random. Computers are fully deterministic and there is nothing inside a computer able to generate random numbers. Those generated by a computer are in fact called pseudorandom, because they come from an algorithm, even if they resemble to be random. A sequence of pseudorandom numbers is perfectly predictable. Once you know the last number generated by the algorithm you can actually compute the next number being generated.

Sequences of pseudorandom numbers just exhibit statistical properties similar to those expected for the distribution of random numbers they were designed for. Usually, pseudorandom generators provides numbers uniformly distributed between 0 and 1. Other distributions can be generated from them using proper techniques.

Most pseudorandom generators use the linear congruential method to yield a sequence of uniformly distributed numbers. A linear congruential method exploits the following recurrence relation:

$$x_{i+1} = (ax_i + b) \bmod c \quad (7.12)$$

where x_i are the pseudorandom numbers to be generated (x_0 is said to be the seed of the generator), a , b and c are integer constants and \bmod represents the modulo operation that returns the remainder after the division of the operands. The quality of the generator depends on the choice of a , b and c . Its period, i.e. the number of extractions after which the sequence will repeat, is an important parameter.

A common choice, used in the GNU C-compiler, is $a = 1\,103\,515\,245$, $b = 12\,345$ and $c = 2^{31}$.

The importance of the subjective definition of probability is made manifest in certain situations, discussed below in the text.

For the time being, we can adopt a combinatorics approach to estimate the probabilities in few important conditions. Bearing that in mind, we can start computing the probabilities of certain events aiming at discovering general rules. Let $P(E_i)$ the probability that event E_i occur. Dealing with dices, for example, $P(E_i)$ is constant and equal to $\frac{1}{6}$, the set contains six events: $\{1, 2, 3, 4, 5, 6\}$ and $E_i = i$. In order to work with the more interesting case of events with different probability, imagine to have a box with six balls: three of them are red (R), two are green (G) and one is blue (B). There is a total of six balls in the box. The probability of extracting a blue ball is $P(B) = \frac{1}{6}$, the one of extracting a green ball is $P(G) = \frac{2}{6} = \frac{1}{3}$, while $P(R) = \frac{3}{6} = \frac{1}{2}$. Let's also suppose that we have two such boxes, with identical content. We define the following concepts.

- The “joint probability” is the probability of the occurrence of two simultaneous events. It is important to realise that here “simultaneous” means that the events are part of the same event, even if the two events happen at different times. For example, we may be interested in the joint probability of extracting a red and a green ball.
- The “marginal probability” is defined as the probability of an event, irrespective of all other events. For example, we can compute the marginal probability of extracting a green ball from box 1.
- The conditional probability is the probability of an event, given the occurrence of another event. If, for example, we extract a green ball from box 1 we can ask ourselves how large is the probability to extract a second green ball from the same box.

Consider the event (G, R) consisting in the extraction of a green ball and of a red ball. In order to compute the probability of occurrence of this event we need to take into account:

- the independence of the two events and
- the ordering of the two events.

Two events are said to be independent if the occurrence of one of them does not change the probability for the other to occur. This is the case when we extract the two balls from different boxes (note that this is equivalent to extracting a ball from one box, putting it back in it and extracting a second ball from the same box).

The joint probability of such an event is the product of the probability $P(G)$ of extracting a green ball from box 1 and the probability $P(R)$ of extracting a red ball from box 2, i.e.,

$$P(G \text{ and } R) = P(G)P(R) = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}. \quad (7.13)$$

It is easy to check that the result is correct. There are 36 possible results in this experiment, because for each ball in box 1 we can extract 6 balls from box 2. The event (G, R) can be any among (G_1, R_1) , (G_2, R_1) , (G_1, R_2) , (G_2, R_2) , (G_1, R_3) , (G_2, R_3) , where (G_i, R_j) represents the event in which the i -th green ball in box 1 is extracted together with red ball j from box 2. There are then six favourable events out of 36 and the probability is

$$P(G, R) = \frac{6}{36} = \frac{1}{6}. \quad (7.14)$$

In the case of dependent events, the probability of extracting the green ball from box 1 is $P(G) = \frac{1}{3}$. If this event occurs, it modifies the probability to extract a second red ball from $P(R) = \frac{1}{2}$ to $P'(R) = \frac{3}{5}$ because there are now three red balls in a box with a total of five balls.

Two events are independent if the occurrence of one does not change the probability for the other to occur.

The joint probability of two events is the product of their individual probabilities.

The joint probability of two events is the product of their individual probabilities.

In general, we can say that the joint probability $P(G, R)$ is equal to the product of the probability for one of the events to occur, times the probability of the second to occur given the first happened. The latter is the conditional probability. Indicating as $P(R|G)$ the probability of event R given G , we can write

$$P(G, R) = P(G)P(R|G). \quad (7.15)$$

In the example, such a probability is $P(G, R) = \frac{1}{3} \times \frac{3}{5} = \frac{1}{5}$. The latter is a general rule. In fact, when the events are independent, $P(R|G) = P(R)$ because the extraction of a green ball has no effect on the probability of extracting the red ball.

Till now we considered the events as if the order in which they occur matters. If we do not care about the order, the event (G, R) is equivalent to (R, G) . Repeating the steps above we arrive at the conclusion that

$$P(G, R) = P(R, G), \quad (7.16)$$

i.e. the joint probabilities are symmetric.

If ~~the events are~~ ~~are~~ mutually exclusive, i.e. if the occurrence of an event forbids the occurrence of the other event, the probability of observing either one or the other is the sum of the probabilities of each event. To check the result let's start from a simple ~~case~~ ~~example~~ ~~in which~~ ~~one~~ ~~box~~ ~~is~~ ~~extracted~~ ~~from~~ ~~one~~ ~~box~~ ~~which~~ ~~of~~ ~~the~~ ~~balls~~ ~~is~~ ~~extracted~~ ~~from~~ ~~one~~ ~~box~~ ~~and vice versa~~ ~~Hence~~, the probability to extract either a green or a red ball is ~~mutually exclusive~~.

$$P(G \text{ or } R) = P(G) + P(R) = \frac{1}{3} + \frac{1}{2} = \frac{5}{6}. \quad (7.17)$$

There are, in fact, five favourable events out of six. The extraction of a green ball from the first box and a red ball from the second makes it impossible to observe the event consisting in the extraction of a red ball from the first box and a green one from the second. In this case, too, the probability of extracting two balls, of which one is red and the other is green, irrespective of the order, is

$$P((G, R) \text{ or } (R, G)) = P(G, R) + P(R, G) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}. \quad (7.18)$$

We can compute this probability considering all the possible outcomes of the experiment. To make the counting easy we can associate to each ball a number from 1 to 6. Balls 1 to 3 are red, 4 and 5 are green and 6 is blue. There are 36 possible events that can be identified as (i, j) , $i, j = 1, \dots, 6$. The favourable events are $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$ and those with i and j swapped. There are, then, 12 favourable events out of 36 and the probability is $P((G, R) \text{ or } (R, G)) = \frac{12}{36} = \frac{1}{3}$.

Consider now the probability of extracting two green balls from the two boxes. In this case, following the above prescriptions, the favourable events are $(4, 4)$, $(4, 5)$ and those in which i and j are swapped. The probability is then

$$P((G, G) \text{ or } (G, G)) = \frac{4}{36} = \frac{1}{9} \neq P(G, G) + P(G, G) = \frac{2}{9}. \quad (7.19)$$

This is because the events are not mutually exclusive. In this case the total probability is the probability computed as if the events were mutually exclusive minus the one that both occur, i.e.

$$\begin{aligned} P((G, G) \text{ or } (G, G)) &= P(G, G) + P(G, G) - P((G, G) \text{ and } (G, G)) = \\ &= P(G, G) + P(G, G) - P(G, G)P((G, G)|(G, G)) = \\ &= \frac{1}{9} + \frac{1}{9} - \frac{1}{9} \times 1 = \frac{1}{9}. \end{aligned} \quad (7.20)$$

In general,

$$P(A \text{ or } B) = P(A) + P(B) - P(A)P(B|A). \quad (7.21)$$

The marginal probability of the event G of extracting a green ball from box 1 is the probability of such an event irrespective of any other event. This is the probability we computed initially $P(G) = \frac{1}{3}$. The reason for which it is called “marginal” is explained considering the marginal probability $P(E)$ as the joint probability $P(E, X)$ where X represents any possible event. Let’s build a table in which we put on each cell the joint probability of the events obtained putting together the event on a row with the event on a column for all the possible events.

	R	G	B
R	$\frac{1}{4}$	$\frac{1}{6}$	$\frac{1}{12}$
G	$\frac{1}{6}$	$\frac{1}{9}$	$\frac{1}{18}$
B	$\frac{1}{12}$	$\frac{1}{18}$	$\frac{1}{36}$

The probability $P(G, R)$ is found looking for the cell corresponding to row G and column R or vice versa (note that the joint probabilities are symmetric, i.e. $P(G, R) = P(R, G)$). Then let’s sum on rows and on columns.

If events are not mutually exclusive, the probability for the occurrence of any of two events is the sum of the marginal probabilities of each minus the one that both occur.

The prob prob
irres even

Joint
sym
 $P(A)$

	R	G	B	
R	$\frac{1}{4}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{2}$
G	$\frac{1}{6}$	$\frac{1}{9}$	$\frac{1}{18}$	$\frac{1}{3}$
B	$\frac{1}{12}$	$\frac{1}{18}$	$\frac{1}{36}$	$\frac{1}{6}$
	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$	

The probabilities on the right and bottom *margins* of the table are the marginal probabilities of the event indicated, respectively, in the row and in the column header.

In fact, in order to compute the probability $P(G)$, we can consider the following joint probabilities:

$$P(G, R) = \frac{1}{6} \quad P(G, G) = \frac{1}{9} \quad P(G, B) = \frac{1}{18}. \quad (7.22)$$

The probability of any of these three events happening is the sum of the probabilities. Remember, in fact, that $P(G, R) = \frac{1}{6}$ because there are six out of 36 possible events giving the wanted result (G, R) . Similarly, there are four out of 36 events leading to the result (G, G) and there are only two events whose result is (G, B) . There is then a total of $6 + 4 + 2 = 12$ events out of 36 for which there is a green ball in the event, then the probability $P(G) = \frac{12}{36} = \frac{1}{3}$.

The conditional probability $P(G|R)$ can be computed starting from the joint probability, as

$$P(R|G) = \frac{P(G, R)}{P(R)}. \quad (7.23)$$

The formula is only valid if $P(R) \neq 0$. Indeed, if $P(R) = 0$ it is impossible that G occurs if R occurred and $P(G|R) = 0$, as well as $P(G, R)$.

Since the joint probability is symmetric we have

$$\begin{cases} P(G, R) = P(G)P(R|G) \\ P(R, G) = P(R)P(G|R) \end{cases} \quad (7.24)$$

and since the LHS of the equations are equal

$$P(G)P(R|G) = P(R)P(G|R) \quad (7.25)$$

from which we find the “Bayes’ Theorem”

$$P(R|G) = \frac{P(R)P(G|R)}{P(G)}. \quad (7.26)$$

The Bayes’ Theorem is a cornerstone in probability theory and states that $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$.

An alternative formulation of the theorem can be made observing that

$$P(G) = P(G|R)P(R) + P(G|G)P(G) + P(G|B)P(B) \quad (7.27)$$

Substituting into the expression of the Bayes' Theorem above,

$$P(R|G) = \frac{P(R)P(G|R)}{P(G|R)P(R) + P(G|G)P(G) + P(G|B)P(B)}. \quad (7.28)$$

In Bayesian terminology the LHS of this equation is called the posterior probability, while $P(R)$ is called the "prior probability". $P(G|R)$ is the likelihood, while the denominator is called the evidence.

Let's check the result in the case of two extractions from the same box, bearing in mind that $P(R|G) = \frac{2}{5}$ as we already computed. We know that $P(R) = \frac{1}{2}$, $P(G) = \frac{1}{3}$, while $P(G|R)$ is the probability of G given R . When R has been extracted from the box, there are two green balls out of five and $P(G|R) = \frac{2}{5}$. Using the first formulation of the theorem we find

$$P(R|G) = \frac{P(R)P(G|R)}{P(G)} = \frac{\frac{1}{2} \times \frac{2}{5}}{\frac{1}{3}} = \frac{3}{5}, \quad (7.29)$$

as expected. For the alternative formulation we have

$$P(R|G) = \frac{\frac{1}{2} \times \frac{2}{5}}{\frac{2}{5} \times \frac{1}{2} + \frac{2}{5} \times \frac{1}{6} + \frac{1}{5} \times \frac{1}{3}} = \frac{3}{5}. \quad (7.30)$$

The most general form of the Bayes' Theorem can be written as

$$P(R|G) = \frac{P(R)P(G|R)}{\sum_i P(G|E_i)P(E_i)}, \quad (7.31)$$

where the sum is extended to all events. We can easily remember it as

$$P(R|G) \propto P(R)P(G|R), \quad (7.32)$$

where marginal and conditional probabilities alternate, as well as the events indicated as the probability arguments: $RGRGR$. the denominator working as a normalisation factor.

This textbook was prepared during the COVID-19 pandemic. Naso/oropharyngeal swabs are performed to diagnose the disease. This kind of tests produces a positive or negative response

depending on whether the virus is detected or not, respectively.

Often, their reliability is not perfect: sometimes, the swabs return a negative result for a sick person (false negative) or a positive result for a healthy one (false positive). Doctors define the PPV (positive predictive value) and the NPV (negative predictive value) as the rate of true positive and true negative responses. The table below reports the PPV and the NPV of a sample of swabs, according to [Castro, 2020].

PPV	95	89	86	95	89	68	85	85	97	94	97	94
NPV	100	94	95	100	94	96	100	96	99	97	99	97

The average PPV is 89.5 %, while the average NPV is 97.25 %. Suppose that we were subject to a swab and we were just randomly chosen within the population (i.e. we do not show symptoms and we have not had any risky behaviour) and that it resulted in a positive response. The probability to be affected by COVID–19 could naively be estimated as $P(\text{COVID}) \simeq 89.5\%$ or, better, considering the possibility of false negatives, as $P(\text{COVID}) \simeq 89.5 + (100 - 92.25) = 92.25\%$. However, when we update our knowledge, the posterior probability is updated, too, and the difference could be dramatic.

According to the Bayes's Theorem,

$$P(\text{COVID}|+) = \frac{P(\text{COVID}) P(+|\text{COVID})}{P(+)} \quad (7.33)$$

where $P(\text{COVID}|+)$ represents the probability of being affected by the disease having been diagnosed with a positive swab, $P(+|\text{COVID})$ the probability that people affected by COVID–19 have been tested as positive with a swab (PPV), while $P(+)$ is the probability to receive a positive diagnose being tested with a swab:

$$P(+) = \text{PPV} \times \text{COVID} + (1 - \text{NPV}) \times \text{healthy} = 3.1\%. \quad (7.34)$$

$P(\text{COVID})$ is the probability to be infected, that can be estimated from the fraction of the population that has been diagnosed as such. In Italy, at the time we were writing, this fraction is 0.35 %.

It is useful to build a table as follows, in which we arrange the probabilities to receive a positive/negative response, depending on your status.

		COVID (0.35 %)	healthy (99.65 %)
		+	-
+		89.5	2.75
-		10.5	97.25

Persons affected by COVID–19 belong to the first column. They have a chance PPV to receive a positive answer from a swab and a chance $1 - \text{PPV} = 10.5\%$ to be negative. Healthy people belong to the second column: they receive a negative diagnosis in $\text{NPV} = 97.25\%$ of the possible cases, however, there is a non null probability, equal to $1 - \text{NPV} = 2.75\%$ to be positive. The sum of all the values in each column is 100.

Putting the values in the Bayes's formula we get

$$P(\text{COVID}|+) = \frac{0.35 \times 89.5}{3.1} = 10\%. \quad (7.35)$$

It has been argued that the actual fraction of infected people could be as large as ten times those declared as such by authorities. Even in this case, the chance to be affected by COVID–19 having being received a positive response from a swab is only 1 %.

It must be noted that the result of the last exercise is not in contradiction with the commonly used frequentist approach. In fact, even if the fraction of true positive is large, the number of people actually healthy is huge. A small fraction of them can then be larger than those truly positive and infected. In fact, using the frequentist approach, we can say that out of N patients, those diagnosed with a positive swab and infected are a fraction $0.895 \times 0.0035 \simeq 0.0031$ of N , while those who just received a positive diagnose from a test are $0.895 \times 0.0035 + 0.9965 \times 0.0275 \simeq 0.031$ times N . The ratio between the two is

$$\frac{0.0031}{0.031} = 0.10 \quad (7.36)$$

corresponding to 10 %. The Bayes' Theorem, in essence, allows us to update our initial estimate for the probability of being ill, taking into account the increase in information represented by the knowledge of the fraction of people infected overall. The probability is subjective in the sense that it is estimated from a limited information. Each time we obtain new information we can update our estimation of the probability.

Despite the apparent impossibility of drawing scientifically sound conclusions from what appears to be a completely arbitrary opinion, the Bayes' Theorem works perfectly and allows for very solid results.

3. Bayes's Theorem and Physics

The content of the previous section seems to be quite far from the topic of this book. In fact, besides being the foundation of the study of probability distributions, its conclusion, the Bayes Theorem, has direct consequences on how to interpret the results of an experiment.

The topic of probability distributions is important also on matters of principles.

We can rewrite its last formulation as

$$P(E|H) \propto P(E)P(H|E), \quad (7.37)$$

where E represents the outcome of an experiment and H the *prior* knowledge on the physics involved in ~~That experiment~~. To remember this formula, think at the arguments in parentheses. The sequence of alternate symbols $EHEHE\dots$ as the arguments of ~~the alternate functions~~ $P_CPPC\dots$ where P_C represents the conditional probability, while P the prior probability.

This formula states that the probability of a certain interpretation of experiment E to be true $P(E)$ is updated when an additional hypothesis H is known to be true. $P(H|E)$ is the probability of H to be true in the context of the experiment E , while $P(E|H)$ is the updated probability for E to be true, when we consider $P(H|E)$.
$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

The Bayes' theorem ~~has important implications when interpreting data, especially for rare processes or searches. From the technical point of view, its application is not always straightforward and many of interesting applications are beyond the scope of this book.~~
$$P(B|A)P(A|B)$$

Nevertheless, even if sometimes it is not possible to compute the probabilities involved in the Bayes' Theorem, its (subjective) approach to probability definition is of great importance in science and in physics, in particular, even when we cannot (realistically) quantify them. Physics, in fact, is not just a matter of collecting numbers during experiments and formulating physics laws in the form of equations. Physicists must interpret the results of an experiment and draw a coherent picture of the Universe. They cannot just write down the results of their experiments, ignoring the complete formal and informal picture, or formulate beautiful theories ignoring the results of experiments.

~~In Feynman's lecture~~ "given by Richard Feynman (1918–1988), whose recording is available to make on YOUTUBE, he explained the essence of the scientific method as follows

difference how beautiful your guess is,
it doesn't matter how smart you are who made the guess, or what his name is. If it disagrees with experiment... it's wrong.

what his name is. If it

To understand the very meaning of such a sentence, we have to discuss the context in which it was formulated. Part of the scientific community believes in the *power of beauty*. In other words they are convinced that the probability for a theory to be true is higher if the resulting equations are *beautiful*. According to this point of view, a theory whose equations are *ugly* is likely to be wrong.

The Bayes's Theorem tells us that the probability for a theory to be true has nothing to do with beauty. Given an *a priori* probability that it is true, such

a probability must be updated taking into account both implicit and explicit knowledge, acquired in the past or that will be acquired in the future. The new posterior probability can become higher or lower, depending on the effect on the prior of $P(H|E)$.

A very instructive example about a non quantitative application of the Bayes' Theorem is the publication of what, at that time, seemed a thrilling discovery by the Opera collaboration [Opera, 2011], claiming that neutrino were found to move with a speed faster than light.

Opera ran at Gran Sasso underground laboratory to investigate the properties of neutrinos coming from CERN. Neutrinos are weakly interacting particles that can travel across the earth practically without interacting with it. After a careful investigation of many possible sources of systematic errors, the collaboration decided it was worth to publish such a result whose statistical significance was quite high. The results obtained in two runs of the experiment, in fact, differ from expectations by about six standard deviations and the probability that they could be ascribed to random fluctuations is in fact extremely low.

Despite such a striking evidence, the scientific community was much reluctant to accept the result as genuine. It is worth noting that, in certain cases, results with less evidence were accepted seamlessly, like in the case of the discovery of the Higgs boson, claimed with a five sigmas evidence, or the existence of dark matter: an unknown kind of matter whose presence can be inferred from cosmological observations.

There was, in fact, a good reason to be skeptical: Einstein's relativity forbids particles to move with a speed faster than light and this theory is among the most successful. A discovery like the one claimed by the Opera collaboration, in fact, opened a problem in the reliability of such a theory that must be resolved in turn. The prior probability $P(v > c)$ that indeed neutrino travels with a speed v greater than the one of the light c is, according to the frequentist evaluations of the physicists of the collaboration, very high. However,

$$P(v > c|\text{relativity}) \propto P(v > c)P(\text{relativity}|v > c), \quad (7.38)$$

i.e. the probability that $v > c$, given the theory of relativity, is much less, due to the fact that it is proportional to $P(\text{relativity}|v > c)$, the probability of the relativity to be correct given the Opera result. Actually, it is zero.

From the story told in the box, it is clear that what makes a discovery acceptable or not by the scientific community is the increased probability that, overall, both prior and posterior knowledge are corroborated. Often, the decision to support or not new discoveries, is driven by subjective evaluations. Contrary to common beliefs, this is not at all far from a scientific point of view (of course, if the evaluations are honest enough), thanks to the Bayesian point of view. The fact that consolidated discoveries have finally reached a state of presumed beauty is not due to a mysterious preference of Nature for beautiful theories, yet to the fact that the supporters of that theories have worked hard to make them look

as good as possible.

The *beauty argument* has probably had its origin in an interview [Dirac, 1962] given by Paul Dirac (1902–1984) to Thomas Kuhn and Eugene Wigner in 1962. In that interview Dirac said

[The idea of spin] came out just from playing with the equations rather than trying to introduce the right physical ideas. A great deal of my work is just playing with equations and see in what they give. Second quantization I know came out from playing with equations. I don't suppose that applies so much to the physicist; I think it's a peculiarity of myself that I like to play about with equations, just looking for beautiful mathematical relations which maybe don't have any physical meaning at all. Sometimes they physical equations was a

In a paper [Dirac, 1982] written in 1982, Dirac wrote his own,

It seems to be one of the fundamental features of nature that fundamental physical laws are described "in terms of a mathematical theory of great beauty and power [...] and we [...] could perhaps describe the situation by saying that God is a mathematician of a very high order, and He used very advanced mathematics in constructing the universe. It seems that if one is working from the point of view of getting beauty in one's equations, and if one has really a sound insight, one is on a sure line of progress.

Not all the scientist share Dirac's view, still they believe in the power of mathematics to which many of them credits something magic. This is mostly due to Galileo Galilei (1564–1642) who wrote

G. Galilei writings, the universe, [...] cannot be understood unless we first learn to understand its language and know the characters in which it is written. It is written in mathematical language, and the characters are triangles, circles, and other geometrical figures, without which it is impossible to understand it humanly; without them it is a vain wandering through a dark labyrinth.

Indeed, our opinion is that there is nothing magic in mathematics, nor it exists a power of beauty. Mathematics is, in fact, a language and, as all languages, is rather flexible. It turns out that mathematics is an extremely flexible language. Its dictionary and its syntax can be redefined almost freely and this accommodates for the possibility of introducing new concepts and structures that lend themselves better and more effective to correctly and unambiguously describe the physical phenomena of interest.

Using the appropriate language, physicists can look deeper into the most profound meaning of equations and the derivation of new results is greatly simplified.

An astonishing example of this is represented by the equations of electromagnetism. Modern textbooks report four beautiful symmetric equations as the Maxwell's Equations (although in slightly different forms). It turns out that those four equations do not look anything like those written by James Maxwell (1831–1879) in his Treatise on Electricity and Magnetism [**Maxwell, 1873**]. The original formulation of that equations was quite cumbersome, while the currently used version is due to Oliver Heaviside (1850–1925). The latter are much more *beautiful* than the original equations written by Maxwell, but they are the same. They just look differently, but their meaning is manifestly the same. The reason for that they look nicer is that, since Maxwell, the mathematics of vector fields was developed in which new operators were introduced to play with vectors, greatly simplifying notation.

A very similar argument can be made for the probably most famous equation in physics as $F = ma$, known as the second Newton's Law. Isaac Newton (1643–1727), in fact, never wrote it. He wrote a long paragraph, whose meaning can be summarised by the above equation, that has to be ascribed [**Euler, 1752**] to Leonhard Euler (1707–1783), not yet in its modern form as a vectorial equation.

Physics laws look beautiful just because physicists write them in the proper way. They are not beautiful ~~because it is however less~~ ~~in making them look finer~~ ~~it is manifest not easier~~. However, the need of reformulating equations is much more than just ~~in making them look finer~~ and easier to remember. A proper look makes some results ~~it is manifest not easier~~ to be derived. We can compare the effect of a clever reformulation ~~of physics laws~~ in terms of a different *language* to what happens in literature, where long, yet descriptive text used to illustrate a scene in a novel, can be surpassed, in clarity, by poetry. Let's examine an excerpt from *The Pickwick Papers* by Charles Dickens (1812–1870)

The modern form of Maxwell's Equations due to Oliver Heaviside.

There is no $F = ma$ equation in Newton's *Principia*. It was Leonhard Euler introduced it.

That punctual servant of all work, the sun, had just risen, and begun to strike a light on the morning of the thirteenth of May, one thousand eight hundred and twenty-seven, when Mr. Samuel Pickwick burst like another sun from his slumbers, threw open his chamber window, and looked out upon the world beneath

Reading this sentence, one can almost see the sun rising and close to the horizon, light filtering throughout a window, Mr. Pickwick being suddenly awakened, lifting from his bed and going to the window to open it completely and looking around.

Compare this rather long description to the following *haiku* by Yosa Buson (1716–1784).



Figure 7.1: Mural reproducing “Guernica”, by Pablo Picasso, installed in the Guernica town. Photograph by Papamanila.

such a moon
the thief pauses to sing.

~~Such is few syllables there is a full description of a scene, even more detailed of the one depicted by Dickens. Yet, each reader can enjoy the picture in a variety of ways that nevertheless, are almost the same. In those two, short, verses, the reader can see the landscape, but also feel the atmosphere, hear the sounds in the air~~

~~that they “create”~~
Moving from the original, ugly formulation of the Maxwell's Equations to the Heaviside's one produces a similar effect. The description of the physics is not only much shorter. While maintaining the same content of information, the latter allows more people to see the relevant characteristics of the electromagnetic fields and technically skilled people can derive new information from them with ~~fact they only make~~
~~(produce) them.~~
~~However, they create~~
~~a lot less effort.~~
~~the interpretation of~~

~~Indeed, science is a creative activity, as art. In art, the language evolves, too. From the very ingenious drawings of the past to the extraordinary techniques achieved by Raffaello Sanzio in painting we can clearly recognise the evolution of physics from the very raw and vague concepts of the ancient Greeks to the perfection of thermodynamics and electromagnetism. However, when the technical perfection reaches its peak, new demands arise. Modern painters no longer need to reproduce Nature as it appears through the senses, but have the need to reproduce things that are impossible to see or touch, such as emotions. They then need to develop a completely new language. Few paintings like “Guernica” by Pablo Picasso (1881–1973) have the power to evoke the feeling of fear and terror felt by the inhabitants of that city during a terrible bombardment. Similarly, when new unexpected phenomena were discovered, like the photoelectric effect, a new mathematics had to be invented (created). Quantum mechanics~~

allowed physicists to see a completely new world, impossible to describe by classical dynamics. That world, in fact, is not less real than the one to which we are used to live in, as fear, pain, love are not less real than a tree in a landscape, a fish in a still life or a woman in a portrait. They just deserve a different language to be expressed.

It is probably no coincidence that many religions attribute to mankind the characteristic of being made in God's image and likeness. For believers, indeed, God is the Creator: men and women, contrary to other living creatures, committed only to feeding and breeding, are, as God, able to "create" through art and science.

4. Statistical distributions of data

From the considerations made while taking data and analysing them, it is clear that we ascribe the fluctuations observed in data collected by an instrument to random effects. We can then understand their distribution studying the distribution of random variables.

Since each instrument always return a value with a limited number of digits, the variables to which we are interested in are always **discrete**. For example, on a ruler we can only read values that differ by 1 mm. The possible results of the measurement of the width of an A4 sheet of paper can be 239, 240 or 241 mm. Results like 239.3, 240.15273, 239.18732846013495861 and so on are not included in the set of the possible results, that always form sets of discrete, numerable elements, often finite.

Let's then start analysing the distribution of simple discrete random variables.

The simplest random variable is a "uniformly distributed" one. In this case all events have the same probability $p_i = p, \forall i$. The normalisation condition implies that

$$\sum_{i=1}^N p_i = \sum_{i=1}^N p = Np = 1 \quad (7.39)$$

and $p = \frac{1}{N}$, where N is the number of possible events. The scores of the launch of a dice is an example of such a distribution with $N = 6$.

In order to work with random variables, it is convenient to simulate random events using computers. As computers are deterministic machines, it is impossible to extract random numbers from it. However, techniques exist to generate lists of numbers appearing as random sequences, called pseudorandom. In the following, when not explicitly stated, we use the words random and pseudorandom as synonyms, for the sake of simplicity.

Fluctuations in the results of a measurement can be ascribed to random effects.

It is
stud
de
rand
Mea
alwa
valu

We ca
numbe
compu
genera
pseudor
numbe
purely

5. Uniform distribution

`numpy.random.uniform()` returns a pseudorandom number between 0 and 1.

It is rather easy to generate uniformly distributed random numbers with Python. We can fill a list of 10 000 events as follows.

```
import numpy as np

x = []

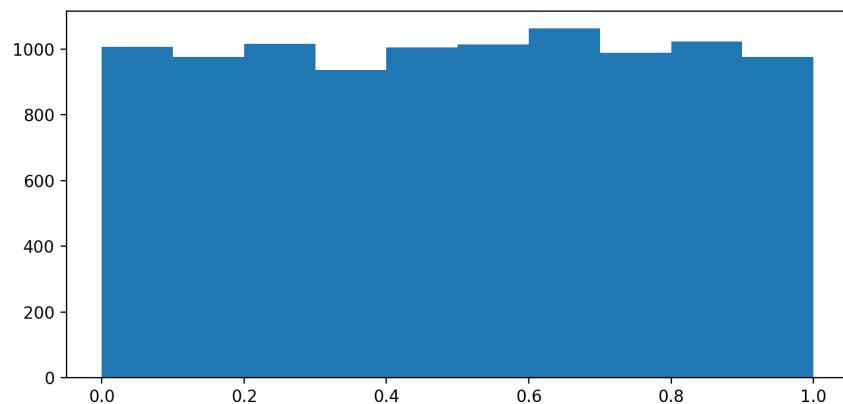
for i in range(10000):
    x.append(np.random.uniform())
```

The method `uniform()` from `numpy.random` generates a uniformly distributed random number between 0 (included) and 1 (excluded). In order to have an idea of the shape of the data distribution we can do a histogram: a plot of the number of occurrences of events against the events themselves. In other words, for each event E_i we count how many times N_i it occurs and make a plot in which we list all the possible events on the abscissa and draw a point at height N_i for each of it (or, more often, a rectangle of height N_i). A histogram of the values in the list can be done and shown using

```
import matplotlib.pyplot as plt

plt.hist(x)
plt.show()
```

The result is



From the graphical point of view, histograms are often represented as bar charts in which each event (or groups of events) is represented by a bar whose height

is proportional to N_i . Note that the frequency f_i of an event E_i is proportional to N_i , too, so histograms give information about the distribution of frequencies, too. Indeed

$$f_i = \frac{N_i}{\sum_i N_i} . \quad (7.40)$$

The shape of the distribution clearly shows that the numbers are uniformly distributed, since all the $f_i \simeq p_i$ are almost the same, the differences being interpreted as statistical fluctuations (remember that, in the frequentists view, $f_i = p_i$ only for $N = \sum_i N_i \rightarrow \infty$).

In fact, `uniform()` returns a list. By default its length is one, however you can change it with the `size` parameter:

```
x = np.random.uniform(size=10000)
```

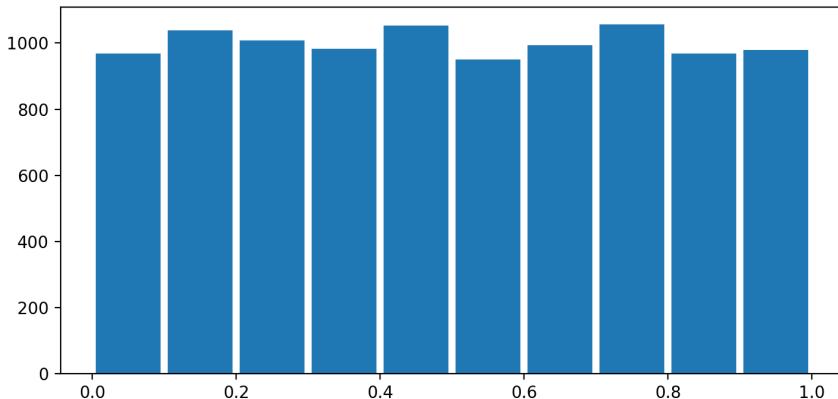
You can also change the appearance of the histogram changing the relative width of each column, as in

```
plt.hist(x, rwidth=0.9)
plt.show()
```

The result is the histogram below, where each column has a width equal to 0.9 times the width of each bin or class.

The optional parameter `size` tells `uniform()` to generate a list.

The `rwidth` optional parameter changes the bars' relative widths in an histogram.



In order to generate numbers between a and b , it is enough to linearly transform the generated numbers in the interval $[0, 1)$, i.e.

$$y_i = a + (b - a)x_i . \quad (7.41)$$

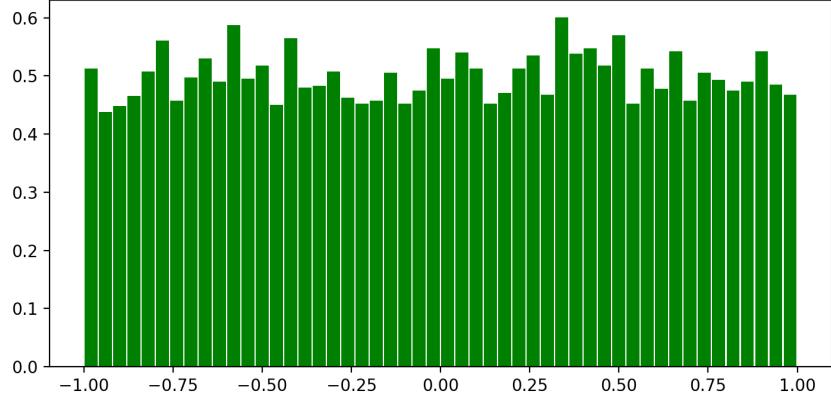
This is done by Python when we pass to `uniform()` the parameters `low` and `high` as in

```
x = np.random.uniform(low=-1, high=1, size=10000)
```

The number of classes or bins in which the interval is divided to build the histogram can be arbitrarily chosen. Moreover, in order to plot the relative frequencies rather than the absolute number of events, we can ask the histogram to normalise data using the keyword `density`. With

```
plt.hist(x, rwidth=0.9, bins=50, density=True,
          color='green')
plt.show()
```

we plot a green coloured histogram with 50 bins between $a = -1$ and $b = +1$, properly normalised such that the area under the histogram is 1.



The three distributions appear very similar and reasonably uniform. Being (pseudo)random, fluctuations in each bin are observed. They appear to be larger for the last histogram, where each bin contains less events. In all cases we generated 10 000 events. While in the first two each bin contains, on average, $\frac{10\,000}{10} = 1\,000$ events, the bins of the last histogram contain, on average, $\frac{10\,000}{50} = 200$ events. The bin width is $\frac{b-a}{n} = 0.04$. The area under the histogram is

$$1 = \sum_{i=1}^n p_i \Delta y_i = \Delta y \sum_{i=1}^n p_i , \quad (7.42)$$

where p_i is the height of the i -th bin and $\Delta y_i = \Delta y = 0.04$ is the bin width and is the same for all i . If $p_i = p$ were constant, $1 = np\Delta y$ and $p = \frac{1}{n\Delta y} = 0.5$, as it can be seen in the plots.

6. Expected value, variance and moments

If we compute the average of x we always obtain a value close to the center of the distribution (0.5 in the first two cases and 0 in the third one). The average random variable is of a sequence of random numbers is in fact an estimator of its expected value, sum of all the defined as

$$E[x] = \mu = \sum_{i=1}^N x_i P_i,$$

The expected value of the random variable is the arithmetic mean (the average) of the values in a random sequence. The corresponding probabilities. It can be estimated from the arithmetic mean (the average) of the values in a random sequence.

where P_i is the probability to obtain a value in the class x_i and N is the total number of classes. For example, in the first set we have ten classes corresponding to the following x_i : 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95 (we defined the classes according to the central value of the interval in which a number fall when counted in the histogram). Since, in this case, all the classes have the same probability $P_i = p = \frac{1}{10}$, the expected value is

$$\mu = \frac{1}{10} \sum_{i=0}^9 0.05 + 0.10 \times i. \quad (7.44)$$

The sum on the right can be easily computed using a formula ascribed to Carl Friedrich Gauss (1777–1855). The sum of the first and the last number in the sum is $0.05 + 0.95 = 1$, as the sum of the second and the penultimate $0.15 + 0.85 = 1$. The same happens for all the possible sums taken between two extremes, once removed those just used. If there are N terms in the sum, there are $\frac{N}{2}$ such sums, each of which gives $x_1 + x_N$. The result is then

$$\mu = \frac{1}{10} \frac{10}{2} (0.05 + 0.95) = \frac{1}{2} = 0.5. \quad (7.45)$$

Note that the Gauss algorithm applies to any sum of a linear transformation of i , even if N is odd. In fact

$$\sum_{i=0}^{N-1} i = \frac{N(N-1)}{2}. \quad (7.46)$$

If N is even, $N+1$ is odd and $\frac{N}{2}$ is integer; if N is odd, $N+1$ is even and

$\frac{N+1}{2}$ is integer. Moreover

$$\sum_i a + b \times i = Na + b \sum_i i = Na + b \frac{N(N-1)}{2}. \quad (7.47)$$

In our case $a = 0.05$, $b = 0.10$, $N = 10$. Clearly, if the sum starts with 1, the Gauss formula becomes

$$\sum_{i=1}^N i = \frac{N(N+1)}{2}. \quad (7.48)$$

If the random variables are continuous, the probability density function $P(x)$ is introduced, such that $P(x)dx$ is the probability that the variable lies between x and $x + dx$.

It should be noted that while μ is perfectly defined and constant for a distribution, being P_i and x_i constant, this is not the case for the average of a sample of random variables. In fact, in this case, while P_i may be thought as constant, the sequence of x_i is random and is different for each sequence. In the examples above we found, respectively, 0.500 and 0.496 for the first two distributions and -0.010 for the third (for which the expected value is zero).

Even if we always deal with discrete random variables, it is useful to extend the reasoning to the case of continuous ones. In fact, if Δy is small enough, the distribution can be approximated by a continuous curve. In this case $P(x)$ is called "probability density function" (PDF). Even if the interval of possible values of x is finite, there are infinite possible values of x in this interval and, from one of the possible definitions of the probability it follows that $P(x) = 0, \forall x$, if we interpret it as in the discrete case. However, from the discrete case, we also know that $\sum_i P_i \Delta y_i$ represents the probability to find y in an interval whose width is Δy_i , then $P(x)dx$ can be thought as the probability of finding the random variable x in the interval $[x, x + dx]$. The normalisation condition becomes

$$\int_{-\infty}^{+\infty} P(x)dx \stackrel{\text{condition}}{=} 1 \quad (7.49)$$

If $P(x)$ is the PDF of a uniformly distributed variable in the range $a \leq x < b$, $P(x) = 0$ for $x < a$ and $x \geq b$, and $P(x) = p = \text{const}$ in the interval, hence

$$\int_{-\infty}^{+\infty} P(x)dx = \int_{-\infty}^a 0 \cdot dx + \int_a^b p dx + \int_b^{+\infty} 0 \cdot dx = p(b-a), \quad (7.50)$$

from which $p = \frac{1}{b-a}$. The expected value is

$$\mu = \int_a^b xp dx = p \int_a^b x dx = \frac{1}{b-a} \frac{x^2}{2} \Big|_a^b = \frac{1}{b-a} \frac{b^2 - a^2}{2} = \frac{b+a}{2}. \quad (7.51)$$

Moments of a
function are defined
as the expected value
of powers of the
differences between
the variable and its
expected value.

It is useful to introduce the so-called moments" of the distribution defined as

$$M_{\mu}^{(k)} = \int_{-\infty}^{+\infty} (x - \mu)^k P(x) dx. \quad (7.52)$$

It is easy to realise that $M_0^{(0)} = M_{\mu}^{(0)} = 1 \forall \mu$ and $M_0^{(1)} = \mu$. Moreover $M_{\mu}^{(1)} = 0$. The smallest interesting moment with respect to μ is

$$M_{\mu}^{(2)} = \int_{-\infty}^{+\infty} (x - \mu)^2 P(x) dx, \quad (7.53)$$

i.e., it is the expected value of $(x - \mu)^2$, $E[(x - \mu)^2]$. Its discrete counterpart is

$$M_{\mu}^{(2)} = \sum_{i=1}^N (x_i - \mu)^2 P(x_i) = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2, \quad (7.54)$$

that, for $\mu = \langle x \rangle$ coincides with our definition of variance of the population.

The variance of the sample, then, is an estimator of the second moment of the distribution with respect to its expected value and is a measure of its width. In fact $(x - \mu)^2$ is the square of the distance of x from μ and taking the integral estimated from the weighted by $P(x)$ is equivalent to compute the expected value of this quantity.

In other words, the variance of a distribution is the average distance squared it. of the values with respect to their expected value. Taking into account that $P(x) = 0$ for $x < a$ and for $x > b$, for a uniform distribution we have

$$M_{\mu}^{(2)} = \int_a^b (x - \mu)^2 P(x) dx. \quad (7.55)$$

Setting $y = x - \mu$, $dx = dy$ and

$$M_{\mu}^{(2)} = \int_{y(a)}^{y(b)} y^2 \frac{1}{b-a} dy = \frac{y^3}{3(b-a)} \Big|_{a-\mu}^{b-\mu}. \quad (7.56)$$

The limits of integration are

$$a - \mu = \frac{a - b}{2} \quad \text{and} \quad b - \mu = \frac{b - a}{2} \quad (7.57)$$

such that

$$M_{\mu}^{(2)} = \frac{(b - a)^2}{12}. \quad (7.58)$$

It is worth noting that the standard deviation of the population $\sqrt{M_\mu^{(2)}} = \frac{b-a}{\sqrt{12}} \simeq \frac{b-a}{3}$, as we empirically obtain in Chapter 4. The uncertainty associated to the reading error, in fact, comes from the fact that we have no reason to prefer a value or another within the interval between two consecutive ticks of the instrument scale. We are then forced to assign to each possible value in that interval a uniform probability, the standard deviation of which is the amplitude of the interval divided by $\sqrt{12} \simeq 3$.

7. Combining errors, revisited

Uncertainties sums in quadrature only if the measured quantities are uncorrelated.

In Section 6, Chapter 6 we show that uncertainties combine **in quadrature**, i.e., if $c = a + b$, $\sigma_c^2 = \sigma_a^2 + \sigma_b^2$. From the definition of the second moment with respect to the mean we can make a formal proof of such a result. In fact

$$\sigma_c^2 = E[(c - E[c])^2] = E[(a + b - E[a + b])^2]. \quad (7.59)$$

To simplify the notation lets' define $E[x] = \bar{x}$, such that $\sigma_c^2 = E[(c - \bar{c})^2]$. The expected value is a linear operator, then $E[a + b] = E[a] + E[b]$ and

$$\begin{aligned} \sigma_c^2 &= E[(a + b - E(a + b))^2] = E[((a - \bar{a}) + (b - \bar{b}))^2] \\ &= E[(a - \bar{a})^2] + E[(b - \bar{b})^2] + 2E(a - \bar{a})E(b - \bar{b}) \\ &= \sigma_a^2 + \sigma_b^2 + 2E[(a - \bar{a})(b - \bar{b})]. \end{aligned} \quad (7.60)$$

The covariance, defined as $E[(a - \bar{a})(b - \bar{b})]$ is null for uncorrelated variables, but not for correlated ones.

The last term in the sum $E[(a - \bar{a})(b - \bar{b})] = \text{Cov}(a, b)$ is called the **covariance** of a and b and

$$\begin{aligned} \text{Cov}(a, b) &= E[ab - a\bar{b} - b\bar{a} + \bar{a}\bar{b}] \\ &= E[ab] - E[a]E[b] - E[b]E[a] + E[a]E[b] = E[ab] - E[a]E[b]. \end{aligned} \quad (7.61)$$

By definition

$$E[a]E[b] = \int aP(a)da \int bP(b)db \quad (7.62)$$

and

$$E[ab] = \iint abP(a, b)dadb, \quad (7.63)$$

where $P(a, b)$ is the joint probability to observe a and b . If a and b are independent (or **uncorrelated**), $P(a, b) = P(a)P(b)$ and the covariance is null. The result found in Section 6 is then a direct consequence of the statistics of

uncorrelated random variables. On the other hand, it is only valid when the variables are uncorrelated. If they are not, we need to correct the error estimation including their covariance. Note that $\text{Cov}(a, a) = \sigma_a^2$. Moreover, being the average an estimator of the expected value,

$$\text{Cov}(a, b) \simeq \langle ab \rangle - \langle a \rangle \langle b \rangle, \quad (7.64)$$

that, for $b = a$, reads

$$\text{Cov}(a, a) = \sigma_a^2 \simeq \langle a^2 \rangle - \langle a \rangle \langle a \rangle, \quad (7.65)$$

providing a formal justification of the rule to estimate σ_a^2 . In general, then, if $c = a + b$,

$$\sigma_c^2 = \sigma_a^2 + \sigma_b^2 + 2\text{Cov}(a, b). \quad (7.66)$$

This rule can be easily extended to any combination of two variables $f(a, b)$. Suppose that we measured N values a_i and b_i , $i = 1, \dots, N$, from which we obtained their averages a and b . From these measurements we can obtain another quantity $f(a, b)$. For example, having measured the height $h = \frac{1}{N} \sum_i h_i$ of a freely falling body in a time $t = \frac{1}{N} \sum_i t_i$, we can compute the gravitational acceleration

$$g = f(h, t) = \frac{2h}{t^2}. \quad (7.67)$$

In fact, if a and b are both affected by errors, then so is $f(a, b)$ and, indeed, we can compute N values of the quantity as $f(a_i, b_i)$ (in the example we will have N values of g , $g_i = \frac{2h_i}{t_i^2}$).

The uncertainty on $f(a, b)$ can be estimated as the average of $f(a_i, b_i) - f(a, b)$. As usual, the uncertainty is defined as a positive quantity, then we could take the average of $|f(a_i, b_i) - f(a, b)|$ as the uncertainty, but the absolute value operator $|\dots|$ is difficult to manipulate. Better to evaluate σ_f^2 as the average of the squares of the differences, whose square root is positive by definition. For the sake of simplicity, let's define $f = f(a, b)$ and $f_i = f(a_i, b_i)$, then

$$\sigma_f^2 \simeq \frac{1}{N} \sum_{i=1}^N (f_i - f)^2 \quad (7.68)$$

and expanding f_i around f , truncating the expansion at the first order,

$$f_i \simeq f + \frac{\partial f}{\partial a} (a_i - a) + \frac{\partial f}{\partial b} (b_i - b) \quad (7.69)$$

The covariance between a and b can be estimated as
 $\text{Cov}(a, b) \simeq \langle ab \rangle - \langle a \rangle \langle b \rangle$.
 $\text{Cov}(a, a) = \sigma_a^2$
 variance of a random variable can be estimated as
 $\sigma_a^2 \simeq \langle a^2 \rangle - \langle a \rangle^2$

such that

$$\begin{aligned}\sigma_f^2 &\simeq \frac{1}{N} \sum_{i=1}^N \left(\frac{\partial f}{\partial a} (a_i - a) + \frac{\partial f}{\partial b} (b_i - b) \right)^2 = \\ &= \left(\frac{\partial f}{\partial a} \right)^2 \sigma_a^2 + \left(\frac{\partial f}{\partial b} \right)^2 \sigma_b^2 + 2 \frac{\partial f}{\partial a} \frac{\partial f}{\partial b} \left(\frac{1}{N} \sum_{i=1}^N (a - a_i)(b - b_i) \right).\end{aligned}\quad (7.70)$$

The last factor in parenthesis is nothing but an estimator of the covariance, quadrature, but if they are correlated we must

include the covariance
in the error
propagation.

$$\sigma_f^2 \simeq \left(\frac{\partial f}{\partial a} \right)^2 \sigma_a^2 + \left(\frac{\partial f}{\partial b} \right)^2 \sigma_b^2 + 2 \frac{\partial f}{\partial a} \frac{\partial f}{\partial b} \text{Cov}(a, b) \quad (7.71)$$

that refutes to the simple sum in quadrature when the covariance is null, i.e., the measurements are uncorrelated.

An important property of the covariance is that
between two variables

$$\begin{array}{ll} \text{is always less than or} & \text{Cov}(a, b) \leq \sigma_a \sigma_b. \\ \text{equal to the product of} & \end{array} \quad (7.72)$$

In fact, given that $\text{Cov}(a, b) = E[(a - \bar{a})(b - \bar{b})]$, its square is deviation:

$$\begin{aligned}\text{Cov}(a, b)^2 &= E^2[(a - \bar{a})(b - \bar{b})] = E^2[(a - \bar{a})(b - \bar{b})] \\ &\leq E[(a - \bar{a})^2]E[(b - \bar{b})^2] = \sigma_a^2 \sigma_b^2.\end{aligned}\quad (7.73)$$

The last step in the demonstration above comes from the fact that

$$E^2(ab) \leq E(a^2)E(b^2). \quad (7.74)$$

The proof of this property is not straightforward. Consider the random variable

$$\begin{array}{ll} E^2(ab) \leq E(a^2)E(b^2) & \\ \text{we build a random} & z = a - \gamma b. \\ \text{variable } z \text{ as a linear} & \end{array} \quad (7.75)$$

Being $b \neq 0$ then $E(z^2) \geq 0$ and

$$\begin{array}{ll} b; \text{ observe that} & \\ E(z^2) \geq 0 \text{ and find } \gamma & E(z^2) = E[(a - \gamma b)^2] = E[a^2 + \gamma^2 b^2 - 2ab\gamma] = \\ \text{such that the latter} & E(a^2) + \gamma^2 E(b^2) - 2\gamma E(ab) \geq 0 \\ \text{condition is true} & \end{array} \quad (7.76)$$

that is verified if

$$\gamma \geq \frac{E(ab)}{E(b^2)} + \Delta \quad \text{or} \quad \gamma \leq \frac{E(ab)}{E(b^2)} - \Delta, \quad (7.77)$$

Δ being the absolute value of the square root of the discriminant of the associated equation divided by $E(b^2)$. Choosing $\gamma = \frac{E(ab)}{E(b^2)}$,

$$E(a^2) + \left(\frac{E(ab)}{E(b^2)}\right)^2 E(b^2) - 2\frac{E(ab)}{E(b^2)} E(ab) = E(a^2) + \frac{E^2(ab)}{E(b^2)} - 2\frac{E(ab)}{E(b^2)} \geq 0, \quad (7.78)$$

from which

$$E^2(ab) \leq E(a^2)E(b^2), \quad (7.79)$$

leading to the result given above.

8. The binomial distribution

Let p the probability that an event occur. Because of the normalisation condition, the probability for that event to not occur is $q = 1 - p$. We want to compute the probability to observe n events in N trials. Since there are only two possible outcomes for this random variable (the event either occurs or not), the corresponding distribution is called **binomial**.

When events can be assigned to two classes, their distribution is binomial.

The simplest binomial experiment consists in tossing a coin, for which there can only be two results. The same happens to the decay of some subatomic particles. For example, the K particle decays (transforms) sometimes into $\pi\pi$, sometimes in $\pi\pi\pi$, while the τ lepton decays into *leptons* or *hadrons*. In general, any system for which one can divide the possible events in two classes follows the binomial distribution.

Using the binomial distribution we can tell the probability of observing a given sequence of events. A very common example is the following: how large the probability of observing n times “3” in a sequence of 10 launches of a dice is? A more physical example is the following. Consider a system of N electrons, whose spin, a quantum number, can only have two values: $s = \pm \frac{1}{2}$. Many measurable quantities of such a system depend on the number of electrons with *spin up*, i.e. $s = +\frac{1}{2}$. Being impossible to measure all the spins for any individual electron, we can only compute such a number statistically. To estimate the probability to observe n spins up we count how many states lead to each configuration and divide by the number of all the possible configurations.

In one configuration, $n = 0$ and all spins are *down* or $s = -\frac{1}{2}$. There are, then, N configurations for which $n = 1$. The only electron with spin up can be the first, the second, the third, ..., the N -th. The number of configurations with $n = 2$ is $\frac{N(N-1)}{2}$. For each of the N configurations with $n = 1$ there are $(N - 1)$ electrons that can have spin up, however two configurations are

In a binomial distribution of N events, the number of configurations in the system that have n events is given by the binomial coefficient:

equivalent (both have $n = 2$), so we have to divide by two. For $n = 3$ we have $\frac{N(N-1)(N-2)}{2 \times 3}$ and so on. In total there are

$$\binom{N}{n} = \frac{N!}{n!(N-n)!} \quad (7.80)$$

such combinations, where $N! = 1 \times 2 \times 3 \times \cdots N$ and $0! = 1$. If p is the probability for an electron to have $s = +\frac{1}{2}$, the joint probability of each single configuration is $p^n (1-p)^{N-n}$, then the total probability is

$$P(n, p) = \binom{N}{n} p^n (1-p)^{N-n}. \quad (7.81)$$

The average number of spin up electrons is

$$\langle n \rangle = \sum_{n=0}^N n \binom{N}{n} p^n (1-p)^{N-n}. \quad (7.82)$$

To compute it we observe that

$$(p+q)^N = \sum_{n=0}^N \binom{N}{n} p^n q^{N-n}, \quad (7.83)$$

and taking the derivative with respect to p we obtain

$$N(p+q)^{N-1} = \sum_{n=0}^N n \binom{N}{n} p^{n-1} q^{N-n}. \quad (7.84)$$

If we multiply both members by p , the right hand side of the equation reproduces our formula for the expected value:

$$Np(p+q)^{N-1} = \sum_{n=0}^N n \binom{N}{n} p^n q^{N-n}, \quad (7.85)$$

The mean of the binomial distribution is given by the product Np .

that, for $q = 1 - p$, becomes

$$Np = \sum_{n=0}^N n \binom{N}{n} p^n (1-p)^{N-n} = \langle n \rangle. \quad (7.86)$$

Similarly, we can compute the variance that is

$$M_\mu^{(2)} = Np(1-p). \quad (7.87)$$

Note that $\sigma = \sqrt{Np(1-p)}$ and that the uncertainty on n increases with N , but only as \sqrt{N} . The relative uncertainty is

$$\frac{\sigma}{\mu} = \frac{\sqrt{Np(1-p)}}{Np} = \sqrt{\frac{1-p}{Np}}, \quad (7.88)$$

i.e., it decreases as $\frac{1}{\sqrt{N}}$.

In order to study the properties of fundamental interactions, physicists make particles collide head-on and look at the products of the collision. One can define the *asymmetry* as

$$A = \frac{N_+ - N_-}{N_+ + N_-}, \quad (7.89)$$

where N_+ and N_- are, respectively, the number of events with a given property (e.g. the electric charge) identified by the + sign and the rest of them, identified with the – sign. Because of charge conservation, in $e^+ + e^-$ collisions (where the total charge is null), we expect the charge asymmetry to be zero.

Suppose that in an experiment we counted $N_+ = 4728$ positive and $N_- = 5021$ negatively charged particles. The charge asymmetry is then

$$A = \frac{4728 - 5021}{4728 + 5021} \simeq -0.03. \quad (7.90)$$

In fact, the experiment is equivalent to drawing a random variable with two possible states +1 and -1, $N = N_+ + N_- = 9749$ times, when $p = q = 0.5$.

The expected number N_+ of positive particles is then $Np = 4875$ with a standard deviation of $\sigma_+ = \sqrt{Npq} = 49$. Of course, the same values apply to N_- , being $q = p$. The uncertainty on the asymmetry is given by uncertainty propagation.

$$\sigma_{A_{th}}^2 = \left(\frac{\sigma_{N_+}}{N}\right)^2 + \left(\frac{\sigma_{N_-}}{N}\right)^2 = 5 \times 10^{-5} \quad (7.91)$$

hence, $\sigma_{A_{th}} = 0.007$. In other words, the expected value for A is

$$A_{th} = 0.000 \pm 0.007. \quad (7.92)$$

Our results differ from this by

$$\frac{A_{th} - A}{\sigma_{A_{th}}} = \frac{0.03}{0.007} \simeq 4, \quad (7.93)$$

i.e. we found a value for A that is four standard deviation out of theoretical predictions (physicists say that there is a *tension* between data and theory). In a successive box we find that such a tension, in fact, does not exist and is just the result of neglecting the fact that even the number of events counted in an experiment is affected by uncertainty.

We emphasise that no physicists will trust you claiming a discovery with such a discrepancy. Even if we observed a discrepancy of a bit more than 5 standard deviations, conventionally established as a threshold to claim a discovery, the discrepancy can be ascribed to many other effects that are not random at all and are called **systematics**. The apparatus used to make the measurements, for example, may not be fully or constantly efficient, for example. You are then required to prove that the discrepancy is genuine and that it does not come from systematic effects. This can be done repeating the experiments with different instruments and methods or studying the distribution of data as a function of the statistics, time, and other variables that in principle can influence the result.

Moreover, we did not consider N_+ and N_- to be affected by any uncertainty. Indeed, they are measurements, too, and as such have an associated uncertainty that must be evaluated. As a consequence $N = N_+ + N_-$ has an uncertainty as well as A . The uncertainties on this number is the topic of the section on Poisson distribution below.

9. The shape of the binomial distribution

In order to understand the shape of this distribution, we can exploit the ability `numpy.random` of Python to generate (pseudo) random numbers with various distributions.

Consider, e.g., the following code.

```
import matplotlib.pyplot as plt
import numpy as np

i = 1
binwidth = 1
while i < 40:
    x = np.random.binomial(100, i*1e-2, 100000)
    plt.hist(x, histtype = 'step', rwidth=1,
              bins=range(min(x), max(x)+binwidth,
                          binwidth), label='p = {}'
                          .format(i*1e-2))
    i += 10

plt.xlim(0,50)
plt.legend()
```

`binomial()` returns lists of random numbers distributed according to the binomial distribution.

```
plt.show()
```

The `np.random.binomial(N, p, m)` returns a list of m numbers according to a binomial distribution with parameters N and p . The expected value of these numbers is Np . Each number in the list is the number of times a binomial random variable assumes its “success” value in a series of N experiments.

At the first iteration, the script draws 100 000 numbers from a binomial distribution with $N = 100$ and $p = 0.01$, each representing the number of successful events occurring in 100 draws, when each event has a probability of 0.01. The expected number of those events is then $100 \times 0.01 = 1$. Manifestly, the possible values of the random number can be 0 (the event never occurs), 1, 2, ..., 100 (the event occurs in all draws). Since each event has a probability of 0.01 it is very unlikely that we can observe high values, while being 1 a relatively high probable value, observing 0 or 2 or some other value close to 1 should not be a too rare event.

Draws are repeated four times with different probabilities besides $p = 0.01$: $p = 0.1$, $p = 0.2$, $p = 0.4$. Correspondingly we expect the drawn values to be distributed around $Np = 10$, $Np = 20$, and $Np = 40$. It is worth noting that the variance (hence the standard deviation) grows with Np . The distribution of values is then expected to become larger.

The distribution can be observed making a histogram of the values, i.e. counting how many times a given number x is drawn from the distribution and plotting such a number as a function of x . This is done via

```
plt.hist(x, histtype = 'step', rwidth=1,
         bins=range(min(x), max(x)+binwidth,
                    binwidth))
```

The only mandatory parameter is `x`, a list containing all the numbers drawn from the distribution. The `histtype` parameter tells how to represent the histogram: data can be shown as bars, stacked bars, filled or unfilled steps. `rwidth` determines the fraction of each bin to be occupied by the bar or the step, while `bins` defines how the horizontal axis is divided in classes. In this case we chose to have one bin per value: `bins` is then a list of integer numbers from 1 to 101 (there is a simpler way to make such a list: we used this as it is very generic and can be used in many cases). Finally, `label` assigns a label to the data set. The labels are used to make the legend with `plt.legend()`.

Plots of the histograms are shown in Fig. 7.2. For low Np the distribution is asymmetric and narrow. It becomes wider and more symmetric as Np increases. It also moves to the right, showing a peak close to Np . The integral of the distribution is clearly equal to the number of values drawn, i.e. 100 000. Since the width of the distribution increases with Np as $\sigma = \sqrt{Np(1 - p)}$, its height decreases.

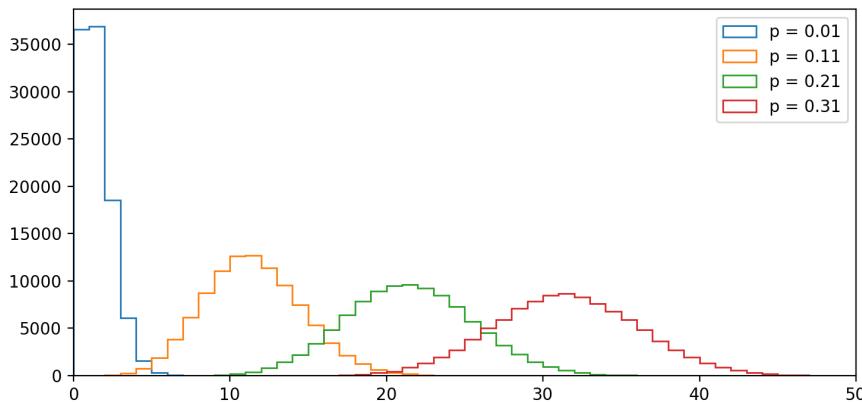


Figure 7.2: Histograms of numbers drawn from binomial distributions with $N = 100$ and various probabilities p

With Python we can easily compute the probability to have observed a number $n_+ < n_b p$ defined in the previous section.

Given a probability distribution $P(x)$, its **cumulative** distribution is defined as

$$C(a) = \int_{-\infty}^a P(x)dx. \quad (7.94)$$

In the case of a discrete distribution like the binomial one the integral becomes a sum, i.e.

$$C(a) = \sum_{i=0}^a P(n). \quad (7.95)$$

Clearly

$$\lim_{a \rightarrow \infty} C(a) = 1 \quad (7.96)$$

is a consequence of the normalisation condition and $C(a)$ is the area under the normalised distribution, i.e. the distribution divided by a constant such that its integral is one, for $x < a$. It represents the probability that $x < a$. For each random number generator defined in `numpy`, the corresponding theoretical distribution is defined in module `scipy`. To compute the value of $P(n, p)$ for $n = 4728$ when the probability $= 10\,000$, we can use the following code.

```
from scipy.stats import binom
P = binom.pmf(4728, 10000, 0.5)
```

variables it returns the
PDF $P(x)$. The
cumulative is given by
the `cdf()` method.

`pmf` stands for **probability mass function**. With this term we identify the function that gives the probability to find the value given as the first parameter in a set of random numbers following the given distribution with the given parameters (in this case $N = 10\,000$ and $p = 0.5$). For continuous distribution what makes sense is the **probability distribution function** $P(x)$, given by the `pdf()` method. In a certain sense PDF and PMF are synonyms and we may confuse them below.

The cumulative $C(4728)$ can be obtained with the method `cdf` (**cumulative distribution function**) as in

```
from scipy.stats import binom

P = binom.pmf(4728, 10000, 0.5)
C = binom.cdf(4728, 10000, 0.5)
print('P(4728; N=10000, p=0.5) = {:.2e}'.format(P))
print('sum_{n=0}^{n=4728} P(4728; N=10000, p=0.5) = ' + str(C))
```

The result is the following:

```
P(4728; N=10000, p=0.5) = 2.97e-09
sum_{n=0}^{n=4728} P(4728; N=10000, p=0.5) =
2.7974014420427173e-08
```

There is then a probability as low as $p \simeq 2.8 \times 10^{-8}$ to obtain such a value from a binomially distributed dataset with $N = 10\,000$ and $p = 0.5$.

Here we show how to format a string, too. The second `print` uses the `+` operator to join two strings: the first is a constant string, written within single quotes, while the second is the result of a conversion from a floating point number `C` into a string (i.e. its character representation) obtained with the casting operator `str()`. In general you can convert a variable or a constant from a type to another using operators identifying the final type like `int()`, `float()` or `str()`. The operation is called **type casting**.

When using the join operator `+`, the strings are represented as they appear by default using `print`. For example, if you call `print(C)` you will see `2.7974014420427173e-08`. The previous line of code **formats** the string according to a given **format**. Each pair of braces in the string is substituted with a value passed as an argument to `format()`: a method applied to the string. If no **format descriptors** are given in the braces, then the value is reproduced in the string as in its default representation, otherwise it is formatted accordingly. In general a format descriptor follows a colon `:` and is composed of an alphabetic character with optional preceding numbers. The alphabetic

character tells the system how to interpret the value passed in the `format()` method. There are many descriptors, the most common being `d` for integers (digit), `f` for floating point numbers, `s` for string and `e` to represent numbers using the scientific notation.

Optionally, the descriptor character can be preceded by one or more numbers depending on it. For `d`, the preceding number represents the number of characters with which the value is going to be written. For non integer numerical descriptors, the descriptor can be preceded by a length specifier in the form `n.m` where both `n` and `m` are integers. The first represent the total length of the field, while the second the number of digits after the decimal point to be shown. Consider the following examples.

```
print('n = {:d}'.format(10000))
print('n = {:8d}'.format(10000))
print('s = {:s}'.format('test'))
print('x = {:.f}'.format(np.exp(1)))
print('x = {:.3f}'.format(np.exp(1)))
print('x = {:.8.4f}'.format(np.exp(1)))
print('x = {:.5.1e}'.format(np.exp(1)))
```

Executing a script containing these lines, we should see something like

```
n = 10000
n =      10000
s = test
x = 2.718282
x = 2.718
x =    2.7183
x = 2.7e+00
```

In the latter case, for example, we asked to print the value of $e \approx 2.7$ using five characters, of which just one after the decimal point and in the scientific notation. The result, being longer than five characters, is represented aligned to the left. The previous line requires to write the same number using eight characters, of which four after the comma. Since the total number of characters needed to write it is six, two characters on the left are left blank.

10. Random walk

Consider a particle uniformly moving along the x -axis of a reference frame, starting from its origin. If such a particle is subject to random symmetric effects, it deviates by ± 1 in the y direction, with equal probability. After N steps along x , it can be found at coordinates (N, y) , where y is a random variable.

The mean coordinate
in a random walk of
length N is null, while
its variance is N .

$$\langle y \rangle = \sum_{i=1}^2 y_i P_i = +1 \times \frac{1}{2} - 1 \times \frac{1}{2} = 0, \quad (7.97)$$

while for the variance we get

$$\sigma_y^2 = \sum_{i=1}^2 (y_i - \langle y \rangle)^2 P_i = \sum_{i=1}^2 y_i^2 P_i = 1. \quad (7.98)$$

For N steps, then

$$\langle y \rangle = 0 \quad (7.99)$$

and

$$\sigma_y^2 = N, \quad (7.100)$$

because the average of a sum is the sum of the averages and the variance of a sum is the sum of the variances (see Section 6, Chapter 6). It is, in fact, straightforward to prove that the expected value is a linear operator, i.e., given a random variable x ,

$$E(ax + b) = aE(x) + b. \quad (7.101)$$

where a and b are constants. After N steps in the x direction, then, our walker, on average, will still be on the x -axis, on average. However, in n experiments, we will find that $y(N)$ is distributed with zero mean and a standard deviation of $\sigma = \sqrt{N}$. Each step in the walk can be considered as a binomially distributed random variable, having just two possible values. Defining a step in the positive direction a “success”, the average number of successes after N steps is $Np = \frac{N}{2}$ with a standard deviation $\sqrt{Npq} = \frac{\sqrt{N}}{2}$. The average coordinate will then be

$$\langle y \rangle = +1 \times P(y > 0) - 1 \times P(y < 0) = +1 \times \frac{N}{2} - 1 \times \frac{N}{2} = 0 \quad (7.102)$$

while

$$\sigma_y^2 = \sigma_{y>0}^2 + \sigma_{y<0}^2 + 2\sigma_{y>0}\sigma_{y<0} = \frac{N}{4} + \frac{N}{4} = \frac{N}{2} + 2\frac{N}{4} = N. \quad (7.103)$$

Indeed, the steps towards positive values of y and those towards negative ones, in this case, are fully correlated. Consider, in fact, a random variable $N = n_+ + n_-$

On av
walker

The
posi
after

(in the example N is the number of steps of the random walk, n_+ the number of steps in the positive direction and $n_- = N - n_+$ those in the opposite one). The variance of N can be written as

$$\sigma_N^2 = \sigma_{n_+}^2 + \sigma_{n_-}^2 + 2\text{Cov}(n_+, n_-). \quad (7.104)$$

Given that the total number of steps is fixed to be N , if there are n_+ steps with $y > 0$, those with $y < 0$ are $n_- = N - n_+$, i.e. $n_- = f(n_+)$ and the covariance attains its maximum value.

In order to verify this *experimentally* we can simulate a random walk using Python as follows. First of all we define a function that, given N , simulates N steps of a random walk with $dy = \pm 1$ as

```
def distance(N):
    d = 0
    for i in range(N):
        dy = 1
        if np.random.uniform() > 0.5:
            dy = -1
        d += dy
    return d
```

Python functions are Functions, in Python, are defined using the keyword **def** followed by the name defined using the given to the function with the list of needed parameters in parenthesis (if the keyword **def** followed function does not require parameters, a pair of parenthesis is mandatory). The by the name given to function declaration is terminated by a colon (:), while its body, representing the function with the the operations to be done on the parameters (if any), is written indented. The list of needed function returns zero or more values via the **return** keyword. It is worth noting parameters in that, contrary to C or C++, variables do not need to be declared in Python.

parenthesis. The body of the function is written indented on the following lines, after a colon (:). Moreover, in C and in C++, variables are always **local** to the **scope** in which they are declared, the scope being identified with the region of code comprised within a pair of braces. That means that a variable *a* defined in a function is different with respect to a variable with the same name defined in another function, irrespective of the fact that they share the name. Variables must be thought, in fact, as containers whose name is arbitrary and irrelevant, but for the programmer. There are as many containers as many declarations. Because of their *locality*, variables cannot be shared among scopes and it is forbidden to use a variable declared in another scope. Consider, for example, the following excerpt: [Variables in C and](#)

```
void loop() {
    int x = analogRead(A0);
    float h = 1.;
    Serial.print(x)
```

nested to it. Two variables with the same name declared in different scopes are different and can be used as if they had

```

if (x < 30) {
    float V0 = 5.;
    float h = V0/1023;
    x *= h;
}
Serial.print("x");
Serial.print(h)
Serial.print("=");
Serial.println(x);
}

```

It reads the value from the A0 analog input of Arduino and it multiply it by $\frac{5}{1023}$ if it is less than 30. Suppose, then, that x contains 51. Using the serial monitor we should see

51x1=51

If, on the other hand, $x = 23$, we would expect to see

23x0.0049=0.1127

but in fact what we get is

23x1=0.1127

The reason being that the variable h is redeclared in the scope of the **if**. Its content is correctly set to $\frac{5}{1023} \approx 0.0049$ and x is changed accordingly, however, exiting from the scope, that h no more exists and **Serial.print(h)** just prints the only accessible h whose value is 1.

Moreover, if we try to print V0, too, outside the **if** scope, the compiler returns an error, being it not existing outside that scope.

Needless to say, this cause a bit of confusion to beginners. Python overrides this behaviour and local variables follow rules that are somewhat more natural. Variables are always considered as local in a scope, but since it is not needed to declare them, their value is taken from those used above, if needed. Function parameters always imply, as in C and in C++, the declaration of local variables whose names coincide with those of the parameters. Consider the following example.

```

a = 3

def fun1():
    b = 2 * a
    return b

```

```

def fun2(a):
    a *= 2
    return a

def fun3(b):
    a = 7
    return a

print('{} {}'.format(a, fun1()))
print('{} {}'.format(a, fun2(a)))
print('{} {}'.format(a, fun3(a)))
print(a)

```

The first `print()` shows

3 6

In fact, `fun1()` multiply `a` by 2 and returns it. Since `a` is not found on the left side of an operator in `fun1()`, it is taken from the **global** scope and its value is assumed to be 3. The second `print` produces the same output. Here, `a` is considered as a parameter and it implies the creation of a new local variable `a` that has anything to do with the global one. In the function, its value is altered, but when we use it in the third `print()` its value is still 3, because the latter is a different container with respect to the one used in the function. In `fun3(b)` the parameter is called `b` and is not used. So, when we pass the content of `a` to the function, it is ignored: the function just assigns the value 7 to `a` and returns it. What we see on the screen is in fact

3 7

Again, even if the content of `a` is assigned in `fun3()`, the change does not reflect into the global `a`. In fact, being `a` on the left of an operator in the function, it implies the creation of a new local variable and the last `print()` shows that in fact `a` is still equal to 3.

This behaviour may sound odd at first glance, but in fact is very natural and does not require the programmer to consider the scope of a variable when using it. It is worth noting that Python allows using the keyword **global** to turn a local variable to be global, however we suggest to completely forget of it, as in the case of the `goto` statement in C: it exists, but using it qualifies the programmer as an uneducated novice.

To simulate $m = 100$ walks with the above function and compute their average and their standard deviation we can write another function as

```

def simulate(n):
    y = []

```

To generate a random event with a given probability p we generate a random number $x \in [0, 1]$. The event occur if $x < p$ or if $x > 1 - p$.

```

for k in range(100):
    y.append(distance(n))
return np.mean(y), np.std(y)

```

We are then ready to simulate random walks with different length, for example as

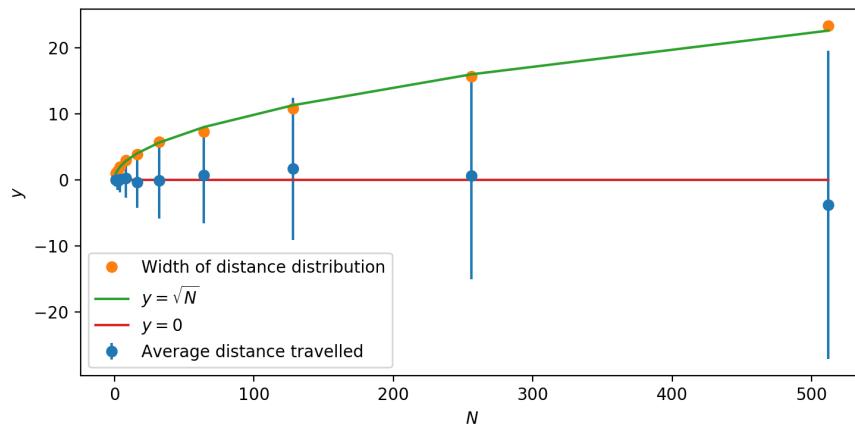
```

nsteps = 1
n = []
t = []
st = []
for k in range(10):
    ymean, sigma = simulate(nsteps)
    n.append(nsteps)
    t.append(ymean)
    st.append(sigma)
    nsteps *= 2

```

Exploiting functions, complex problems become easier and less error prone to implement, because we decompose them into smaller and simpler chunks.

At the first iteration we compute the average $\langle y \rangle$ and its σ after just $N = 1$ step, then after $N = 2, N = 4, N = 8, \dots, N = 1024$ steps. For each simulation we store N , $\langle y \rangle$ and σ in as many lists we can use to make the following plot



where blue dots represent $\langle y \rangle$. While $\langle y \rangle$ does not depend on N and is always close to zero, the standard deviation of the distribution (shown by the orange dots as well as by vertical bars whose amplitude is σ) grows as \sqrt{N} , as expected.

The random walk is a relatively common phenomenon in physics. As an example, consider the multiple scattering of a particle A after the collision with other particles, where the x -axis is defined as the direction of the initial momentum of particle A .

After N collisions, on average, the particle continues moving in the same direction of its initial momentum. However, the distribution of momenta of m particles will have a width proportional to \sqrt{N} : the higher the number of collisions, the larger the width.

This result can be used to estimate the density of scattering centres in a material, for example. When a relativistic particle of momentum p collides with a particle at rest, its momentum direction changes, on average, of an angle $\theta \propto \frac{1}{p}$ (the larger the momentum, the lower the scattering angle). After many collisions, then, the average angle between the initial momentum and the final one will be zero with a distribution whose width is proportional to

$$\sigma_\theta \propto \sqrt{\theta} \propto \sqrt{\frac{1}{p}}. \quad (7.105)$$

It is instructive to see alternative ways of computing $\langle y \rangle$ and its standard deviation. The distance traveled in the $+y$ direction is a binomially distributed variable: the probability of going in the positive direction is $p = \frac{1}{2}$ and after N trials the average number of steps with $y > 0$ is $E[n_+] = Np = \frac{N}{2}$. Exactly the same applies to n_- , the number of steps with $y < 0$. On average, then, the sum of positive and negative steps give

A random walk can be considered a sequence of binomially distributed random variables.

$$E[y] = E[n_+ - n_-] = E[n_+] - E[n_-] = \frac{N}{2} - \frac{N}{2} = 0. \quad (7.106)$$

The variance of positive steps is $\sigma_+^2 = Np(1-p) = \frac{N}{4}$, equal to the one of the number of negative steps. Remembering that $\sigma_x^2 = E[x^2] - E[x]^2$, we can write

$$\sigma_+^2 = \frac{N}{4} = E[n_+^2] - E[n_+]^2 = E[n_+^2] - \frac{N^2}{4} \quad (7.107)$$

and

$$E[n_+^2] = \frac{N}{4} + \frac{N^2}{4}. \quad (7.108)$$

The same result applies to $E[n_-^2] = \frac{N}{4} + \frac{N^2}{4}$. Similarly, with $y = n_+ - n_-$,

$$\begin{aligned} \sigma_y^2 &= E[y^2] - E[y]^2 = E[y^2] = E[(n_+ - n_-)^2] = \\ &= E[n_+^2] + E[n_-^2] - 2E[n_+ n_-] = \frac{N^2 + N}{2} - 2E[n_+ n_-]. \end{aligned} \quad (7.109)$$

By definition

$$E[n_+n_-] = \sum n_+n_- P_{n_+n_-} \quad (7.110)$$

where the sum is extended to all the possible values of n_+n_- and $P_{n_+n_-}$ is the joint probability to observe n_+ and n_- counts at the same time. Since $N = n_+ + n_-$, if we observe n_+ events with $y > 0$, the probability to observe $n_- = N - n_+$ events with $y < 0$ is equal to 1 and

$$\begin{aligned} E[n_+n_-] &= \sum n_+(N - n_+)P_{n_+} = N \sum n_+P_{n_+} - \sum n_+^2 P_{n_+} = \\ NE[n_+] - E[N_+^2] &= N \frac{N}{2} - \frac{N^2 + N}{4} = \frac{N^2 - N}{4}. \end{aligned} \quad (7.111)$$

Substituting in the equation for σ_y^2 ,

$$\sigma_y^2 = \frac{N^2 + N}{2} - 2 \frac{N^2 - N}{4} = N. \quad (7.112)$$

11. The Poisson distribution

In many cases the probability of success p in a binomial experiment is extremely low. In those cases, in order to observe a statistically significant number of events, N has to be large, such that $M = Np$ is not negligible. In this case, the binomial distribution tends to a continuous distribution called the **Poisson** distribution, named after Siméon-Denis Poisson (1781–1840) who introduced it. To obtain the Poisson distribution we need to let $N \rightarrow \infty$, keeping $p \rightarrow 0$ such that $Np = M = \text{const}$ and $p = \frac{M}{N}$.

The binomial distribution can be rewritten as

$$P(n, p) = \binom{N}{n} p^n (1-p)^{N-n}. \quad (7.113)$$

Substituting p it becomes

$$P(n, p) = \frac{N!}{n!(N-n)!} \left(\frac{M}{N}\right)^n \left(1 - \frac{M}{N}\right)^{N-n}. \quad (7.114)$$

and we take the limit to $n \rightarrow \infty$:

$$\lim_{N \rightarrow \infty} P(n, p) = \left(\frac{M^n}{n!}\right) \lim_{n \rightarrow \infty} \frac{N!}{(N-n)!} \left(\frac{1}{N^n}\right) \left(1 - \frac{M}{N}\right)^N \left(1 - \frac{M}{N}\right)^{-n}. \quad (7.115)$$

The first factor is

To evaluate a limit of a product, we compute the limit of each

$$\frac{N!}{(N-n)!} = \frac{N(N-1)(N-2) \cdots (N-n)(N-n-1)(N-n-2) \cdots 2 \cdot 1}{(N-n)(N-n-1)(N-n-2) \cdots 2 \cdot 1} \cdot \text{factor 1} \quad (7.116)$$

and only the terms $N(N-1)(N-2) \cdots (N-n+1)$ survives. There are n factors in this product, so this number grows as N^n . The product of this number times $\frac{1}{N^n}$, then, tends to 1 as $N \rightarrow \infty$.

Defining $x = -\frac{N}{M}$, the third factor in the limit,

$$\lim_{N \rightarrow \infty} \left(1 - \frac{M}{N}\right)^N, \quad (7.117)$$

becomes

$$\lim_{N \rightarrow \infty} \left(1 + \frac{1}{x}\right)^{-xM}. \quad (7.118)$$

Letting N going to infinity is equivalent to let x to go to infinity and

$$\lim_{x \rightarrow \infty} \left(\left(1 + \frac{1}{x}\right)^x\right)^{-M} = e^{-M}. \quad (7.119)$$

The latter term, when $N \rightarrow \infty$, approaches 1, hence

$$\lim_{N \rightarrow \infty} P(n, p) = \left(\frac{M^n}{n!}\right) \times 1 \times e^{-M} \times 1 = \frac{M^n}{n!} e^{-M}. \quad (7.120)$$

Such a probability does not depend on p anymore and is the probability density function for the Poisson distribution:

The Poisson probability mass function $P(M, n) = \frac{M^n}{n!} e^{-M}. \quad (7.121)$

The probability mass distribution is correctly normalised. In fact, given that the Taylor expansion of e^M is

observe n events,
given an expectation of M , is $\frac{M^n}{n!} e^{-M}.$ $e^M = \sum_{n=0}^{\infty} \frac{M^n}{n!}, \quad (7.122)$

it is straightforward to check that

$$\sum_{n=0}^{\infty} \frac{M^n}{n!} e^{-M} = e^M e^{-M} = 1. \quad (7.123)$$

We can thus use the same result to compute the mean and the variance of the distribution. The mean is

$M = Np$. Its variance

$$\sum_{n=0}^{\infty} n \frac{M^n}{n!} e^{-M} = M \sum_{n=1}^{\infty} n \frac{M^{n-1}}{(n-1)!} e^{-M} = M, \quad (7.124)$$

while the variance can be obtained as $\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$. We must then compute

$$\langle x^2 \rangle = \sum_{n=0}^{\infty} n^2 \frac{M^n}{n!} e^{-M}. \quad (7.125)$$

We write $n^2 = n^2 + n - n = n + n(n-1)$ and substitute:

$$\langle x^2 \rangle = \sum_{n=0}^{\infty} (n + n(n-1)) \frac{M^n}{n!} e^{-M} = \sum_{n=0}^{\infty} n \frac{M^n}{n!} e^{-M} + \sum_{n=0}^{\infty} n(n-1) \frac{M^n}{n!} e^{-M}. \quad (7.126)$$

The first term is just M , while the second can be rewritten, as before,

$$\sum_{n=0}^{\infty} n(n-1) \frac{M^n}{n(n-1)(n-2)!} e^{-M} = M^2 \sum_{n=2}^{\infty} \frac{M^{n-2}}{(n-2)!} e^{-M} = M^2. \quad (7.127)$$

Finally,

$$\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2 = (M + M^2) - M^2 = M. \quad (7.128)$$

The Poisson distribution represents the probability to observe n low probability events in an infinitely large sample, when the expected number of observed events is M . In general, data from any experiment in which something is counted follows the Poisson statistics, provided that the observed process is relatively rare.

This distribution is very useful in a number of cases. For example, radioactive decay is a process that happens on certain nuclei that spontaneously transform into a different species emitting particles like electrons (β rays) or helium nuclei (α particles). Given a sample of ^{60}Co , for example, you can observe a certain number of decays detecting β rays exiting the sample. The **activity** of the sample is defined as the number of decays per second and is measured in Becquerel (Bq, after Antoine Henri Becquerel - 1852–1908 - who discovered radioactivity). A typical value for a small radioactive source used in schools, for example, is under 200 kBq: it emits less than 200 000 particles per second, on average. The number of atoms in the sample is of the order of the Avogadro number $N_A \simeq 6 \times 10^{23}$. Radioactive decays happen randomly with a fixed rate. In other

Poisson
application
counting

Radioactivity
and decay
descriptions
Poisson

words, only a small number M of decays is observed out of an almost infinite number of possible events ($N_A \simeq \infty$), such that $p = \frac{M}{N}$ is small and constant.

With a radioactive source whose activity is 200 kBq, we expect to observe, on average, 2×10^5 counts per second, with an uncertainty $\sqrt{N} = 450$, i.e.

$$N = (2.0000 \pm 0.0045) \times 10^5 \text{ Bq}. \quad (7.129)$$

The real number of counts will likely be less, considering that the detector has an efficiency $\epsilon < 1$, the latter being the ratio between the detected particles and those that have passed through it. Also, the source emits particles in the full solid angle $\Omega = 4\pi$, while the detector only covers part of it: ω . The ratio $A = \frac{\omega}{\Omega}$ is called *acceptance* and the experimental number of counts is expected to be $N_{exp} = NA\epsilon$.

Consider the experiment outlined in Section 8, where we counted $N_+ = 4728$ and $N_- = 5021$ particles. We can assume that these numbers follow the Poisson statistics, since the production of a particle is a relatively low probability event happening during a long enough experiment.

The uncertainties of the numbers are then $\sigma_+ = \sqrt{N_+} = 69$ and $\sigma_- = \sqrt{N_-} = 71$ (note that we kept two significant figures in this case, while we usually keep only one of them, in which case they are the same).

As a result the errors on the numerator and on the denominator of the asymmetry are:

$$\sigma_n = \sigma_d = \sqrt{\sigma_+^2 + \sigma_-^2} \simeq 100 \quad (7.130)$$

(it is worth noting that, since uncertainties add in quadrature, the two standard deviations are the same). The variance on the ratio is

$$\sigma_A^2 = \left(\frac{\sigma_n}{N^2} \right)^2 + \left(\frac{N_+ - N_-}{N^2} \sigma_d \right)^2 = 1 \times 10^{-12} + 1 \times 10^{-7} \simeq 1 \times 10^{-7}, \quad (7.131)$$

and the standard deviation $\sigma_A = 0.0003$, negligible with respect to the one computed in Section 8. The assumption made there, that those numbers have no errors, is justified by this observation.

Another example of experiments following the Poisson statistics is the observation of cosmic [muons](#): the particles produced when high energy extraterrestrial protons [interact with the atmosphere](#) of the atmosphere, about 10 km above the ground. Their rate is [not too strong](#) but it varies slowly and their rate is well approximated [by Poisson statistics](#). Observing cosmic muons requires expensive and complex [detectors](#) called [muon telescopes](#), however there is a number of universities and [research centres](#) that make data collected by these instruments publicly [available](#) also through Apps for smartphone, giving public access to real [laboratories](#) around the world.

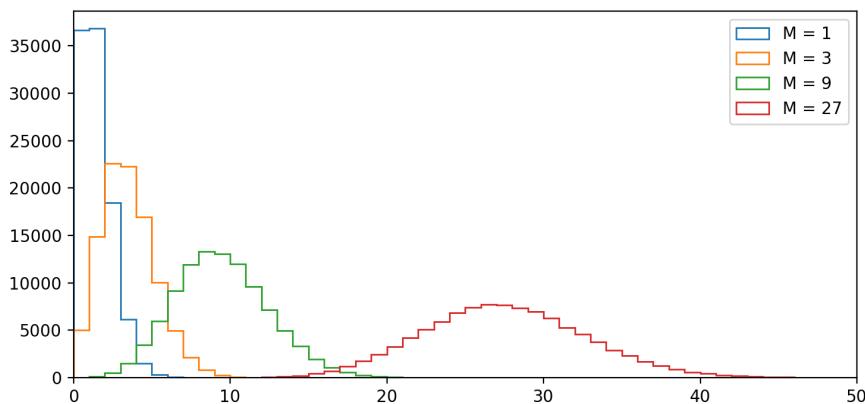
A ~~less exotic example~~ can be the distribution of the number of rain drops falling ~~on a given surface per unit time, or the number of cars passing through a toll booth or crossing a junction, provided that in all these cases the events are not too much~~ (~~eng.~~ rain, as well as traffic, must be light).

~~the Poisson statistics~~

Often, population statistics, political or market polls, etc. are well described by the Poisson distribution, but in few cases when the fraction of counted events is too high. The reliability of polls, for example, strongly depends on the size of the sample. If a party claims to have 12.2 % of consensus in a poll involving 1 000 voter, it means that 122 of them are willing to vote for it. However, such a number have an uncertainty of $\sqrt{122} \approx 11$, then the real consensus in the population is between $89/1\,000 = 8.9\%$ and $155/1,000 = 15.5\%$ with a **confidence level** of 99 %. Any change of few percent from week to week is then statistically not significant, unless there is a clear trend distributed over few weeks. In this case the binomial statistics applies, to compute the probability that the trend is positive or negative by chance, in a number of trials.

12. The shape of the Poisson distribution

Plots of the Poisson distribution for various values of M are shown below.



They are not much different from that of the binomial distributions. After all, they are related. As for the binomial distribution, the shape of the Poisson one becomes lower, wider and more symmetric as M increases.

The above plot has been made using the following script.

```
import matplotlib.pyplot as plt
import numpy as np
```

```
i = 1
binwidth = 1
while i < 30:
    x = np.random.poisson(i, 100000)
    plt.hist(x, histtype = 'step', rwidth=1,
              bins=range(min(x), max(x) + binwidth
                          , binwidth),
              label='M = {}'.format(i))
    i *= 3

plt.xlim(0,50)
plt.legend()
plt.show()
```

Using Python one can obtain the probability and the cumulative for a given observed values N and expected value M , as in

```
import numpy as np
from scipy.stats import poisson

Ntot = 9749
N = 4728
M = Ntot * 0.5
print('N = {}'.format(Ntot))
print('Expected value: {:.0f} +- {:.0f}'.format(M, np.sqrt(M)))
print('Observed value: {:.0f} +- {:.0f}'.format(N, np.sqrt(N)))
d = np.fabs(M-N)
sd = np.sqrt(M+N)
print('Difference : {:.0f} +- {:.0f}'.format(d, sd))
P = poisson.pmf(N, M)
C = poisson.cdf(N, M)
print('P({:.0f}, {:.0f}) = {:.2e}'.format(N, M, P))
print('C({:.0f}) = '.format(N) + str(C))
```

`pmf()` and `cdf()` define methods common to all the available distributions in the `scipy.stats` package. The result of the script is

```
N = 9749
```

corresponding probability mass function, as well as `cmf()` that returns the cumulative.

```
Expected value: 4874 +- 70
Observed value: 4728 +- 69
Difference      : 146 +- 98
P(4728, 4874) = 6.28e-04
C(4728)        = 0.01789369643918747
```

You are invited to compare the required format of the numbers and their appearance. The probability to observe 4728 events when 4874 are expected appears to be quite low, being 6.28×10^{-4} . In fact it is not so low, considering that the Poisson distribution is quite wide. The probability to observe any number individually is low because there are many different numbers for which the chance to observe them is relatively large. In fact, the cumulative of the distribution up to 4728 is 0.02, meaning that there is a probability of 2 % of observing a number lower than that (and, remember, 2 % is a large probability for a physicist, corresponding to more than two standard deviations).

Summary

The result of an experiment rarely is just a number. In most cases it is a set distributed over a range of values.

Data randomly fluctuate around their mean value, often taken as their *true value*. The latter is only an abstract, yet useful, concept to compare experimental data with predictions. Both data and predictions, in fact, are always affected by uncertainties.

Fluctuations in measurements are ascribed to random effects.

The uncertainty of a linear combination of variables $c = \alpha x + \beta y$ is the square root of its variance $\sigma_c^2 = \alpha\sigma_x^2 + \beta\sigma_y^2 + 2\alpha\beta\text{Cov}(a, b)$. If the variables are uncorrelated their covariance is null and the formula reduces to the sum in quadrature.

The mean position reached by a random walker after N steps is null, with a variance N .

Statistics There are various definitions of probabilities. The classical definitions are the combinatorial or analytical one and the frequentist one. In the first, the probability of an event is defined by the ratio between the favourable cases to the possible ones. In the frequentist approach, the observed frequency of events is taken as an estimation of the probability. Frequency and probabilities tend to match as the number of experiments tend to infinity.

The Bayesian probability is subjective. It must be

estimated from the current knowledge and can be updated as long as new information are available. In most cases the subjective probability coincides, as a matter of fact, with those computed from the frequentist or combinatorial approaches.

The probability for an event to occur irrespective of other events is called the prior or marginal probability.

The probability of an event that happens with certainty is equal to 1. That of an impossible event is 0.

The joint probability $P(A \text{ and } B)$ for the occurrence of two simultaneous events is $P(A \text{ and } B) = P(A)P(B)$ if the events are independent. If not, $P(A \text{ and } B) = P(A)P(B|A)$. The latter, $P(A|B)$, is the conditional probability of A given B has occurred.

$$P(A \text{ and } B) = P(A)P(B|A)$$

If A and B are independent on each other, then $P(B|A) = P(B)$.

Joint probabilities are symmetric, i.e. $P(A, B) = P(B, A)$.

If events are mutually exclusive, the probability that $P(A \text{ or } B) = P(A) + P(B)$. If not, $P(A \text{ or } B) = P(A) + P(B) - P(A)P(B|A)$.

$P(A \text{ or } B) = P(A) + P(B) - P(A)P(B|A)$
 If A and B are mutually exclusive, the probability $P(A \text{ and } B) = P(A)P(B|A) = 0$, then $P(A \text{ or } B) = P(A) + P(B)$.

The posterior probability of A is updated by new evidence B according to the Bayes' Theorem:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Bayes' Theorem:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

The expected value (or the mean) of a random variable x distributed according to the probability mass function $P(x)$ is defined as $E[x] = \sum_i x_i P(x_i)$. It can be estimated as the average. If the random variable is continuous, the sum becomes an integral and the probability mass function becomes the probability distribution function (pdf): $E[x] = \int x P(x) dx$. The pdf $P(x)$ represents the probability that the random variable lies between x and $x + dx$.

The n -th moment with respect to a value μ is defined as $M_\mu^{(n)} = \int (x - \mu)^n P(x) dx$.

The variance of the population is expected value of the second moment of its distribution with respect to the mean and can be estimated by the variance of the sample.

The variance of a uniform distribution is the amplitude of the interval of the possible values divided by 12.

The covariance of two random variables x and y is defined as $\text{Cov}(a, b) = E[(x - \bar{x})(y - \bar{y})]$, \bar{x} and \bar{y} being, respectively, the average values of x and y . It can be estimated as $\langle ab \rangle - \langle x \rangle \langle y \rangle$. As a consequence, the variance of x can be estimated as $\sigma_x^2 = \langle x^2 \rangle - \langle x \rangle^2$.

The covariance of two variables is always less or equal the product of their standard deviation: $\text{Cov}(a, b) \leq \sigma_a \sigma_b$.

The binomial distribution describes events with only two outcomes: success and failure. The mean of the binomial distribution is $\langle n \rangle = Np$ where N is the number of trials and p the probability of success. Its variance is $\sigma^2 = Np(1 - p)$.

The cumulative distribution $C(a)$ is defined as $C(a) = \int_{-\infty}^a P(x) dx$. It represents the probability that the random variable is less or equal to a .

The Poisson distribution represents the probability of rare events with probability p in large samples of size N . Its mean is $M = Np$, equal to its variance.

The Poisson distribution is not very different from the binomial one: it is asymmetrical and becomes wider and more symmetrical as its mean increases.

Python

Computers cannot provide random numbers, but deterministic lists of numbers having the properties of random ones: they are called pseudorandom.

The `numpy.random` package contains pseudorandom numbers generators.

A histogram is a plot of the number of times an event occur versus the event itself. It can be rendered graphically using `matplotlib.pyplot.hist()`.

`scipy.stats` is a package useful for dealing with probability distribution.

Strings can be concatenated in Python using the `+` operator. Numeric data can be converted into strings casting them using the `str()` function. Strings can be rendered according to a specified format, using the `format()` method of the `string` class.

To generate a random event with a given probability p , generate a random number $x \in [0, 1)$, then check if $x < p$.

It is useful to decompose a problems in small chunks and write a function for each of them. Python functions are defined using the keyword `def` followed by the name given to the function

with the list of needed parameters in parenthesis. The body of the function is written indented on the following lines, after a colon (:).

In Python new local variables are automatically allocated when they are found on the left of an operator. In all other cases, the content of a variable is taken from the outer scope.

Arduino

Variables in C and in C++ must be declared and

exist only within the scope in which they are declared and in those nested to it. Two variables with the same name declared in different scopes are different and can be used as if they had different names. Variables are containers and their names are only meaningful for the programmer.

Distributions defined in `scipy.stats` offer the `pmf()` method that returns the corresponding probability mass function, as well as `cumf()` that returns the cumulative.

Bibliography

- [1] Laplace, P. (1814) "Essai philosophique sur les probabilités". Cambridge Library Collection - Mathematics, pp. 1–67. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511693182.001
- [2] The OPERA Collaboration, "Measurement of the neutrino velocity with the OPERA detector in the CNGS beam", *J. High Energ. Phys.*, 2012, 93 (2012). [https://doi.org/10.1007/JHEP10\(2012\)093](https://doi.org/10.1007/JHEP10(2012)093)
- [3] "Interview of P. A. M. Dirac" by Thomas S. Kuhn and Eugene Wigner on 1962 April 1, Niels Bohr Library & Archives, American Institute of Physics, College Park, MD USA, www.aip.org/history-programs/niels-bohr-library/oral-histories/4575-1
- [4] Dirac, P.A.M. Pretty mathematics. *Int J Theor Phys* 21, 603–605 (1982). <https://doi.org/10.1007/BF02650229>
- [5] Maxwell, James Clerk. "A Treatise on Electricity and Magnetism". Oxford, Clarendon press, 1873.
- [6] "Mémoires de l'académie des sciences de Berlin" 6, 1752, pp. 185–217.

Chapter 8

Counting experiments

Any measurement consisting in counting events is analysed using the statistical distributions illustrated in the previous chapter, even in fields, such as economy or sociology, that have nothing to do with physics (at least as it is traditionally intended: remember that physics is useful in any field in which we measure something). For example, knowing how the number of car accidents depend on the age of the driver, his/her job, the town in which he/she resides, etc. is of capital importance for insurance companies to estimate insurance premiums. In many cases experiments consist in counting the occurrences of an event. This chapter is devoted to this kind of experiments.

1. Experiments with binomial and Poisson statistics

We can perform some experiment with the above mentioned distributions using data collected with PHYPHOX in Section 7, Chapter 6 using a smartphone at rest.

Data appear to be distributed around different values in the three directions. We can ask ourselves how often data lies beyond a certain threshold. In other words, we want to estimate the probability to observe $a_i > a_i^{th}$ where a_i represents the component of the acceleration measured along the i -th direction and a_i^{th} a given constant. This is an example of a binomial distribution, where we define a success the observation of an event in which $a_i > a_i^{th}$.

In order to increase the statistics we should consider a sample made merging all the data from the three directions and those of the modulus of the acceleration vector. Since each subsample has its own average and its own width σ_i , we can transform data such that they have zero mean and unitary standard deviation with

$$a_i \rightarrow \frac{a_i - \langle a_i \rangle}{\sigma_i} \quad (8.1)$$

It is straightforward to show that now a_i have zero mean and unitary standard deviation.

The collection of data contains now M measurements that can be thought as

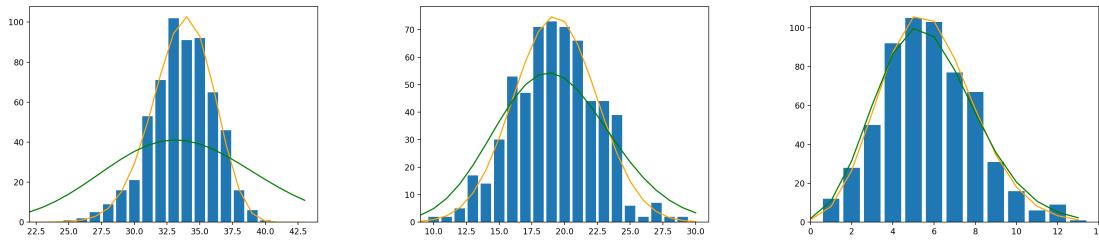


Figure 8.1: Distribution of counts of successes in experiments where the probability of success is (from left to right) 0.84, 0.50 and 0.16.

random variables. Dividing the measurements in groups of m is equivalent to perform $N = \frac{M}{m}$ experiments in each of which we measure m times a quantity and count how many times c_n that quantity exceeds a threshold c_0 , with $n = 1 \dots N$. We then make a histogram of the counts. From the histogram, the average number of successful events is

$$\mu = \frac{1}{N} \sum_{n=1}^N c_n = \frac{\sum_{n=1}^N c_n}{\sum_{n=1}^N n c_n} \quad (8.2)$$

and the probability to observe $c_n > c_0$ is $p = \frac{\mu}{m}$. If p is low, the distribution of the c_n is predicted to follow the binomial distribution with parameters N and p . As p tends to zero, the values of c_n tend to distribute as predicted by the Poisson distribution with mean μ .

Fig. 8.1 shows the distributions when c_0 has been set to -1 , 0 and $+1$, respectively, from left to right. Correspondingly, the expected probability p of success is expected to be 0.84, 0.50 and 0.16.

The superimposed orange curve is the shape of binomial distribution $P(n, p)$ with $N = \frac{M}{m}$, while the green one represents a Poisson distribution with mean Np . The binomial distribution always represents well the data, while the Poisson one do that only for low p .

2. Operations on lists

Data can be **normalised**, i.e. transformed such that their mean is null and their width is unitary, as follows.

```
import numpy as np

mu = np.mean(ax)
sigma = np.std(ax)
ax = [(x - mu)/sigma for x in ax]
mu = np.mean(ay)
```

```

sigma = np.std(ay)
ay = [(x - mu)/sigma for x in ay]
mu = np.mean(az)
sigma = np.std(az)
az = [(x - mu)/sigma for x in az]
mu = np.mean(a)
sigma = np.std(a)
a = [(x - mu)/sigma for x in a]

```

where `ax`, `ay`, `az` and `a` are lists built reading the CSV raw data file from PHYPHOX.

Merging the four lists in a unique one is as simple as

```
ax += ay + az + a
```

thanks to the **operator overloading** consisting in the fact that the result of the application of an operator to one or more operands depends on the nature of the operands. In this case operands are lists and the `+` operator consists in returning a list containing all the elements of its operands. Operator overloading is an important feature of object oriented programming languages.

The list `ax` now contains normalised data from all the columns of the CSV file. We can even shuffle the elements of the list using

```

import random

random.shuffle(ax)

```

The order of the elements of the resulting list is changed randomly with respect to the original one.

3. The Chauvenet's criterion

The observations of the previous section is the basis for the so called **Chauvenet's criterion**, named after William Chauvenet (1820–1870), sometimes used to make a decision as to whether to discard one or more measures in a set of many.

To apply the criterion in a set of N measurements x_i , $i = 1, \dots, N$, one computes

$$d_i = \frac{|x_i - \langle x \rangle|}{\sigma} \quad (8.3)$$

and compare it with a number D taken as a threshold. If $d_i > D$, x_i is considered as an outlier and discarded.

Lists can be joined thanks to the overloading of the `+` operator. Operator overloading is typical of OOP.

The Chauvenet's criterion consists in discarding data that deviates more than a given threshold with respect to the mean.

Generally speaking, rejecting data has to be considered as a bad practice in all experiments. Outliers, in fact, can sometimes be the sign of new physics and as such they must be carefully scrutinised to be absolutely certain that they can be ascribed to known effects. Indeed, new physics is often discovered as a tiny deviation from expectations. Discarding data may lead to loose a big opportunity.

As usual, also in the case of the Chauvenet's criterion, the decision to use it or not is a matter of how confident we are in the physics we are dealing with. A measurement of the gravitational acceleration using readily available materials can hardly lead to a discovery and the application of this rule is meaningful. The same rule applied to a collider experiment or to satellite observations is totally inappropriate.

In general, it is preferable to avoid adopting arbitrary decisions. If there is a couple of outliers in a set of measurements, their weight is of the order of $\frac{1}{N}$ on physics results. If they are important, the experiment has serious problems that must be corrected before proceeding. Hence, in general there is no need for using any rejection criteria. It is important to know about them, because sometimes they are used, but they are strongly discouraged.

4. Simulating advanced experiments

Many modern physiSeveral isotopes undergo radioactive decays consisting in the transformation of a experiments, especially nucleus in another one together with the emission of one or more particles. ^{14}C

those involving nuclei, for example, transform into ^{14}N emitting an electron and an antineutrino quantum mechanics ($^{14}\text{C} \rightarrow ^{14}\text{N} + e^- + \bar{\nu}_e$). This process can be characterised by the number requires counting the number of events as physicists developed the theory of weak interactions, originally formulated by function of several Enrico Fermi (1901–1954) in 1933.

variables (time, electric charge, direction, etc.) Our planet is constantly targeted by high-energy particles with a frequency, at sea level, of about 100 particles per square meter per second: the cosmic rays, discovered by Victor Hess (1883–1964) in 1912. This phenomenon can be for which we have to use the appropriate statistical distributions characterised by the number of particles as a function of the time, the surface, their electric charge, direction, etc..

These experiments need specialised, often expensive, detectors or collide head-on resulting in a number of secondary particles produced by the

When those conversion of the collision energy into matter, according to Einstein's relation instruments are $E = mc^2$. The number of the products, their angular and momentum distribution, missing, one cabin, etc. are investigated to understand the properties of the interactions simulate the events governing the processes.

practise with them All these experiments require advanced, expensive and complex detectors and cannot be easily implemented at home and in many laboratories. In this chapter

we simulate these kinds of processes.

In this section we show how to perform an experiment about radioactivity in environments where radioactive sources and detectors are not available, such as at home or in not equipped laboratories. This experiment, in fact, is instructive also in the case in which experiments with real radioactive sources can be done. In fact, while in the case of a real experiment, the behaviour of the source can be only inferred from results, in the case of a simulated experiment one knows exactly how it behaves and can then compare expectations with results, providing a way to *look inside* things that cannot be seen, such as atomic nuclei.

Radioactivity consists in the emission of particles from active atomic nuclei. There can be different types of radioactivity, historically classified as α , β and γ . α -particles are just helium nuclei. They are emitted by heavy isotopes, like bismuth (Bi), radon (Rn), radium (Ra), uranium (U), plutonium (Pu), americium (Am) and californium (Cf). α -particles are electrically charged with charge +2 in units of the proton charge. β -particles are nothing but electrons or positrons (their antiparticles, whose electric charge is the same of the proton). The most known β -emitter is ^{14}C , an isotope of carbon, used for dating ancient samples of living bodies. ^{60}Co , ^{137}Cs and ^{210}Bi are common radioactive sources found in laboratories. ^{22}Na is a β^+ emitter. Finally, γ -particles are photons. ^{60}Co , besides being a β -emitter, is a very common source of this type of radiation, too, often used in cancer radiotherapy.

The Geiger–Müller counter was one of the first instruments invented to detect radiation. It consists in a gas filled tube, with a conducting wire at the center kept at high voltage with respect to the walls. Charged particles passing through the gas ionise it, i.e. extract one or more electron from the gas molecule. Electrons and ions drift towards the wire and the walls, respectively, hitting other gas molecules producing secondary ionisation resulting in a detectable electric current flowing from the wire. An electronic circuit counts the electric pulses and show them on a display. Usually an audible *click* is produced, too.

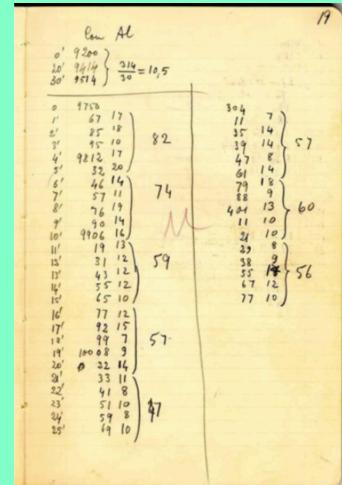
In this experiment we simulate the radioactive source with an Arduino producing random audible *clicks*, using a smartphone as a Geiger counter counting the latter. The details about the tools are given below. In this section we concentrate on the measurements.

The discovery of artificial radioactivity

In 1938 Enrico Fermi (1901–1954) was awarded the Nobel prize for his discovery of artificial radioactivity. In his experiments Fermi used a natural source of neutrons to irradiate samples of non-radioactive material. He then exposed recently irradiated materials to a Geiger counter to

check if they became radioactive.

The figure on the side shows the page of his logbook where he recorded the discovery, after irradiating a sample of Al. The first column reports the time in minutes since the beginning of the experiment. The second column is the number shown on the counter when the measurement were performed. Fermi reported the difference in counts between adjacent lines as a third column, then sum up them every five minutes. He measured 82 counts during the first five minutes, 74 in the next interval, 59 during the third, etc.. Counts drop exponentially, as expected for a radioactive source. The first three lines on the top left of the page is a measurement of the background: in 30 minutes he measured 314 counts with no irradiated samples close to the counter, for an expectation of about 10 counts per minute, such that one need to subtract 50 to each number in the last column to obtain the counts to be ascribed to the irradiated Al sample.



It is quite surprising to see that, in fact, a Nobel prize can be obtained with such a simple measurement. Fitting data with the techniques shown in this and the following chapters, one can easily obtain a decay time of 9 ± 3 minutes for a half-life of about 6 minutes. It is also remarkable the fact that the measurements were done using an extremely poor set of devices: the Geiger counter were built by Fermi himself using a pills' metallic container. Its amplified electrical signal drove a counter used in telephony to account for the duration of the calls. The whole, original data acquisition chain is still on exhibit at the Physics Museum of Sapienza Università di Roma.

The activity of a First of all, we need to assess the decay rate, also called **activity**. The activity radioactive source of a sample is measured as the number of decays per unit time. Its unit is the measures the number bequerel (Bq), after Henri Becquerel (1852–1908) who discovered radioactivity.

of decays per unit time. It is measured in bequerel (Bq). Another interesting quantity to study is how the activity changes over time.

The observed behaviour is described in terms of the **decay time** of the radioactive isotope. If N is the number of nuclei in a sample, the number of those who decays dN must be proportional to N and to the elapsed time dt , such that

$$dN = -\alpha N dt. \quad (8.4)$$

The solution of this equation is

$$N(t) = N(0) \exp(-\alpha t) = N(0) \exp\left(-\frac{t}{\tau}\right), \quad (8.5)$$

where $\tau = \frac{1}{\alpha}$ has been introduced as the **decay time** or the **lifetime** of the radioactive sample. The meaning of τ is the following: despite radioactivity is a random process, on average about one third ($1/e$) of the nuclei present at time $t = 0$ are still in the sample at $t = \tau$, as

$$N(\tau) = N(0)e^{-1}. \quad (8.6)$$

Lifetime is often expressed in terms of the **half-life** $t_{1/2}$: the time needed for a sample to reduce by one half. There is a precise connection between lifetime and half-life. In fact

$$N(t_{1/2}) = \frac{N(0)}{2} \quad (8.7)$$

and

$$N(t_{1/2}) = N(0) \exp\left(-\frac{t_{1/2}}{\tau}\right). \quad (8.8)$$

Comparing the two equations leads to

$$t_{1/2} = \tau \log 2. \quad (8.9)$$

The half-life $t_{1/2}$ of an element is related to its lifetime τ by $t_{1/2} = \tau \log 2$.

5. Using Arduino pins

In order to simulate a system in which atoms have a fixed probability ~~to decay~~
in a given interval of time Δt , we can write a program that generates ~~a random~~
number $x \in [0, 1)$, check if $x < p$ and, is so, let the atom decay, repeating the
above process in an infinite loop.

In order to generate a *click* sound, we use an **actuator**, namely a loudspeaker.
Actuators are usually driven by Arduino digital pins. The latter can have two
states: LOW corresponding to 0 V and HIGH corresponding to 5 V, and can be
used either as input pins or as output ones. If used as input pins, their status
can be read by Arduino that can tell if they are at low (0 V) or high (5 V)
voltage. When used as output pins, Arduino can set their state. Some of them
are marked with a *tilde* ~.

To prepare Arduino to drive a loudspeaker using a PWM pin we need to configure it as an output pin and set its value initially to 0, as follows.

```
void setup() {
    pinMode(3, OUTPUT);
    digitalWrite(3, 0);
    Serial.begin(9600);
```

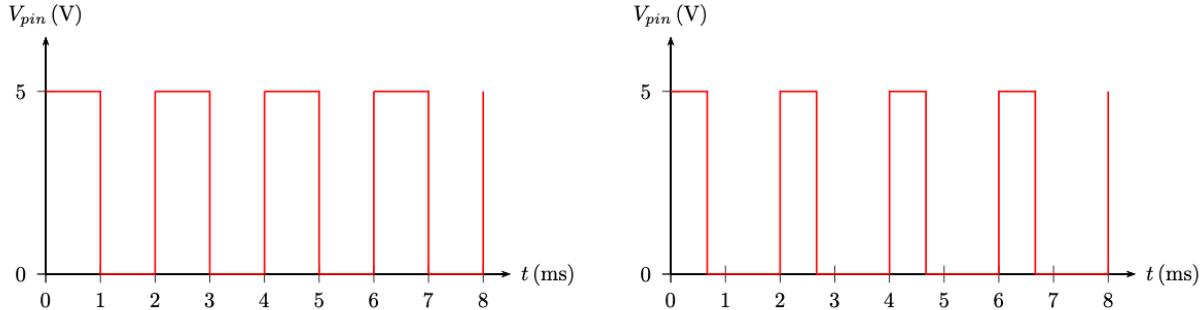


Figure 8.2: When the duty cycle of a PWM pin is set at 50 % it oscillates between 0 V and 5 V such that it remains in the HIGH state for 50 % of its time (left). When the duty cycle is 30 % the time spent in the HIGH state is reduced (right).

}

```
pinMode(pin, mode
) assigns the mode
mode to pin pin.
mode can either be
INPUT or OUTPUT.
```

The above piece of code shows how to setup a digital pin as an output pin. The mode of operation is set with `pinMode(PIN, OUTPUT)`, where the first parameter is the pin number. To set a pin as an input pin, just use `pinMode(PIN, INPUT)`. If set as output, the state of a digital pin can be set via `digitalWrite(PIN, VALUE)`. In this case the second parameter can either be `LOW` or `HIGH`. In the case of PWM pins, their status can be set as an integer between 0 and 255. The digital state of a PWM pin oscillates continuously between `LOW` and `HIGH` with a frequency that depends on the value to which they are set.

If set to 0, they are permanently in the `LOW` state and a measurement of its voltage always returns 0 V. Similarly, if set to 255 they stay permanently in the `HIGH` state and its voltage is 5 V. Assigning `The value of a PWM` 255, the pin stays in the `HIGH` state for a fraction $f = \frac{p}{255}$ of its time. For example, if $p = 128$, $f = \frac{128}{255} \approx 0.5$ and the pin remains in the `HIGH` state half of its time. In this case we say that the pin's duty cycle is 50 % and measuring the voltage of the pin as a function of time we see something like that shown in Fig. 8.2 left.

A duty cycle of 30 % can be achieved setting the pin to $p = 85$. In this case, a graph of the voltage as a function of the time is the one shown in Fig. 8.2 right.

The time actually spent in the `HIGH` state depends on the Arduino board, since it is a function of the PWM clock, and on the pin number. For an Arduino UNO the cycle has a duration of about 2 ms for pins 3, 9, 10 and 11. Pins 5 and 6 exhibit a frequency a factor two higher (then, the cycle duration is half of the latter).

PWM pins are then used when we need a square wave to drive an actuator, while non-PWM ones when we just have to switch between two values. Below, documentation, if you need to know it.

the `loop()` function is shown, using PWM pin 3 to produce a click sound.

```
void loop() {
    float p = (float)random(10000)/10000;
    if (p < 1.e-4) {
        digitalWrite(3, 127);
        delay(10);
        digitalWrite(3, 0);
    }
}
```

In order for a loudspeaker to emit a sound it is necessary to drive it with a variable voltage. The shape of the sound waveform almost strictly follows that of the driving voltage. The latter cannot be modulated on Arduino pins that can only assume two values: 0 V and 5 V. However, exploiting PWM pins we can put a non constant voltage on the loudspeaker contacts, such that it reacts and emit a sound as soon as the voltage changes. If the voltage changes rapidly, our ears cannot resolve the single pulses. In order to make a click-like sound, we can drive the loudspeaker for a very short time (10 ms) with a square wave at 50 % duty cycle (the exact values are manifestly irrelevant and you are invited to experiment with different values). After 10 ms, we silence the system by setting the voltage to zero.

The three lines of code that makes the sound are enclosed in a selection structure: they are executed only if p is less than 10^{-4} . The value of p is set randomly by means of the Arduino `random()` function. The latter returns an integer number picked randomly between 0 and the value passed as an argument (10 000 in our case), uniformly distributed. p , then, is a random number between 0 and 1. Note that, before dividing the number returned by `random()` by 10 000, we need to cast it into a `float`, otherwise, being the numerator always lower than the denominator, the ratio will always be null. In the end, the probability that $p < p_t$ is exactly p_t (note that if $p_t = 0$ the probability is null and if $p_t = 1$, the probability is 1). Hence, in order to execute a piece of code with a probability P , it is enough to draw a random number $0 \leq p \leq 1$ and check if $p < P$.

The above code makes a loudspeaker to emit a short sound, on average, once every 10 000 executions of the `loop()` function. The number of sounds emitted per unit time, then, is constant, similarly to what happen when a Geiger counter is placed close to a radioactive source whose activity is stable.

The activity of a radioactive source changes with time because, once decayed, nuclei can no more contribute to it. This can happen either if the source contains a very small number of radioactive nuclei or because many of them decay per unit time such that the available number of them decreases sensibly with time. The number of nuclei contained in a source, even in a small one, is always quite

In order to execute a piece of code with a probability P , it is enough to draw a random number $0 \leq p \leq 1$ and check if $p < P$. With Arduino, random numbers can be drawn using `random()`.

Global variables are shared among functions. They are declared outside any function block.

large (remember that one mole of substance contains a number of elementary units equal to the Avogadro constants $N_A \simeq 6 \times 10^{23}$; even if the abundance of the radioactive isotopes is small, their number is still huge). If, however, the nucleus' lifetime is short enough, one can appreciate the reduction of the activity of the sample that follows the same exponential law given above.

We can take that into account in our Arduino sketch, producing a click sound only with a given probability only for atoms not yet decayed. Atoms can be represented as an array of integers, that can be defined as a **global** variable. Global variables are variables declared outside the functions. They are shared among all the functions, unlike variables declared within them that can only be used in the function within which are defined.

We then add, before the `setup()` function, the following code.

```
#define N 750

int n = 0;
int atoms[N] = {0};
```

The `#define` directive, already encountered in Chapter 6, defines a constant `N` whose value is `750`. Note that, in this case, the assignment of the value to `N` is not done via the assignment operator `=` and there are no semicolons at the end of the line. A directive, in fact, is not a language statement. They are meant to simplify the program maintenance. In this case, it is used to define constants that are not susceptible to change during the execution of the program. Using constants have the advantage that the programmer can change them *only* in one place (usually at the top of the program). For the same reason we could add the following constants:

```
#define SPKRPIN 3
#define DUTYCYCLE 127
```

to define the pin to which the loudspeaker is attached and its duty cycle. Even the maximum integer to be drawn by `random()`, the duration of the pause and the probability with which nuclei decay can be defined through constants.

The integer variable `n` is used to count the elapsed time in units of “number of `loop()` executions”. It will be useful to observe the exponential decay of the number of clicks. `atoms` is defined as an array of integers. Arrays are declared in the Arduino language specifying their type (`int`), their name (`atoms`) and their size (`N`, equivalent to 750) between square brackets. With the declaration, a number of consecutive locations is reserved in Arduino's memory to accomodate its content. In this case, since an Arduino integer takes 16 bits (2 bytes) to be represented, $750 \times 2 = 1500$ bytes are used. Note that the available memory on an Arduino UNO is 2 kB. With this declaration, half of the available memory will be used to hold the array.

Once declared, the elements of an array can be assigned at the same time listing them within a pair of braces, separated by a comma. If the explicitly assigned integers are less than the size of the array, the rest of them are automatically assigned to zero. Note that, if no elements are assigned, no automatic assignment is made and there is no guarantee that all the elements will be zero at the time of the declaration. This is why we need to assign at least the first element to zero to be sure that all of them are zero. In our program each element of the array represents a radioactive nucleus. A value of zero indicates a not yet decayed nucleus. The loop() function is modified as follows.

and the third to 8. All

```
void loop() {
    n++;
    int k = 0;
    for (int i = 0; i < N; i++) {
        float p = (float)random(10000)/10000;
        if ((p < 1.e-4) && (atoms[i] == 0)) {
            digitalWrite(3, 127);
            delay(10);
            digitalWrite(3, 0);
            atoms[i] = 1;
            k++;
        }
    }
    Serial.print(n);
    Serial.print(" ");
    Serial.println(k);
}
```

Each time the function is executed, it increases the `n` counter: a measure of time. Then, it loops over each atom. If atom i has not yet decayed, the corresponding array element is zero. It can then decay with a probability of 10^{-4} . `&&` is a logical operator realising the AND boolean expression: the condition in the `if` statement is true only if both expressions (`p < 1.e-4`) and (`atoms[i] == 0`) are true (note the `==` operator, not to be confused with the `=` one). In this case, besides simulating the Geiger counter click, we update the corresponding array location setting it to 1 and increase the `k` counter. The latter is a **local** variable (i.e., it can be used only within the function in which it is defined) used to count the number of atoms decayed within a unit time (a single `loop()` execution). At the end of the loop over atoms we send both `n` and `k` over the serial line, such that we can monitor the number of decays as a function of time.

Even if the probability to decay is constant, the fact that the number of nuclei that can decay diminishes with time, produces an exponential decrease of the activity. In fact, the number of decays dN is proportional to time t and to the

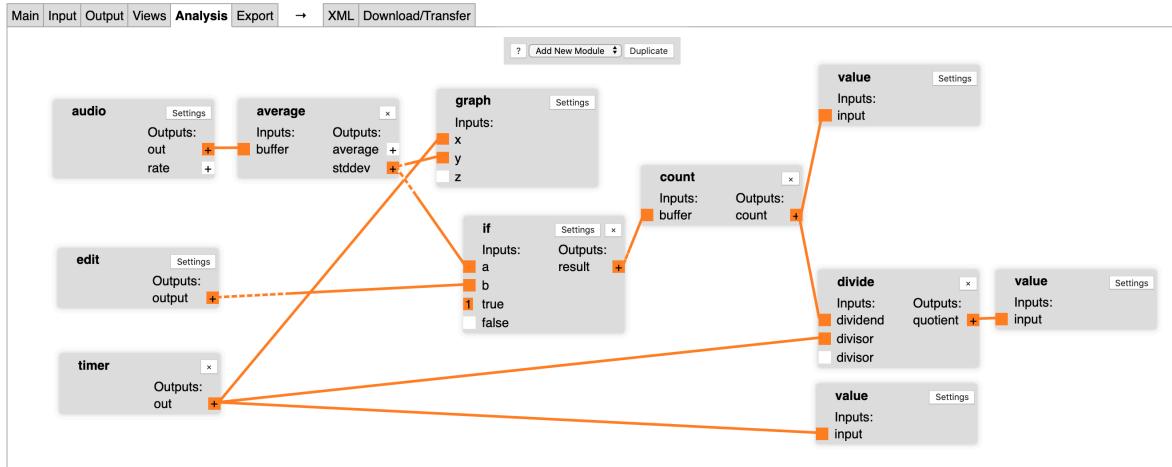


Figure 8.3: The PHYPHOX editor allows users to define their own experiments by manipulating data coming from input modules and showing the results to output modules.

number of not yet decayed atoms N , such that

$$dN = -\alpha N dt \quad (8.10)$$

and, as a consequence, $N(t)$ decreases as $\exp(-\alpha t)$.

6. The PHYPHOX editor

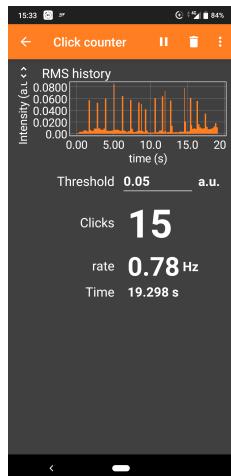
Custom experiments can be built with PHYPHOX using a web editor available at <https://phyphox.org/editor>. Understanding the PHYPHOX editor is beyond the scope of this textbook, however, a brief introduction to it is worth. Here we briefly describe an application used to simulate the behaviour of a Geiger counter using the smartphone's microphone as the detector. The description given below cannot be exhaustive, however, looking at how the experiment is realised using the editor provides enough information to start playing with it.

The microphone is defined as the data source, defining it as an input module in the INPUT tab. The VIEWS tab contains the definition of the output modules that can be seen on the smartphone display. Each element in this tab is represented by a rectangle in the ANALYSIS tab. Data sent to their inputs is rendered on the display of the phone. The core of the custom experiment is in the ANALYSIS tab. Here, modules and operators are represented by rectangles connected by lines (called buffers) representing the flux of data.

A program to count the number of clicks detected by the microphone is illustrated in Fig. 8.3. The example shows a possible way of realising a click counter. The “audio” input provides microphone data at its output as a list of numbers representing the amplitude of the sound waves. These data are used as input

for the “average” module that computes their standard deviation and send its value to both the graph module and to an “if” module. The graph module is an output module showing a graph of its inputs on the smartphone’s display. In order to plot the standard deviation of the input signal as a function of time, we send the output of a “timer” module to its x -input. The standard deviation is larger for louder sounds. The “if” module compares its a -input with its b one. The operation to be done is defined in its settings. In this case, if $a > b$, the output result is set to 1, specified as a fixed value in the “true” input of the module. Its b -input contains the value set as a threshold by the user, represented by the “edit” module, rendered as an input box on the smartphone display.

The “count” module just counts the number of its inputs and output the result to the “value” module, rendered as a number on the smartphone’s display. The same number n is used to compute $\frac{n}{\Delta t}$, Δt being the time elapsed since the beginning of the experiment obtained from the “timer” module. The ratio is shown on the smartphone’s display, too, together with its units defined in the settings. The elapsed time is also shown on the display. The result is shown below.



With such an experiment, each time the standard deviation of the audio input exceeds the threshold, the counter is incremented by 1 and shown on the display.

The performance of the system depend on the microphone sensitivity and on the smartphone processor’s speed. The higher the speed, the less time is spent in computing and showing the results, reducing the dead time. It also depends on the size of the audio input buffer: if it is too small, only a small fraction of the audio input is processed at each time; if it is too large, sound is averaged over a longer period. In the first case, the same *click* can be counted more than once, since the sound may persist for a certain time beyond the threshold. In the latter, many *clicks* may be counted just once. Such an effect can be regarded as an instrument limitation and contributes to the finite resolution and to the precision that can be attained with it.

The
desc
down
from
spr
In or
your
the f
PHY
and
QR—
with
by c
“plus
lowe
PHY

7. Readily available particle detectors

There exist, in fact, a number of Apps exploiting smartphone's cameras as particle detectors. Unfortunately (or, better, fortunately enough), most of them suffer from the fact that the sensitive element of a smartphone camera is extremely small, hence the experiment must last for a huge amount of time to observe few counts.

Most of these Apps can detect cosmic muons, but they are barely sensitive to most common products of radioactive nuclei, usually found in the environment. One of the most abundant radioactive isotopes, in fact, is ^{40}K whose activity is about 30 Bq per gram of potassium [Bin Samat, 1997]. Human body contains about 180 g of potassium, hence it is a source of $180 \times 30 = 5\,400$ Bq. ^{40}K emits β -rays of 1.3 MeV energy. Most of them are absorbed in the smartphone body before reaching the sensitive element and cannot be detected.

Public laboratories often release public data that can be used to make experiments similar to those described in this chapter. In particular, there are several cosmic ray observatories that publish data, sometimes together with a graphical rendering of their tracks. A comprehensive list is maintained by the DESY laboratory in Germany (<https://icd.desy.de/>) under their PROJECTS page.

A good alternative can be a DSLR camera. The sensitive element of the latter is about the size of its analog counterpart: $24 \times 36 = 864$ mm 2 , to be compared to the typical area of a smartphone's camera of the order of 100 mm 2 . Many DSLR camera allows to shoot pictures with a manual exposure. Among them various exposure settings there can be the Bulb (B) and the Time (T) exposures. The particle first opens the shutter when the shutter button is pressed and closes it when released; the second opens the shutter at the first pressure on the shutter button and closes it once the button is pressed again. With this latter option one can exposed take a very long shot of few hours keeping the camera in the dark with the cap enough. on the lens. Cosmic rays traversing the sensor leave a track on it. An example can be seen on Fig. 8.4, taken with an exposure of about 7 hours and saved as a JPEG image.

Each pixel of the sensor is made of three silicon photodiodes, each for one of the primary colours red, green and blue (RGB). The image is obtained as a combination of the intensities of each color. The colours of the hits in the picture depend on the energy deposited on each pixel by the cosmic rays. Most of the hits appear as small clusters of pixels, because the energy released by a cosmic ray particle in the sensor is distributed over an area a bit larger than the one of the single pixel and most of them cross the sensor with a relatively large angle with respect to the sensor surface. However, if a particle enters the sensor with a small angle with respect to its surface, it can leave a relatively long track in it. These events are rare, because the sensitive thickness of the

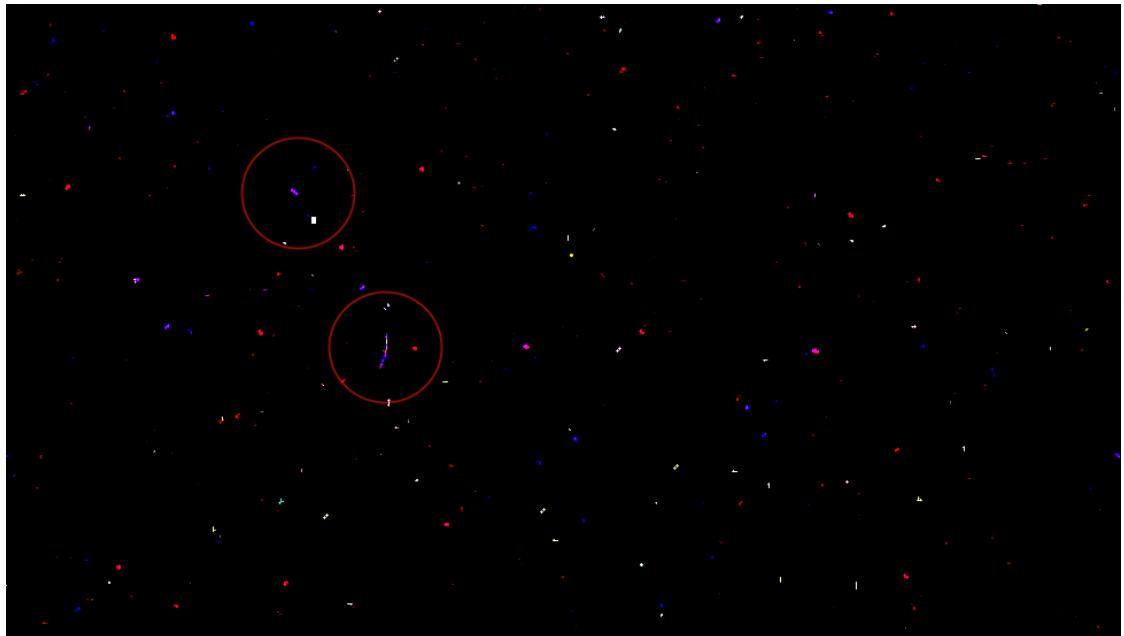
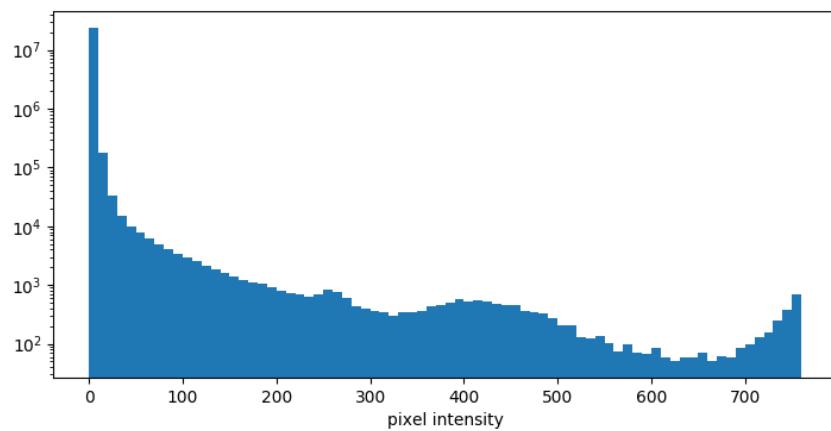


Figure 8.4: A small portion of a picture taken with a long exposure in the dark using a DSLR camera. With a little luck, you can even see tracks as long as the highlighted ones, corresponding to particles traversing the sensor with a small angle.

sensor is small (tens of microns) and the solid angle covered is a small fraction of 4π .

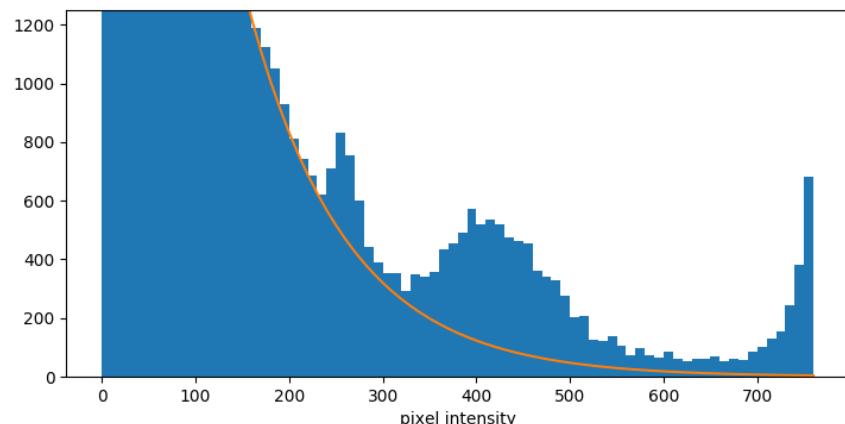
A histogram of the sum of the RGB pixels, defined as the intensity of each single sensitive element, is shown below for the full picture from which Fig. 8.4 has been extracted. The histogram is a plot of the number of times the pixel intensity is within a given interval.



The plot is in **logarithmic scale**. Such a scale is useful when the numbers span over a large interval. If we plotted the histogram normally, we could barely see its features: the first bins contains, in fact, about 10^7 events, while the counts in interesting ones is of the order of 10^3 or less and would be completely washed out. It is obtained plotting the logarithm of the counts as a function of the pixel intensity, using the counts to label the axis. In other words, for each value on the x -axis, we plot $\log y(x)$, while the ticks on the y -axis are labelled as $y(x)$.

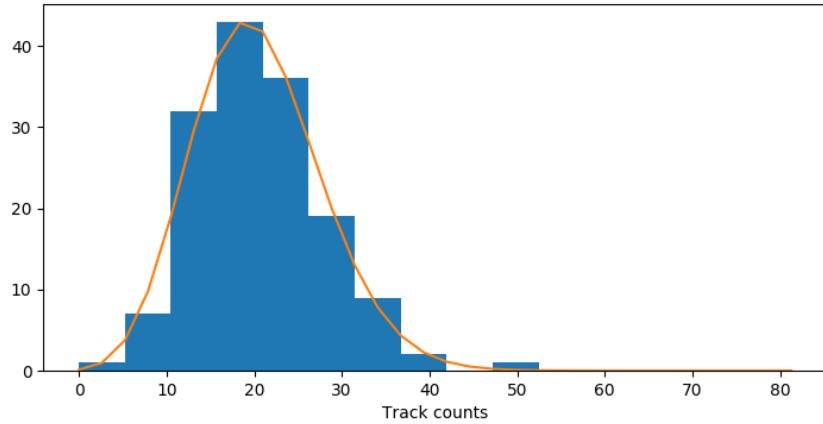
In this plot a long exponential tail can be seen on the left side. It is worth noting that an exponential function appears as linear in a logarithmic scale and that, in fact, the tail is described by a sum of exponentials. Such a tail is due to noise, mostly of thermal nature. Even if the pixel has not been hit by a particle, some current may be drawn from it. The energy gained by electrons in the pixel because of their temperature may fluctuate enough to make them free and be collected by the electronics.

Restricting the y -axis to values below a given value, as shown below, it is possible to better appreciate the features of the histogram, in a linear scale.



The exponential tail, shown in the picture, goes rapidly to zero. On top of it we can appreciate at least three peaks. Note that the first peak is located around 255, while the last is truncated, the maximum intensity for each photodiode being set to 255 and $255 \times 3 = 765$. The intermediate peak is naively expected at $255 \times 2 = 512$, but it is found around $255 + 127 = 382$ due to the way in which photodiodes are arranged in the sensor (as the so called "Bayer array"). They must correspond to some physics event happening in the silicon of which the sensor is made. In fact they can be ascribed to the energy released in the sensor by particles. Counting only the high energy hits (those on the right whose energy in units of pixel intensity is higher than about 600) allows to ignore the effects of the noise, that is negligible in this region.

The format of the picture is $4\,000 \times 6\,000$ pixels. Dividing the picture in chunks of 16×10^4 pixels, a distribution of the number of counts per chunk can be made, as shown below.



The average number of tracks found is $n = 20$, with a standard deviation $\sigma = 7$. The process can be described by the Poisson statistics, the probability p for a cosmic ray to leave a track being small, while the exposure is long (a long exposure corresponds to make a huge number of short experiments). If the distribution of the tracks follows the Poisson statistics, the ratio between σ and n must be

$$\frac{1}{\sqrt{M}} = \frac{\sigma}{n} = \frac{7}{20} \simeq 0.35, \quad (8.11)$$

resulting in $M \simeq 8$. A Poisson distribution with mean $M = 8$, properly normalised, is superimposed to the histogram. The distribution agrees with data, confirming that the process is following the Poisson statistics. The fact that the number of hits whose intensity larger than the threshold is, on average, $n = 20$ means that, on average, each cosmic ray fires $\frac{20}{8} = 2.5$ pixels.

To properly normalise the distribution we observe that the area under the histogram is given by

$$A[\text{px}] = \Delta w \sum_{i=1}^N n_i, \quad (8.12)$$

Δw being the bin width in units of pixels and n_i the number of events in bin i . In order to make A dimensionless, we need to divide it by the *calibration*

factor $c_0 = 2.5$ pixels, i.e. the average number of fired pixels per track. Then, the number of events (tracks) under the histogram is

$$A = \frac{\Delta w}{c_0} \sum_{i=1}^N n_i. \quad (8.13)$$

Since $P(N, M)$ is properly normalised to 1, it is enough to multiply it for A to obtain the shape shown in the figure.

8. Images manipulation with Python

Pillow is a useful Python module to work with images. It is not among the standard modules installed with the language and it has to be installed upon request. Installing external packages depends on the operating system, but it is often straightforward. Follow the appropriate instructions for your own operating system. On UNIX-like ones this is as simple as

```
pip3 install Pillow
```

in a terminal (the command prompt on Windows). Once installed it allows to play with image files. They are loaded into the memory using the `open()` method and several information about them can be easily accessed like

```
from PIL import Image
image = Image.open('tracksmall.jpg')

print(image.format)
print(image.size)
print(image.mode)
```

A picture of $n \times m$ pixels can be thought as an $n \times m$ -matrix. Each element of the matrix represents a pixel, that in turn can be represented as a list of three numbers in the interval $[0, 255]$, each one representing the intensity of the corresponding primary color red (R), green (G) and blue (B).

Using numpy, it is easy to cast any picture in a list of numbers to be interpreted as above:

```
data = numpy.asarray(image)
print(len(data))
print(len(data[0]))
print(len(data[0][0]))
```

The length of `data` is m . It can be thought to be made of m arrays, each of length n . The first array in `data`, `data[0]`, is then made of n elements, each of which is, in turn, a three-components array whose length is 3. In other words

`data[i][j]` is an array of three components containing the intensities of each primary color of the pixel at coordinates (i, j) .

To loop over the pixels and count the number of those whose intensity is large enough we can use the following code.

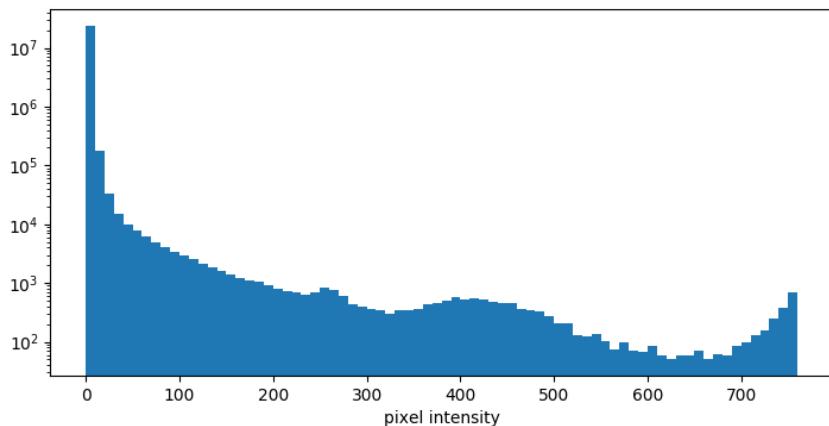
```
i = 0
N2 = 16000
k = 0
c = []
p = []
threshold = 600
for row in data:
    for pixel in row:
        i += 1
        if (i % N2) == 0:
            c.append(k)
            k = 0
        h = int(pixel[0]) + int(pixel[1]) +
            int(pixel[2])
        p.append(h)
        if h > threshold:
            k += 1
```

Here, `row` and `pixel` are *iterators* traversing the whole data structure and returning the appropriate type. `data` is made of arrays, then `row` is an array. As such, we can iterate over it and since it contains arrays, too, even `pixel` is an array. The variable `i` is a local counter holding the index of each pixel. The modulus operator `%` returns the remainder of the division between `i` and `N2`. Every time `i` is an integer multiple of `N2` the remainder is zero and we reset the `k` counter after appending its last value to a list.

`h` contains the sum of the values of each subpixel. The casting to an `int` is needed because each subpixel is in fact defined as a `byte`, an 8-bits integer whose maximum value is 255. Summing two or more bytes may result in a number greater than 255 and the result cannot be accommodated in a byte. All the values are appended to the list `p` and if `h` exceeds the threshold, we increment `k` by one. The latter, then, represents the number of pixels whose intensity is above the threshold in the given chunk, whose distribution is reproduced below for convenience.

Lists
using

The
ret
of
op



The **spectrum** of the pixel intensities, i.e. their distribution, can be made in a logarithmic scale as

```
plt.hist(p, bins)
plt.yscale('log')
plt.xlabel('pixel intensity')
```

The **logarithmic scale** has been selected using the `yscale()` method of `matplotlib.pyplot`. The default behaviour is restored using 'linear' as the argument for `yscale()`. The same applies to the horizontal axis for which the `xscale()` method is used.

The relevant measures from the plot are easy to compute, remembering that `c` contains the number of hits found in each chunk of the picture.

```
avg = np.average(c)
sigma = np.std(c)
print('Average = {}'.format(avg))
print('RMS = {}'.format(sigma))
m = (avg/sigma)**2
print('Mean = {}'.format(m))
cal = avg/m
print('Calibration = {}'.format(cal))
```

Finally, the plot of the Poisson distribution on top of the experimental one can be made using the following code.

```
ch, bins, patch = plt.hist(c, bins = np.arange(0,
                                              cal*m*4, cal * 2))
binwidth = bins[1] - bins[0]
S = sum(ch)
```

```
xx = range(int(math.ceil(m*4)))
yy = poisson.pmf(xx, m)
plt.plot(xx*cal, S*yy*binwidth/cal, '-')
plt.xlabel('pixel counts')
```

The first line makes a histogram of the content of `c` defining the classes as an interval of amplitude equal to twice `cal`, starting from 0 and up to four times `m` (the rightmost point on the x -axis corresponds to $4c_0M$). The content of each bin is returned as a list `ch`, together with a list containing the bins' limits and a list of *patches* (a set of properties describing the graphical appearance of the histogram).

From `bins` we get the bin width and from the the content of the histogram we obtain the sum of them, such that we can compute the integral, to be used in the plot of the Poisson distribution below.

Summary

Normalisation consists in shifting and stretching data such that their mean is null and their standard deviation is unity. The transformation is $a_i \rightarrow \frac{a_i - \langle a_i \rangle}{\sigma_i}$.

The Chauvenet's criterion consists in discarding data that deviates more than a given threshold with respect to the mean. To be used with great care and *cum grano salis*.

Even if we can make experiments with the appropriate instruments, it is often useful to simulate it because in this case the behaviour of the system is perfectly known and one can compare directly with the results of the experiment.

The activity of a radioactive source measures the number of decays per unit time. It is measured in bequerel (Bq).

Radioactive decay is yet another example of exponential law, characterised by the decay time (also called lifetime). The lifetime τ of a sample is related to its half-life $t_{1/2}$ by

$$t_{1/2} = \tau \log 2$$

To generate an event with a fixed probability p , generate a uniformly distributed random number in the interval $[0, 1)$, then check if it is less than p .

There are many sources of natural radioactivity in the environment. Even our body is radioactive, due to the presence of ^{40}K .

Public data about cosmic rays are released by many laboratories. Check out the list of them on the International Cosmic Day website at <https://icd.desy.de/>.

Camera sensors are good particle detectors, provided they are large enough and can be exposed long enough. They may be affected by thermal noise consisting in fluctuations of the intensity of each pixel due to thermal agitation.

Logarithmic scales are used when data span over several orders of magnitude. In logarithmic scales one plot $\log y$, while labelling the axis with the values of y . In this scale an exponential tail looks like a straight line.

Python

Lists can be joined thanks to the overloading of the `+` operator. Operator overloading is typical of OOP.

Lists can be traversed using iterators. They are more efficient than navigating the list using their index.

Axis in logarithmic scales can be done using the `xscale()` and/or `yscale()` methods.

The `%` operator returns the remainder of the division of the operands.

`pip3` is the package installer for Python-3 (the latest version of Python used at the time this book was written). It simplifies the management of installed packages. To install a package use `pip3`

install <package name>.

Arduino

`pinMode(pin, mode)` assigns the mode `mode` to pin `pin`. `mode` can either be `INPUT` or `OUTPUT`

.

To set the state of a digital pin use `digitalWrite(pin, state)`, where `state` can either be `LOW` or `HIGH`.

The state of a PWM pin can be set to a value between 0 and 255. It then oscillates between the two states with a duty cycle proportional to its value.

The time actually spent by a PWM pin in the `HIGH` state depends on the board and on the pin number. Check the documentation, if you need to know it.

In order to execute a piece of code with a probability P , it is enough to draw a random number $0 \leq p \leq 1$ and check if $p < P$. With Arduino, random numbers can be drawn using `random()`.

Global variables are shared among functions. They are declared outside any function block.

Constants are defined using the `#define` directive. No assignment operator, nor a semicolon

must be used to this purpose. Constants make program maintenance simpler.

Arrays are data structures composed of an ordered list of variables of the same type. Each element of an array is indexed by an integer.

Values to elements of an array can be assigned at the declaration time as in `int array[10] = {4, 12, 8};` where the first element `array[0]` is equal to 4, the second to 12 and the third to 8. All the other elements are automatically set to zero. To access the i -th component of an array, specify its index in square brackets as in `array[i]`.

The `&&` operator realises the AND logical operator: an expression A AND B is true if and only if both A and B are true.

Do not confuse the assignment operator `=` with the comparison one `==`.

phyphox

Custom experiments can be added to the PHYPHOX collection using its web editor. Once defined, they can be loaded on the phone framing the QR-code generated on the editor.

Bibliography

- [1] Supian Bin Samat *et al.* (1997) "The ^{40}K activity of one gram of potassium" Phys. Med. Biol. 42 407

Chapter 9

The normal distribution

When the number of trials increases in the discrete distributions illustrated in the previous chapters, the probability mass distribution tends to become symmetric around a central value. For large enough numbers the shape of these distributions are not far from what we observe in many experimental data distribution. They also become wider and the width of each single bin becomes negligible with respect to the width of the distribution, such that the latter can be considered to be a continuous distribution. In this chapter we study the properties of the Gaussian distribution, a distribution of capital importance in statistics, mostly because it is the one to which all other distributions tend, as stated by the central limit theorem. We also learn other important results.

1. A distribution depending on the distance

Uncertainties in physics represent the average distance between the central value and the rest of the measurements. If they are ascribed to random fluctuations, they must be distributed as random distances. Let's then derive the probability density function for a random variable $r = \sqrt{x^2 + y^2}$ with null average, assuming that

- the probability depends only on r ;
- the probability decreases with r .

Indicating with $P(x)$ and $P(y)$ the probability density function of x and y , respectively, the probability of finding $r = \sqrt{x^2 + y^2}$ is the product of the probabilities to find x and y , i.e.

$$P(r) = P(x)P(y). \quad (9.1)$$

We can always represent r in a cartesian plane as a vector starting from the origin of length r and with coordinates $x = r \cos \theta$ and $y = r \sin \theta$. Then,

Physics data usually distribute such that the probability of a measurement x_i to be different from the *true* value μ decreases with the distance $|x_i - \mu|$. Finding the distribution of random distances whose probability decreases with it is a way to understand the origin of the distribution of physics data.

differentiating both sides of this equation,

$$0 = \frac{dP(x)}{dx} \frac{dx}{d\theta} P(y) + P(x) \frac{dP(y)}{dy} \frac{dy}{d\theta} = -P(y) \frac{dP(x)}{dx} r \sin \theta + P(x) \frac{dP(y)}{dy} r \cos \theta \quad (9.2)$$

that can be rewritten as

$$xP(x) \frac{dP(y)}{dy} = yP(y) \frac{dP(x)}{dx}. \quad (9.3)$$

Separating the variables we get

$$xP(x) \frac{dx}{dP(x)} = yP(y) \frac{dy}{dP(y)}. \quad (9.4)$$

Since the equality must be true for all x and all y , such ratios must be constant, i.e.

$$xP(x) \frac{dx}{dP(x)} = yP(y) \frac{dy}{dP(y)} = A. \quad (9.5)$$

Defining $C = \frac{1}{A}$,

$$\frac{dP(x)}{P(x)} = Cx dx. \quad (9.6)$$

Integrating both members we obtain

$$\log P(x) + c = \frac{C}{2} x^2 \quad (9.7)$$

from which we can conclude that

$$P(x) = P_0 \exp \left(\frac{C}{2} x^2 \right). \quad (9.8)$$

Because the probability must decrease with r ,

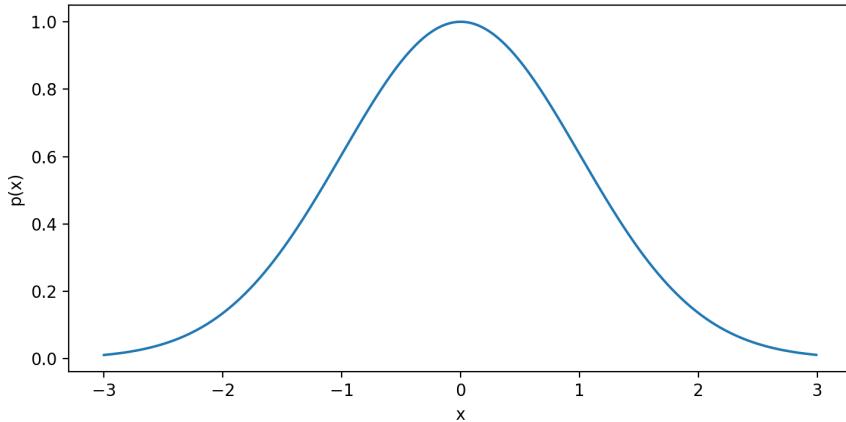
probabilities are always

$$\text{non-negative and } \frac{dP(x)}{dr} \frac{dx}{dr} = \frac{dP(x)}{dx} \cos \theta < 0. \quad (9.9)$$

For positive x , $\cos \theta > 0$ and $\frac{dP(x)}{dx} < 0$, and, as a consequence, $C < 0$, because $P(x) \geq 0$, being a probability density function. It is convenient to write $C = -k$ with $k > 0$ and rewrite

$$P(x, k) = P_0 \exp \left(-\frac{k}{2} x^2 \right), \quad (9.10)$$

where we added k among the parameters upon which $P(x)$ depends, its shape depending on it (it depends on P_0 , too, but below we show that P_0 depends, in turn, on k). The function $P(x, k)$ attains its maximum at $x = 0$, it is symmetric with respect to $x = 0$ and its shape is shown below for $k = P_0 = 1$.



This shape is interesting because it resembles very much the frequently observed distribution of experimental data. The mean of such a distribution is, by definition,

$$\mu = \int xP(x, k)dx = 0 \quad (9.11)$$

because $xP(x, k)$ is an odd function. The variance is given by

$$\sigma^2 = \int (x - \mu)^2 P(x, k)dx = \int x^2 P(x, k)dx = P_0 \int x^2 \exp\left(-\frac{k}{2}x^2\right)dx. \quad (9.12)$$

Let's observe that

$$d \exp\left(-\frac{k}{2}x^2\right) = -kx \exp\left(-\frac{k}{2}x^2\right)dx. \quad (9.13)$$

We can then integrate by parts writing $z = \exp\left(-\frac{k}{2}x^2\right)$ such that

$$\sigma^2 = -\frac{P_0}{k} \int x dz = -\frac{P_0}{k} \left(xz \Big|_{-\infty}^{+\infty} - \int z dx \right). \quad (9.14)$$

The first term in the parenthesis is null, since z goes to zero faster than x , as can be clearly seen from its graphical representation and

$$\sigma^2 = \frac{P_0}{k} \int \exp\left(-\frac{k}{2}x^2\right) dx. \quad (9.15)$$

The normalisation of $P(x)$ imposes that

$$P_0 \int \exp\left(-\frac{k}{2}x^2\right) dx = 1 \quad (9.16)$$

and

$$\sigma^2 = \frac{1}{k}. \quad (9.17)$$

The value of P_0 can be evaluated observing that

$$\left(\int \exp\left(-\frac{k}{2}x^2\right) dx \right)^2 = \frac{1}{P_0^2} = \int \exp\left(-\frac{k}{2}x^2\right) dx \int \exp\left(-\frac{k}{2}x^2\right) dx. \quad (9.18)$$

Making the substitution $y = x$ in one of the integrals we can rewrite the equation as

$$\frac{1}{P_0^2} = \int \exp\left(-\frac{k}{2}x^2\right) dx \int \exp\left(-\frac{k}{2}y^2\right) dy = \int \int \exp\left(-\frac{k}{2}(x^2 + y^2)\right) dx dy. \quad (9.19)$$

We can compute the last integral in polar coordinates, observing that $dx dy = r dr d\theta$ such that the integral can be rewritten as

$$\frac{1}{P_0^2} = \int \int \exp\left(-\frac{k}{2}r^2\right) r dr d\theta, \quad (9.20)$$

where the integrals are extended in r from 0 to ∞ and in θ for 0 to 2π . If we write $r^2 = u$, $2rdr = du$ and remembering that

$$\int_0^{+\infty} \exp\left(-\frac{k}{2}u\right) du = \frac{2}{k}, \quad (9.21)$$

the equation becomes

$$\frac{1}{P_0^2} = \frac{1}{2} \int \int \exp\left(-\frac{k}{2}u\right) du d\theta = \frac{1}{k} \int d\theta = \frac{2\pi}{k} \quad (9.22)$$

and

$$P_0 = \frac{1}{\sqrt{2\pi k}} = \frac{1}{\sqrt{2\pi}\sigma}. \quad (9.23)$$

Finally we get

$$P(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right). \quad (9.24)$$

For $\sigma = 1$ such a distribution is called the **normal** distribution. Its maximum can be easily translated from x to μ writing

$$P(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \quad (9.25)$$

that represents the distribution of a random variable with mean μ and width σ . In general, $P(x, \mu, \sigma)$ is more often called a **Gaussian**, after Carl Friedrich Gauss (1777–1855), though the names “normal” and “Gaussian” are sometimes used as synonyms. It is straightforward to show that if a variable x is distributed as a Gaussian of mean μ and width σ , the variable

$$z = \frac{x - \mu}{\sigma} \quad (9.26)$$

is normally distributed. It can be proven that, for $\sigma = 1$,

$$\int_{-1}^{+1} P(x)dx \simeq 0.68. \quad (9.27)$$

Consequently, this is the probability to find a random variable distributed as a Gaussian of width σ in the interval $[-\sigma, +\sigma]$ around μ .

2. The central limit theorem

Experimental data distribute as Gaussians because of the central limit theorem. Simply stated, this theorem says that the sum of independent random variables x_1, x_2, \dots, x_n tends to be distributed as a Gaussian whatever the distribution of x_i .

The proof of this theorem is beyond the scope of this book and is left to more advanced courses on statistics. Here we prove the theorem *experimentally*. Before we go any further we observe that random fluctuations in experimentally obtained data are due to many uncontrolled causes and that, whatever their distribution, their effect sum up such that the final distribution is the one expected for the sum of many random variables.

The normalisation factor of the normal distribution is $\frac{1}{\sqrt{2\pi}}$.

A no
whos
trans
and v
chara
para
a Ga
norm

In order to experimentally prove the theorem, we use Python to generate uniformly distributed integer random numbers from 1 to 6, simulating the launch of a dice. The average score is $\frac{6+1}{2} = 3.5$ and the probability to obtain the values 3 or 4, closest to it, is constant and equal to $p = \frac{1}{6}$. Launching two dices (i.e. summing two uniformly distributed random numbers from 1 to 6) we expect the average score to be $\frac{2+12}{2} = 7$. However, the probability is no more uniform. In fact, while there is only one way of obtaining, e.g., 2 ($1+1$), there are three ways to obtain 7, namely: $1+6$, $2+5$, $3+4$. The probability to obtain 7, then, is three times higher than that of obtaining 2. Summing the scores of three dices, the average score is $\frac{3+18}{2} = 10.5$ and the probability to obtain 10, for example, is much higher than that of observing $3 = 1+1+1$. In fact, 10 can be obtained in a variety of ways.

The higher the number of dices to sum, the higher the probability to obtain a number close to the mean score. At the same time, the shape of the distribution increasingly resembles a Gaussian.

Note that the same behaviour has been observed in analysing the binomial and the Poisson distribution: as the number of trials increases, these distributions tend to Gaussians.

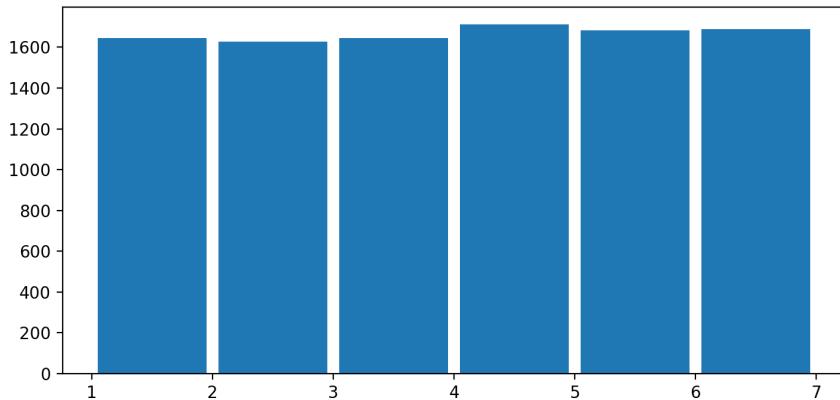
3. Experimental proof of the central limit theorem

In order to prove the central limit theorem experimentally we generate uniformly distributed numbers using Python, starting with `randint` variable as in

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.randint(1, 7, 10000)
plt.hist(x, bins = range(1,8), rwidth = 0.9)
plt.show()
```

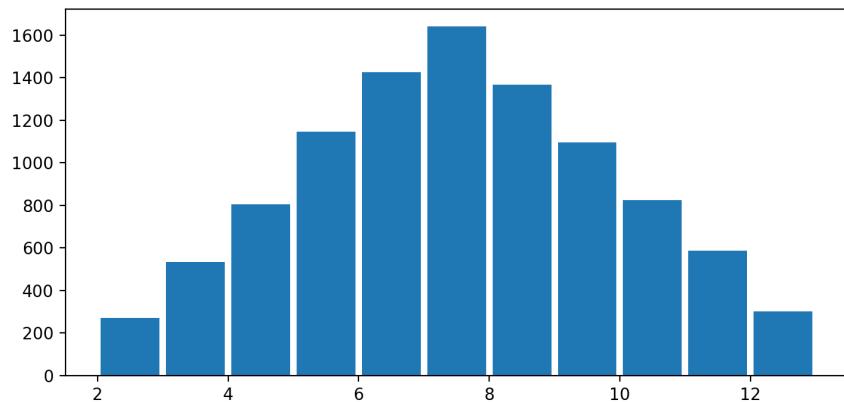
where we generate 10 000 uniformly distributed random numbers between 1 and 6 and we show the distribution as an histogram with bars whose relative width is 90 % that of each bin. The result is shown below.



In the above code the `randint(a, b, n)` method returns n uniformly distributed random numbers in the interval $[a, b)$. We also forced the histogram to have one bin per score. The elements of the `bin` list, in fact, are the edges of each tick in the horizontal axis: this list must contain the bin edges, including the left edge of the first bin (1) and the right edge of the last one (7, remembering that `range(n, m)` returns a sequence of integers from n to $m - 1$). All the bins are open on the right side, but the last. Stated differently, the first bin includes values in the $[1, 2)$ interval, the second in $[2, 3)$, etc., up to $[5, 6)$, while the latter is $[6, 7]$. Adding

```
y = np.random.randint(1, 7, 10000)
z = [x + y for x, y in zip(x, y)]
plt.hist(z, bins = range(2, 14), rwidth = 0.9)
plt.show()
```

to the above script, we generate a second list of 10 000 random variables and we put the sum of them and the previously generated ones into a list called `z`, whose histogram is shown below.



A tuple is an ordered collection of data. Aggregating two lists side by side using `zip` is a way to obtain a tuple. The `zip()` function aggregates the two lists such that the result is a list of tuples. A tuple is an ordered collection of data. In this case the tuple consists of two values: each element of the resulting list consists of a pair obtained aggregating the i -th element of list x with the i -th element of list y . The `for` statement applied to the list returned by `zip(x, y)` navigates through all the tuples and sum their values, filling a new list called z . Let's now sum the scores of ten dices.

```
x = []
for i in range(10):
    x.append(np.random.randint(1, 7, 10000))
z = [sum(i) for i in zip(*x)]
plt.hist(z, rwidth = 0.9)
plt.show()
```

Here x is now a list of 10 lists. In each component of x we put a list of 10 000 random numbers, such that $x[i][j]$ represents the score of dice i at launch j . To sum up the scores of each dice we exploit the `sum()` function, whose meaning is straightforward, still using `zip()` to build a list of tuples. Following what illustrated above we would ideally write

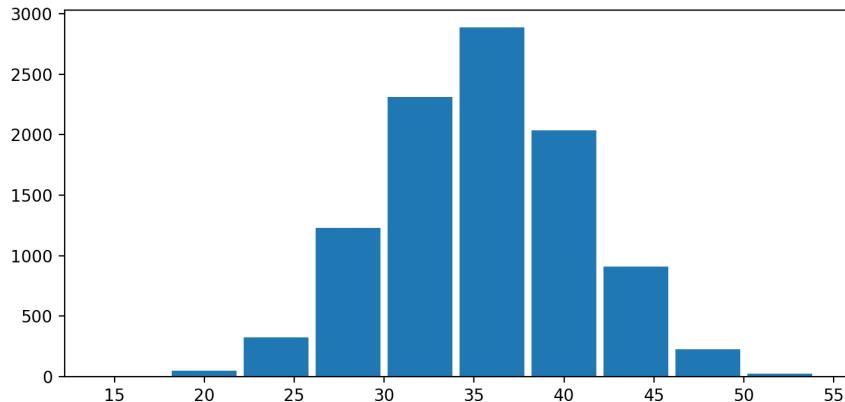
```
z = [sum(i) for i in zip(x[0], x[1], x[2], x[3],
                          x[4], x[5], x[6], x[7],
                          x[8], x[9])]
```

The asterisk operator `*`, applied to a list, unpack it, i.e. split it into a list of its elements.

In order to be generic enough, we exploit the **asterisk operator** that, applied to a list, *unpack* it in its elements. The above line of code is then equivalent to

```
z = [sum(i) for i in zip(*x)]
```

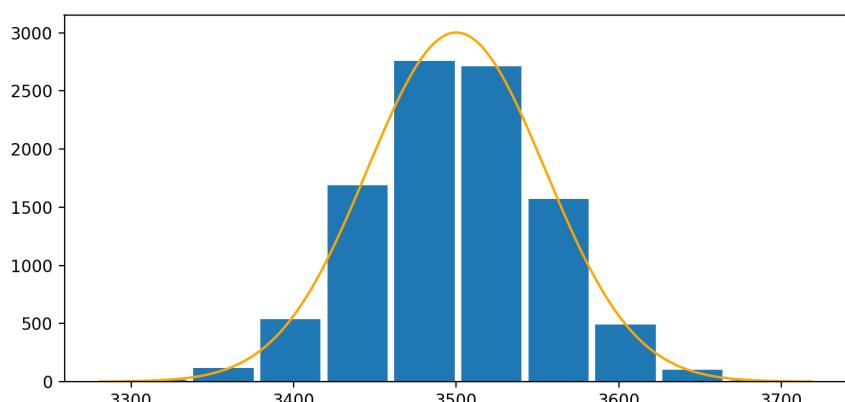
that is much more compact and generic. The result is shown in the histogram below, that clearly show how the distribution tends to a Gaussian one.



The peak became more pronounced and data are relatively more close to the average value. While, in the case of two dices, the minimum possible value 2 still has a non negligible probability, in this case the minimum possible value of 10 is absent from the plot (the probability of obtaining it is lower than $\frac{1}{10\,000}$).

Repeating the experiment with 1 000 dices we obtain a distribution whose values can, in principle, span from 1 000 to 6 000, however, the histogram below shows that most of the data are distributed around the average value 3 500 within a relatively small interval.

As the number of random variables increase, the distribution of their sum becomes more pronounced and peaked around their mean value. While the width of the distribution increases in absolute value, its relative value with respect to the mean comes smaller and smaller. Correspondingly, the probability of the tails comes negligible.



A Gaussian with the proper parameters have been superimposed to the experimental data. Needless to say that the Gaussian describe almost perfectly the experimental distribution.

The average value of the Gaussian μ has been set to $\mu = \frac{7000}{2} = 3500$. Given that the variance σ_1^2 of the distribution of the score of one dice is equal to the square of the width of the interval divided by 12, the variance of the sum of 1000 dices is

$$\sigma^2 = \sum_{i=1}^{1000} \frac{6^2}{12} = \frac{36000}{12} \simeq 3000. \quad (9.28)$$

The width of the Gaussian is then $\sigma = \sqrt{\sigma^2} \simeq 55$ as it can be seen from the plot.

To compute the normalisation factor C in front of the Gaussian we observe that the integral of the histogram is

$$C \int P(x, \mu, \sigma) dx \simeq \sum_{i=1}^n n_i \Delta w \quad (9.29)$$

where n_i is the number of events in bin i , Δw is the bin width (assumed constant), and the sum is extended on all the bins, such that

$$C \simeq \Delta w \sum_{i=1}^n n_i = N \Delta w \quad (9.30)$$

where N is the number of trials, i.e. the number of events under the histogram.

If the reading uncertainty Δw dominates, the uncertainty on the measurement is $\frac{\Delta w}{\sqrt{12}}$. If statistical fluctuations are important the uncertainty depends on the width of the distribution.

The origin of the shape of the distribution of experimental errors then, depends on the type of uncertainty. If the reading uncertainty dominates, any value between two consecutive values in the instrument scale has the same probability of being the *true* value and the corresponding uncertainty is the instrument resolution divided by the square root of 12. If random fluctuations are larger than the resolution of the instrument, many random effects sum up giving rise to a distribution tending to the Gaussian one and the uncertainty is given by the width of the latter.

4. The Markov's and Chebyschev's Inequalities

Even if in most cases experimental data distribute as Gaussians, they sometimes exhibit different behaviours. Moreover, there can be cases in which we are not interested in attaching a value to a physics quantity like mass, time, pressure, or something similar. In certain cases we measure distributions, in fact. In β -decays, for example, electrons are emitted from a radioactive material (e.g. ^{60}Co). These electrons carry a momentum p distributed over a continuous range from $p = 0$ to $p = p_{max}$ whose distribution is such that, indicating

with $N(p)$ the number of electrons with momentum p ,

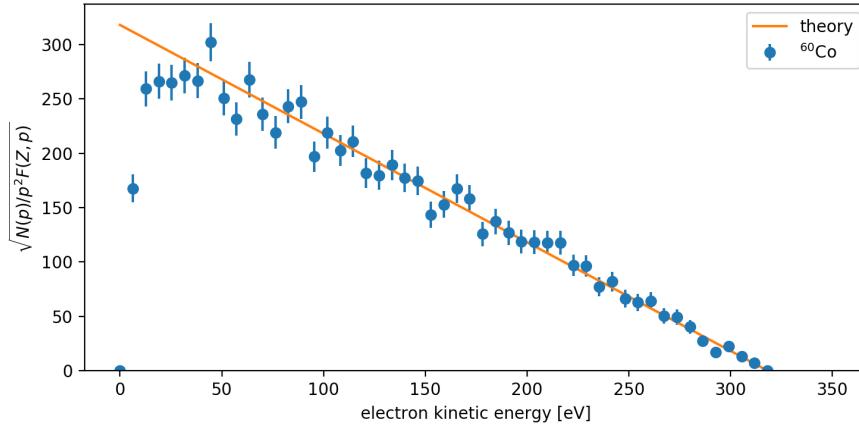
$$\sqrt{\frac{N(p)}{p^2 F(Z, p)}} = a(Q - E) \quad (9.31)$$

where $F(Z, p)$ is called the Fermi function, accounting for the effects of the electrostatic attraction between the electron and the final state nucleus whose atomic number is Z , E is the electron kinetic energy and Q , called the Q -value, is characteristic of the atomic species, has the dimensions of an energy and a value of the order of 1 MeV. a is a normalisation constant, such that,

$$\int_0^Q a(Q - E) dE = aQ^2 - a\frac{Q^2}{2} = a\frac{Q^2}{2} = N, \quad (9.32)$$

N being the number of events detected.

The distribution of the quantity on the LHS of equation (9.31) as a function of E is shown below and can be described by a straight line that intercepts the E -axis where $E = Q$.



A graph like this tells us that the probability to observe an electron with a kinetic energy of, e.g., 100 eV, in cobalt β -decays is about one half of the probability to observe an electron of about 200 eV. The probability to observe electrons with energy greater than 300 eV is very low and is zero for $E \gtrsim 320$ eV.

If we knew nothing about the distribution of data, we could in any case perform some useful statistical test. For example, we can estimate the probability of observing an event with $X = \sqrt{\frac{N(p)}{p^2 F(Z, p)}} \geq k$ just counting the events n_k for

which $X \geq k$ and dividing by the total number of events N :

$$P(X \geq k) \simeq \frac{n_k}{N}. \quad (9.33)$$

The **Markov's Inequality** states that, for a random variable $X \geq 0$,

Markov's Inequality,

$$P(X \geq k) \leq \frac{E[X]}{k}. \quad P(X \geq k) \leq \frac{E[X]}{k}. \quad (9.34)$$

The proof can be easily obtained observing that

$$E[X] = \int_0^\infty X P(X) dX \geq \int_k^\infty X P(X) dX \geq k \int_k^\infty P(X) dX \quad (9.35)$$

The last integral is nothing but the probability that $X \geq k$. Dividing everything by k we obtain the Markov's Inequality.

The **Chebyshev's Inequality** follows from the latter. It states that
Inequality guarantees

that

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \quad (9.36)$$

where μ and σ are, respectively, the mean and standard deviation of X . In fact, defining $Y = (X - \mu)^2$, $Y > 0$ and we can apply the Markov's Inequality such that

$$P((X - \mu)^2 \geq K) \leq \frac{E[Y]}{K}, \quad (9.37)$$

but $E[Y] = \sigma^2$, then

$$P((X - \mu)^2 \geq K) \leq \frac{\sigma^2}{K}. \quad (9.38)$$

The probability that $(X - \mu)^2$ is greater than K is the probability that $|X - \mu|$ is greater than \sqrt{K} , and defining $K = k^2\sigma^2$,

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}. \quad (9.39)$$

5. Testing the Chebyschev's Inequality

One of the ~~T~~^{interesting} features of the Python programming language is that we can ~~refer to functions by~~ using their names as variables. We can, for example, build a ~~list of~~^{containing} statistical functions, importing the `scipy.stats` module and constructing the list as if we do with ordinary numbers or strings, as in

```
continues to represent
from scipy import stats

distributions = [stats.binom(n = 100, p = 0.5),
                  stats.poisson(mu = 5),
                  stats.norm()]
```

The list `distributions` contains three functions representing a binomial distribution with $N = 100$ and $p = 0.5$ (corresponding to the distribution of the only two values of 100 coin tosses), a Poisson distribution with $M = 5$ and a normal distribution with $\mu = 0$ and $\sigma = 1$.

We can loop over the elements of the list, generate, for each distribution, $n = 10\,000$ values and compute their average `m` and standard deviation `s`.

```
for d in distributions:
    x = d.rvs(size = 10000)
    m = np.mean(x)
    s = np.std(x)
```

The `rvs` method of the `scipy.stats` distributions returns a sample of random variables properly distributed (`rvs` stands for *random variables*). In order to estimate the probability that $|x_i - \mu|$ is greater than $k\sigma$ we iterate over the sample and count the number of events for which the above inequality is satisfied for a given value of k .

```
n = 0
k = 1
for i in range(len(x)):
    diff = np.fabs(x[i]-m)
    if diff >= k*s:
        n += 1
p = n/N
```

We can then compare `p`, that is a function of `k`, with $1/k^{**2}$ for various values of `k` (only $k > 1$ makes sense, because probabilities are always less than or equal to one) and superimpose a graph of $1/k^2$ to the plot of $P(k) = P(|x_i - \mu| \geq k\sigma)$, as shown in Fig. 9.1 for the three distributions in the list.

As predicted by the Chebyschev's Inequality, $\frac{1}{k^2}$ is always higher than $P(k)$.

The
a
scip
a list
distri
to it.

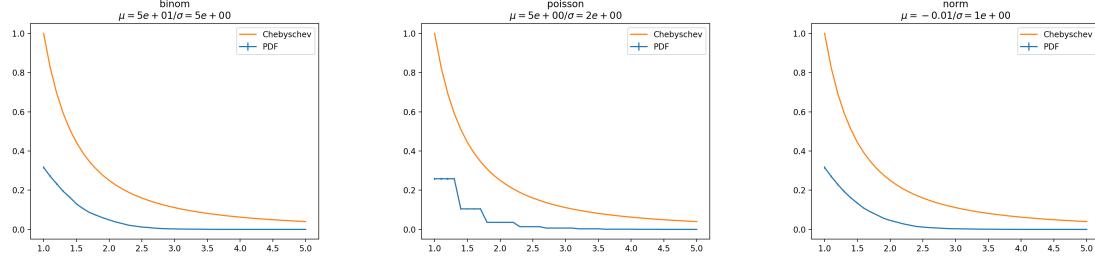


Figure 9.1: Plots of $P(k) = P(|x_i - \mu| \geq k\sigma)$ (blue) and $1/k^2$ (orange) as a function of k for (left to right) the binomial, Poisson and normal distributions.

6. The Law of large numbers

The “experiments” of Section 3 show another important feature of random variables known as the law of large numbers. Again, there are various formulation of this law that can be formally proven. For our purposes, a good formulation is the so-called **weak Law of large numbers**, according to which the average $\langle x \rangle$ of N samples of N random numbers is likely to be equal to the expected value μ of their distribution when $N \rightarrow \infty$.

It can also be expressed as
the value μ of their
distribution

when $n \rightarrow \infty$.

$$\lim_{n \rightarrow \infty} P(|\langle x \rangle - \mu| > \epsilon) = 0, \quad (9.40)$$

where $P(|\langle x \rangle - \mu| > \epsilon)$ represents the probability that the difference $|\langle x \rangle - \mu|$ is greater than ϵ . The limit operation should not be taken as a formal limit as in calculus. It just means that, in order for the law to be true, N must be large and that the larger is N , the more likely is that $\langle x \rangle \simeq \mu$. In practice, $N \geq 30$ is considered to be large.

This law can easily be demonstrated using the **Chebyshev's inequality**. In fact, being a random variable, the average $\langle x \rangle = \frac{1}{N} \sum x_i$ is in turn a random variable and its variance can be computed using uncertainty propagation as

as $\frac{1}{\sqrt{N}}$. If σ is the common uncertainty on x_i , the uncertainty on its average is $\frac{\sigma}{\sqrt{N}}$.

$$\sigma_{\langle x \rangle}^2 = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N}, \quad (9.41)$$

assuming that all x_i has the same variance σ^2 . According to the Chebyshev's Inequality

$$P(|\langle x \rangle - \mu| < \epsilon\sigma) = 1 - P(|\langle x \rangle - \mu| \geq \epsilon\sigma) \geq 1 - \frac{\sigma^2}{N\epsilon^2} \quad (9.42)$$

Taking the limit for $N \rightarrow \infty$ leads to the Law of large numbers.

In deriving the above result we obtained another important result. Equation (9.41) states that the variance of the average value of a set of measurements decreases as $\frac{1}{N}$, then the uncertainty on $\langle x \rangle$ decreases as $\frac{1}{\sqrt{N}}$. Averaging over many measurements, then, is good because it decreases the uncertainty of its value.

When we take N measurements of a physics quantity x_i , $i = 1, \dots, N$, we compute their average $\langle x \rangle$ and their standard deviation σ . The average has then an uncertainty that can be estimated as

$$\sigma_{\langle x \rangle} = \frac{\sigma}{\sqrt{N}}. \quad (9.43)$$

This means that, if we repeat the measurement of x , we have 68 % of chances to observe a value between $\langle x \rangle - \sigma$ and $\langle x \rangle + \sigma$. However, if we take N new measurements, there will be a probability of 68 % that $\langle x \rangle$ will lie between $\langle x \rangle - \frac{\sigma}{\sqrt{N}}$ and $\langle x \rangle + \frac{\sigma}{\sqrt{N}}$.

7. The uncertainty on the average

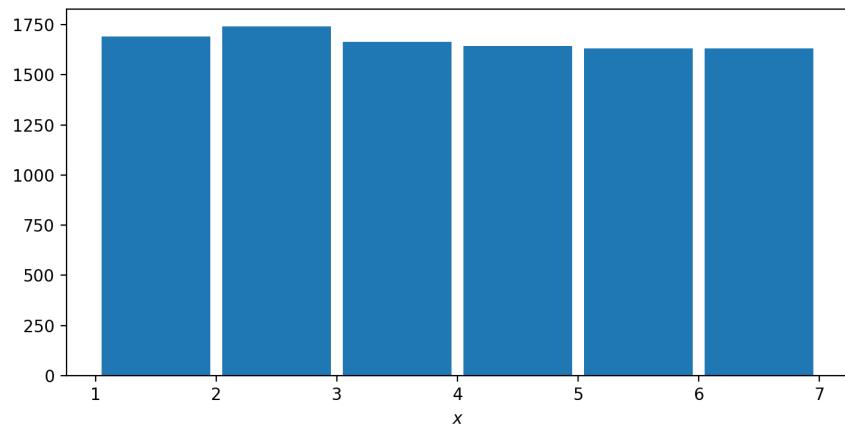
We can test this behaviour generating a sequence of n random variables according to any distribution, compute their average $\langle x \rangle$ and repeating the experiment N times, each time collecting the value of $\langle x \rangle$ in a list.

```
import numpy as np

n = 10000
N = 10000

avg = []
for i in range(N):
    x = np.random.randint(1, 7, size = n)
    m = np.mean(x)
    avg.append(m)
```

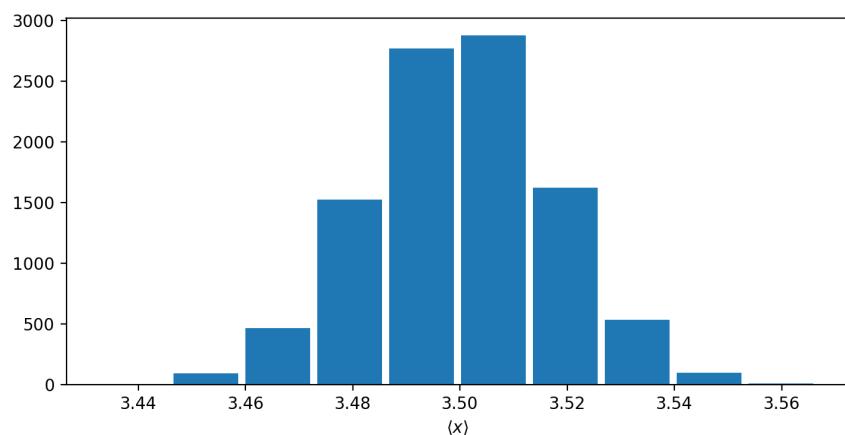
The `randint()` method returns a list of integers uniformly distributed between the given limits. We are then simulating N times n launches of a dice of which we compute the average. Each set of n launches has a distribution similar to the one shown below.



The expected value of the score of each dice is 3.5, while the standard deviation of the uniform distribution is $\frac{6}{\sqrt{12}} \simeq 1.7$. In one run we obtained $\langle x \rangle = 3.462$ with $\sigma = 1.7$, compatible with the expectations.

In another run of n launches we are likely going to find that $\langle x \rangle \simeq 3.5$, while the variance of the distribution is still about 3. With the above script we make N such runs and compute N averages and variances.

We can then plot the distribution of the averages that, as expected, is Gaussian and shown below.



The statistics of the averages are given by

```
m = np.mean(avg)
s = np.std(avg)
print('mu = {} sigma = {}'.format(m,s))
```

In our run of 10 000 runs we obtained $\mu = 3.500$ and $\sigma = 0.017$. Manifestly, the average of the *measurements* is known with a better precision than the single averages. In a single run, we have an uncertainty on the value of x of 1.7, but the average over $N = 10\,000$ measurements has an uncertainty of $\frac{1.7}{\sqrt{10\,000}} = 0.017$. Indeed, $3.500 - 3.462 = 0.038$, much less than 1.7 and corresponding to a bit more than two standard deviations on the average.

In other words, the standard deviation represents the spread of the measurements: in a two- σ wide interval around the average value we will likely find 68 % of the measurements. The uncertainty on the average, however, is $\frac{1}{\sqrt{N}}$ times the latter.

As a consequence, when we make a measurement N times, the value of the measured quantity is given by the average $\frac{1}{N} \sum_i x_i$, while its uncertainty should be quoted as $\frac{\sigma}{\sqrt{N}}$ rather than the standard deviation σ of the measurements.

Summary

Physics data usually distribute such that the probability of a measurement x_i to be different from the *true* value μ decreases with the distance $|x_i - \mu|$. Finding the distribution of random distances whose probability decreases with it is a way to understand the origin of the distribution of physics data. It turns out that such a distribution is a Gaussian.

When the mean value of the Gaussian is $\mu = 1$ and its variance $\sigma^2 = 1$, the distribution is called normal.

The normalisation factor is needed to make the integral of the distribution equal to 1.

$$G(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

In quoting the uncertainty of a measurements, if the reading uncertainty Δw dominates, the uncertainty on the measurement is $\frac{\Delta w}{\sqrt{12}}$. If statistical fluctuations are important the uncertainty depends on the width of the distribution.

The uncertainty on the average $\langle x \rangle$ on N values x_i decreases as $\frac{1}{\sqrt{N}}$. If σ is the common uncertainty on x_i , the uncertainty on its average is $\frac{\sigma}{\sqrt{N}}$.

Statistics

According to the central limit theorem, the sum of many independent random variables follows a Gaussian distribution, irrespective of the distribution of each of them.

The distribution of the sum of random variables,

whatever their distribution is, becomes more pronounced and peaked around their mean value as the number of variables grows. While the width of the distribution increases in absolute value, its relative value with respect to the mean becomes smaller and smaller. Correspondingly, the probability of the tails becomes negligible.

According to the Markov's Inequality, $P(X \geq k) \leq \frac{E[X]}{k}$.

The Chebyshev's Inequality guarantees that $P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$.

The weak Law of large numbers states that the average $\langle x \rangle$ of a sample of n random numbers is likely to be equal to the expected value μ of their distribution when $n \rightarrow \infty$.

Python

`randint(a, b, n)`, in the `numpy.random` package, returns a list of n uniformly distributed integers between a (included) and b (excluded).

A tuple is an ordered collection of data. Its elements can be addressed as if they were the elements of an array. Differently from an array, a tuple can contain objects of different types. Moreover, while lists and arrays are mutable (i.e., you can change their content), tuples are not. Aggregating two lists side by side using `zip` is a way to obtain a tuple.

The asterisk operator `*`, applied to a list, unpack it, i.e. splits it into a list of its elements.

The names of functions in Python can be assigned to a variable and still continue to represent the corresponding functions, instead of merely being represented as strings.

The method `rvs()` applied to any distribution in `scipy.stats` returns a list of random values distributed according to it.

Chapter 10

Kinematics

This chapter is devoted to the experimental study of the motion of a projectile along an inclined plane. Despite its simplicity, characterising such a motion is very instructive and shed some light on the very meaning of the equations of motion that can be found on physics textbooks. Those equations are always the result of (often) crude approximations and the behaviour of a real system may differ from the expected one. Besides learning how to make interesting measurements in the domain of kinematics, we will then learn how to model systems in a more realistic situation.

1. Designing the experiment

In order to experimentally study the motion of something we need to obtain its Experiments in position \mathbf{x} as a function of time t , $\mathbf{x}(t)$. Here $\mathbf{x}(t)$ is a three-component vector whose coordinates depend on time: $\mathbf{x}(t) = (x(t), y(t), z(t))$.

Experiments in kinematics require the ability to measure coordinates as a function of time.

The description of the motion can be greatly simplified with a clever choice of the reference frame in which it is expressed. If, for example, the trajectory of the moving body lies on a plane, choosing a reference frame such that one of the axis is perpendicular to that plane, makes one of the coordinates constant. Thanks to the freedom to choose the position of the origin of the reference frame, we can always set this constant to zero and one of the coordinates becomes superfluous. Similarly, if the motion lies along a line, one can choose a 2-dimensional reference frame oriented such that one of the axis is parallel to that line. In that case, a second coordinate becomes pleonastics and the motion is fully characterised by just one number x as a function of time t , $x = x(t)$, representing the distance between the origin of the reference frame and the position of the body.

Choosing a reference frame with an axis parallel to the velocity in rectilinear motions, greatly simplify the equations.

It must be noted that, while ideally the body can be a pointlike particle, in practice this cannot be true. Any body has a non null size along all the three dimensions. $x(t)$, then, must represent the position of an arbitrary chosen point of it, provided that the body can be considered rigid enough such that all its points stay at fixed positions with respect to any other points of it. It may seem obvious, but there can be cases in which this observation is not so obvious.

As usual, the description of a system in terms of a mathematical model, neglects many effects existing in real systems.

Our goal is then to measure the position $x(t)$ of a body as a function of time t as it travels along a rectilinear trajectory. A free falling object clearly follows such a path, as well as any object falling along an incline. It is easy to realise that the speed of anything sliding along an incline moves slower than the same object freely falling vertically. An interesting experiment could then be to characterise the motion (i.e. to measure $x(t)$) in different conditions, e.g., as a function of the inclination angle θ of the plane. We can expect that the motion when $\theta = \frac{\pi}{2}$ has something to do with free fall.

To perform such an experiment we need something sliding along a plane, whose distance from one end of the plane can be measured as long as time goes by.

2. Measuring time and distances with Arduino

Arduino can measure time with good precision. Arduino provides simple, but interesting tools to perform both the measurement time with good time and distance.

Like any microprocessor, the Arduino ATmega 328, requires a *clock* to operate. Each elementary operation is performed at regular intervals by a device (the **clock**) that emits a regular sequence of electrical pulses.

How CPU works

The Arduino UNO microprocessor contains a CPU (Central Processing Unit) just as any computer. It provides a very clean example on how they work, not needing an operating system. When a CPU is powered up (or reset), it always loads from a fixed location in its memory a fixed number of bytes (just one in the case of the Atmega 328), called a *word*. Loading happens at the first clock pulse. Such a word is interpreted as an *instruction*: each CPU has in fact a one-to-one correspondence between a string of bits and an instruction. The instructions set is composed of operations consisting in doing basic arithmetics on internal memory locations called *registers*, moving data from one location to another in the attached memory (RAM: random access memory) and to or from memory and internal registers, as well as logical operations on data in registers.

Each instruction may or may not require certain parameters to be executed. For example, an instruction to add the contents of two registers clearly needs the two registers on which to execute the required operation. In this case, the parameters are loaded into the CPU unit at the next clock pulses, from the adjacent memory locations. The operation is actually performed at the next clock pulse (more than one if needed).

Once done, at the next clock pulse the CPU loads the next word from the RAM and interpret it again as an instruction, such that the cycle described above restarts.

Computers appear to be able to make extremely complex operations, however their CPU's just

performs only very basic operations on data that are always in the form on binary strings. The apparent complexity of their work is translated into long sequences of elementary operations by the *compilers*: software that translate programmers' code into machine code.

The Arduino UNO clock runs at 16 MHz, i.e. it provides 16 millions pulses per second. Clearly, counting the number of clocks elapsed since the beginning of the run provides a way to measure time with a precision that, in principle, can be as good as

$$\sigma_t = \frac{1}{16 \times 10^6} \text{ s} = 62.5 \text{ ns}. \quad (10.1)$$

Distances can be measured with the help of an ultrasonic device, like in a **sonar**. Sound waves travel in air at constant speed c and can be reflected by surfaces. If a sonic pulse is reflected back by an obstacle at distance d from its source, it returns to it in a time

$$t = \frac{2d}{c}. \quad (10.2)$$

A measurement of such a time, then, can be turned into a distance measurement as

$$d = \frac{ct}{2}. \quad (10.3)$$

The ability of Arduino to measure times can then be exploited to measure both time and distances.

A time measurement can be used to obtain a distance using the definition of velocity.

3. Ultrasonic sensors

A rather popular ultrasonic sensor for Arduino is the HCSR-04. It comprises two piezoelectric crystals working as a speaker and a microphone. Piezoelectric crystals change their shape and size depending on the application of a voltage drop at their ends. Applying an oscillating electric signal to a piezoelectric crystal, makes it to oscillate at the same frequency. Such an oscillation is transmitted to the surrounding medium in which it propagates as a sound wave, like in a speaker.

Similarly, a piezoelectric crystal exhibits a voltage when a mechanical stress is applied, as it happens when sound waves hit them. Voltage variations follow the sound pressure and provides a way to record it as in a microphone.

In an ultrasonic sensor (see Fig. 10.1), a burst of eight acoustic pulses at 40 kHz is generated by the device in response of a trigger signal with a minimum duration of $10 \mu\text{s}$ of at least 2 V. At the same time, an electrical line is raised

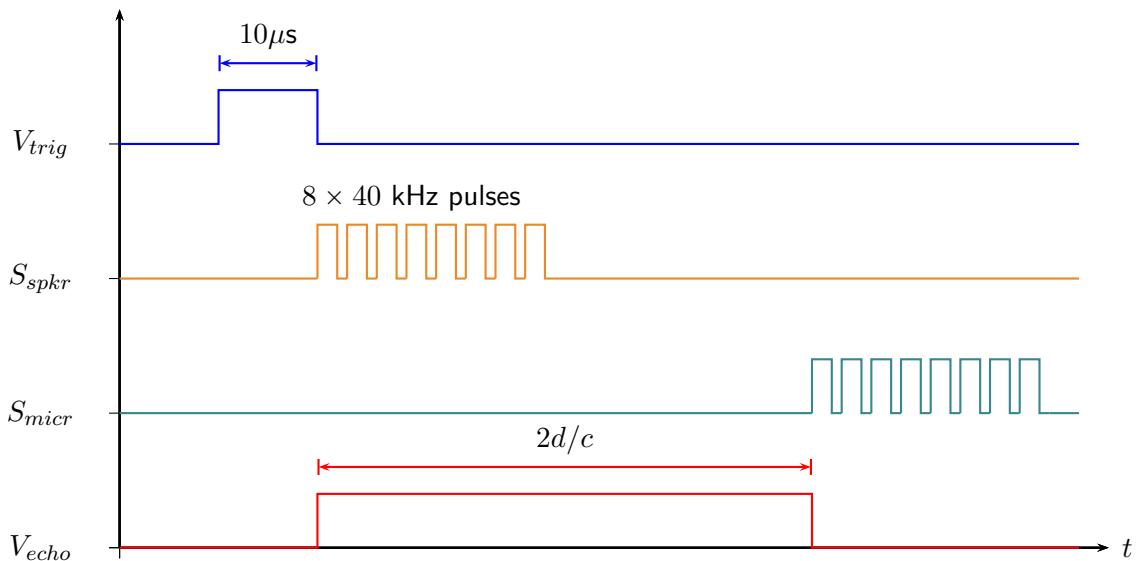


Figure 10.1: Applying a $10\mu s$ voltage pulse (in blue) to the trigger pin of the HC-SR04, causes the emission of eight ultrasonic pulses from its speaker (in orange) and sets its echo pin (in red) to 5 V. The pin goes back to 0 V once the microphone (in green) detects the signal echo.

from 0 to 5 V and stays in this state until the microphone detects an echo of the signal.

The sensor looks like the one on the margin and has four legs: two (marked VCC and GND) are for power and must be connected, respectively, to the 5V and GND Arduino pins; the trigger can be provided by a dedicated leg (marked TRIG), while the fourth leg (marked ECHO) can be used to get the echo digital signal in red in Fig 10.1.



The HC-SR04 is a variant of this sensor exists with three legs (HC-SR03) in which the same leg is used as both the trigger and the echo pin. The HC-SR05 has five pins: four are the same as the HC-SR04, while the fifth, marked OUT is reserved for future uses. According to the data sheet, it can provide meaningful data every 50 ms, such that we can get coordinate measurements with a rate

in the two cylindric structures on it.

$$f = \frac{1}{50 \times 10^{-3}} = 20 \text{ Hz}. \quad (10.4)$$

The useful range of the HC-SR04 is between 3 mm and 3 m.

The range within which these devices can measure a distance is usually within 3 mm and 3 m, with a relatively large variability among different samples.

Most colleges and universities own devices such as air tracks, low-friction cart tracks and magnetic tracks. In air tracks, air is pumped from the bottom and emerges from a series of fine holes such that a cart can glide on an air

cushion with ~~T~~₁ ~~is no friction~~. Low-friction carts are designed to exhibit very low friction ~~while sliding along~~ aluminium tracks, while magnetic tracks exploit magnetic repulsion ~~not made by~~ magnetic carts levitate, avoiding contact between the cart and the track ~~kinematics~~.

As usual in this textbook, we propose an alternative that can be realised ~~there is indeed no home using readily available very common materials, with which we took data to use expensive~~ used below to show the performance that can be achieved with that. For ~~the~~ materials to perform experiment, the rails of an electric train for kids can be used as the track, ~~while~~ an experiment. one of the wagons of the train can be used as a low-friction cart. Indeed ~~Y~~ou can use the rails ~~friction will not be as low as in professionally designed carts, but it will be down~~ electric train as a ~~low-friction track and a wagon as a cart.~~ enough to allow us to obtain nice results.

The track, whose total length must be at least 1 m, must be kept flat when inclined. Rails, then, must be fixed on a rigid support like a wooden plank. They have to be fixed on it such that they are as straight as possible, with almost no bends. Mark their location with a pencil before you stare at them. You can use small wide-headed nails to fix them on the outer sides of the rails, taking care that they do not interfere with the movement of the wagon. The track can be inclined raising one of its ends using boxes or other supports.

As an alternative you can use a duct used for electrical system as a ~~guide and~~, take your any object able to slip (but not roll) inside it. In order to reduce friction ~~you build a stable, you can even put the cart, upside down, in the freezer with a thin layer of water on top. Make its underside so that it becomes ice. When the ice starts to melt it will make the early stages the cart quite slippery. In this case you have to chose properly the cart such that the setup that its size efficiently reflects sound waves, while the guide does not affect their that the setup propagation.~~ works well.

Another possibility is to suspend an object to an inclined smooth bar (e.g., using two metal rings around a broomstick).

The ultrasonic sensor ~~must~~ be fixed at one end of the rail, oriented such that the ultrasonic beam ~~is directed towards the wagon. Securing a cardboard screen on the side of the wagon facing the sensor helps in reflecting waves back. The sensor must then be connected to Arduino using jumpers, as in Fig. 10.2: VCC and GND legs ~~must be connected~~, respectively, to 5V and GND pins; the TRIG and ECHO legs ~~can be connected~~ to any digital pin. As usual, remember that a stable, neat setup guarantees much better results: keep the sensor on a breadboard glued on the track's support and the cables running neatly behind the sensor. Fix firmly the Arduino board on the support such that you can easily plug the USB cables when needed.~~

The ultrasonic beam is ~~Ret~~₁ ~~very belimated~~ (~~its opening angle is about 30°~~) and part of it will not intercept the ~~diagon~~. As a result it can be reflected back by other obstacles placed along its direction. Putting a V-shaped screen made, e.g., by two cardboard or polystyrene sheets at the opposite end of the track

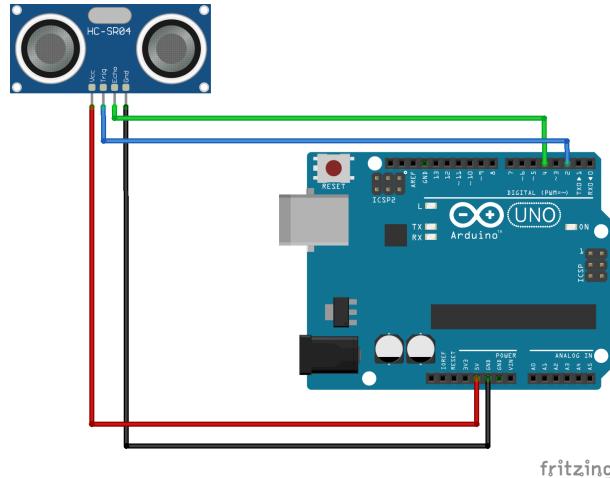


Figure 10.2: The TRIG and ECHO legs of the HC-SR04 sensor must be connected to digital pins, while the other two leads serve for power.

with respect to the sensor, reduces this effect improving the reliability of the measurements.

4. Arduino data acquisition

An Arduino sketch for using an ultrasonic sensor need to trigger the device to emit the ultrasonic beam and wait for its echo.

The *sketch* for data acquisition is shown below. TRIG and ECHO are constants representing the Arduino pins to which the trigger and echo leads of the sensor, respectively, are connected.

```
#define TRIG 2
#define ECHO 4

#define V 343.
#define CONV (1.e-6*100.)

void setup() {
    Serial.begin(9600);
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);
    digitalWrite(TRIG, LOW);
}

void doMeasurement() {
    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    unsigned long t0 = micros();
```

```

    unsigned long ut = pulseIn(ECHO, HIGH);
    unsigned long t1 = micros();
    float d = V*ut*CONV/2.;
    Serial.print((t0+t1)/2.);
    Serial.print(" ");
    Serial.println(d);
}

void loop() {
    doMeasurement();
    delay(50);
}

```

The `V` constant represents the speed of sound in air and is set as $c = 343$ m/s.
The distance is obtained as

Constants are useful
for several reasons.

$$d = \frac{ct}{2} \quad (10.5)$$

and, since t will be measured in microseconds, we need to multiply it by 10^{-6} to express it in seconds. To show the distance in cm, we need to multiply the result by 100. This is done through the constant `CONV`. Note that its value is enclosed in parenthesis, even if there is no need for them in this case. Using the parenthesis is a good habit that you have to take every time a constant is defined by mathematical operations. In fact, constants are not interpreted by the compiler, i.e., the translator from the Arduino programming language to machine code for the Atmega 328. Constants are interpreted by **preprocessors**: software that helps the programmer manipulating their source code before passing it to the compiler. The preprocessor substitute each occurrence of the constants found in the `#define` directives with the corresponding values in the source code.

Suppose, then, you defined a constant as

```
#define CONV 1000+100
```

such that `CONV` is supposed to represent the number 1100. Writing

```
x = CONV*3;
```

will not make $x = 3300$, but $x = 1300$, since such a line is interpreted by the compiler as

```
x = 1000+100*3;
```

due to the fact that the string `CONV` is just substituted to its value by the preprocessor. Enclosing the value of any constant in parenthesis prevents such

Always
values
written
expressions
parentheses
unnecessary

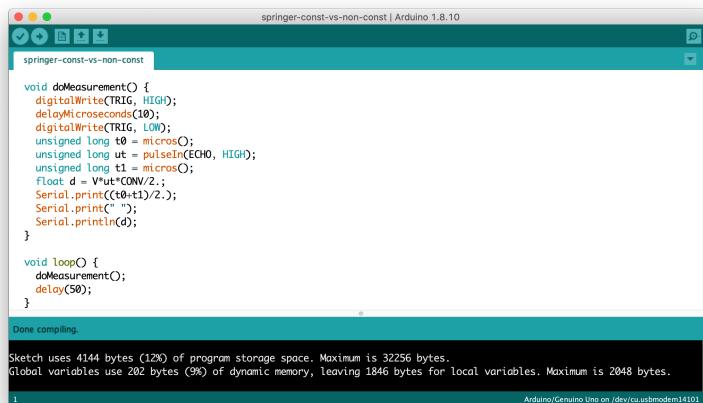


Figure 10.3: The size of the compiled *sketch* is shown in the black part at the bottom of the window of the IDE.

a harmful effect. On the other hand, an excess of parenthesis does not harm your *sketch*.

Many tutorials on Arduino programming use global variable or *constant variables*, instead of constants. The term *constant variable* is only apparently an oxymoron. In computer language the term *variable* indicates just a container for data. If such data are constant or does not depend on how the variable is used, the variables that represent them are said to be constant. Ordinary variables, on the contrary, can contain data subject to change. A variable can be forced to be constant using the `const` qualifier, as in

```
const float v = 343.;
```

that makes the content of a variable constant. Any attempt to modify the content of a constant variable results in an error at compile time.

Constants help ~~that~~ said, it is clear that any variable in the *sketch* may require more memory reducing the ~~memory~~ to be allocated to hold the data it contains. As a result, a compiled *sketch* in occupation of ~~which~~ variables are used instead of constants requires a bit more memory, as compiled *sketch*. You can check compiling it and looking at the report at the bottom of the IDE window (Fig. 10.3). As the memory onboard is very limited, it is not a good idea to waste it.

Programming ~~does not just mean~~ writing code technically correct. It is also a literary exercise, ~~in which you must~~ exploit your communication skills, writing code easy to maintain and whose interpretation is crystal clear, unless you are participating ~~to~~ *The International Obfuscated C Code Contest – ioccc.org* – where you ~~are supposed to~~ write the most incomprehensible, though working, computer program.

Constant names, unlike those of the variable, are by convention completely constants' names all capitals.

written in capitals. Just looking at how a string is capitalised, tells the reader if he/she is looking at a constant or not.

Moreover, constants help in maintaining code in a simple, yet effective way. Suppose, for example, that you are going to use pin 9 for the trigger, instead of pin 2. Without constants you have to look for every occurrence of the character 2 in the code and decide if you need to change it into 9 or not. For example, you may want to change the line containing `pinMode(2, OUTPUT)`, but you don't want to change the 2 in the formula `float d = V*ut*CONV/2`. By using the `TRIG` constant allows you to change just one line to have a perfectly working *sketch*.

Constants help maintaining code.

In the `setup()` function, besides initialising the serial connection, we define the mode of operation of the digital pins. Remembering that digital pins can act both as input and output pin, the trigger pin is set as an `OUTPUT` pin using

```
pinMode(TRIG, OUTPUT);
```

while the echo pin is set as an input one. We also set the state of the trigger pin to `LOW` with

```
digitalWrite(TRIG, LOW);
```

such that there are 0 V on it.

The `loop()` consists in repeating two instructions: `doMeasurement()` sends over the serial line the result of the execution of a distance measurement, then Arduino waits 50 ms before attempting to run another `loop()`. Here `doMeasurement()` is a function as shown by the parenthesis at the end of its name. Such a function is not part of the Arduino programming language, however we can define our own functions if needed. In order to define a function we need to assign a name (`doMeasurement()`) and a type to it. `doMeasurement()` is not supposed to return something and is defined as `void`. This is the reason for defining the function as

```
void doMeasurement() {  
    ...  
}
```

The programmer can define his/her own functions.

Note that its definition must precede its usage in the program. Calling `doMeasurement()` results in the execution of all the code enclosed in braces. The first three lines of code are needed to ~~generate the trigger pulse~~ when a voltage pulse of at least 10 μ s duration is applied to ~~the angular leg of~~ the sensor. Writing HIGH on that pin makes it to exhibit a voltage of 5 V. The `delayMicrosecond()` function pauses the execution of the *sketch* for the number of microseconds passed as its argument (10), then putting back to `LOW` the pin produces the desired result.

At this time the ultrasonic trail of pulses is emitted and the ECHO pin is set to HIGH. We then get the current time with `micros()` and record it in `t0` and call the `pulseIn()` function. The latter continuously monitor the pin passed as the first argument and, when the pin switches from the state passed as the second argument (HIGH) to the complementary one, it returns the time elapsed since its beginning in microseconds. Such a number is put in the `ut` variable defined as an unsigned `long`, i.e. a 32 bits long non-negative integer.

After getting the current time and recording it in `t1`, the value of `ut` is used to compute the distance in cm as

$$d = \frac{v \text{ [m/s]} t \text{ [\mu s]}}{2} \times 10^{-6} \frac{\text{s}}{\mu\text{s}} \times 100 \frac{\text{cm}}{\text{m}}, \quad (10.6)$$

Once time and distance are measured, we send them over the serial line to be stored on a computer.

With the last three lines we send over the serial line: the average time between `t0` and `t1`, corresponding to the time at which the ultrasonic beam hit the obstacle before being reflected, a blank and the distance d .

This way Arduino provides a measurement of the position of an obstacle, with respect to a reference frame whose origin is on the ultrasonic sensor, as a function of time every 50 ms.

5. Taking data

We can now get the data letting the wagon slide along the inclined track. First of all, we need to measure the angle of the track, for which we can choose among a protractor, a ruler or a smartphone. The first gives us directly the angle; with a ruler one can measure the height h of the end of the track and, knowing its length ℓ obtain the angle as

$$\theta = \sin^{-1} \frac{h}{\ell}. \quad (10.7)$$

If the plane supporting the track is not horizontal, both methods can be affected by a systematic error. This error can be mitigated using a smartphone, whose accelerometer can be exploited to tell the inclination of a plane by measuring the components of the gravity acceleration along the device axis. Apps like PHYPHOX provides an easy way to measure θ .

For ~~and~~ to avoid any push or pull on the wagon when launched, we can tie it to the upper part of the board using sewing thread. Burning the thread with a lighter will make the cart free to move with null initial speed.

Using the `arduinoReader.py` script developed at Chapter 4 we get the data from the serial line and record them on a file for each inclination angle θ . In order to keep only interesting data, we can setup the Arduino reader such that we only keep data such that the distance is within a reasonable interval named after the angle for easier bookkeeping.

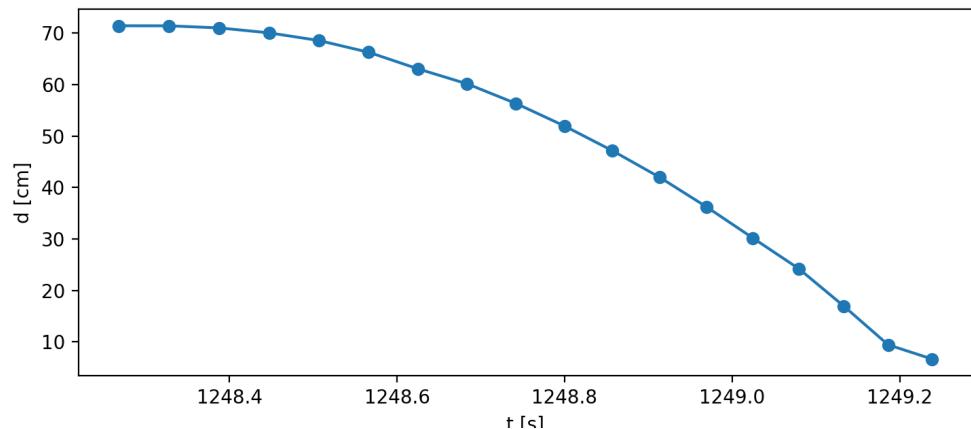


Figure 10.4: The position of the cart along the track, in cm, as a function of the time, in μs .

Data must be taken at angles that are not too large, nor too small. If angles are too large, the cart can go fast enough to be damaged as a result of the collision at the end of the track; if it is too small, static friction prevents the cart to move equipment. Keep conditions in a range

6. Data analysis

your model. Data are represented such that we can easily appreciate their features. FigD10a4 look like the shows the coordinate x of the cart along the track as a function of time t : position versus time is The overall path followed by the cart was a bit more than 60 cm and the duration of the descent was about 1 s. The plot appears to be parabolic, so we can easily argue that

$$x(t) = At^2 + Bt + C, \quad (10.8)$$

where $C \simeq 60$ cm, while $B \simeq 0$ because, with an appropriate translation of the origin of the axes, the vertex of the parabola is at coordinates $(t = 0, x = 60)$. The last point may not be usable: it is probably too close to the sensor to give a reliable result.

The derivative of the position with respect to time $v = \frac{dx}{dt}$ is, by definition, its velocity. It can be obtained by data remembering that

$$v = \lim_{\Delta t \rightarrow 0} \frac{\Delta x}{\Delta t} \simeq \frac{x_{i+1} - x_i}{t_{i+1} - t_i}, \quad (10.9)$$

where x_i and t_i are, respectively, the i -th coordinate of the cart and its corresponding time. Similarly, one can get the acceleration as

he derivative of a
ysics quantity can
tained from data,
ing the difference
quotient.

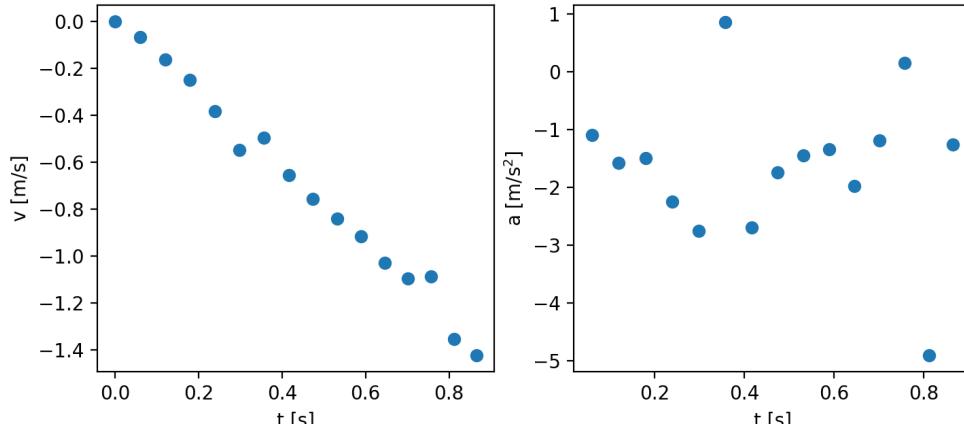


Figure 10.5: Velocity and acceleration of the cart along the track, in SI units, as a function of the time.

$$a = \lim_{\Delta t \rightarrow 0} \frac{\Delta v}{\Delta t} \simeq \frac{v_{i+1} - v_i}{t_{i+1} - t_i}, \quad (10.10)$$

with an obvious meaning of the symbols. Results are shown in Fig. 10.5 (after removal of the last point). From the figure on the left one can see that the absolute value of the velocity of the cart increases almost linearly with time, as expected from the fact that $x(t)$ is a parabola, i.e.

$$v(t) = 2At + B, \quad (10.11)$$

with $B \simeq 0$ consistent with what can be inferred from the graph of $x(t)$. A first guess for A can be easily obtained as

$$2A \simeq \frac{\Delta v}{\Delta t} = \frac{-1.4}{0.8} \simeq -1.8 \frac{\text{m}}{\text{s}^2}. \quad (10.12)$$

The least squares A better estimation can be obtained averaging over various pairs of points, method provides a as seen in Section 4 Chapter 5. Here we learn a new technique to estimate convenient way of the parameters of a function that best fit data. We assume a linear model, estimating function i.e. $v = \alpha t + \beta$ and use the **least squares** method, by means of the Python parameters from data, `curve_fit` from `scipy.optimize`, illustrated in the next section. In short, the given a model for least squares method consists in finding the parameters \vec{p} of a function $y = f(\vec{p}, t)$ that minimises them.

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(\vec{p}, t_i)}{\sigma_i} \right)^2, \quad (10.13)$$

(pronounce ki-square) where y_i are the experimental data taken when $t = t_i$ (in the example above, $y_i = v_i$, $\vec{p} = (\alpha, \beta)$ and $f(\vec{p}, t) = \alpha t + \beta$), N is the number

of points and σ_i the uncertainty on y_i . The uncertainty on t_i is supposed to be negligible. It is easy to recognise in each term of the sum the squared distance between each data point y_i from the fit function, measured in units of their uncertainty. Finding the best fit function means, in fact, to minimise the sum of such distances.

The reason why we minimise the sum of the squares of the distances and not that of the distances themselves is that in the latter case we would be forced to use the modulus operator, defining the total distance as

$$\chi = \sum_{i=1}^N \left| \frac{y_i - f(\vec{p}, t_i)}{\sigma_i} \right|. \quad (10.14)$$

In order to perform operations on this expression we should always consider the sign of each term in the sum and this is very uncomfortable. The squares, instead, are defined positive and if the distance is minimal, so is its square.

Finding the minimum of χ^2 is relatively easy, since in this case χ^2 , seen as a function of α and β , is a parabola. One can then find the vertex of such parabolas to compute the values of \vec{p} . Details about this technique are given below (see Section 10). Here we discuss the results of the minimisation.

We have not evaluated the uncertainties on experimental data, however, we can assume that they are all of the same order of magnitude and put $\sigma_i = 1 \forall i$. On the other hand, if all the uncertainties are equal to each other, the result of the minimisation will be independent on the value of σ_i and we can get rid of them for the moment. The result of the fit is

$$\begin{aligned} \alpha &= -1.61 \frac{\text{m}}{\text{s}^2} \\ \beta &= 0.017 \frac{\text{m}}{\text{s}}, \end{aligned} \quad (10.15)$$

where α is not far from our first raw estimation and β is relatively close to zero (however, think about how reasonable is this statement). As a result of a measurement, both α and β must be affected by some uncertainty that, in turn, depends on σ_i .

Few points, in fact, appear to fluctuate around the best fit, such that the best fit itself can be seen as susceptible to fluctuate around each point. If we trust the fit, data points give us the opportunity to estimate the uncertainty of the measurements. The difference between the velocities at $t \simeq 0.36$ s and at $t \simeq 0.42$ s (those that exhibit the largest deviation from a straight line) is of the order of 0.05 m/s. Given $v \simeq 0.5$ m/s, the relative uncertainty is about 10 %.

From our model, we expect $a \simeq \text{const}$, while from the plot on the right in Fig. 10.5 it seems that a is far from being constant. However, we should not

It w
natu
sum
betw
point
theo
it is

Takin
not a
the m
provid
uncer
to ea
least,
order

The
unce
can
at th

forget that data are affected by a relatively large uncertainty: if the uncertainty of v , σ_v , is of the order of 10 %, σ_a , the uncertainty on a , can be as large as 20 %. We found that

$$\begin{aligned}\langle a \rangle &= -1.65 \text{ ms}^{-2} \\ \sigma_a &= 1.3 \text{ ms}^{-2} \\ \frac{\sigma_a}{\sqrt{N}} &= 0.32 \text{ ms}^{-2}\end{aligned}\tag{10.16}$$

The latter is the uncertainty on the average value, so, assuming $a = \text{const}$,

$$a = -1.65 \pm 0.32 \frac{\text{m}}{\text{s}^2},\tag{10.17}$$

still compatible with the fit result, for a relative uncertainty of 20 %,

Using the least square method it is possible to exploit data distribution to infer parameters.

the least square method it is possible to exploit data distribution to infer parameters.

Using the least square method, one can take a similar approach to evaluate the uncertainties on parameters that takes into account the shape of data distribution. Assuming that the model is correct, one can suppose that the uncertainty is such that data fluctuate around the best fit line according to a gaussian distribution, inferring the uncertainty from the width of the distribution of **residuals**, defined as

$$r_i = y_i - f(\vec{p}, t_i) = y_i - \alpha x_i - \beta.\tag{10.18}$$

The result, obtained with this technique, detailed in Section 7, is

$$\begin{aligned}\alpha &= -1.61 \pm 0.05 \frac{\text{m}}{\text{s}^2} \\ \beta &= 0.017 \pm 0.023 \frac{\text{m}}{\text{s}},\end{aligned}\tag{10.19}$$

consistently with the above results. If we fit the acceleration data with a constant, i.e. with a polynomial of grade 0, using the least squares method we obtain exactly the same value $a = -1.65 \text{ ms}^{-2}$ found averaging the slopes. It can be shown, in fact (see Section 10), that finding the minimum of the χ^2 in this case is equivalent to take the average of data.

Fig. 10.6 shows the same data as Fig. 10.5 with the results of the fits superimposed. The one- and two-sigma intervals are shown in the case of the acceleration, as coloured bands.

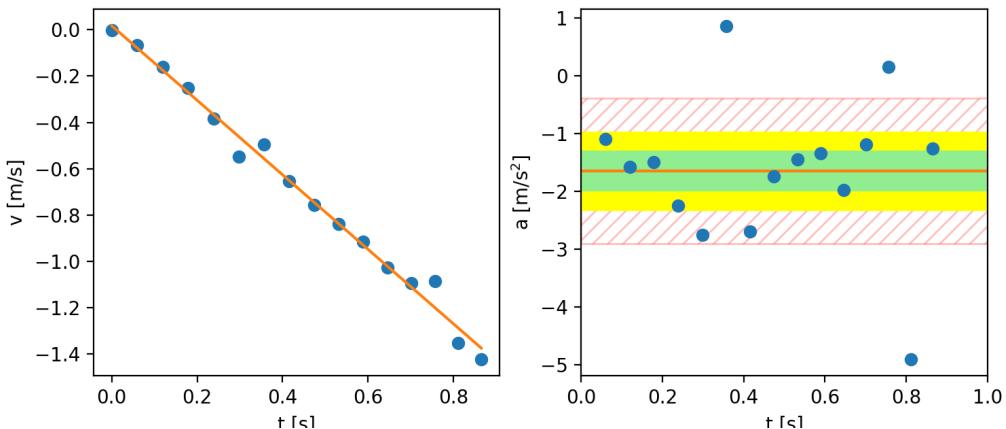


Figure 10.6: Velocity and acceleration of the cart along the track, in SI units, as a function of the time, with the results of the fits superimposed. The green and yellow bands represent, respectively, the one- and two-sigma uncertainty intervals as from the linear fit. The hatched band shows the one-standard deviation interval amplitude.

7. Evaluating the goodness of a fit

Once the parameters \vec{p} have been determined, the χ^2 is a number that can be computed substituting the parameters where appropriate. With the proper normalisation of the terms in the sum, the value of χ^2 gives an indication of how good the fit is. Minimising the χ^2 assuming a quantity of 10%, leads to the following result.

given by the value
of χ^2 , if it is properly
normalised.

$$\begin{aligned} \alpha &= -1.61 \pm 0.06 \frac{\text{m}}{\text{s}^2} \\ \beta &= 0.017 \pm 0.033 \frac{\text{m}}{\text{s}}. \end{aligned} \quad (10.20)$$

Substituting $\alpha = -1.61$ and $\beta = 0.017$ in the expression of χ^2 , its resulting value is $\chi^2 = 6.8$. As expected, the result of the weighted fit (the one in which we normalised the χ^2 using the σ_i) is compatible with that of unweighted one.

Data points are random variables distributed as gaussians and then each term in the χ^2 is a random variable distributed as a gaussian with zero mean and unitary sigma. As a consequence, the χ^2 itself is a random variable distributed as a sum of gaussians. Needless to say, it follows the χ^2 -distribution. As any other distribution, the χ^2 -distribution has an expected value that can be proven to be $E[\chi^2] = \nu$, and a variance $\sigma^2(\chi^2) = 2\nu$ where ν , called the number of degrees of freedom, is given by $\nu = N - m$, m being the number of parameters estimated from data. The distribution, in fact, depend upon ν . In the above example $\nu = 16 - 2 = 14$.

of freedom.

The goodness of the fit, also called *p*-value, is given by the area of

the tail of the χ^2 -distribution,

computed after its cumulative distribution function, i.e. the probability that a randomly picked χ^2 for ν degrees of freedom is greater than a given value χ_0^2 . This probability is also called a *p*-value. The integral on the RHS of the equation is the cumulative distribution function and gives the probability that the variable is between the integral limits (note that by definition $\chi^2 \geq 0$, then $P(\chi^2 < 0, \nu) = 0 \forall \nu$).

The value of the cumulative can be computed numerically and, as usual, there are functions in Python that return it. In our case, the *p*-value of the fit is 0.94: the probability that our experiment lead to a $\chi_0^2 = 6.8$ or higher is then quite high. This usually gives physicists confidence that the model well describes data. As a rule of thumb a *good* fit is one returning a **reduced** χ^2 , i.e. the ratio $\frac{\chi^2}{\nu} \simeq 1$. In fact, $E[\chi^2] = \nu$, because each term in the sum is a normally distributed random variable and, on average, contributes about 1 to the sum. Since there are ν independent terms in the sum (there are N terms, of which m are constrained by the values of \vec{p}), the latter is expected to be ν . A much larger value of χ^2 , as well as an extremely low value, should draw your attention. In summary, a *good* fit should return a reduced χ^2

$$\frac{\chi^2}{\nu} = \frac{\nu \pm \sqrt{2\nu}}{\nu} = 1 \pm \sqrt{\frac{2}{\nu}}. \quad (10.22)$$

It is important to observe that statistics cannot give absolute answers about the goodness of a fit. It can only evaluate the probability that the observed χ^2 can reasonably be ascribed to statistical fluctuations. It is always possible that the χ^2 of a fit is large because of unfortunate circumstances or, conversely, that the χ^2 of a fit is extremely low because the model *overfit* data (any $(n - 1)$ -degree polynomial fit perfectly a set of n data points).

When we affirm that a fit is *good* because of its χ^2 , we implicitly assume that the model is correct. Quoting the *p*-value of a fit like in the above paragraphs, in fact, we implicitly provide the conditional probability that, given a certain hypothesis H_0 (the acceleration is constant) the data distribute according to our expectations: $P(\text{data}|H_0)$. As a matter of fact, the *p*-values are often misinterpreted as the probability of an hypothesis H_0 given the data: $P(H_0|\text{data})$. Remember that conditional probabilities are not symmetric!

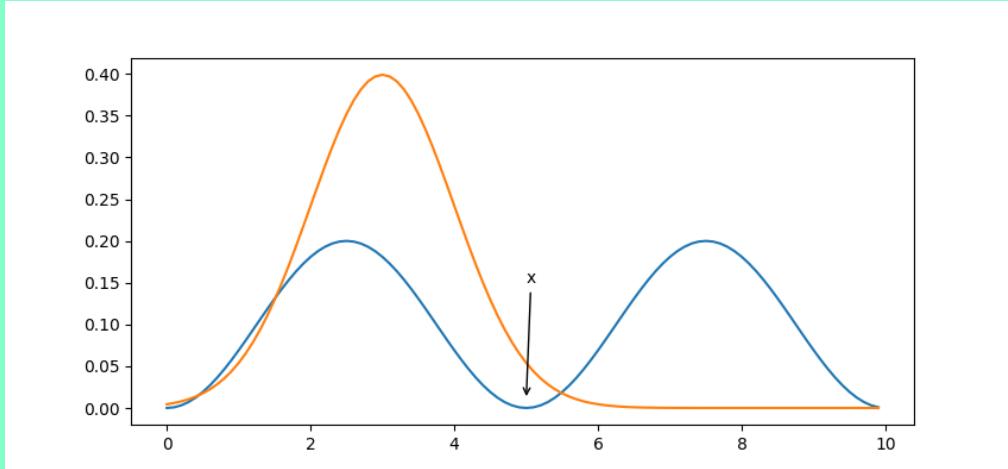
Bearing that in mind, there is nothing really wrong in quoting *p*-values as above, even if care must be taken in using this metric as a measure of the quality of the fit in itself. On the other hand, observing very large or very low χ^2 , especially when the model is manifestly correct, should draw your attention: often it is

a signal that either you made a mistake or that you are over/underestimating uncertainties or neglecting some systematic effect.

Never take rules as laws

Statistics often provides rules to take decisions that, however, cannot be taken as universal laws. Usually they can be applied more or less blindly in most situations, but in subtle cases they can lead to completely wrong conclusions.

Consider, for example, the case in which you found a value $x = 5.0$ during an experiment and you want to compare this value with two alternative models. In one model x is distributed such that it peaks around $x \simeq 3$ (in orange in the plot below); in the alternative model, the x distribution is *bimodal* (in blue).



Manifestly the p -value of the alternative is much larger than that of the other. Adopting naively the p -value rule, we would conclude that $x = 5$ is compatible with the hypothesis that it is distributed according to the blue curve. Unfortunately, the probability of $x = 5$ in this case is predicted to be zero!

It is instructive to work out a goodness of fit test comparing two models in a poorly modelled experiment, considering the acceleration as a function of time. Our hypothesis H_0 , stemming from mechanics, is that $a = \text{const}$, further corroborated by the fact that the observed velocity is linearly dependent on time.

The hypothesis is also compatible with the result of the measurement of $\langle a \rangle$ leading (eq. (10.16)) to a relative uncertainty $\frac{\sigma_a}{a} \simeq 20\%$ consistent with the error on v .

Manifestly, however, we could have obtained the same results even if $a \neq \text{const}$.

An appropriate change of a with time would lead to similar results, in fact. So, fitting the a vs t distribution is worth and permits to test the hypothesis directly, exploiting the way in which data distribute over time. Performing a weighted least square minimisation with $f(\vec{p}, t) = a = \text{const}$ the result is

$$a = -1.65 \pm 0.08 \text{ ms}^{-2}, \quad (10.23)$$

A wide $\chi^2 = 219$ for $\nu = 14$ degrees of freedom. The result is perfectly corresponding compatible with the one obtained by simply averaging data. In fact, minimising p -value, it is equivalent, in this case, to take the average, as it can be seen by just that either imposing that

does not describe data,

or that uncertainties
on data are
underestimated.

Consequently for

excessively high

p -value probably
means that

uncertainties are

i.e. for $a = \langle y_i \rangle$. The uncertainty is a bit larger, due to the fact that few data points deviates significantly from the predicted behaviour. The fact that the model overfit data,

the χ^2 of the fit is so large ($\chi^2 = 219$) tells us that the hypothesis H_0 that the acceleration is a constant is unlikely to be supported by data. The p -value of the fit is practically zero, in fact.

Such a result would naively lead to the conclusion that the hypothesis is wrong, so we could test a different hypothesis. Since any function $f(\vec{p}, v)$ can always be rewritten as a Taylor series, we can try with a truncated series. A fit with a straight line gives a similar result, with $\chi^2 = 216$. Even if the χ^2 is smaller than the previous one, the difference is not significant (remember that the variance on its values is about twice the number of degrees of freedom) and in fact even in this case the p -value is in practice null. Increasing the order to which the Taylor series is truncated is not worth because it is unlikely that a reasonable function describing the data cannot be approximated by a low degree polynomial and, on the other hand, a 14th-degree polynomial will lead to a *perfect* fit with $\chi^2 = 0$.

The correct way to interpret the result is the following: a *bad* χ^2 may have different interpretation.

- The hypothesis H_0 is wrong. In the case we are considering this is very unlikely, since H_0 stems directly from newtonian mechanics that has been extensively tested over centuries and unless we are doing an extremely precise experiment in very unusual conditions none will bet on that. Following the Bayes' definition of probability, the probability that H_0 is wrong is almost null.

- Data are affected by large statistical fluctuations. Such an hypothesis cannot be ruled out completely, because the probability that the χ^2 is large is never 0, though small. After all, the p -value is greater than zero for any $\chi^2 \in (0, \infty)$. However, this is unlikely to happen. If this is in fact a fluctuation, performing the experiment a second time should lead to a much better result.
- Data do not follow the predicted distribution. Such an hypothesis can be true, in fact. However, the experiment we made is simple enough that even considering the presence of friction, we would expect data to be compatible with H_0 , still according to our previous (*a priori*) knowledge of the laws of mechanics. On the other hand, the velocity measured during the same experiment is compatible with H_0 .
- Data are affected by large systematic errors, not properly included in the χ^2 .

The last explanation is indeed the most credible one. Data about acceleration clearly exhibit large fluctuations in selecteds point along the track. Those points coincide with those in which also the values of v show larger fluctuations at times corresponding to points in which two rails join. It is then possible that, when the wagon crosses the joins, its motion is affected in a difficult to predict way. The result is that its position changes abruptly, and that reflects on both v and a .

In summary, while a low χ^2 is often translated into “data follows the predicted model” or “the model is correct”, one should never forget that statistics can only predict probabilities for purely random variables. In particular, it cannot predict how good a model is. Hence, we cannot ask ourselves how large is the probability that a fit is good, but how large is the probability that the fit is not too bad.

8. Data processing

A least square fit can be performed using Python thanks to the `curve_fit()` function defined by `scipy.optimize`, as follows.

```
from scipy.optimize import curve_fit
...
res, cov = curve_fit(f, t, v)
```

where `res`, `cov`, `t` and `v` are arrays, and `f` is the fitting function. When the latter is a straight line, it is defined as

```
def f(x, alpha, beta):
    return alpha*x+beta
```

`curve_fit`
imported from
`scipy.optimize`
used to perform
fit to data
function.

The returned objects `res` and `cov` contains, respectively, the parameters \vec{p} of the fit, in the order in which they are listed in the function definition, and the covariance matrix. The slope of the line is then given by `res[0]`, while the intercept by `res[1]`. The variances of parameters are given by `cov[0][0]` and `cov[1][1]`. Off diagonal elements `cov[0][1]` and `cov[1][0]` are the covariances $\text{Cov}(\alpha, \beta)$ and $\text{Cov}(\beta, \alpha)$ (and they are equal to each other, being the matrix symmetric).

Data collected by Arduino consists of the list of positions `x` collected at times `tx`. From them, the list of velocities `v` can be obtained as

```
for i in range(len(x) - 1):
    v.append((x[i+1]-x[i])/(tx[i+1]-tx[i])/100.)
```

(don't forget that `x` contains positions in cm). When doing the plot of v vs t and the corresponding fit, we have to take into account that the length of `v` is the one of `x` minus one. The corresponding list of times must then be shorter.

In particular, the last time must not be included in the list. A copy of the list

To create `oftimes` can be done as

object uses the
copy() method. The
= operator does not
create a copy of the
operand `t.pop()`

From the latter, the last element can be removed with
but a reference to it.

It is interesting to observe that a statement like `t = tx` does not actually create a list `t` whose elements are equal to those of `tx`, but just a reference to the original list. In this case, a change in the elements of `tx` is reflected into the elements of `t` and viceversa, so `t.pop()` would remove the last element of `tx`, too. The `copy()` method, on the contrary, creates a *shallow copy* of the list, constructing a new list and inserting in it the elements found in the original one (there is also a `deepcopy()` method useful when the list contains complex objects).

`curve_fit(f, t, v)` perform an unweighted fit for which $\sigma_i = 1$ is assumed for all data. The elements of the covariance matrix are evaluated from the width of the distributions such that when the parameters change by one standard deviation, the χ^2 increases by the corresponding number of degrees of freedom.

In order for the fit to take into account the proper normalisation, we need to pass an array `w` containing the values of σ_i to `curve_fit()`, as in

```
res, cov = curve_fit(f, t, v, sigma = w)
```

In this case the minimisation is performed weighting each term in the sum by $\frac{1}{\sigma_i^2}$, such that the points with a smaller uncertainties dominate while looking for the minimum. However, the uncertainties are still evaluated as above, assuming

that the model is correct and looking for the values of the parameters that make the χ^2 to increase by ν . In order to compute the uncertainties from σ_i , we need to state that explicitly as in

```
res, cov = curve_fit(f, t, v, sigma = w,
                      absolute_sigma = True)
```

After a weighted fit, the χ^2 of the observed data can be used to evaluate its goodness. Unfortunately, there is no way to ask Python to return that value and we need to compute it explicitly by ourselves as in

```
def chisquare(x, y, fun, par, sigma = None):
    chi2 = 0.
    for i in range(len(x)):
        s = 1
        if sigma:
            s = sigma[i]
        chi2 += ((y[i]-fun(x[i], par[0], par[1]))/s)**2
    return chi2
```

When defining a user function, besides positional parameters, one can define optional parameters that can be omitted when calling the function; in this case function can be those parameters get their default value specified after the = sign (it can be optional if they are None). In the case of chisquare, the parameter sigma can be omitted. If assigned a default so, its value is None and the weight of each term in the sum is taken to be 1. Otherwise, it is assumed to be a list of as many elements as x and y.

function is defined.

Note that positional parameters, on the contrary, are mandatory and their values are assigned taking into account the order in which arguments are listed in the function call, such that, in order to call the above function to compute the weighted χ^2 of the fit to v vs t with a function linear using res as \vec{p} , we write

Arguments to function
are either passed in the
order in which it
expects them, or as a
comma separated list
of keyword/value pairs.

```
chi2 = chisquare(t, v, linear, res, wv)
```

However, values to parameters can be assigned in any order explicitly indicating the corresponding parameter name (also called keyword) together with the argument, such as in

```
chi2 = chisquare(y = v, x = t, sigma = wv,
                  fun = linear, par = res)
```

A mix of the techniques is allowed when there is no ambiguity. This technique is used when the first arguments are passed in the default order, as in

```
chi2 = chisquare(t, v, par = fitres,
                  fun = linear, sigma = wv)
```

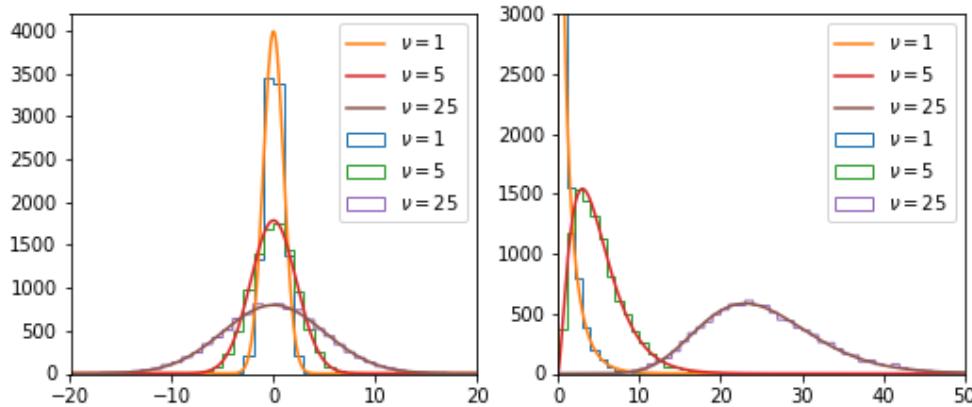


Figure 10.7: The distribution of 10 000 random variables obtained summing up ν normally distributed variables (left) and the one of as many variables obtained summing up their square (right).

For a clean code we recommend to always call the functions using positional arguments when they are mandatory and keyword arguments when they are optional.

The above function returns

$$\chi^2 = \sum_i \left(\frac{x_i - f(x_i, \alpha, \beta)}{\sigma_i} \right)^2 \quad (10.26)$$

where `fun` is supposed to be the name of a function accepting two arguments, besides `x`, contained in the `par` list. The returned χ^2 can be used to evaluate the goodness of the fit.

The χ^2 -distribution is defined in the `scipy.stats` package and is named `chi2`. The corresponding cumulative distribution function is given by `scipy.stats.chi2.cdf(X2, NDF)`, where `X2` represents the value of χ^2 and `NDF` ν .

9. The χ^2 -distribution

The sum of ν normally distributed variables x_i , $i = 1 \dots \nu$. Their sum $\chi = \sum_i x_i$ is distributed as a gaussian with zero mean and standard deviation $\sigma = \sqrt{\nu}$. Fig 10.7 left shows the distribution of 10 000 random variables for $\nu = 1, 5$ and $\nu = 25$, with their PDF superimposed as a continuous curve.

The distribution of $\chi^2 = \sum_i x_i^2$ is shown on the right side of the same figure, for various ν . Manifestly $\chi^2 > 0$ and its distribution is asymmetric. It exhibits a peak around ν and a tail on the right side. The tail becomes less pronounced as ν increases.

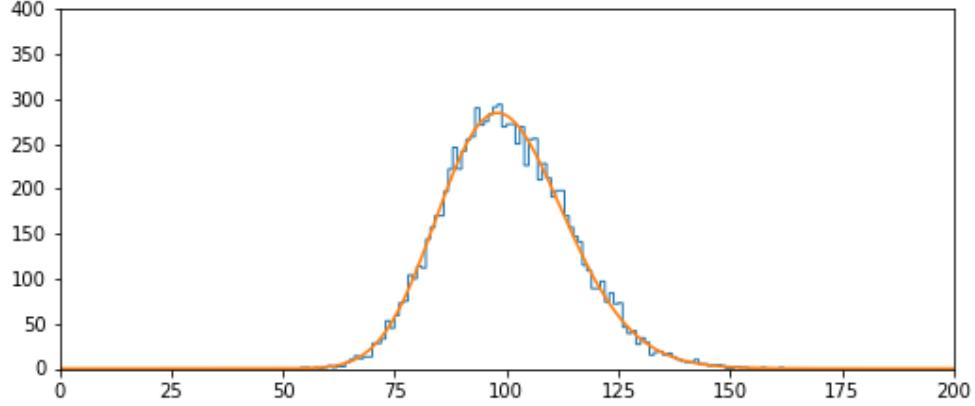


Figure 10.8: The distribution of a χ^2 random variable obtained summing up $\nu = 100$ normally distributed variables. One can hardly distinguish it from a gaussian distribution with mean 100.

as ν increases, as predicted by the central limit theorem. In fact, for $\nu = 100$ (Fig. 10.8) the difference between the χ^2 distribution and a gaussian is hard to spot. The number ν is called the **number of degrees of freedom** (NDF).

From the figures, we can easily spot few features of the χ^2 distribution, that can be rigorously proven. The mean of a χ^2 variable with ν degrees of freedom is ν , while its variance is 2ν . From these properties it is easy to show that the mean of the **reduced χ^2**

$$\chi_{\nu}^2 = \frac{\chi^2}{\nu} \quad (10.27)$$

is 1, while its variance is $\frac{2}{\nu}$. The PDF of χ^2 can be obtained with Python from `scipy.stats.chi2.pdf(x, NDF)`. Its analytical expression is

$$P(\chi^2, \nu) = \frac{(\chi^2)^{\frac{\nu}{2}-1} e^{-\frac{\chi^2}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)}, \quad (10.28)$$

$\Gamma\left(\frac{\nu}{2}\right)$ being the Gamma function, defined as

$$\Gamma(x) = \int_0^\infty y^{x-1} e^{-y} dy \quad (10.29)$$

for which

$$\Gamma(n) = (n - 1)! . \quad (10.30)$$

In most cases there is no need to memorise it. Values of $P(\chi^2, \nu)$ are either tabulated in printed books or can be obtained using library functions as from Python's `scipy.stats.chi2.pdf()`.

10. The least squares method

In this section we review the least square method in a more formal and general way with respect to what is done in the examples.

The parameters of the function that best describes data can be found imposing that its distance from experimental points is the minimum possible one.

Having collected data of $x(t)$, $v(t)$ and $a(t)$ as shown in Sections 5 and 6 we need to find the function $f(\vec{p}, t)$ that best describes our data. A possible approach is to find the function for which the distance between the function itself and the experimental points is minimum. Such a distance can be defined as the sum of the distances of each experimental point to the corresponding value of the function of the variable on which it is supposed to depend upon, e.g., for $x(t)$,

$$d_i = |x_i - f(\vec{p}, t)| \quad (10.31)$$

such that we want to minimise

$$D = \sum_{i=1}^N d_i = \sum_{i=1}^N |x_i - f(\vec{p}, t_i)|. \quad (10.32)$$

The need for the absolute value $|\cdot|$ comes from the fact that each experimental point can lie below or above the function. However, such an operation is complicated to handle, so we should try to avoid it. A possibility is to minimise the sum of the squares of the distances d_i . In fact, if $\sum d_i$ is minimum, so is $\sum d_i^2$. The advantage of this technique, is that we don't care about the sign of each term in the sum being all non-negative by definition. We then find the **best fit** minimising its minimum value,

then so is the distance

$$D^2 = \sum_{i=1}^N d_i^2 = \sum_{i=1}^N (x_i - f(\vec{p}, t_i))^2. \quad (10.33)$$

However, we should take into account that each x_i is affected by an uncertainty σ_i and each term in the sum should then be weighted by their uncertainty. We take that into account minimising the sum of the squared distances in units of their uncertainties

$$\chi^2 = \sum_{i=1}^N \left(\frac{d_i}{\sigma_i} \right)^2 = \sum_{i=1}^N \left(\frac{x_i - f(\vec{p}, t_i)}{\sigma_i} \right)^2, \quad (10.34)$$

such that the minimum location is determined mostly by more precise points (those whose σ_i is lower) and less by less precise ones.

Finding the best fit means to find the values of parameters \vec{p} such that χ^2 is minimum. Since each term in the sum is a normal distributed random variable, the sum is distributed as a χ^2 variable with ν degrees of freedom, where $\nu = N - m$ and m is the number of parameters \vec{p} estimated from the fit. In fact, for each parameter estimated from data we introduce a constraint among the variables such that the number of random variables decreases by one.

Consider, for example, the fit to $a(t)$, where we supposed that $f(\vec{p}, t) = C = \text{const}$. In this case $\vec{p} = (C)$ is a vector with just one component (i.e. a scalar) and the χ^2 can be computed as

$$\chi^2 = \sum_{i=1}^N \left(\frac{a_i - C}{\sigma_i} \right)^2. \quad (10.35)$$

Clearly, χ^2 , as a function of C , attains its minimum when

$$\sum_{i=1}^N a_i - C = 0 \quad (10.36)$$

i.e., for

$$C = \frac{1}{N} \sum_{i=1}^N a_i. \quad (10.37)$$

Indeed, we already observed that finding the best fit in this case is equivalent to take the average of the measurements. We can also realise that, once C is determined by the above formula, the N normally distributed random variables represented by data are no more independent on each other. They are constrained such that C is given by the equation above. That is why the number of degrees of freedom of such a χ^2 is not N , but $N - 1$.

Best fitting $v(t)$ is more interesting. In this case, $f(\vec{p}, t) = \alpha t + \beta$ and $\vec{p} = (\alpha, \beta)$, then

$$\chi^2 = \sum_{i=1}^N \left(\frac{v_i - (\alpha t_i + \beta)}{\sigma_i} \right)^2. \quad (10.38)$$

In this case the minimum of χ^2 as a function of α and β is attained when its derivatives with respect to α and β vanish. In fact, expanding the square in the

Fitting with a constant
is equivalent to take
the average of the
data.

In general, χ^2 is
defined as a function
 \vec{p} and reaches its
minimum when the
derivatives with
respect to the
components of \vec{p}
vanish.

sum we have

$$\begin{aligned}\chi^2 &= \sum_{i=1}^N \frac{v_i^2 + (\alpha t_i + \beta)^2 - 2v_i(\alpha t_i + \beta)}{\sigma_i^2} \\ &= \sum_{i=1}^N \frac{v_i^2 + \alpha^2 t_i^2 + \beta^2 + 2\alpha\beta t_i - 2\alpha v_i t_i - 2\beta v_i}{\sigma_i^2}.\end{aligned}\quad (10.39)$$

Fitting simple case we can avoid taking derivatives observing that the numerator is a parabolic function of α and β . Finding the minimum means finding the **abscissa of the vertex** of each parabola. Writing the expression above ordering in powers of α gives

parabolas.

$$\chi^2 = \alpha^2 \sum_{i=1}^N \frac{t_i^2}{\sigma_i^2} + 2\alpha \sum_{i=1}^N \frac{\beta t_i - v_i t_i}{\sigma_i^2} + \sum_{i=1}^N \frac{v_i^2}{\sigma_i^2} + \beta^2 \sum_{i=1}^N \frac{1}{\sigma_i^2} - 2\beta \sum_{i=1}^N \frac{v_i}{\sigma_i^2}. \quad (10.40)$$

We can improve readability introducing the following definitions:

the data (t and v),
 their product vt and $S_{vt} = \sum_{i=1}^N \frac{t_i v_i}{\sigma_i^2}$,
 their squares ($t^2 S_{tt} = \sum_{i=1}^N \frac{t_i^2}{\sigma_i^2}$) is useful to simplify notation.
 $S_{vv} = \sum_{i=1}^N \frac{v_i^2}{\sigma_i^2}$, $S_{11} = \sum_{i=1}^N \frac{1}{\sigma_i^2}$
 $S_{1t} = \sum_{i=1}^N \frac{t_i}{\sigma_i^2}$, $S_{vt} = \sum_{i=1}^N \frac{v_i t_i}{\sigma_i^2}$, $S_{1v} = \sum_{i=1}^N \frac{v_i}{\sigma_i^2}$

$$\quad (10.41)$$

such that

$$\chi^2 = S_{tt}\alpha^2 + 2\alpha(S_{1t} - S_{vt}) + (S_{vv} + \beta^2 S_{11} - 2\beta S_{1v}). \quad (10.42)$$

The abscissa of the minimum (the vertex) of this parabola is at

$$\alpha = \frac{S_{vt} - \beta S_{1t}}{S_{tt}}. \quad (10.43)$$

Seen as a function of β the χ^2 has a minimum at

$$\beta = \frac{S_{1v} - \alpha S_{1t}}{S_{11}}, \quad (10.44)$$

then, substituting the latter in the expression of α :

$$\alpha = \frac{S_{vt}S_{11} - S_{1v}S_{1t}}{S_{tt}S_{11} - S_{1t}S_{1v}}, \quad (10.45)$$

that **is not so difficult** to remember, taking into account that α should have the **dimensions of a velocity divided by a time**. The numerator is made up of a way to remember formulas.

combination having the dimensions of vt (ignoring the σ_i^2 part that is common to numerator and denominator), and to obtain the right dimensions we need to divide by something having the dimensions of t^2 .

Knowing α , it is straightforward to obtain β from eq. (10.44). Since we constrained the data such that they satisfy both equations for α and β , the number of degrees of freedom of the χ^2 is $N - 2$.

We can also fit $x(t)$ to data. In this case $f(\vec{p}, t) = At^2 + Bt + C$ and $\vec{p} = (A, B, C)$. Finding the values of A , B and C is a bit trickier, but in principle we just need to compute the derivatives of the χ^2 with respect to A , B and C and set them to zero. The same happens for any function $f(\vec{p}, t)$.

In these cases we profit from the power of a computer to numerically solve equations. There is plenty of modules to find the coordinates of the minimum of a function of q variables. Some of them adopt a classic, deterministic method consisting in exploring the q -dimensional space *moving* in the direction along which the gradient of the function diminishes (the **gradient descent** method). Such a method is simple to implement, but it relies on the fact that the function exhibits just one minimum within the search interval. More powerful techniques use stochastic methods to find the minimum, often stolen from physics. Optimisation methods are discussed in the next chapter.

Coming back to the problem of evaluating the parameters of a function, we still need to find the uncertainties with which we know them. Given that data are affected by uncertainties, so are the parameters derived from them. The uncertainty of each term in the numerator in the expression of χ^2 is σ_i . As seen above, since each term has a σ_i in the denominator, each term contribute as 1 to the sum. Since there are N terms in the sum, a one-sigma fluctuation of the data makes χ^2 increase by N . However, two parameters are constrained when we choose the slope and the intercept of the line, then, in fact, only $\nu = N - 2$ of the terms in the sum actually may fluctuate, i.e.

$$\chi^2 \rightarrow \chi^2 + \nu, \quad (10.46)$$

upon a one-sigma fluctuation of the data. We then expect that the error with which we know the parameters of the fit function is such that it increases χ^2 by ν . The way in which this is done is clearly illustrated when the \vec{p} has actually one component only, as in the case of $a = C = \text{const}$ case. In this case, the χ^2 as a function of the C has a minimum for $C = \langle a \rangle$. In a relatively narrow interval around the minimum the χ^2 has the shape of a parabola. Finding the intersection between $\chi^2(C)$ and an horizontal straight line $y = \chi_{\min}^2 + 1$ identifies two points whose abscissa determine the uncertainty on C .

In the case of two free parameters, the χ^2 , as a function of the parameters, is a function of two variables: $\chi^2 = \chi^2(A, B)$ and is graphically represented as a paraboloid. To find the uncertainties on A and B we find the plane at $z =$

Fitting
difficu
function
simple
line. C
help, t
ability
of com
short t

$\chi^2_{min} + 2$ whose intersection with the χ^2 gives an ellipse. The intersection of such an ellipse with the A and B -axis determines the uncertainties on parameters.

When the χ^2 cannot be well approximated by a Taylor series truncated at the second order it may happen that errors are asymmetric. If, for example, we find that the minimum is obtained for $C = 1.65$ in some units, while the interceptions between the χ^2 and the line $\chi^2 + 1$ are found at $C = 1.60$ and at 1.82 , we express our result as

$$y = 1.65^{+0.17}_{-0.05}. \quad (10.47)$$

Then when the ~~uncertainties~~ of data are not known, assuming that the model is a good ~~description~~ of data we can estimate the uncertainties on the parameters \vec{p} finding the values of ~~parameters~~ that make χ^2 increase by ν . This is how the uncertainties of the parameters ~~are found~~ in the residuals method.

parameters.

11. Discarding bad data

When the ~~goodness of fit~~ test returns a low p -value, students are often tempted to just ~~discard~~ data that behaves as outliers. The first reaction is to assume that ~~they find some mistake in collecting them, then it is better to delete them such that their professor cannot ask them why those points behave that way.~~

This habit can turn in a big mistake, in fact. First of all, instructors are often smart enough to spot that just by performing statistical tests on the distribution, often by eye. Secondly, and most importantly, keeping this habit may lead to miss a big opportunity.

In ~~1967~~ Jocelyn Bell, observing the signals of the radio telescope she was operating, had simply ignored the small deviations from the average signal visible in Fig. ~~10.9~~. She would not be remembered today as the discoverer of the pulsars. Unfortunately, at that time concerns about gender equality were not yet popular and indeed the contribution of female scientists were often ascribed to their (mostly male) supervisors. For that discovery the Nobel Prize was awarded to Antony Hewish. Jocelyn Bell Burnell were fair enough to recognise his merits, still she had a major role in the discovery that was recognised only recently by the attribution of the Breakthrough Prize in Fundamental Physics in 2018.

We report this story here in the hope that none of you, in the near future, will miss an opportunity due to carelessness in data analysis or because of his/her gender, race or political views. The message is: never discard data you do not understand. Try hard to explain their origin. Even if you do not succeed, report them to your community. It is very possible that you will be blamed for that, but if those data turns out to be interesting your career will get a boost. On the other hand, do not trust too much the results of your data analysis, based

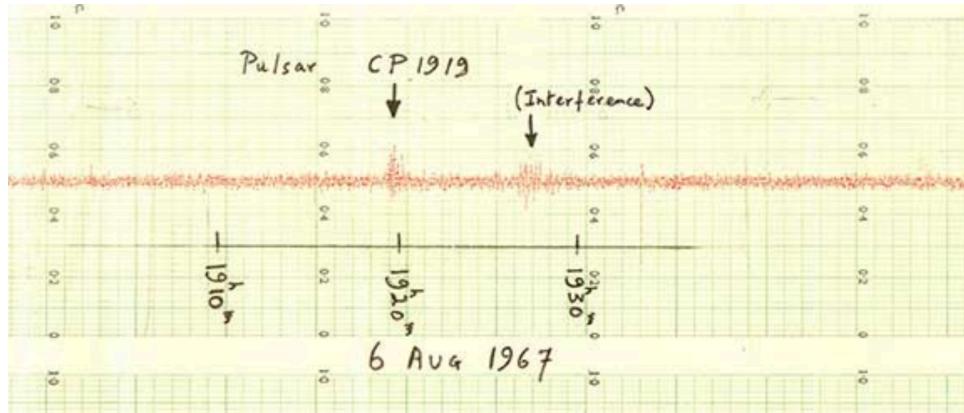


Figure 10.9: The original chart of the recordings of the Jocelyn Bell's radiotelescope in Cambridge, with her annotations.

only on p -values. Statistics ~~can~~ ~~not~~ only provide the probability that some effect can be produced by ~~an~~ ~~random fluctuation~~. It can never make predictions about the reliability ~~of your model~~ ~~of~~ ~~your~~ ~~models~~ ~~at~~ ~~any~~ very low probability that an observation is due to statistical fluctuations ~~is not~~ ~~equivalent~~ to a high probability that the same observation is due to a real effect!

12. Measuring the gravity acceleration

Newtonian mechanics predicts that an object sliding along an incline falls with an acceleration

$$a = g \sin \theta. \quad (10.48)$$

Knowing θ one can easily measure g from data collected in the experiment described above. The result could be disappointing, if g , expected to be close to 9.8 turns out to be quite far from it. The reason is that the above formula neglects completely the effects of friction. Taking into account it, the acceleration is found to be

$$a = g (\sin \theta - \mu \cos \theta), \quad (10.49)$$

μ being the friction coefficient. We need at least two measurement to obtain both g and μ , so we need to repeat the experiment twice at different angles. Repeating the experiment several times at several angles allows us to fit the distribution of a vs θ to obtain both g and μ with a better precision. It is interesting to observe that if θ is small enough, $\sin \theta \simeq \theta$ and $\cos \theta \simeq 1$, such that

$$a \simeq g (\theta - \mu). \quad (10.50)$$

Taking
friction
get ready
from a
made u
availab

This way, a is a linear function of θ , whose slope is g , while μ can be computed after the intercept, dividing it by g .

In doing such an experiment, θ cannot be too small, otherwise static friction prevents the object to slide, nor too high, otherwise the approximation is no more valid. For θ below 35° , however, $\sin \theta \simeq \theta$ is still valid within few percent, being $\theta \simeq 0.61$ and $\sin \theta \simeq 0.57$.

Summary

Experiments on kinematics require to measure the position of an object as a function of time. A clever choice of the reference frame may lead to great simplifications.

Even if a physics quantity can be obtained by a single measurement, exploiting the relationship existing between different quantities helps in reducing the uncertainties and identifying possible sources of systematic errors. For example, in the case of the motion along an incline, the gravity acceleration g can be inferred, in principle, by a single measurement. However, measuring the accelerations of the object varying the angle of the incline leads to a much better result.

The position of an object falling along an incline as a function of time can be well described as a parabola.

Knowing the position at different times, velocities can be computed taking the numerical derivative $\frac{\Delta x}{\Delta t}$. Similarly, the acceleration can be computed as $\frac{\Delta v}{\Delta t}$.

The least squares method provides a convenient way of estimating function parameters from data, given a model. It consists in minimising

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(x, \vec{p})}{\sigma_i} \right)^2$$

Once the minimum is found, the χ^2 is a number that can be computed substituting the values of \vec{p}

in its expression. The value of the χ^2 is a measurement of the goodness of the fit.

The χ^2 is a random variable with $E[\chi^2] = \nu$ and $\sigma^2 = 2\nu$, where $\nu = N - m$, m being the number of parameters estimated from the fit. ν is called the number of degrees of freedom.

The goodness of the fit, also called p -value, is given by the area of the tail of the χ^2 -distribution, computed after its cumulative distribution function.

$$p\text{-value} = 1 - \int_0^{\chi^2_0} P(\chi^2, \nu) d\chi^2$$

As a rule of thumb, a good fit returns a reduced χ^2 , defined as $\frac{\chi^2}{\nu}$, close to 1 with a standard deviation $\sqrt{\frac{2}{\nu}}$.

Statistics can only provide the probability that a given result is consistent with random fluctuations. It cannot tell us how good a model is, nor how data are consistent with the model. It can only tell us how consistent the deviations from the model are with random fluctuations.

A large χ^2 , corresponding to a low p -value, is a signal that either the model does not describe data, or that uncertainties on data are underestimated. Conversely, an excessively high p -value probably means that uncertainties are overestimated or that the model overfit data.

Arduino

With Arduino times can be measured exploiting

the fact that its operation is regulated by a clock running at 16 MHz.

Distances can be measured with Arduino using ultrasonic sensors, like the HCSR-04. It is composed by two piezoelectric crystals: one emits ultrasonic pulses, the other detects their echo working as a microphone. Measuring the time t needed to the ultrasonic beam to travel back and forth we can infer the distance d , knowing the speed of sound c .

$$d = \frac{ct}{2}$$

Ultrasonic sensors can measure distances over an interval of few meters. It produces a train of ultrasonic pulses when triggered by a positive voltage signal on its TRIG lead whose duration is at least $10\ \mu s$. It then sets its ECHO lead to HIGH and put it back to LOW when an echo is detected.

The function `pulseIn(pin, state)` monitor a given pin to change its state and returns the time elapsed since it has been called, in microseconds.

Constants in Arduino sketches are useful for maintenance. They are introduced by the `#define` directive, interpreted by the preprocessor that substitute its value to any occurrence of them in the program before sending it to the compiler. Remember that the substitution happens literally and before compilation.

Variables, in programming, are data containers. Each variable corresponds to one or more memory locations. Variables can be made constant (i.e immutable) adding the `const` qualifier in their declaration. This way, the compiler can detect any place in the code in which the content of a variable can be potentially be modified and exit with an error.

It is important to write programs in a clean, easy to maintain fashion. When writing a program always try to transmit implicit messages to other programmers. Adopt conventions like writing variables in lowercase and constants in uppercase.

Phyphox

PHYPHOX has a function to measure angles with respect to its axis. The function exploits the accelerometer and the angles are measured comparing the components of the gravity acceleration.

Python

`curve_fit()` imported from `scipy.optimize` is used to perform a best fit to data with a user function. It returns the list of parameters estimated by the fit and their covariance matrix.

To create a copy of an object, use the `copy()` method. The `=` operator does not create a copy of the operand on its right, but a reference to it.

By default, `curve_fit()` performs an unweighted fit. To properly weight the terms in the sum of the χ^2 , a list of the uncertainties must be passed as an optional argument.

Parameters in a function can be optional if they are assigned a default value when the function is defined. Arguments to function are either passed in the order in which it expects them, or as a comma separated list of keyword/value pairs.

Statistics The sum of the squares of ν normally distributed random variables is distributed as a χ^2 with ν degrees of freedom.

The mean of χ^2 variable with ν degrees of freedom is ν , while its variance is 2ν .

In general, χ^2 is regarded as a function of \vec{p} and reaches its minimum when the derivatives with respect to the components of \vec{p} vanish. The parameters that best describe data are taken to be those that lead to the minimum possible χ^2 .

Fitting data to a constant is equivalent to average data. Fitting to a straight line consists in finding the vertex of the parabola represented by the χ^2 as a function of the fit parameters.

There exist explicit formula to compute the fit parameters in the case of a linear fit (or regression). In general, a computer finds the minimum of the χ^2 numerically.

Uncertainties can be computed propagating them to the fit parameters or using the method of residuals, i.e. finding the values for which the χ^2 increases by ν .

Chapter 11

Oscillations

This chapter is devoted to the experimental study of the motion of a spring, resulting in the Hooke's Law. In fact, there is nothing particularly interesting about the motion of the end of a spring, except that it has a certain regularity. On the other hand, flipping through a physics textbook, it seems that physicist are almost obsessed by them. The reason why physicists are so interested in springs is that the elastic force that governs their motion is, in the first approximation, similar to the forces that govern other much more interesting phenomena. Understanding how a spring works allows us to investigate a variety of physics phenomena from subnuclear to cosmological scales.

1. An experiment to study elasticity

Elasticity can be studied using very simple tools: a smartphone with PHYPHOX, a flask and one or two rubber bands. You may have noticed that there are no springs in the list. In fact, the role of the spring is taken, in this case, by the rubber bands. On the other hands, they behave almost like springs, at least qualitatively. From now on, then, "spring" and "rubber band" will be synonyms.

A very simple experimental setup is shown in the margin, where we suspended the flask to a clothes liner by means of a rubber band.

In order to start a quantitative study of the spring motion we need to identify the relevant physical quantities. The state of the system under investigation is given by the spring length $y = y(t)$, that is a function of time t . The length of the spring varies according to the external force applied to one of its extrema, keeping the other one fixed. Its shape, the material of which is made, its weight, etc., have some influence on its motion. However, given a spring, they all remain constant under all circumstances, then they can be modelled by simply identifying one or more constants with the proper dimensions.

From Newton's second Law we can write that

$$\frac{d^2y}{dt^2} = \frac{F}{m}. \quad (11.1)$$

where m is the mass attached to one of the spring's ends. The spring itself is



~~The flask is massless in this model. In order to reproduce this condition in the analysis of the system we need to make sure that the mass of the flask is much higher than the mass of the spring.~~ Integrating twice this equation we should obtain $y(t)$ ~~as experimentally observed when the spring is subject to force F , i.e., allows us neglecting~~

~~the effects of parameters that are difficult to be taken into account.~~

Here, F can be a function of time, too, i.e. $F = F(t)$ and, since $y = y(t)$, $F = F(y)$. In principle, we know almost nothing about F .

Even if we do not know much more than that F must be continuous and derivable we can go further in our investigation assuming that, if F is smooth enough (and we have no reasons to believe it is not), then we can approximate

~~Irrespective of using Taylor expansion and write~~

~~complicated the~~

~~expression of the elastic force is, for small elongation ($y - y_0$)~~

~~and $F''(y_0)$ being, respectively, the first and second derivative of $F(y)$, always be written as computed for $y = y_0$. If we stretch the spring only a little, $y - y_0$ is small, then linearly dependent $(y - y_0)^2$ can be considered as negligible and it. The elastic force is~~

~~the Taylor's expansion~~

~~of any force $F(y)$~~

$$F(y) - F(y_0) \simeq F'(y_0)(y - y_0). \quad (11.4)$$

~~This formula can be rewritten defining y_0 the length of the spring for $F = 0$, such that $F(y_0) = 0$. Being $F'(y_0) = k$ a constant and given that the force exhibited by the spring is always opposite to the external force,~~

$$F(y) = -k(y - y_0). \quad (11.5)$$

~~We can further simplify its expression defining the elongation $\Delta y = y - y_0$, such that~~ $F = -k\Delta y$.

$$F(y) = -k\Delta y \quad (11.6)$$

also known as the Hooke's Law, after Robert Hooke (1635–1703), who first formulated such a Law as *ut tensio, sic vis*: the force (*vis*) depends (*sic*) on the elongation (*tensio*, from which the term *extension* comes). the constant k is called the **elastic constant** and represents the spring's stiffness: the higher k , the stronger the force needed to change its length by Δy . The elastic constant's dimensions can be easily computed as

$$[k] = \left[\frac{F}{\Delta y} \right] = \left[\frac{MLT^{-2}}{L} \right] = [MT^{-2}]. \quad (11.7)$$

It can then be measured in units of kg/s^2 , but it is usually given in N/m (the unit of a force divided by the unit of a length).

As ~~it can be seen~~ the Hooke's Law is nothing but a law that states that, in first approximation, every force depending on the position y is proportional to the displacement from the equilibrium position. A toy model of a solid, for example, is made of a regular mesh of points representing atoms attached to each other by springs. It is manifest that physicists do not believe that in fact atoms are attached by microscopic springs to each other. In fact they believe that there must be a force that keeps the atoms close to each other in a solid. ~~There is no way to measure such a force directly using a dynamometer, however we can argue that the force must depend on the distance Δx between atoms (at least because we can break a solid if we take two ends far apart with enough intensity)~~ Repeating the above reasoning, we can conclude that, for small displacements Δx , such that those presumably taking place in normal conditions, the interatomic force F must be proportional to them, and $F = -k\Delta x$.

In order to find the equation of motion we need to integrate the Newton's second Law and solve the following differential equation.

$$\frac{d^2y}{dt^2} = -\frac{k}{m}(y - y_0). \quad (11.8)$$

In other words, we need to find a function $y(t)$, whose second derivative is proportional to the function itself. Trigonometric functions have this property, i.e.,

$$\begin{aligned}\frac{d^2}{dt^2} \sin \omega t &= -\omega^2 \sin \omega t \\ \frac{d^2}{dt^2} \cos \omega t &= -\omega^2 \cos \omega t\end{aligned}$$

(11.9) *A trigonometric function is always a solution of differential equations in which the second derivative of a function is equal to the function itself.*

and since they differ just by a constant angle (the **phase**), we can try with

$$y(t) = y_0 + A \sin(\omega t + \phi). \quad (11.10)$$

Substituting in the differential equation we obtain

$$-\omega^2 \sin(\omega t + \phi) = -\frac{k}{m} \sin(\omega t + \phi) \quad (11.11)$$

that is satisfied for

$$\omega = \sqrt{\frac{k}{m}}. \quad (11.12)$$

The phase ϕ depends on our choice on the origin of times. For $t = 0$,

$$y(t) = y_0 + A \sin \phi. \quad (11.13)$$

If we choose $t = 0$ when the spring is at rest, $y(0) = y_0$ and $\phi = 0$. If, instead, $t = 0$ when the spring reaches its maximum elongation A , $y = y_0 + A$ and $\phi = \frac{\pi}{2}$.

We are now ready to make an experiment to test our theory. We can, for example, measure y as a function of t and compare with predictions. Since the second derivative of y is proportional to y itself, we can also measure the acceleration of the weight as a function of time, to see if it behaves as expected.

2. A study of spring dynamics with smartphones

A smartphone's accelerometer can be exploited to measure the acceleration of the weight attached to the spring. Among the PHYPHOX experiments, the "Spring" one uses it to compute the period of oscillations, in fact.

To collect data from the accelerometer, ~~use a "Tethered run"~~, setting the delay and the experiment duration to few seconds (~~typically 3 and 6 s respectively~~), such that we can avoid measuring the ~~acceleration during the~~ experiment preparation and we can catch few oscillations of ~~the spring after~~ Data can be collected using either the "Acceleration" experiments (~~both with or without g~~) as well as using the dedicated "Spring" experiment. In this case, the experiment take data and try to determine the period of oscillations, shown at the end of the experiment in the main tab. It also shows its inverse (the frequency).

A smartphone accelerometer can be used to collect data about the spring acceleration. Start the run, stretch the spring, let it go just before the data acquisition starts. Wait for the chosen duration, then export data as a CSV file. Try to avoid giving boosts to the weight when the experiment start and make sure that the system oscillates in the vertical direction only as much as possible, avoiding oscillations on other axis. Repeat the experiment several times, such that you can choose the one less affected by systematic effects later, each time deleting previous data from the phone memory after exporting them.

Ideally, we should observe the horizontal components of the acceleration (a_x and a_z) to be zero, while the vertical one (a_y) is expected to oscillate as

$$a_y = \frac{d^2y}{dt^2} = -\frac{k}{m} A \sin(\omega t + \phi) \quad (11.14)$$

Let's have a look to the data taken using the "Spring" experiment with PHYPHOX. The raw data CSV file contains four columns: t , a_x , a_y and a_z . A plot of a_y vs t is shown in Fig. 11.1 for a 5 s run.

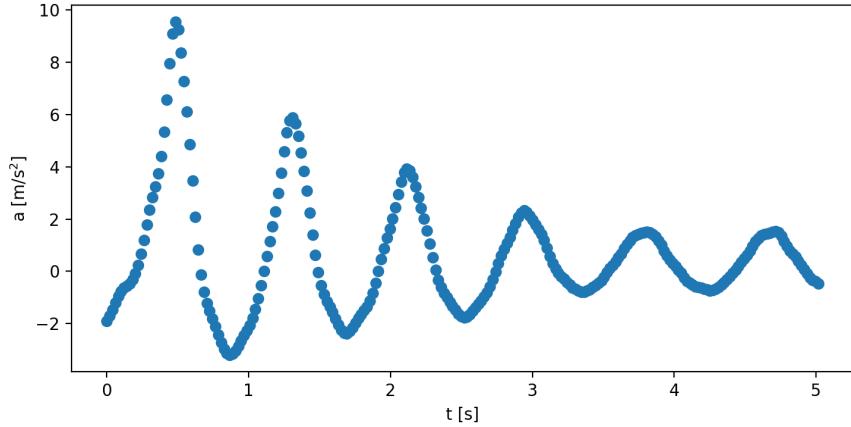


Figure 11.1: Raw data taken during a PHYPHOX run using the “Spring” experiment with a rubber band and a flask with $m = 0.930 \pm 0.001$ kg. The acceleration of the mass is shown as a function of time.

We can clearly spot few interesting features. The acceleration, in fact, oscillates, but the amplitude of the oscillations is not constant: it decreases with time. Oscillations do not seem to be quite as expected: they are not harmonic.

Our simple model of a harmonic oscillator only included the elastic force $F = -k\Delta y$. In a real experiment, non conservative forces such as dissipative ones can be relevant and must be taken into account. Usually, this is a difficult task, however, in this case, it is relatively easy to include them in our model. In fact, ideally the mass attached to the spring moves as

$$y(t) = y_0 + A \sin(\omega t + \phi). \quad (11.15)$$

The speed of the mass is then

$$v_y(t) = \frac{dy}{dt} = \omega A \cos(\omega t + \phi). \quad (11.16)$$

Neglecting its gravitational potential energy, the mechanical energy E of the mass can be written taking its elastic potential energy $U = 0$ when $y = y_0$, such that

$$\begin{aligned} E = U + K &= \frac{1}{2}ky^2(t) + \frac{1}{2}mv_y^2(t) = \\ &= \frac{k}{2}A^2 \sin^2(\omega t + \phi) + \frac{m\omega^2 A^2}{2} \cos^2(\omega t + \phi) \end{aligned} \quad (11.17)$$

Substituting $\omega^2 = \frac{k}{m}$,

$$E = \frac{k}{2}A^2. \quad (11.18)$$

Even if the acceleration is not well described by a harmonic function, it is periodic. The Fourier Theorem guarantees that its shape can be represented by a series of harmonic functions with appropriate frequencies. Despite its anharmonicity, a rubber band behaves as a set of springs with different elastic constants acting together.

If the energy of the system is conserved, $A = \sqrt{\frac{2E}{k}}$ is constant, as expected. The decrease in amplitude is due to friction. The latter subtract mechanical energy to the system.

If not, the mechanical energy is reduced by the work ΔL done by dissipative forces and

$$\Delta E = \Delta L. \quad (11.19)$$

These forces can only subtract a fraction of the system energy at any time, i.e. $\Delta E = \alpha E$. Moreover, the longer the time they act, the larger ΔE , then

$$\Delta E = -\alpha E \Delta t \quad (11.20)$$

where the minus sign tells us that E decreases with time. As a result, E decreases with time as an exponential with a characteristic time $\tau = \frac{1}{\alpha}$, so makes the amplitude, i.e.,

$$A = A(t) = \sqrt{\frac{2E(t)}{k}} = \sqrt{\frac{2E(0)}{k}} \exp\left(-\frac{t}{2\tau}\right). \quad (11.21)$$

Taking $t = 0$ when the amplitude is at its maximum $y(0) = A_0$, $E(0) = \frac{1}{2}kA_0^2$, $\phi = \frac{\pi}{2}$ and, including non-conservative forces, the theoretical prediction about the amplitude of the oscillations as a function of time is modified into

$$y(t) = y_0 + A_0 \exp\left(-\frac{t}{2\tau}\right) \cos(\omega t), \quad (11.22)$$

The net result is that the acceleration oscillates sinusoidally with an exponentially decreasing amplitude. Velocity and position follows this behaviour.

The predicted velocity is then

$$v_y(t) = \frac{dy}{dt} = -\frac{A}{2\tau} e^{-\frac{t}{2\tau}} \cos \omega t - A \omega e^{-\frac{t}{2\tau}} \sin \omega t \quad (11.23)$$

while the acceleration is

$$a_y(t) = \frac{d^2y}{dt^2} = A e^{-\frac{t}{2\tau}} \left(\frac{1}{4\tau^2} \cos \omega t - \omega^2 \cos \omega t + \frac{\omega}{\tau} \sin \omega t \right) \quad (11.24)$$

that can be written as

$$a_y(t) = A e^{-\frac{t}{2\tau}} (C \cos \omega t + S \sin \omega t) \quad (11.25)$$

with

$$C = \frac{1}{4\tau^2} - \omega^2 \quad (11.26)$$

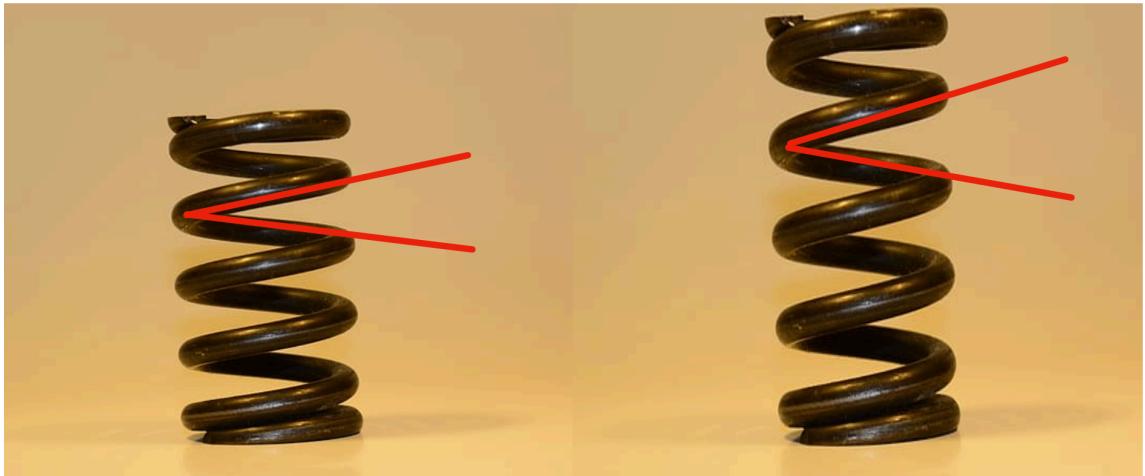


Figure 11.2: Stretching a spring does not change its length. In fact it makes the turns of the spring to slightly rotate.

and

$$S = \frac{\omega}{\tau}. \quad (11.27)$$

The latter is an oscillating function whose period is $T = \frac{2\pi}{\omega}$ and whose amplitude decreases exponentially, as observed in experimental data.

The reason for which the oscillations do not appear to be harmonic is that the conditions under which we derived the solutions are not perfectly matched in the real setup. We assumed the validity of the Hooke's Law that, however, as seen, it is a first order approximation of the expression of the elastic force that can be much more complex. Hooke's Law is valid only for small elongation and at the beginning of the experiment that condition may not be valid. In fact, as the amplitude diminishes, the shape of the curve is more and more approximated by an harmonic motion.

Using a spring rather than a rubber band, makes the experimental data look more similar to the expectations, because in a spring the effective elongation is much less than in a rubber band. Imagine the turn of a spring as composed by flexible curved sections like those shown on the left in Fig. 11.2

Stretching the spring does not cause the spring to change its length, in fact. It makes its turns rotate slightly such that a tiny increase of the angle between turns causes a relatively large change of its effective length. As a result, the conditions for the validity of the Hooke's Law are well satisfied with springs.

Observing closely the data, we can see that minima and maxima show a sort of cusp. This happens because, when the length of a rubber band becomes too long (or too short), it becomes more rigid and does not behave as an harmonic oscillator at all.

Springs behave much like as predicted by Hooke's Law because their length practically changes infinitesimally: they become longer or shorter because the turns rotate around the wire axis.

Despite the conditions in which the experiment has been done do not match well the mathematical model, it is instructive to analyse systems like these. They provide lot of useful insights about the correct interpretation of analytical models, as we already see above. Making the experiment with a spring carefully selected to behave as expected, in fact, is not that interesting.

3. Obtaining parameters from data

In order to test our model against data we can do a *best fit* of the model against the data using the technique introduced in Chapter 10.

Even if linearisation is not possible (or difficult), with the help of a computer it is possible to find the minimum of a χ^2 almost with no effort.

In this case it is not possible to linearise data and we must rely on the ability of computers to perform lot of elementary operations in a small amount of time, to find the parameters that minimises the χ^2 defined as

$$\chi^2 = \sum_{i=1}^N \left(\frac{a_i - a_y(t_i)}{\sigma_i} \right)^2 \quad (11.28)$$

where the sum is extended to the N points collected during the experiment and a_i are the experimental values of the acceleration of the mass; $a_y(t_i)$ are the values computed from eq. (11.25), giving the acceleration as a function of the time, for $t = t_i$, the times at which data were collected. σ_i are the uncertainties on a_i . In order to find the parameters (A , ω and τ) that best describe data, we need to minimise the value of χ^2 .

The fit procedure returns the fit parameters, together with the covariance matrix: a 3×3 symmetric matrix, whose diagonal elements represent the variances on the free parameters of the fit.

Especially in the first experiments on a given topic, systematic uncertainties on the data are often large and the χ^2 is not properly weighted. This is not probably the case. In fact, obtaining a reasonable estimation of σ_i is difficult in this case. Moreover, even if it was possible to evaluate the statistical error on each individual point, the systematic uncertainties on the data are in any case going to dominate the uncertainty. As already observed, initially the motion is far from being harmonic and even if the fit converged, the uncertainty on the reliability of the parameters has to be carefully evaluated.

Physicists refine their experiments over time, trying to eliminate any source of systematic error, either by refining the mathematical model of the system or improving the correspondence between the model and the experimental apparatus. Comparing uncertainties estimated with different methods provides a mean to evaluate systematic uncertainties due to the adopted procedure. Moreover, comparing the results of different fits provides a mean to evaluate the relative

The uncertainties evaluated taking the square root of the diagonal elements of the covariance matrix are then not very meaningful from a statistical point of view. In fact, they are evaluated under the assumption that the uncertainties on data are correctly estimated. Nevertheless, we can extract meaningful information from the procedure.

goodness of them, even if an absolute figure of merit for their quality cannot be established. We can then perform an iterative procedure in which we start by fitting the whole range of data, then we progressively reduce the number of periods included in the fit. At each iteration we can fit only the data collected after each maximum in the plot. At each step the conditions of the experiment match more and more those in the model and the fit is expected to be better and better.

We don't need the granularity of the original data to perform a good fit. In Section 7, Chapter 7, we show that averaging is good to reduce fluctuations in data. If σ is the uncertainty on a single point, the uncertainty on the average of N points is $\frac{\sigma}{\sqrt{N}}$. From Fig. 11.1 we can estimate a period of the order of 1 s. Reducing the density of points such that we keep about ten points per period is then possible, keeping the main features of the model clearly visible. To reduce data of a factor n , each data point can then be defined as (t'_k, a'_k) where t'_k is the average of the time of few adjacent points as

$$t'_k = \frac{1}{n} \sum_{i=(k-1)n+1}^{kn} t_i. \quad (11.29)$$

Similarly

$$a'_k = \frac{1}{n} \sum_{i=(k-1)n+1}^{kn} a_i. \quad (11.30)$$

For example, for $n = 5$, t'_1 is the average of t_1, t_2, t_3, t_4 and t_5 . Similarly, a'_2 is the average of a_6, a_7, a_8, a_9 and a_{10} . This way we can attribute to each point of the fit an uncertainty as the standard deviation of the group of n measurements.

Fitting our data we find, for ω , the following values depending on the starting point of the fit.

the system behaves as an ideal spring only for

run no.	t_{min} [s]	ω [s^{-1}]	<small>relatively large times, χ^2/N_{dof}</small>
0	0.44	7.47 ± 0.04	<small>we perform several fits</small>
1	1.35	7.42 ± 0.07	<small>considering only data</small>
2	2.16	7.39 ± 0.10	<small>collected at $t > t_{min}$.</small>
3	2.96	7.29 ± 0.12	7.5
4	3.77	7.11 ± 0.10	1.9

Data reduction is a way to improve statistics. It does not really reduces the information. Instead, it aggregates data such that their statistical fluctuations are reduced averaging them.

The corresponding graphs are shown in Fig. 11.3 for the first iterations. According to our rule of thumb, only the last row in the table provides a *good fit* and we should take this value as $\omega = 7.11 \pm 0.10$ Hz. It is worth noting that the

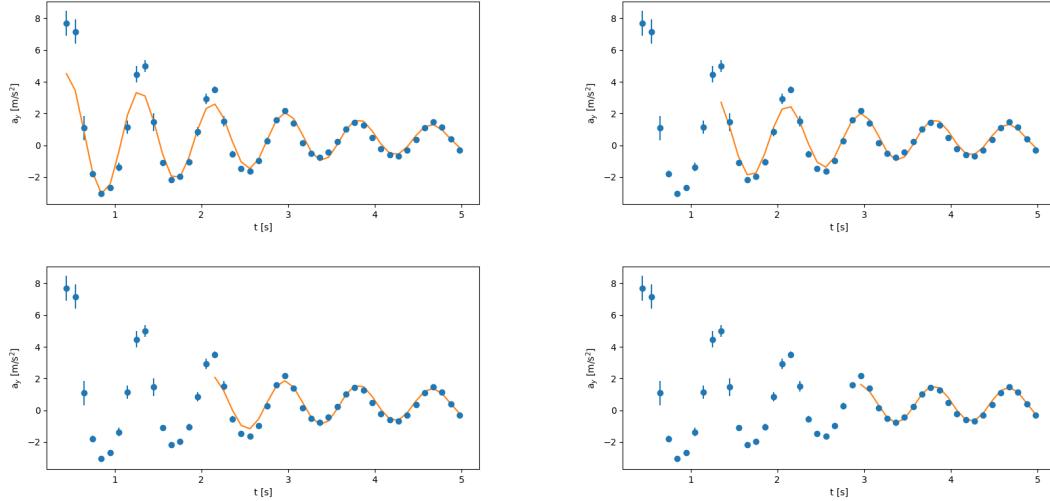
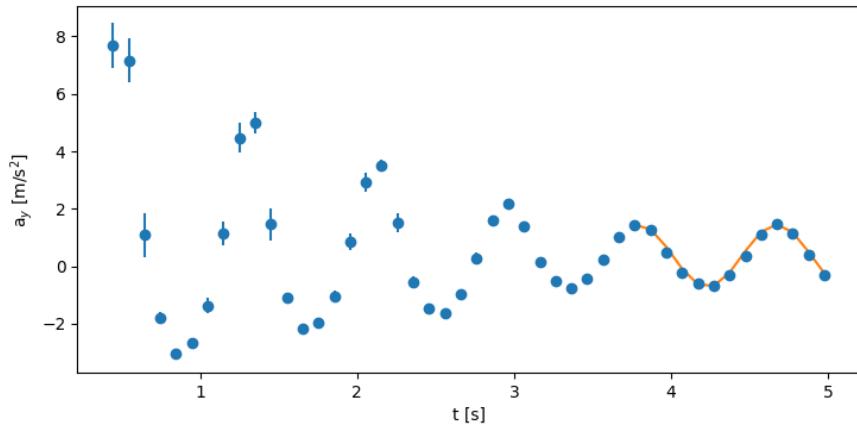


Figure 11.3: Iterative fits made on data. The result of the fit is shown as an orange curve.

various values of τ are not compatible with each other and becomes systematically lower. The reduced χ^2 , too, becomes lower and lower, as a consequence of the fact that the model tends to represent better and better the data, as long as the conditions under which it was assumed are matched in the real system.

With time, the system matches the model better and better. The result of the last iteration is shown below.

because the elongation of the rubber band becomes shorter ; shorter. Discarding first period improves the precision of $d\omega$



We can associate a systematic uncertainty to ω observing that, even when the model is not so good, the value of ω is not too far from the one obtained in ideal conditions. The evaluation of a systematic uncertainty is somewhat

more arbitrary than the statistical one and depends on how confident we are in the apparatus, too. A conservative estimation could be given by the difference between the biggest and smallest values of ω obtained in different conditions, i.e.

$$\sigma_{sys} \simeq \frac{\omega_{max} - \omega_{min}}{2} = \frac{7.47 - 7.11}{2} = 0.18 \text{ Hz} \quad (11.31)$$

and write

$$\omega = 7.11 \pm 0.10 \pm 0.18 \text{ Hz}. \quad (11.32)$$

In this case we keep two significant figures, to mitigate the effects of rounding when determining further data. Indeed, knowing the relationship between ω , the period T and the elastic constant k we can derive

$$T = \frac{2\pi}{\omega} \simeq 0.884 \text{ s} \quad (11.33)$$

and, for $m = 930 \text{ g}$

$$k = m\omega^2 \simeq 47 \text{ Nm}^{-1}. \quad (11.34)$$

The evaluation of the corresponding uncertainties is left as an exercise.

When a systematic error is quoted, the first number represents the statistical error, while the second the systematic one. The latter has no statistical content, even if sometimes is added in quadrature with the statistical uncertainty for convenience.

4. Extracting and manipulating data

As discussed in the previous section, when looking for the parameters of a function that best fit data there is no need for the latter to be much more denser than necessary. Data should be enough to represent the main features of the model and, for sure, more than m , the number of free parameters in the fit.

Data look like the following.

```
"Time (s)", "Acceleration x (m/s^2)", "Acceleration y (m/s^2)", "Acceleration z (m/s^2)"
0.00000000E0, 3.376618028E-1, -1.893230915E0, 9.337148070E-1
2.011108400E-2, 2.025120258E-1, -1.701057911E0, 7.484606504E-1
4.022216800E-2, 8.397972584E-2, -1.461338520E0, 5.465144515E-1
6.033325200E-2, -2.084195614E-2, -1.195132732E0, 3.279784322E-1
8.047485300E-2, -8.745026588E-2, -9.496812820E-1, 1.668487489E-1
1.005859370E-1, -6.927591562E-2, -7.615351677E-1, 9.249490499E-2
```

From the first column we get that data are taken every 20 ms. Averaging every 5 points will make the shape smoother without affecting the capability to catch the features of the model: the acceleration oscillates and its amplitude decreases.

Using PANDAS we extract the data from the CSV file as follows.

```
f = pd.read_csv('rawdata-930g.csv',
                 usecols=['Time (s)',
                           'Acceleration y (m/s^2)'])
print(f.columns)
t = f['Time (s)']
a = f['Acceleration y (m/s^2)']
```

There is often no need to load the whole set of ddata into the memory. We can select interesting columns using the `usecols` parameter in `read_csv()`.

The `usecols` optional parameter in `read_csv()` allows us to extract only the columns to which we are interested from the file. Note that the name we gave to the file contains some relevant information such as the mass of the weight. This way, we do not need to look at our logbook each time we analyse these data. We can easily spot the most important information from the file itself. The names of the columns to be extracted must match those given in the file and included in a comma separated list defined by the pair or square brackets [...]. The following line just print the list of columns names on the screen, as a cross check.

Time and corresponding acceleration can then be defined as **Series** just indicating the corresponding column name in the DataFrame. A Series is almost like a list, in PANDAS. Series can be thought as lists of key-value pairs. By default, the keys of a Series returned by `read_csv()` is an integer from 0 to $N - 1$, N being the number of rows in the DataFrame. As such, in most cases Series and lists can be confused. A plot like the one shown in Fig. 11.1 is done with

```
plt.plot(t, a, 'o')
plt.xlabel('t [s]')
plt.ylabel('a [m/s$^2$]')
plt.show()
```

In order to average data in groups of 5 we can exploit the `groupby()`, `mean()` and `std()` methods as follows.

```
in
N = 5
t = t.groupby(t.index // N).mean()
e = a.groupby(a.index // N).std() / np.sqrt(N)
a = a.groupby(a.index // N).mean()  
an argument.
```

The `//` operator is the **floor division** operator. It returns the integer part of the division between the operands.

`t.index` is a list containing the row numbers of a Series, hence can be thought as a list of integers from 0 to $N - 1$, N being the number of rows in the Series. `division operator //`.

A list divided by a scalar is a list, too, and `t.index // N` is a list containing the floor division of each row number by N . The first five elements of the resulting list will then be equal to zero, the following group of five elements are equal to one, and so on.

The `groupby()` method returns a list of lists. Each element of the resulting list is in turn a list of five numbers extracted from the original list based on the fact that they share the same value of `t.index // N`. If the original list `t` contained 250 rows, the resulting one contains 50 elements, each of which contains five consecutive data points.

Applying the `mean()` method to them results in a list containing 50 numbers, each of which is the average of each group of five. We can put the result back to the original list, such that `t` and `a` are a factor five shorter and contain the data collected during the experiment averaged every 0.1 s. The `e` list contains the standard deviations of each group of five accelerations.

As described in the previous section, the fit is performed iteratively on a subset of the available data. Let's then start dividing the time interval into `phases` by starting from the moment in which the weight reaches its maximum acceleration. `signal`, is a per oscillation. In other words we need to identify the positions of the `maxima` to identify of Fig. 11.1. This is as simple as

```
p, pdict = find_peaks(a)
print('==== peaks found at ====')
print(t[p])
```

`find_peaks()` is a function defined in the `scipy.signal` module, included using

```
from scipy.signal import find_peaks
```

that looks for local maxima in the list passed as an argument (the accelerations) and returns an array containing their index and a **dictionary**: a data structure composed by key-value pairs. Given that `p` contains the indexes of maxima, the times at which they occur can be selected as `t[p]`, that returns the list of corresponding times. The result is something like

```
===== peaks found at =====
4      0.443048
13     1.349774
21     2.155756
29     2.961743
37     3.767725
46     4.674457
```

Compare the list with the position of the maxima in Fig. 11.1. For each interval starting at the time given in the list we fit the data using

```
for k in range(len(p) - 1):
    res, cov = curve_fit(spring, t[p[k]:],
                          a[p[k]:], maxfev=100000)
```

`curve_fit` can be used if imported from the `scipy.optimize` module. The first argument (`spring`) is the name of a user defined function representing the model against which we are fitting the data. Data follows as a pair of arrays of the same size, the first containing the values of the independent variable (t_i) and the second those of the dependent one (a_i). The optional argument `maxfev=100000` sets the maximum number of calls to the function, used either to limit or to extend the number of iterations performed by the minimisation algorithm to find the minimum.

In the above example, the first fit is performed within the interval starting at $t = 0.443048$ s up to the last data point. Then, we repeat the fit considering only data with $t \geq 1.349774$ s and so on. In total we are going to make five fits, indexed from 0 to 4 (the last maximum is not used: we cannot fit our model using only data after it).

The model function is defined as follows.

```
def spring(x, b, A, tau, omega, t0):
    C = 1/(4*tau**2)-omega**2
    S = omega/tau
    t = x-t0
    y = b + A*np.exp(-t/(2*tau))*
        (C*np.cos(omega*t) + S*np.sin(omega*t))
    return y
```

The function takes the independent variable as the first argument and the free parameters as separate remaining arguments. In this function we define `C` and `S` as the corresponding symbols in eq. (11.25). Since, in this equation, $t = 0$ at the beginning of the motion, we need to translate the time axis by an offset `t0`. We consider the possibility of a constant bias `b` to the measured acceleration, due to the limited accuracy of the smartphone's accelerometer. Trigonometric functions `sin` and `cos` are defined in the `numpy` module aliased to `np`.

The χ^2 of the fit is computed *manually* as the result of the function

```
def computeChi2(y, e, f, dof):
    chi2 = 0
    i = 0
    y = y.to_list()
    f = f.to_list()
    e = e.to_list()
    for i in range(len(y)):
        chi2 += ((y[i]-f[i])/e[i])**2
    return chi2
```

to which we pass, at each iteration, `a[p[k]:]` and `e[p[k]:]`. `f` is computed as

```
f = spring(t[p[k]:], res[0], res[1], res[2],
           res[3], res[4])
```

where `res` is the list of the free parameters of the fit returned by `curve_fit()`. It is interesting to note that, in order to loop on data to compute the χ^2 , we need to transform Series into lists by means of the `to_list()` method. In fact, this is a case where a Series cannot be considered as equivalent to a list. When extracting the subset of the data from the original Series, as in `a[p[k]:]` (that selects only the components whose key follows `p[k]`), the resulting one keeps the same list of keys. Moreover, being `t[p[k]:]` a series, `f` is a series too. Consider, for example, the Series resulting from `a[p[0]:]`. The first element of `p` is 4, then `a[p[0]:]` is equivalent to `a[4:]` and only data whose key is greater or equal than 4 is kept in the resulting Series. The first key of the resulting Series is then 4 and when looping on `i` from 0 to the length of `y` the first four elements (whose keys are expected to be from 0 to 3) are not found in the Series, resulting in a runtime error. Converting the Series into a list, the indexing is based on the position of the value rather than its key.

To plot the error bars on data, together with the result of the fit, as in Fig. 11.3, we use the following code.

```
plt.errorbar(t[p[0]:], a[p[0]:], e[p[0]:],
             fmt = 'o')
plt.plot(t[p[k]:], spring(t[p[k]:], res[0],
                           res[1], res[2], res[3], res[4]), '-')
plt.xlabel('t [s]')
plt.ylabel('a$_y$ [m/s$^2$]')
plt.show()
```

The `errorbar()` function plots the values in the list passed as its second argument as a function of those in the first, taking the values of the third argument as the corresponding uncertainties. The format of the plot (circles) is chosen using the optional parameter `fmt = 'o'`.

As
the
are
in
the
de
the
to
ot
to

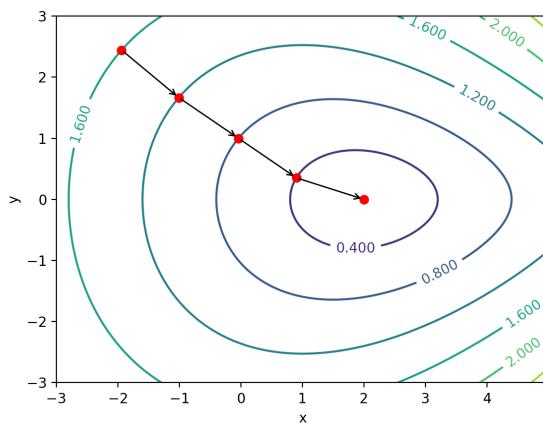
The
metho
matp
shows
togeth
points

5. Optimisation methods

Computer algorithms minimises the χ^2 considering it a surface in a $(m + 1)$ -dimensional space, m being the number of the free parameters in the fit, namely, A , ω and τ in our example. This way, χ^2 is a function of $\mathbf{x} = (x_1, x_2, \dots, x_m)$: $\chi^2(\mathbf{x})$. A common minimisation scheme is the so called **gradient descent method**. It consists in evaluating $\chi^2(\mathbf{x}_0)$ in a random point $\mathbf{x}_0 = (A_0, \omega_0, \tau_0)$ in that space and comparing it with the value of the χ^2 evaluated in $\mathbf{x}_1 = \mathbf{x}_0 - \delta \nabla \chi^2(\mathbf{x}_0)$ and iterating the procedure such that

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \delta \nabla \chi^2(\mathbf{x}_n) \quad (11.35)$$

until $F(\mathbf{x}_{n+1}) < F(\mathbf{x}_n)$. δ is the amplitude of each step that can be adjusted during the minimisation process. We can easily visualise the method in three dimensions, representing the surface $z = f(x, y)$ as a set of contour levels as in the figure below.



Suppose that we start from the top left point at coordinates (x_0, y_0) . The gradient of a function is null at its minima.

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \Big|_{x=x_0, y=y_0} \quad (11.36)$$

and find the new point according to the rule above. Moving along the gradient direction consists in moving along the arrow shown in the figure to the next red point. Here we compute the new values for the gradient, descend it along the next arrow and reach the third red point and so on, until we reach the minimum value for which the gradient is null.

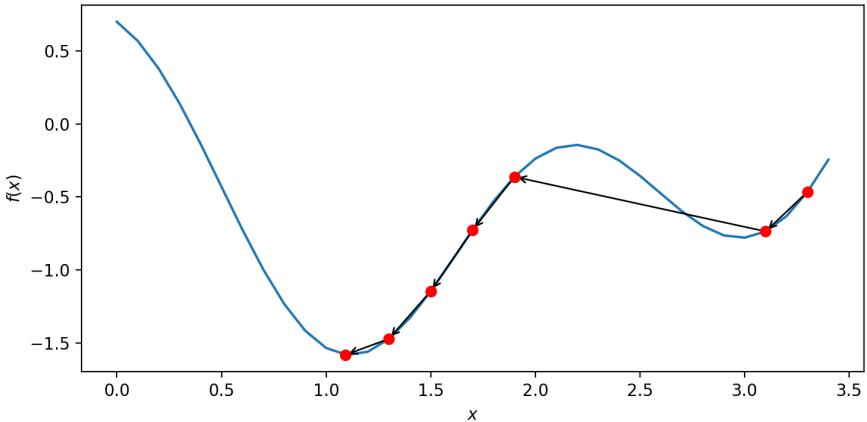


Figure 11.4: In simulated annealing there is a non null probability to jump from a position x_n to another position x_{n+1} where $f(x_{n+1}) > f(x_n)$. This allows the walker (the red point) to jump from a local to an absolute valley.

The gradient descend method is also called the hill-climbing method, especially when the problem is reversed, i.e. we look for the maximum rather than the minimum.

Sometimes stochastic methods are more efficient, especially if the function to be minimised is not smooth. Stochastic methods explore the space of the parameters randomly, guided by some criteria. In **simulated annealing** methods [Kirkpatrick, 1983], for example, the points at iteration n , \mathbf{x}_n is tested against the one obtained at the previous iteration $n-1$, \mathbf{x}_{n-1} . Before accepting it as the new starting point, the algorithm select another random point \mathbf{x}_{rnd} . If $f(\mathbf{x}_{rnd}) < f(\mathbf{x}_n)$, the randomly chosen point is selected as the new starting point instead of \mathbf{x}_n . Otherwise, the random point is chosen instead of \mathbf{x}_n with a probability $P \propto \exp\left(\frac{-\Delta f}{kT}\right)$, where k is a constant, $\Delta f = f(\mathbf{x}_{rnd}) - f(\mathbf{x}_n)$ and T a parameter that decreases with a given scheme as the algorithm proceed. **but there is a non null probability to jump on**

The method is inspired by physics, where $\exp\left(-\frac{\Delta E}{k_B T}\right)$ is the **Boltzmann distribution** giving the PDF of finding a system in an energy state ΔE at temperature T , k_B being the Boltzmann's constant. To understand how the method works consider a very simple case of a function depending on only one parameter $f(x)$, as shown in Fig 11.4. **is explored by the algorithm,**

The walker is represented by the red point and starts on the rightmost part of the figure. Initially it descends the gradient of the function towards the local minimum. However, there is a certain non null probability to jump on the left valley, from where it can descend to the real minimum. The probability to go back to the local minimum is lower because, as the walker proceeds, the temperature T reduces and the difference Δf increases.

Various methods are implemented in `curve_fit` and are selectable by the user,

the default one being a variant of the gradient descend method.

6. A harmonic oscillator with Arduino

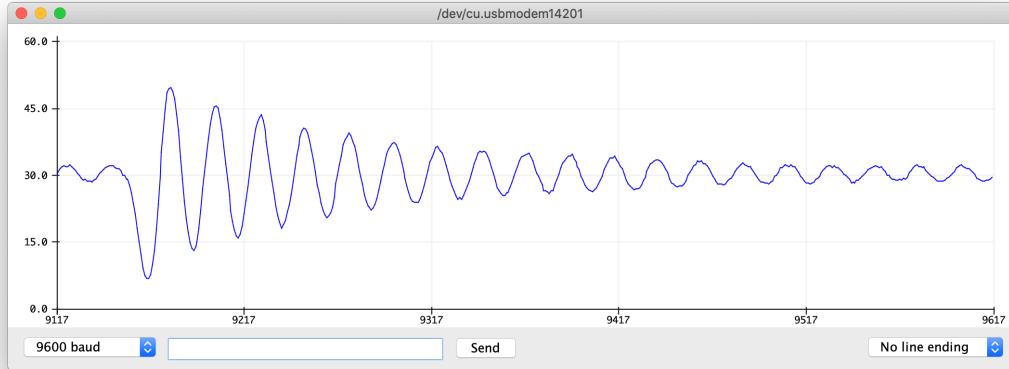
Only few extremely simple systems can be solved analytically rather than being a problem, the opportunity to study a physical system that behaves only partially as a harmonic oscillator is welcome as it allows us to understand that modelling a real system can be a tough task, yet it is an invaluable tool to learn why physicists need to learn computing and programming.

Most of interesting problems in real life applications cannot be solved analytically: only numerical solutions are possible. In fact, solving physics problems numerically is not necessarily worse than solving them analytically, nor they are less instructive. In many cases, numerical approaches can be even more illustrative than analytical ones and we encourage you to compare numerical solutions to known analytical solutions.

On the other hand, it is important to realise systems that behave as expected. This ability, too, is one of those that a physicist must develop. In this section we will try to build a system that behaves as much as a harmonic oscillator. We already discussed the fact that springs, in fact, do not really stretch when a (not too much strong) force is applied to their ends: their coils just twist and the conditions under which the Hooke's Law is valid are fulfilled. For example, it is difficult to realise the validity of the second Newton's Law without friction from experiments.

We can then realise a system suspending a slinky spring with a screen on its free end aiming at reflecting the ultrasound signal emitted by an ultrasonic sensor connected to Arduino, as shown in the figure on the side. The ultrasonic sensor must be oriented towards the screen. We can then measure the distance of the screen as a function of the time after pulling the spring and letting it move under the elastic force. A typical plot of the distance versus time obtained with the Arduino serial plotter is shown below.





The extinction of the amplitude of the oscillations due to dissipative forces is clearly visible in this case, too. With respect to the rubber band case, the shape of the graph is smoother and more similar to the theoretical predictions.

The analysis of the motion of the spring end is almost identical to the one made in the case of the rubber band, in which we measured the acceleration with the smartphone. In this case we measure the position y of the oscillating body and we expect that

$$y = y(t) = y_0 + A_0 \exp\left(-\frac{t}{2\tau}\right) \cos(\omega t). \quad (11.37)$$

Finding the values of y_0 , A_0 , τ and ω consists in minimising the χ^2 defined as

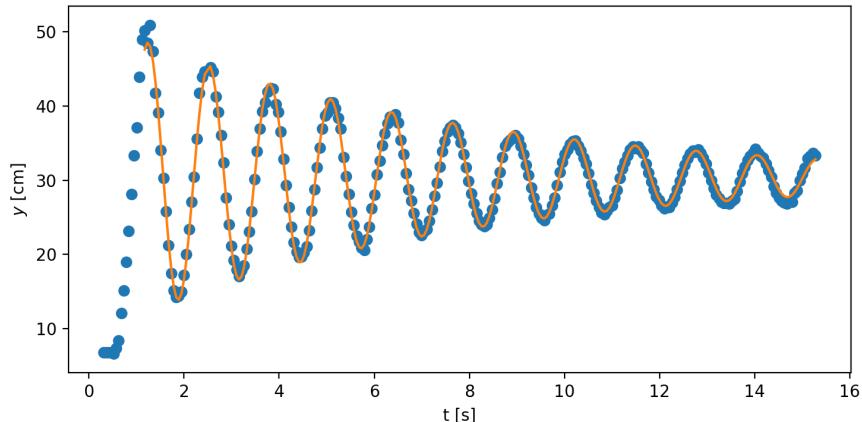
$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - y(t)}{\sigma_i} \right)^2, \quad (11.38)$$

where y_i are the measured distances and σ_i the corresponding uncertainties. According to the manufacturer of the sensor, its resolution is about 3 mm, then we can assume $\sigma_i \simeq 0.3$ cm $\forall i$.

It is instructive to perform the fit with different starting point t_{min} , as in the previous section. The table below shows the main results of various fits with various t_{min} .

i	t_{min} [s]	A_0 [cm]	τ [s]	ω [Hz]	χ^2/ν	$P(\chi^2_\nu > \chi^2_0)$
0	1.2	21.7 ± 0.2	3.54 ± 0.05	4.904 ± 0.002	3.73	0.00
2	2.6	20.0 ± 0.2	3.81 ± 0.05	4.914 ± 0.002	1.65	0.00
3	3.8	19.3 ± 0.3	3.94 ± 0.05	4.920 ± 0.002	1.27	0.01
4	5.1	18.4 ± 0.3	4.10 ± 0.07	4.925 ± 0.002	1.09	0.19
5	6.4	16.7 ± 0.4	4.4 ± 0.1	4.926 ± 0.003	0.88	0.86
6	7.7	14.8 ± 0.5	4.9 ± 0.1	4.931 ± 0.003	0.71	1.00
7	9.0	12.7 ± 0.6	5.5 ± 0.2	4.935 ± 0.004	0.61	1.00
8	10.2	10.2 ± 0.7	6.8 ± 0.5	4.947 ± 0.006	0.50	1.00
10	11.5	9 ± 1	8 ± 1	4.967 ± 0.009	0.46	1.00
12	12.8	7 ± 2	12 ± 5	4.97 ± 0.02	0.52	1.00
13	14.0	8 ± 5	9 ± 7	5.1 ± 0.1	0.54	0.94

A typical fit is shown in the figure below.



The values of ω are consistent among them, contrary to what happens with rubber bands. The extinction time τ becomes longer with time. This behaviour can be interpreted in two ways: either the fit is not enough sensitive when moving the starting point to the right, or, in fact, the dissipative forces are not constant. In this case we can lean towards the second hypothesis, because we observe a rather clear pattern towards larger values of τ . The drift in the value of A_0 is a consequence of the latter effect.

Another feature that can be observed in the table is that the uncertainty on the free parameters increases with t_{min} . This is reasonable, because with increasing t_{min} there are less points to exploit to fit the data.

The quality of the fit can be inferred from the reduced χ^2 that is almost always as good as expected. A more quantitative evaluation of the quality of the fit

can be done considering the cumulative distribution function of a χ^2 -distributed variable. The fit appears to be quite good only for $i \geq 4$. The number in the last column of the table represents the probability that a variable distributed as a χ^2 with ν degrees of freedom attains the value found in the column on its left just because of statistical fluctuation: in other words, the p -value of the fit. If this probability is high, it means that we cannot distinguish between a statistical fluctuation from genuine, systematic deviations from the model and we can ascribe the value of χ^2 to them.

Usually, we prefer to quote the best result that is obtained using the highest possible number of data providing a good fit. In this case, for $i = 4$,

$$\omega = 4.925 \pm 0.002 \text{ Hz}. \quad (11.39)$$

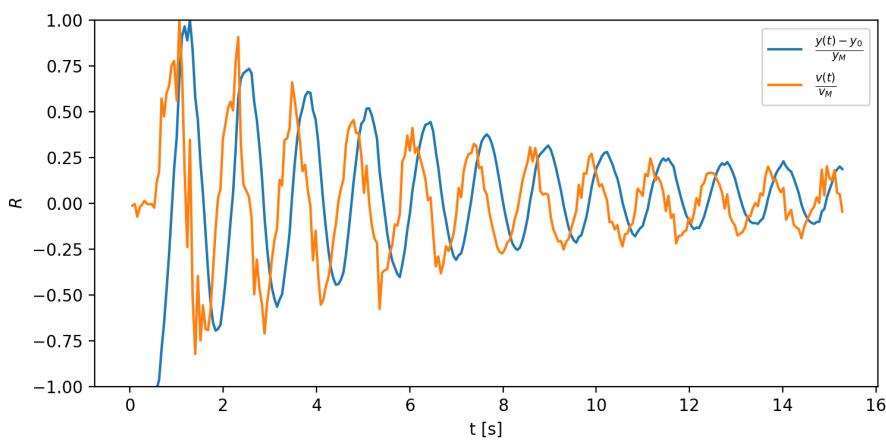
As mass, we can take the one of the system composed by the spring and the cardboard screen, $m = 40 \pm 1 \text{ g}$, such that

$$k = m\omega^2 = 40 \times 10^{-3} \times 4.925^2 \simeq 0.970 \text{ Nm}^{-1}. \quad (11.40)$$

The computation of the uncertainty is left as an exercise, too. With Arduino we can make further observations. We measure $y(t)$, then we can compute $v(t)$ as

$$v(t_i) \simeq \frac{y(t_i) - y(t_{i-1})}{t_i - t_{i-1}}. \quad (11.41)$$

The figure below shows how $v(t)$ and $y(t)$ depend on time. Both are normalised such that they are divided for their maximum $v_M = \max_i v(t_i)$, $y_M = \max_i y(t_i)$.

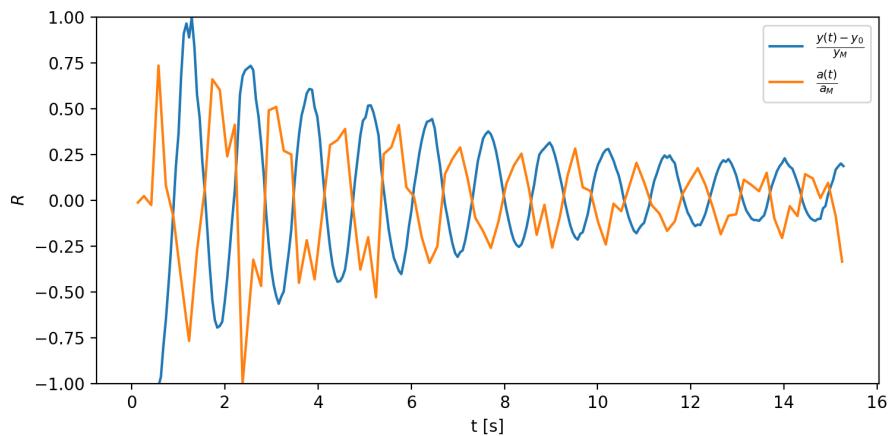


As expected, the graph of $v(t)$ is out of phase by $\frac{\pi}{2}$ with respect to $y(t)$. Similarly, we can obtain a measurement of $a(t_i)$ as

$$a(t_i) \simeq \frac{v(t_i) - v(t_{i-1})}{t_i - t_{i-1}}. \quad (11.42)$$

Due to the relatively large statistical fluctuations, the graph of $a(t)$ is much noisier. As usual, averaging over few points mitigates the effects of fluctuations. The result is shown below.

The plot of the velocity vs time of harmonic oscillator out of phase by $\frac{\pi}{2}$ respect to the one the position. Acceleration is out phase by π with respect to the latter

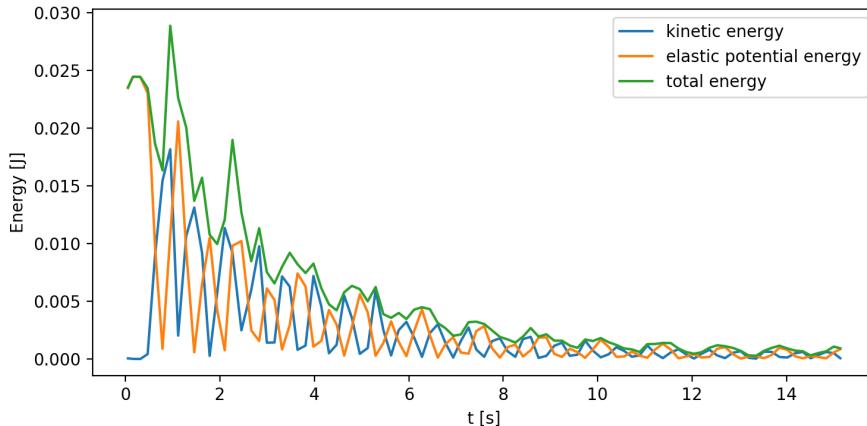


In this case $a(t)$ is out of phase by π , as expected from the fact that $\frac{d^2y}{dt^2} \propto -y(t)$. A plot of the energy versus time can also be done, computing

Energy is easy to measure starting from velocity and position of the oscillator.

$$E(t) = \frac{1}{2}mv^2 + \frac{1}{2}k(y - y_0)^2, \quad (11.43)$$

and is shown below.

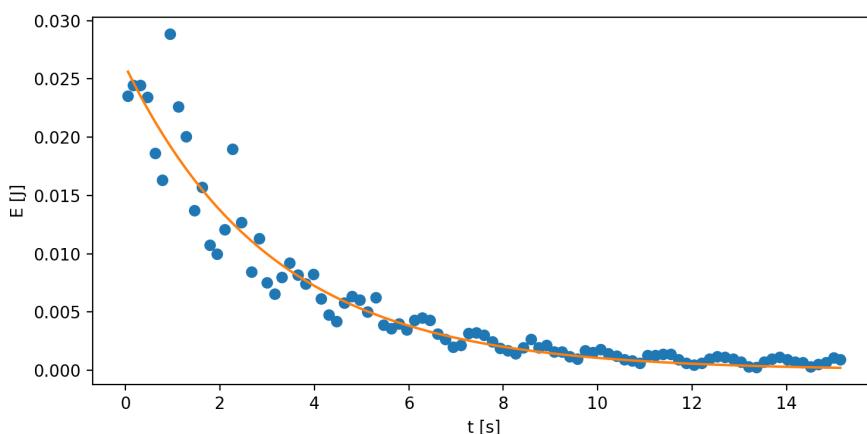


Neglecting fluctuations, the total energy appears to decrease exponentially, as expected from the fact that dissipative forces subtract part of it from the system. The total energy is the sum of kinetic and potential energy that oscillate with opposite phase. If there were no dissipative forces, the total energy was conserved and equal to $\frac{1}{2}kA(0)^2$. Fitting the total energy with an exponential

The work done by friction is proportional to time and to the energy of the system, then the latter decreases exponentially.

$$E(t) = E_0 \exp\left(-\frac{t}{\tau}\right) \quad (11.44)$$

returns $E_0 \simeq 0.026$ J and $\tau \simeq 3.1$ s. The fitting curve is shown below.



The extinction coefficient τ is consistent with the value that can be predicted from the fit to $y(t)$. In fact, assuming that $\tau = \tau(t)$ we can extrapolate to $t = 0$ assuming that, at first approximation, $\tau \simeq \tau_0 + \eta t$ and find that $\tau_0 \simeq 3.3$ s.

From $E(0)$ we can infer $A(0)$ as

$$A(0) = \sqrt{\frac{2E(0)}{k}} \simeq 23 \text{ cm}. \quad (11.45)$$

Data for $y(t)$ can be extrapolated to $t = 0$, too. Identifying the position t_i of the peaks in $y(t)$ one can fit $y(t_i)$ vs t_i with an exponential

$$A(t_i) \simeq A(0) \exp\left(-\frac{t}{2\tau}\right) \quad (11.46)$$

obtaining $A(0) \simeq 22 \text{ cm}$ and $\tau \simeq 4 \text{ s}$, not far from the values guessed from the analysis of energy. In the real system, gravity is not negligible, however, including its effects, the solution of the second Newton's Law is

$$y(t) = A(t) \cos(\omega t) + \frac{g}{\omega^2} + y_0 \quad (11.47)$$

and for $t = 0$

$$y(0) = A_0 + \frac{g}{\omega^2} + y_0. \quad (11.48)$$

Note that

$$\frac{g}{\omega^2} = \frac{mg}{k} \simeq \frac{40 \times 10^{-3} \times 9.8}{0.970} \simeq 0.40 \text{ m} \quad (11.49)$$

is the length of the spring when subject to its own weight $m = 40 \times 10^{-3} \text{ kg}$ that was, in fact, about 40 cm. Correspondingly, neglecting friction, the energy can be written as

$$E(0) = \frac{1}{2}m\omega^2 A_0^2 - mg\left(\frac{g}{2\omega^2} + y_0\right). \quad (11.50)$$

It only differs from the previous one for a constant term. Being the energy defined up to an arbitrary constant, the previous analysis remains valid: it is enough to add the constant

$$C = -mg\left(\frac{g}{2\omega^2} + y_0\right) \quad (11.51)$$

to the expression of the energy to make it consistent with the one computed above.

In summary, using Arduino it is possible to perform detailed experiments on the topic of oscillations.

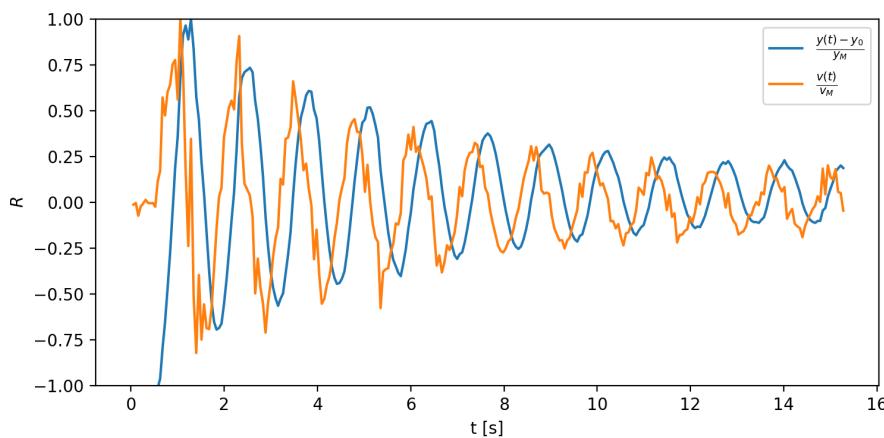
7. Newton's Laws

Using data collected with Arduino we can easily make an experimental verification of the Newton's first and second Law.

According to the first Law, a particle at rest stays at rest unless some force acts on it. As a consequence of the arbitrariness in the choice of the reference frame, even a particle moving at constant speed continues moving at the same speed unless a force acts on it.

The validity of such a statement is clearly shown in the plot with both $y(t)$ and $v(t)$

Data collected with Arduino give us the opportunity to experimentally test Newton's laws.



A spring can be viewed as a **dynamometer**: knowing $\Delta y = y - y_0$, y_0 being the value of y when the dynamometer is in the equilibrium position, we can obtain the intensity of F as $F = -k\Delta y$.

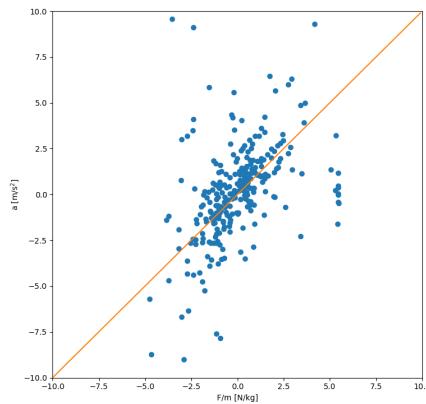
In this plot $y(t) = 0$ corresponds to the case in which the spring is at rest in equilibrium. Here $F = 0$. As soon as $y(t) \neq 0$, the force $F = -ky \neq 0$ and the free end of the spring starts moving. The highest force intensity is reached where $y(t)$ attains its maximum or its minimum. In these positions, the speed is null. In fact, the elastic force progressively reduces the speed of the spring tip till it becomes zero. Due to the elastic force, the speed starts increasing soon after, in the opposite direction. At a certain point $y(t) = 0$ and $F = 0$. Despite $F = 0$, $v(t)$ reaches its maximum in absolute value and, according to Newton's first Law, the spring tip continues moving such that it compresses or stretches the spring, that reacts applying a force to it.

A plot of the harmonically equivalent of $F(t)$ versus t shows its maximum amplitude occurring when the spring is moving by New

Having measured ω from oscillations, we can obtain the values of

$$\frac{F}{m} = -\frac{k}{m}\Delta y = -\omega^2(y - y_0) \quad (11.52)$$

and make a plot of this quantity as a function of a , as below.



The orange line is not a fit, but the prediction made using the Newton's second Law, according to which $a = \frac{F}{m}$. The system, then, allows a direct experimental verification of Newton's Laws.

Indeed, fitting the data with a straight line of the form $a = \alpha \frac{k\Delta y}{m} + \beta$ we obtain $\alpha \simeq 0.67$ and β compatible with zero. The slope of the effective line is lower, indicating that only part of the energy transferred to the system by the elastic force is used to accelerate the mass (a bit less than 70 %). The rest is lost via dissipative forces.

8. A widely applicable model

The Hooke's law studied in this chapter, despite its simplicity, can be applied to a variety of systems. As shown in the chapter opening, the Hooke's Law is nothing but Newton's second Law applied to forces that depend on the distance between the latter is small, i.e. when the function describing the dependency of the intensity of the force on distance can be expanded in a Taylor series and the expansion is truncated at the first order.

In general, the same behaviour is exhibited by any force F described by a function of some parameter x , $F(x)$, such that

harmonic.

$$F(x) \simeq F(x_0) + F'(x_0)(x - x_0). \quad (11.53)$$

For example, the Newton's Law for a pendulum, neglecting friction, is

$$\mathbf{a} = \frac{m\mathbf{g} + \mathbf{T}}{m}, \quad (11.54)$$

\mathbf{T} being the tension of the suspension wire. It must be noted that the mathematical model of a pendulum consists of a pointlike particle of mass m attached to a massless wire of length ℓ . Gravity, then, applies only to the suspended mass and ℓ represents simply the distance between the latter and the points to which it is suspended. Eq. (11.54) can be casted in three scalar equations writing the components of the vectors in a reference frame where the x -axis is horizontal and lying in the oscillation plane, y -axis is vertical and upward, while z is perpendicular to both x and y . In this frame $\mathbf{g} = (0, -g, 0)$ and $\mathbf{T} = (-T \sin \theta, T \cos \theta, 0)$, θ being the angle made by the suspending rope and the vertical direction. Eq. (11.54) reads

$$\begin{cases} a_x = -\frac{T}{m} \sin \theta \\ a_y = -g + \frac{T}{m} \cos \theta \\ a_z = 0 \end{cases} \quad (11.55)$$

If θ is small, $\sin \theta \simeq \theta$, $x \simeq \ell \theta$ and $\cos \theta \simeq 1$, such that

$$\begin{cases} a_x = -\frac{T}{m} \frac{x}{\ell} \\ a_y = -g + \frac{T}{m} a_z = 0 \end{cases} \quad (11.56)$$

Moreover, the acceleration vector $\mathbf{a} \simeq (a_x, 0, 0)$ is parallel to the x -axis, from which we obtain

$$\frac{T}{m} \simeq g \quad (11.57)$$

and

$$a_x \simeq -g \frac{x}{\ell}. \quad (11.58)$$

It is straightforward to show that this equation must have exactly the same solution for that obtained in the case of the Hooke's Law, then the pendulum oscillates with a frequency

$$\omega = \sqrt{\frac{g}{\ell}} \quad (11.59)$$

A real pendulum often includes a model: a massless wire to a string, any system, both gravitational tension and suspensions, a negligible respect

corresponding to a period

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{\ell}{g}}. \quad (11.60)$$

An experiment similar to those made above can be done suspending a mass M to a “wire” of mass $m \ll M$. To avoid rotations, the weight can be suspended using at least two wires. In this case ℓ is the distance between the suspension axis and the center of mass of the weight. Note that one can always write $\ell = \ell_0 + d$ where ℓ_0 is the distance between the axis to which M is suspended and its upper edge, while d is the distance between the latter and its barycentre. Then

$$\frac{T^2}{4\pi^2} = \frac{1}{g} (\ell_0 + d). \quad (11.61)$$

Taking the measurement of T for different ℓ_0 allows us to compute both g and d , respectively, from the slope and the intercept of T^2 versus ℓ_0 .

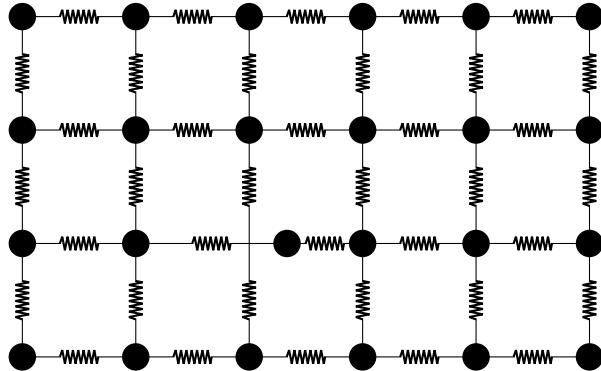
The precision that can be achieved with Arduino is sufficient to appreciate the deviations from the predicted approximated solution. The solution found to the Newton’s equation is such that the period of the pendulum is independent on the angle. This is known as the pendulum’s **isochronism** and is the basis for the construction of mechanical clocks. However, this is only a first order approximation solution valid until $\theta \simeq \theta_0$. It can be shown that, for larger angles

Their period depend,
even if slightly, on the
amplitude of
oscillations

$$T \simeq 2\pi\sqrt{\frac{\ell}{g}} \left(1 + \frac{\theta_0^2}{16}\right). \quad (11.62)$$

This relation, too, is not exact, though is valid for a wider interval of angles (it is obtained expanding $\sin \theta$ to the next order). In this formula θ_0 is the starting angle that is supposed to remain constant because of energy conservation. In practice it will not be such and must be determined extrapolating the amplitude to $t = 0$.

Another model for which the Hooke’s Law turns to be useful is a model for a solid or a liquid, described as an array of pointlike particles (the atoms) bound together by springs, as shown below for a 2D arrangement.



The idea behind this model is that the forces that keeps the atoms bound together in a solid depend clearly on the distance between the atoms. In fact, keeping two solids far apart no interactions is observed. On the other hand, if we press them enough one against the other we can make them merge. As long as there is no way to measure the force between the atoms, the only conclusion to which we can arrive is that $F = F(\Delta x)$, Δx being the interatomic distance. If Δx is small,

$$F \simeq F(\Delta x_0) + F'(\Delta x_0)(x - x_0) + \frac{F''(\Delta x_0)}{2}(x - x_0)^2 + \dots \quad (11.63)$$

Truncating the series at first order, whatever the force between atoms looks like, it exhibits a behaviour close to that of a spring. This explains why ordinary materials increase in size when heated. Heating a body is equivalent to transfer energy to it. The increase in energy to the springs modelling interatomic forces makes them stretching and, as a result, we observe a macroscopic increase of the size.

Few materials (e.g., water) do not *obey this law*, meaning that the above approximation is not appropriate for them.

A so
atom
force
the
distanc
order
behav
and
solid
point
conn
Desp
appre
crude
obse
such
size
heat

Summary

Objects exhibiting the property of elasticity can be treated, at least at first approximation, as a spring.

The elastic force is described by the Hooke's Law.

$$F = -k\Delta x$$

In fact, the Hooke's Law describes any force $F(x)$ that depends on a parameter x , when x is small enough to allow $F(x)$ to be written as a truncated Taylor' series. It is strictly valid only for small Δx .

For example, in solids atoms are held together by a force that manifestly depends on the interatomic distance Δx . Irrespective of how complicated it is as a function of Δx , the latter being small, the force can be written as a truncated Taylor series and behave like an elastic force.

Rubber bands, for example, do not behave like springs, however it is interesting to make experiments aiming at determining their dynamics. Though the motion of an object attached to a rubber band is not harmonic, it is periodic and can be described by a series of harmonic terms.

On the contrary, springs behave much like as predicted by Hooke's Law because their length practically changes infinitesimally: they do not really stretch or compress; they become longer or shorter mostly because the turns rotate around the wire axis.

The solution of Newton's Law for elastic forces is always a combination of trigonometric functions,

being, in this case, the second derivative of the position proportional to the position itself.

The decrease in amplitude observed in real oscillating systems is due to friction. The latter subtract mechanical energy to the system, proportionally to time and energy. The amplitude, then, decreases exponentially.

Even if the system under investigation is not well described by the simplest models, the results obtained making experiments on it helps physicists in identifying the parameters to be optimised in order to make the system more and more similar to that represented by the mathematical model.

A way to improve statistics is to reduce data. It does not consists in reducing the information. Instead, it aggregates data such that their statistical fluctuations are reduced averaging them.

When a systematic error is quoted, the first number represents the statistical error, while the second the systematic one. The latter has no statistical content, even if sometimes is added in quadrature with the statistical uncertainty for convenience.

If the p -value of a statistical test is high we cannot distinguish between statistical fluctuation and systematic deviations from the model, hence we ascribe the differences to statistics.

From position measurements, velocity and acceleration are computed numerically. Also energy can

be easily computed and studied as a function of time and other variables.

Measuring the position of a mass attached to a spring is equivalent to measure the force applied by the spring to the mass. This gives us the opportunity to directly measure a force using a dynamometer. This way, a study of a harmonic oscillator can be turned into an experiment about Newton's Laws.

Never take a model literally: a real pendulum is often imagined as its model, i.e. as a ball attached to a string. If you try to realise a pendulum like this, however, you often have troubles of various kind. In fact, a pendulum is any system subject to both gravity and a tension, where the suspension device (the system applying the tension) has a negligible mass with respect to the rest.

Pendulums are isochronous only in first approximation. Their period depend, even if slightly, on the amplitude of oscillations.

Phyphox

A PHYPHOX timed run automatically starts after the given delay and stops after the specified time elapsed.

A smartphone accelerometer can be used to study springs' dynamics.

Python

Computer programming is important, also because only few extremely simple systems can be solved

analytically in physics. Most of the time, numerical methods are employed.

When linearisation is not possible (or difficult), with the help of a computer it is possible to find the minimum of a χ^2 almost with no effort. Computers look for minima adopting numerical techniques that can be either deterministic or stochastic. Python's `curve_fit()` adopts, by default, a modified gradient descent method.

Reading a CSV file there is often no need to load the whole set of data into the memory. We can select interesting columns using the `usecols` parameter in `read_csv()`.

The `groupby()` method applied to a list returns a list of lists grouped such that each sublist share the same value passed as an argument.

The integer part of the division between two integer numbers is given by the floor division operator `//`.

`find_peaks()`, provided by `scipy.signal`, is a method to identify peaks in a distribution and return their indexes.

When the index of a list `lst` is a list itself, `indx`, the result `lst[indx]` is still a list containing the elements of `lst` whose indices are listed in `indx`.

The `errorbar()` method of `matplotlib.pyplot` shows the error bars together with data points.

Bibliography

- [1] Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983). "Optimization by Simulated Annealing". *Science*. 220 (4598): 671–680.

Chapter 12

Maximum likelihood

Experimental data often distribute as gaussians. The purpose of measuring quantities is to increase our confidence on the knowledge of a particular phenomena. It is important to recognise that we do not always need a measurement to know the value of a quantity. For example, the period T of a pendulum can be easily estimated by eye and we will never consider the possibility to use an hourglass to get it. If we want to increase our knowledge on its value, we must choose an instrument, like a stopwatch, able to provide it with a better resolution, i.e., with less uncertainty, increasing the probability that the measured value T_0 is closer to its *true* value. Thanks to the Bayes' Theorem, the probability attached to each estimation, though subjective, can be formally computed and even if it remains subjective, its value tends to be agreed by everyone. Most importantly, asymptotic results will be independent on our prior subjective knowledge. This chapter deals with the review of the meaning of the measurement process, leading to a formal justification of formulas used in the previous chapters.

1. Bayes' Theorem application to measurements.

According to the Bayes' Theorem, we may know something ~~(the period T of which pendulum)~~ with an a priori knowledge to which we assign a ~~subjective probability~~ $P(T)$ that can be updated through a measurement, making ~~it less subjective~~

The Bayes's Theorem states that

$$P(T|\text{data}) \propto P(T) P(\text{data}|T). \quad (12.1)$$

Here, $P(\text{data}|T)$ is the probability to observe the collected data, given T as the *true* value. $P(T|\text{data})$ is the probability that the period is in fact T , given the observed data. In this textbook, when quoting the words “*true* value”, the word “*true*” is always in italics, because the existence of a *true* value is a prejudice of us: we are not even able to define it for most of the measurements. Bayes' Theorem provides a clear definition of what should be intended for it: a *true* value is a parameter of the system under investigation that determines the data we can collect during a measurement. The probability of obtaining that value for the parameter from our data is proportional to the probability that it determines exactly our set of data.

Measuring the period of a pendulum with a stopwatch whose resolution is σ ends up with a distribution of times that likely resembles a gaussian

$$P(x|T) = Ce^{-\frac{1}{2}(\frac{x-T}{\sigma})^2}, \quad (12.2)$$

representing the probability to obtain x in a trial, while the period is T . The probability that the period is T , given a single measurement T_1 , is then

$$P(T|T_1) \propto P(T) Ce^{-\frac{1}{2}(\frac{T_1-T}{\sigma})^2}. \quad (12.3)$$

We do not know $P(T)$, but in fact we do not need to know it. Manifestly the period of a 1 m long pendulum, cannot be ms or hours, neither minutes. It is enough to look at it to realise that T is of the order of two seconds. This constitutes our prior knowledge from which we can conclude that $P(T)$ is about constant in a range of few seconds centred on 2 s ($P(T) = \alpha = \text{const}$ is a uniform distribution). On the other hand there is no reason to prefer a value to another in the given range, then

$$P(T|T_1) \propto Ae^{-\frac{1}{2}(\frac{T_1-T}{\sigma})^2}, \quad (12.4)$$

with $A = \alpha C$, whose value is fixed by the normalisation condition. The measurement is then expressed as

$$T = T_1 \pm \sigma. \quad (12.5)$$

It is worth noting that now the roles of T_1 and T are inverted: T is the unknown period of the pendulum, while T_1 is the known value of its measurement. In other words, the assigned value to T is the one (T_1) that maximises the probability that the period is the chosen one given that we measured it as T_1 .

Clearly, with such an estimation, we cannot include any statistical effect leading to measurement fluctuations and σ could be underestimated. In this sense, the evaluation of T is subjective, though it is the result of a well defined procedure. To increase our confidence in its value we need to perform more measurements.

Using again the Bayes' Theorem, after taking a second measurement T_2 we can write the probability that the period is T given T_2 as

always subjective, but

$$P(T|T_2) \propto P(T) e^{-\frac{1}{2}(\frac{T_2-T}{\sigma})^2} \quad (12.6)$$

Now our prior $P(T)$ is known with a better confidence to be more reliable.

$$P(T) = Ae^{-\frac{1}{2}(\frac{T-T_1}{\sigma})^2} \quad (12.7)$$

When we make a measurement of a quantity we update our knowledge of the system that follows Bayes' Theorem. (known) values of the measurement something we assign to the (unknown) value with a given probability to assign the value to a range such that the probability measure it

and

$$P(T|T_2) \propto Ae^{-\frac{1}{2}\left(\frac{T-T_1}{\sigma}\right)^2} A'e^{-\frac{1}{2}\left(\frac{T_2-T}{\sigma}\right)^2}. \quad (12.8)$$

A formal proof is a bit tedious and laborious, but it is not too difficult to convince yourself that the maximum probability is attained when

$$T = \frac{T_1 + T_2}{2} \quad (12.9)$$

(see next section, too) and that the resulting probability density function (PDF) is a gaussian whose variance is twice the variance of each, supposed to be the same. In practice, the best estimate of T is

$$T = \frac{T_1 + T_2}{2} \pm \frac{\sigma}{\sqrt{2}}. \quad (12.10)$$

Adding a third, a fourth, a fifth, an N^{th} measurement leads to a sequence that, not surprisingly, provides the average of the measurements as the best estimate of the *true* value of the period with an uncertainty given by $\frac{\sigma}{\sqrt{N}}$. Interestingly enough, the asymptotic result is independent on the choice of the initial prior. If we started with a prior different with respect to a uniform PDF, we arrived at the same conclusion.

In summary, the reason why we average data is that the average ~~is the best~~ is the estimator of the *true* value of a quantity, where the latter is ~~defined as the~~ likelihood one having the highest probability to represent it. This principle ~~is called~~ is the result of **maximum likelihood principle**. According to it, the sequence of ~~measurements~~ is the most probable one given the *true* value of the quantity and ~~the boundary~~ is the conditions (instruments used, environmental conditions, technology adopted, highest etc.). ~~possible probability to happen.~~

2. An experimental proof

Instead of manipulating eq. (12.8) algebraically to prove that the resulting distribution is a gaussian centred on $T = \frac{T_1+T_2}{2}$, we can setup a numerical experiment to observe how the product behaves upon changing the parameters.

After defining a function `gauss(x, mu, sigma)` that returns

$$P(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad (12.11)$$

a plot of the product of two gaussians with mean $\mu_1 = 1$ and $\mu_2 = 2$, respectively, and with unitary σ , can be made defining the function

```

def plot(mu2):
    mu1 = 1
    mu2 += 1
    mu = (mu1+mu2)/2
    x = np.arange(0, 10, .1)
    y = gauss(x, mu1, 1)*gauss(x, mu2, 1)
    label = f'$\\mu_1={mu1};\\mu_2={mu2}$'
    fig.clf()
    plt.plot(x, y, label = label)
    plt.legend()
    arr = dict(facecolor='orange')
    M = np.max(y)*0.2
    plt.annotate('f$\\mu = {mu}$',
                 xy = (mu, M/0.2*0.95),
                 xytext = (mu, M/0.2*0.70),
                 arrowprops = arr, ha = 'center')
    plt.show()

```

and calling it as

```
plot(1)
```

f-strings are evaluated. Here we introduce a new technique to format strings, useful especially when the at runtime. They aformat of the variables to be displayed does not require to specify the length preceded by aof the field and/or the precision, as in this case. *f*-strings are preceded by the *f*-character. Curly character and are evaluated at runtime. Curly brackets contains the variable brackets contains the part of the string as the name of the variable whose content must be displayed variable part of the when printed. In the above example the string is rendered as

string expressed the \$\\mu_1=1\\,\\mu_2=2\$
name of the variable

whose content has tNote that double backslashes (\\) are interpreted as a single backslash (\) being be displayedthe latter a meta-character used as an escape character. It let the following character to be interpreted non-literally.

In turn, this string is interpreted by the Python's LATEX engine and rendered as

$$\mu_1 = 1 \mu_2 = 2 \quad (12.12)$$

The \\, A characters are used in LATEX to typeset a short space in math mode. The annotation is used to define the properties of the arrows used to indicate the relevant points on the plot (their color), while ha = 'center' tells Python to draw the annotation centred with respect to the given coordinates.

The plot can be made for various values of μ_2 and to see how $P(x; \mu_1, \sigma)P(x; \mu_2, \sigma)$ evolves with μ_2 , one can animate the plot showing its content for a short time, clean the figure and drawing a new plot again with different μ_2 .

The Python's `matplotlib` a `Figure` is the top level container for all the plot elements. It can be accessed via the `gcf()` (get current figure) method of a `plotObject` and can be

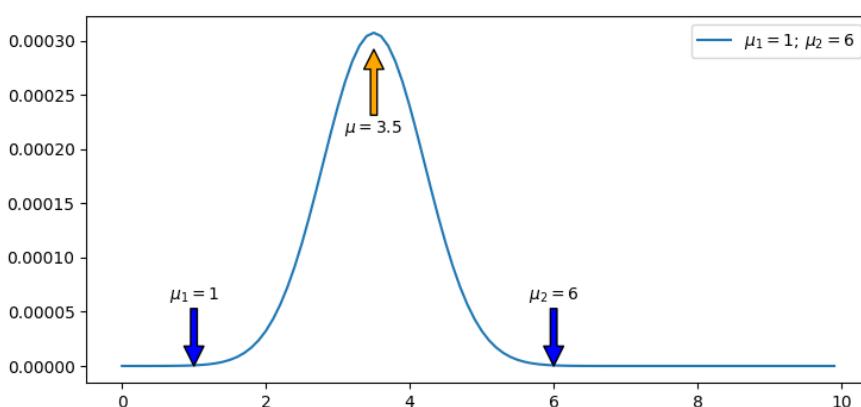
```
fig = plt.gcf()
        ^ obtained via the
        ^ get current
        ^ figure) method
```

We can then profit of the animation subpackage of `matplotlib`. It includes a method `FuncAnimation()` that requires at least two parameters: the first is the figure object to update at each step of animation; the second is the name of a function that is called before each step. The latter must have at least one integer parameter representing the animation step. The maximum number of steps (frames) can be specified as an optional parameter, as well as the interval between two frames, expressed in ms. For example

```
import matplotlib.animation as animation
anim = animation.FuncAnimation(fig, plot,
                               frames = 10,
                               interval = 1000)
```

repeatedly calls the function `plot()` defined above passing, as its arguments, the integer numbers from 0 (included) to 10 (excluded), such that $\mu_2 \in [1, 10]$ (this explains why we used `mu2+=1` in the function code). Each step in the animation lasts for 1 s. Including the `fig.clf()` (clear figure) statement in the `plot()` function, clear the figure content before drawing a new plot, such that the user has the impression that its content actually moves within the figure.

A single step of the animation is shown below for $\mu_2 = 6$.



We leave as an exercise the evaluation of the resulting width of the gaussian in

the cases above and to study what happens when more than two gaussians are multiplied to each other.

3. Parameter estimation

The maximum likelihood principle can be used to estimate unknown parameters from a set of observations x_1, x_2, \dots, x_N . Consider the law relating the length ℓ of a pendulum to its period T

squares method in

searching the parameters that best describe a set of data.

$$\frac{T^2}{4\pi^2} = \frac{\ell}{g} \quad (12.13)$$

and suppose we measured T as a function of ℓ for $\ell \in \{\ell_1, \ell_2, \dots, \ell_N\}$. The value of g can be obtained as the inverse of the slope of the line that describes the data. Using the principle of maximum likelihood we can say that the set of measurements is such that their probability is the highest possible in the given conditions. The single measurement is a gaussian centred at $\frac{\ell}{g}$ and is then value in the given

measurement.

Maximising it is equivalent to minimise $P(T_i|\ell_i, g) = A_i e^{-\frac{1}{2\sigma^2} \left(\frac{T_i^2}{4\pi^2} - \frac{\ell_i}{g} \right)^2}$ $-2 \log \mathcal{L}$, i.e., to

$$(12.14)$$

such that the probability to observe the whole set of data is the product of each single probability of each

measurement is a gaussian.

$$\mathcal{L} = P(\{T_i\}|\{\ell_i\}, g) = \prod_{i=1}^N P(T_i|\ell_i, g). \quad (12.15)$$

More generally, if $y = f(\vec{p}, x)$, the likelihood \mathcal{L} is

$$\mathcal{L} = \prod_{i=1}^N A_i e^{-\frac{1}{2} \left(\frac{y_i - f(\vec{p}, x_i)}{\sigma_i} \right)^2}, \quad (12.16)$$

\vec{p} representing the unknown parameters (in this case $\vec{p} = (g)$ is unidimensional). Taking the logarithm of both members, the product of exponentials turns into a sum of their exponents and

$$\log \mathcal{L} = \sum_{i=1}^N \log A_i - \frac{1}{2} \sum_{i=1}^N \left(\frac{y_i - f(\vec{p}, x_i)}{\sigma_i} \right)^2. \quad (12.17)$$

The first term is a constant and is irrelevant in the search for the maximum of \mathcal{L} , attained where $-2 \log \mathcal{L}$ is minimum, then it can be dropped and the

maximum of \mathcal{L} is found where

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(\vec{p}, x_i)}{\sigma_i} \right)^2 \quad (12.18)$$

is minimum. The latter can be found exploring the space of the parameters upon which $f(\vec{p}, x)$ depends. For example, if $y = Ax$, as in the case in which $y = \frac{T^2}{4\pi^2}$ and $x = \ell$, a plot of

$$\chi^2 = \chi^2(A) = \sum_{i=1}^N \left(\frac{y_i - Ax_i}{\sigma_i} \right)^2 \quad (12.19)$$

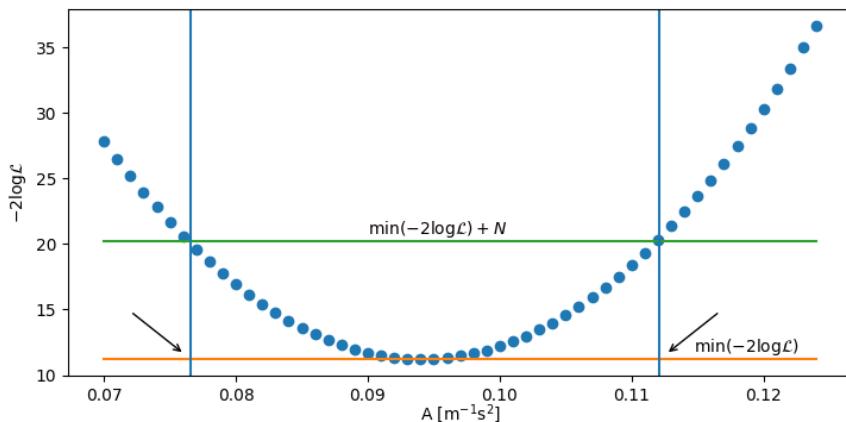
reveals that χ^2 is a parabola as a function of A . Locating its vertex is then equivalent to find the value of A that maximise the probability to have observed the given sequence, under the assumption that $y = Ax$. Substituting A with $A \pm \sigma_A$:

$$-2 \log \mathcal{L} = \sum_{i=1}^N \left(\frac{y_i - (A \pm \sigma_a)x_i}{\sigma_i} \right)^2 = L_0^2 + \sum_{i=1}^N \left(\frac{\sigma_a x_i}{\sigma_i} \right)^2 \pm 2 \sum_{i=1}^N \frac{y_i - Ax_i}{\sigma_i^2} \sigma_a x_i, \quad (12.20)$$

and, observing that the last term in the sum averages to zero and that $\sigma_i = A\sigma_{x_i}$, each term contributes 1 to the sum such that

$$-2 \log \mathcal{L} = L_0 + N, \quad (12.21)$$

L_0 being the minimum of $-2 \log \mathcal{L}$. Locating the abscissa of the points in which $-2 \log \mathcal{L} = L_0 + N$ we can identify the values of A that differ by σ_A from the minimum, as shown below.



The χ^2 , equal to $-2 \log \mathcal{L}$, as a function of the free parameters is a paraboloid in an $(m+1)$ -dimensional space. Looking for the values of the parameters for which $\chi^2 \rightarrow \chi^2 + \nu$ allows us to find the uncertainties on the latter.

In this figure the value of $-2 \log \mathcal{L}$ is plotted against A . It follows a parabola whose minimum is reached for $A_{min} \simeq 0.094$, corresponding to $g \simeq 10.6 \text{ ms}^{-2}$. The arrows indicate the abscissa of the points defining σ_A . Since the shape of $-2 \log \mathcal{L}$ is a parabola they are symmetric with respect to A_{min} and are found at a distance of about 0.017 in SI units. Finally, then,

$$A = 0.09 \pm 0.02 \text{ m}^{-1}\text{s}^2 \quad (12.22)$$

corresponding to

$$g = 11 \pm 2 \text{ ms}^{-2}. \quad (12.23)$$

Sometimes the profile of the log likelihood is not a parabola. If its shape is not symmetric around the minimum, the uncertainty band may be asymmetric, too.

In fact, when deriving the parameters from the data, they are not independent on each other and the likelihood cannot be expressed as the product of independent probabilities. There are as many relationships between data as many estimated parameters (m) and the number of independent PDF is $N - m = \nu$.

In most practical cases, $-2 \log \mathcal{L}$ can always be well approximated with a parabola in the vicinity of its minimum and the uncertainty on the parameters can be evaluated as above using the number of degree of freedom ν . Sometimes the minimum is not symmetric. In this case the uncertainties can be different on each side and the result is expressed using asymmetric uncertainties as, e.g.,

$$g = 11^{+1}_{-2} \text{ ms}^{-2}. \quad (12.24)$$

In the previous chapters we exploit the principle of maximum likelihood defining the χ^2 in an alternative way that, as a matter of fact, is equivalent to the one given above. The above discussion can be regarded as a formal justification of the application of the method, in the framework of Bayesian statistics.

Summary

The reason for which we believe we are able to measure the *true* value of a quantity resides in Bayes' Theorem, according to which the probability to obtain a quantity T from data is proportional to the probability to obtain that data when T is its *true* value. In fact Bayes' Theorem defines what we intend for *true* value.

When we take a measurement of a quantity we update our knowledge on that system thanks to Bayes' Theorem. The (known) value of the measurement of something is the one we assign to the (unknown) *true* value with a given probability. We choose to assign the measured value to a parameter such that we maximise the probability to measure it.

Measurements are repeated to increase our confidence on the results. The probability content of a measurement is always subjective, but adding information makes its estimation more reliable.

According to the maximum likelihood principle, the result of a series of measurements is the one with the highest possible probability to happen.

The product of two gaussian PDF with mean μ_1 and μ_2 and with the same variance is a gaussian with mean $\frac{\mu_1+\mu_2}{2}$.

The maximum likelihood principle is equivalent to the least squares method in searching the parame-

ters that best describe a set of data. The probability \mathcal{L} to observe a set of data is in fact the product of the probabilities to obtain each single value in the given measurement. Maximising it is equivalent to minimise $-2 \log \mathcal{L}$, i.e., to minimise the χ^2 if the PDF of each measurement is a gaussian.

The χ^2 , equal to $-2 \log \mathcal{L}$, as a function of the free parameters, is a paraboloid in an $(m+1)$ -dimensional space. Looking for the values of the parameters for which $\chi^2 \rightarrow \chi^2 + \nu$ allows us to find the uncertainties on the latter.

Sometimes the profile of the log likelihood is not a parabola. If its shape is not symmetric around the minimum, the uncertainty band may be asymmetric, too.

Python

f-strings are evaluated at runtime. They are preceded by an *f*-character. Curly brackets contains the variable part of the string expressed the name of the variable whose content has to be displayed.

A dictionary is a collection of key, value pairs.

`FuncAnimation()` repeatedly calls a function with a variable argument. That argument can be used to draw something in a figure. Cleaning the figure with `clf()` before redrawing it gives the impression that its graphical elements move on the screen.

Chapter 13

Physics in non-inertial systems

Despite appearances, Newton's dynamics is not simple at all, nor very intuitive. The three fundamental principles, for example, took centuries before being recognised as true, as we are used to experience that bodies don't move unless a force is applied to them. Galileo Galilei first realised that friction is responsible for that and only in absence of it Newton's first Law is straightforward. Many conceptual difficulties in classical mechanics, such as those connected to the Newton's third Law or to the usage of non-inertial reference frames, can be at least partly ascribed to the confusion between the term *force* and the term *interaction*. Forces are the result of an interaction and interactions are only effective between at least two entities. To this respect, what we call *fictitious* or apparent forces can be considered, in fact, as real as gravity or the elastic force. On the other hand we can measure them, then they exist. The confusion comes from the fact that, in the past, forces and interactions were in fact synonyms, as any interaction gave rise to a force, represented as a vector in Newtonian mechanics. Today, we know that this is not always the case. The weak interaction, for example, is responsible for radioactive decays. The interaction cause the *change of state* of a system composed of an atom of a given species into another in which the state is composed of an atom of different species, together with one or more particles (a photon, an α -particle or an electron and a neutrino). There is no place to which we can attach a vector, in this case, hence there is no force, in the Newtonian sense. We better say that interactions are responsible for the change of state of a system and the effects of interactions can be often, but not always, represented by forces. In this chapter, we perform experiments in which we explore the physics in non-inertial frames, to dissipate doubts and to let you to familiarise with them.

1. Dynamics in non-inertial systems

Consider an accelerometer at rest in a gravitational field, such as those found in smartphones or in Arduino sensors. As shown in Chapter 6, irrespective of its actual shape and technology, an accelerometer can be thought as a pair of masses m connected with a spring. Assuming a massless spring (i.e. a reference frame. If it is a spring whose mass m_S is much less than m), each mass is subject to its weight mg , directed downward. If the accelerometer is at rest on a table, with its reference frame horizontal, gravity produces on both masses a force directed downward experiences fictitious forces. and, as a result, the acceleration \mathbf{a} of both masses is null. Consequently, the

distance between the masses remains constant and the accelerometer output is zero. If, on the other hand, the spring is kept vertical, while for the mass at the bottom $\mathbf{a} = 0$ for the same reasons, on the mass at the top gravity is cancelled by the elastic force provided by the spring, such that

$$m\mathbf{g} = -k\Delta x \hat{y} \quad (13.1)$$

\hat{y} being a unit vector directed downward. In this case $\Delta x \neq 0$ and, being m and k known, the system provides a reading of the acceleration of the system \mathbf{g} . If the accelerometer were in a non-inertial frame, the acceleration experienced by the system is

$$\mathbf{a}' = \mathbf{a} - \mathbf{A} - \boldsymbol{\omega} \wedge (\boldsymbol{\omega} \wedge \mathbf{r}) - 2\boldsymbol{\omega} \wedge \mathbf{v} - \frac{d\boldsymbol{\omega}}{dt} \wedge \mathbf{r}, \quad (13.2)$$

The most general expression of the acceleration of a system includes the linear acceleration of the reference frame, the centripetal acceleration, the Coriolis term and the Euler's acceleration.

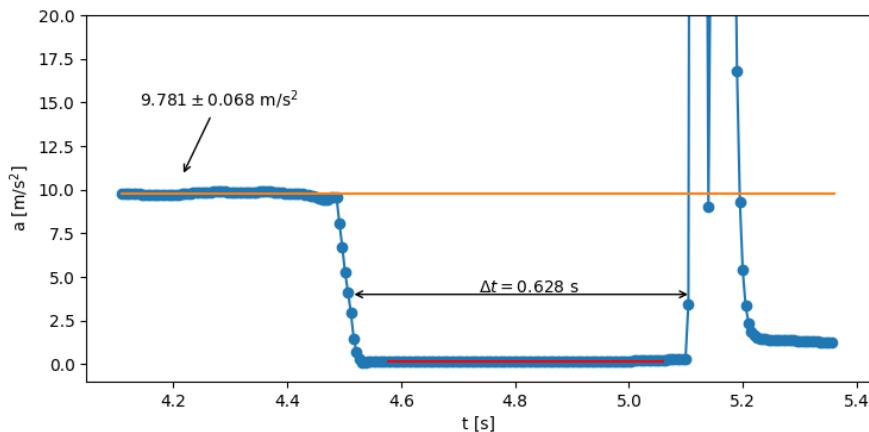
An accelerometer at rest always measures the gravity acceleration, since gravity affects the distance between the two proof masses kept separated by an elastic device.

2. Free fall

If $\omega \approx 0$, $\mathbf{a}' = \mathbf{a} - \mathbf{A}$. When a smartphone falls freely, its accelerometer is in an accelerated reference frame with $\mathbf{A} = \mathbf{g}$. Since it moves with acceleration $\mathbf{a} = \mathbf{g}$ in the laboratory frame, its accelerometer measures $\mathbf{a}' = 0$.

The acceleration measured in a freely falling reference frame is null. This is the foundation of General Relativity. An experiment to demonstrate that is super easy. It is enough to start the acceleration experiment in PHYPHOX keeping the phone at some height (e.g., above your head) for few seconds, then leave it falling on a soft surface, as a bed (be sure not to damage your device: choose a large bed to perform the experiment, such that the smartphone does not fall on the floor after bouncing on the mattress surface).

During the fall the absolute value of \mathbf{a} is expected to be null. In fact, the result of such an experiment is shown below.



At the beginning of the experiment $a' = 9.78 \pm 0.07 \text{ m/s}^2$, being $A = 0$. When the phone falls it drops to almost zero for $\Delta t \approx 0.63 \text{ s}$, corresponding to an height

$$h = \frac{1}{2}g\Delta t^2 \approx 1.94 \text{ m}. \quad (13.3)$$

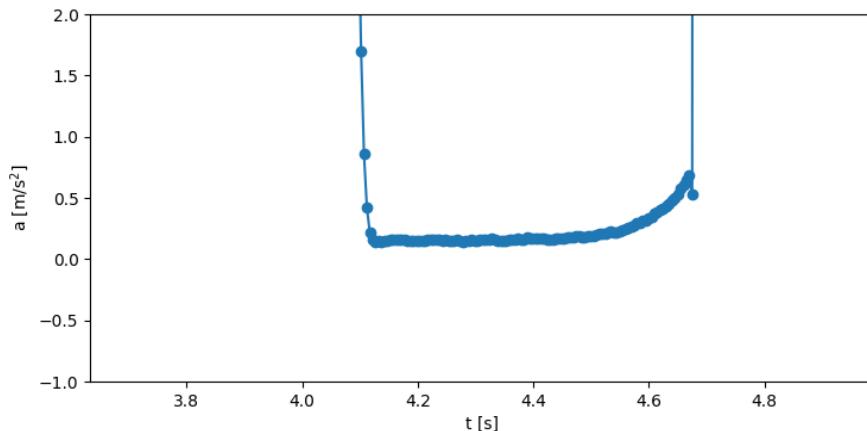
Looking carefully at the data one can see that $a' \neq 0$, and it can even slightly increase with time. The average of the first experimental points after the fall has started is shown in red and can be significantly different from zero, due to mechanical stresses or electronic biases introduced by the manufacturer while mounting the phone that can often depend on temperature, too.

Such an experiment appears to be trivial, but in fact it was the experimental foundation for the General Relativity. Remarkably enough, at the time of its

In fact, a cheap accelerometer can be affected by systematic biases and the measured acceleration can be not null while falling. Sometimes, the acceleration measured during free fall may not be constant, due to the fact that the device rotates around one or more axis.

formulation, the available technology prevented Einstein to actually make the experiment and his results are based on what he called a *gedankenexperiment*, a German word for “thought experiment”.

It is worth noting that, often, the absolute acceleration of the phone during free fall is not constant, but slightly increase with time, as shown below, where a zoomed portion of the previous graph is reported.



The increase is due to the fact that, while falling, the phone may tend to rotate mostly around its x -axis, leading to the development of further accelerations in its reference frame.

3. Custom experiments with PHYPHOX

PHYPHOX allows collecting data from several sensors at the same time using custom experiments.

A very interesting feature of PHYPHOX is its ability to collect data from more sensors at the same time. Custom experiments can be defined clicking on the on the bottom right angle of the display (see Fig. 13.1).

The simplest method to define a custom experiment is to click on ADD SIMPLE EXPERIMENT. This opens a dialogue in which the user can select the available sensors to read during the experiment. The resulting experiment has as many tabs as the number of sensors included in it.

In an experiment that includes the accelerometer and the gyroscope, both sensors are read at (almost) the same time. In fact, the readings are made sequentially, but setting the acquisition rate to zero, sensor readings are done at the fastest possible rate and each set of data can be considered, in many cases, simultaneous.

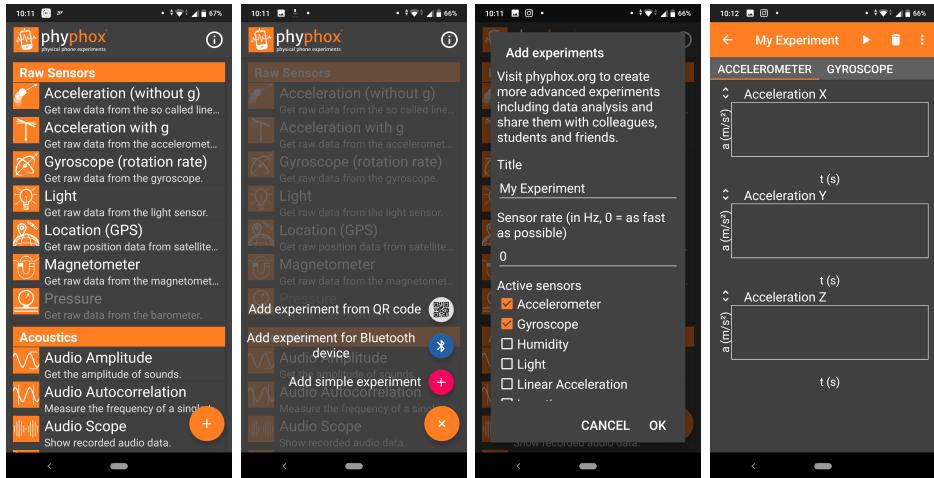


Figure 13.1: Screenshots taken during the definition of a PHYPHOX custom experiment. After choosing ADD SIMPLE EXPERIMENT, the user has the ability to select the desired sensors. The experiment has as many tabs as the number of selected sensors.

Once exported, data will be saved in as many files as there are sensors included in the experiment.

4. Centripetal acceleration

Centripetal acceleration manifests when a system finds itself in a rotating reference frame. Remembering that forces cause accelerations according to the second Newton's Law, if a body is in an accelerated frame, it experiences a centrifugal force. The latter is said to be **fictitious** (apparent) or **inertial** because there is no source for it. While the source of gravity is mass and the elastic force is exerted by a spring, fictitious forces have no source: they appear just because the system experiencing it is in a non-inertial reference frame, such that the subject can measure an acceleration a that can be interpreted as the presence of a force whose intensity is $F = ma$. The centripetal acceleration correspond to the term

$$\mathbf{a}_c = -\boldsymbol{\omega} \wedge (\boldsymbol{\omega} \wedge \mathbf{r}). \quad (13.4)$$

For a point-like particle rotating around an axis perpendicular to the xy -plane at distance r from it, $\boldsymbol{\omega} \wedge \mathbf{r}$ is a vector lying on the rotation plane directed as the velocity of the particle, as given by the right-hand rule. The cross product of $\boldsymbol{\omega}$ and the latter is still a vector lying on the xy -plane, antiparallel to \mathbf{r} .

The computation can be done using vector coordinates, as well, remembering that the cross product of two vectors \mathbf{a} and \mathbf{b} can be written as the determinant of a 3×3 matrix whose elements are the unit vectors in the first row and the

Centrifugal force is as real as other forces, because it can be measured. It is said to be fictitious because there is no source for the observer, while the centripetal acceleration (its counterpart in an inertial reference frame), is the result of an interaction.

To be produced better

coordinates of the factors in the product as the second and third row. In our case $\omega = (0, 0, \omega)$ and $\mathbf{r} = (r_x, r_y, 0)$, then

$$\boldsymbol{\omega} \wedge \mathbf{r} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ 0 & 0 & \omega \\ r_x & r_y & 0 \end{vmatrix} = -\omega r_y \hat{x} + \omega r_x \hat{y} \quad (13.5)$$

and

$$\mathbf{a}_c = \boldsymbol{\omega} \wedge (\boldsymbol{\omega} \wedge \mathbf{r}) = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ 0 & 0 & \omega \\ -\omega r_y & \omega r_x & 0 \end{vmatrix} = -\omega^2 r_x \hat{x} - \omega^2 r_y \hat{y} \quad (13.6)$$

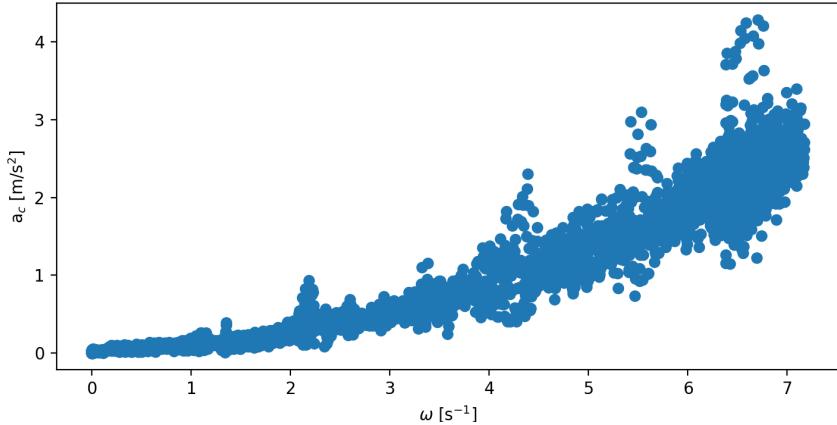
~~This such that product~~ $-\omega^2 r$. Finding the cross product this way is particularly useful ~~does when doing it with~~ a computer. Applying this latter method, it is easy to show that the cross product is not commutative, and that $\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a}$

An experiment means be set up positioning an accelerometer and a gyroscope in a rotating frame. ~~be~~ A smartphone with both sensors can be placed into a ~~salad spinner~~ keeping its width aligned with its radius and kept in position with ~~adhesive tape or cable ties~~. Record players or turntables like those used to support ~~TVs and monitors~~ are valid alternatives, as tightening it to the spokes of a bicycle wheel.

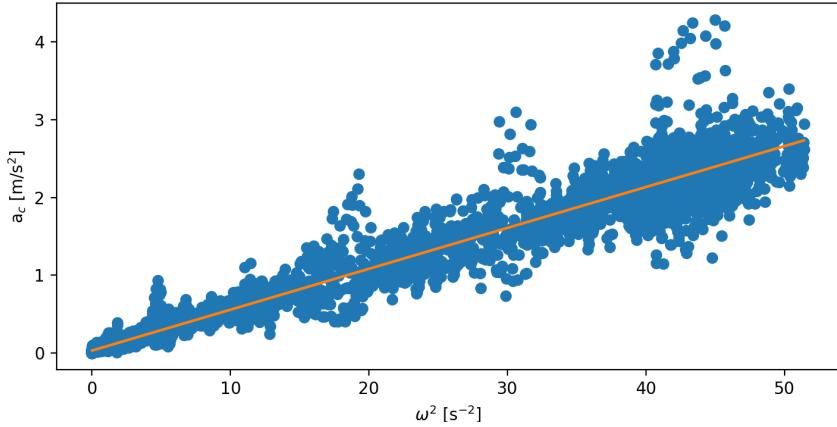
The experiment consists in collecting the acceleration \mathbf{a} and the angular velocity $\boldsymbol{\omega}$ while keeping the device rotating at different speeds. As soon as the experiment starts, wait few seconds to measure the steady state conditions, then make the device rotate varying the speed continuously.

In the above configuration the device rotates along its z -axis, such that $a_z \simeq 0$ ~~as well as it may be~~ 0. Indeed, the support may not be perfectly horizontal ~~and in order to obtain~~ the centripetal acceleration we need to subtract the components of ~~g~~ from a_x , a_y and a_z . Using data collected during the first ~~seconds~~ ~~when the system was at rest~~, we get g_x , g_y and g_z as the average of ~~the corresponding~~ ~~rest~~ accelerations, and subtract them from all the points.

A plot of $a = \sqrt{a_x^2 + a_y^2 + a_z^2}$ as a function of $\omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$ is shown below (this plot is done assuming that $\boldsymbol{\omega}$ is measured simultaneously with \mathbf{a}).



A clear parabolic dependence of the type $a \simeq r\omega^2$, as expected, can be spotted ~~by eye~~ ~~arising~~ ~~from the plot~~. An unweighted fit to a vs ω^2 with a straight line ~~allows~~ ~~to~~ ~~simplifies~~ ~~the way to~~ ~~find unknown~~



Here, r is the distance of the accelerometer from the axis of rotation. Our result means that the device is placed in one of the points of a circumference of radius r from the rotation axis. In order to identify which of them, we observe that the centripetal acceleration makes an angle θ with the phone axis equal to the angle between the latter and the vector position of the accelerometer, such that

$$\theta = \tan^{-1} \frac{a_y}{a_x}, \quad (13.7)$$

as shown in Fig. 13.2.

Meas...
ident...
when...
acc...
insid...
dist...
of ro...
circu...
whic...
place...
in wh...
latter...
the a...

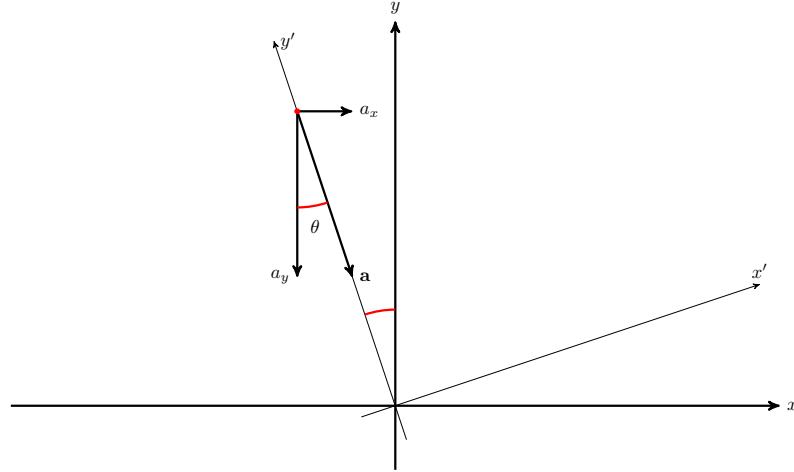
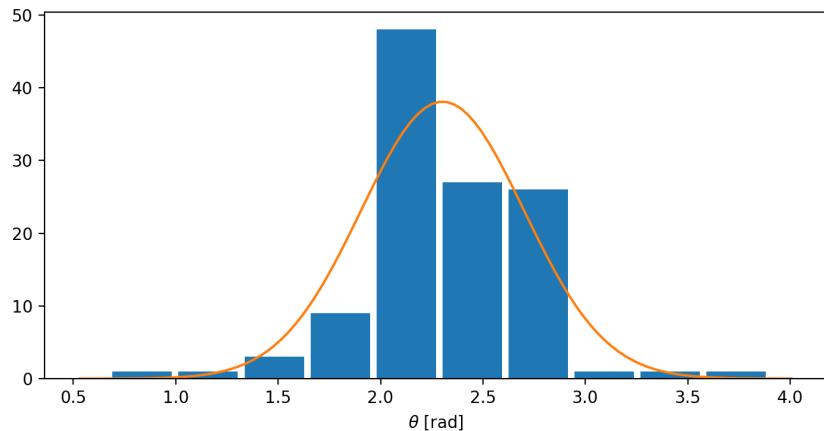


Figure 13.2: If the sensor, represented by the red dot, is not aligned with the phone axis, its acceleration makes an angle θ with respect to it.

A histogram of θ is shown below, with a gaussian superimposed, such that its mean μ is equal to $\langle \theta \rangle$ and its width σ is equal to the standard deviation of the measurements σ_θ .



The normalisation of a gaussian is given by its integral $N\Delta\theta$, divided by $\sqrt{2\pi}\sigma$.

The proper normalisation constant is computed as

$$C = \frac{N\Delta\theta}{\sqrt{2\pi}\sigma_\theta} \quad (13.8)$$

where N is the total number of events in the histogram and $\Delta\theta$ is the bin width.

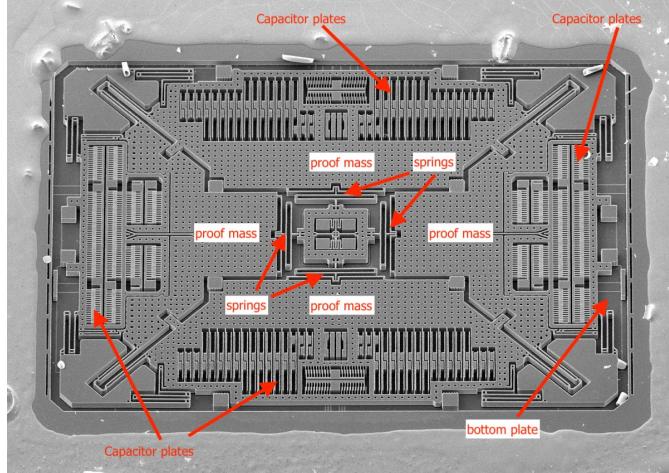


Figure 13.3: Microphotograph of a tuning fork gyroscope. The x -axis is directed to the right, the y -axis upward and the z -axis is perpendicular to the page, exiting from it. Courtesy of Adam McCombs.

5. Coriolis' acceleration

Smartphone's accelerometers always measure the acceleration in their own reference frame. The accelerometer is clearly at rest in that frame and it is then difficult to measure the term $-2\omega \wedge \mathbf{v}$, unless the phone itself is put in a rotating frame and moves within it.

It is instead interesting to see how gyroscopes work, exploiting just the Coriolis's force. As seen above, inertial forces develop in accelerated frames. Fig. 13.3 shows a microphotograph of a MEMS gyroscope, known as a tuning fork gyroscope, taken using an electron microscope.

Consider a reference frame whose origin is at the center of the device, such that its x -axis is directed to the right, the y -axis points upward and the z -axis exit from the page. The four pitted wings vibrates at a frequency f as the tines of a tuning fork, driven by an external force impressed through the elastic structures attached to the central body. The velocity $\mathbf{v} = \mathbf{v}_0 \cos(2\pi ft)$ at which each wing vibrates is known: when the right wing moves to the right, the left one moves to the left; when the top wing moves upward, the bottom one moves downward, and viceversa.

If the system rotates along the x -axis, the device experiences a Coriolis force

$$\mathbf{F}_C = -2\omega \wedge \mathbf{v}. \quad (13.9)$$

Such a force is null on the left and right wings, and has opposite directions for the top and bottom wings, is directed as \hat{z} and has an absolute intensity equal to $F_C(t) = 2\omega_x v(t)$. The system, then, is subject to a torque and the top wings is pushed or pulled against the plate beneath it, while the bottom one

Gyroscope
Coriolis
on a de
non-in
frame.
are for
such th
device
experi
force. I
gyrosc
masses
that vi
frequency
force is
the dist
the pro
fixed pl
by an e
to them

tends to move in the opposite direction. Both the plate beneath the wings and the latter form a capacitor: when their distance $d \propto 2\omega_x v_0 \cos(2\pi ft)$ varies, its capacitance do so and a current can be measured proportional to d and, in turn, to ω_x .

Similarly, if the device rotates around the y -axis, the force on top and bottom wings is null, while the one acting on the left and right wings is again directed along \hat{z} . ω_y is measured from the current drawn from the capacitors made of the left and right wings, together with the plate beneath them.

Finally, if the system rotates around \hat{z} , the four wings experience a force. Assuming the directions of \mathbf{v} as above, the top wing experience a force to the right, the bottom one is subject to a force directed to the left, the right wing is pushed downward, while the left one upward. In this case the device measures the variation of the capacitance of the system made of the proof masses and fixed plates visible at the four sides of the device. This way, the system measures ω_z .

In order to observe the effects of a Coriolis' acceleration, the accelerometer must move (for simplicity, better to let it move along a straight line) with respect to a rotating frame like, e.g., moving across a rotating carousel.

6. Euler's acceleration

Accelerometers can be used to experiment with accelerations not usually explicitly treated in physics courses, like the Euler's acceleration. It is enough to measure accelerations in a frame that rotates at non-constant velocity.

The last term in eq. (13.2) is called the Euler's acceleration, named after Leonhard Euler (1707–1783), and represents the variation of the velocity with respect to time due to a change in the rotational speed of a point that rotates around an axis. This term is quite easy to understand, being analogous to the linear acceleration. Indeed, a particle moving along a circular track with non constant speed, experience an acceleration along the tangent of its trajectory, as a particle moving along a linear track feels an acceleration directed parallel to its velocity.

In order to spot this term in experimental data a gyroscope and an accelerometer can be put close to the inner rim of a bucket, suspended by means of two twisted ropes such that it starts to rotate. In an experiment like this, if alignment was perfect, the angular velocity $\boldsymbol{\omega}$ is vertical, as \mathbf{g} . Its orientation changes from up do down and viceversa when the direction of the rotation changes. The centripetal acceleration is expected to be horizontal and perpendicular to the angular velocity, while the Euler acceleration is expected to be tangent to the trajectory of the accelerometer. In our experiment, the relevant directions are shown in Fig. 13.4.

Plots of the components of the acceleration, the angular velocity and acceleration is shown below as a function of time.

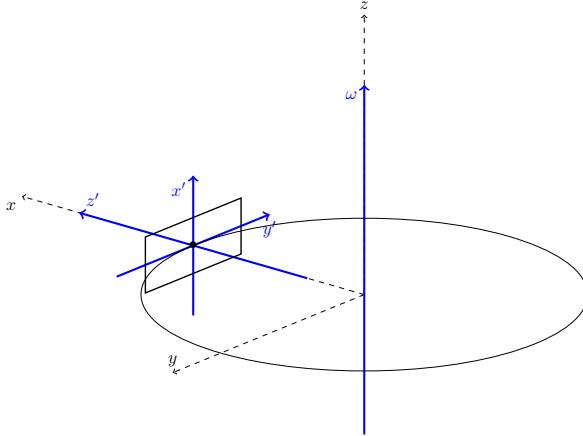
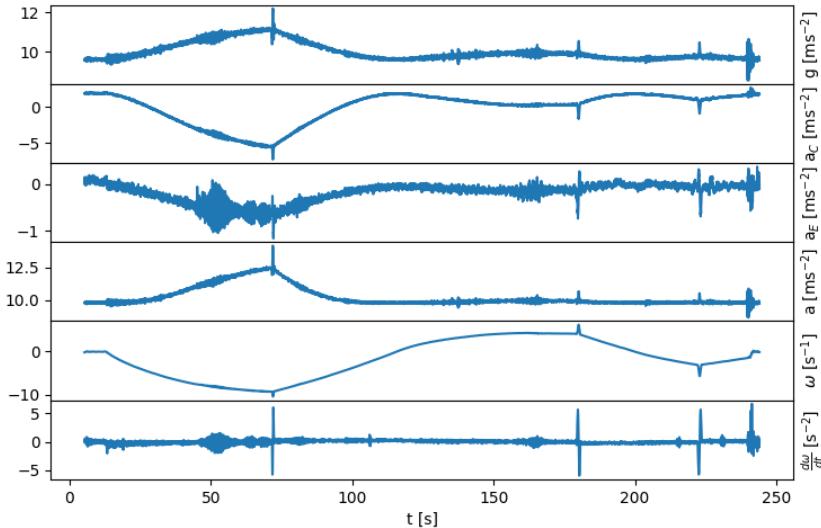


Figure 13.4: Reference frames used in an experiment in which a device with a gyroscope and an accelerometer rotates around a vertical axis. Primed axis are those of the device, while non-primed ones are those of the inertial laboratory frame.

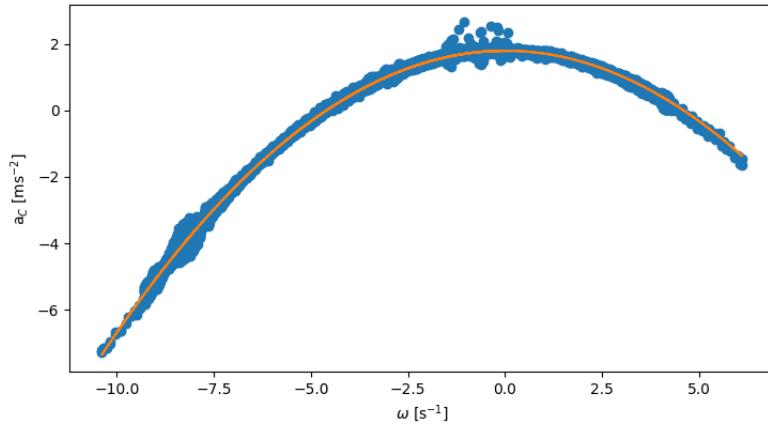


The rotation is almost uniform for the whole duration of the experiment (the angular acceleration $\frac{d\omega}{dt} \approx 0$), but when the rotation changes sign at $t \approx 70, 180, 220$ s.

Correspondingly, we observe a jump in the acceleration's components. The centripetal acceleration a_C follows the graph of ω , being $a_C = \omega^2 r$, with opposite sign. A similar, milder, behaviour is observed for the component tangent to the trajectory a_E . The latter is not expected, being $a_E = \frac{d\omega}{dt} r$, however, the experiment is not perfectly aligned and the accelerometer detects a component of

the centripetal acceleration along its y' -axis. The same happens for the vertical component (the one along the z' -axis).

A plot of the acceleration along z' as a function of the angular velocity ω is shown below.



Data ~~Distributed along~~ follows a parabola, as expected from the fact that $a_C = -\omega^2 r$. The vertex ~~located where~~ where $\omega = 0$, is not at $a_C = 0$ because of the imperfect alignment of the device. In fact

approximations done
during the analysis.

$$\text{Refining the analysis, } a_{z'} = -\omega^2 r \sin \theta_{\omega,r} + g_{z'}, \quad (13.10)$$

~~however, may lead to better results or at least, to a better understanding of possible systematics.~~

$$a_{z'} \simeq -\omega^2 r + g_{z'}. \quad (13.11)$$

Fitting data against this model provides the following results.

$$\begin{aligned} r &= 8.468 \pm 0.004 \text{ cm} \\ g_x &= 1.793 \pm 0.001 \text{ ms}^{-2}. \end{aligned} \quad (13.12)$$

The value of r is consistent with the diameter of the bucket $D = 2r = 16.5 \pm 0.5$ cm even if $\theta_{\omega,r} \neq \frac{\pi}{2}$. In fact

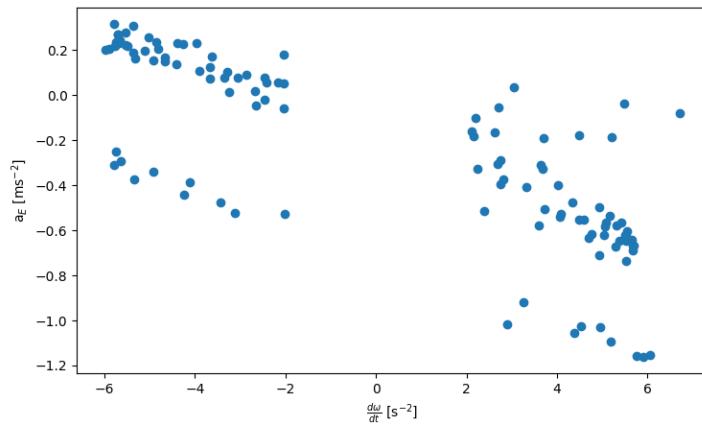
$$\theta_{\omega,r} = \frac{\pi}{2} + \alpha \quad (13.13)$$

and for small α we can expand the sine in a Taylor series truncated at the first order such that

$$a_{z'} \simeq -\omega^2 r (1 + \alpha^2) + g_{z'}. \quad (13.14)$$

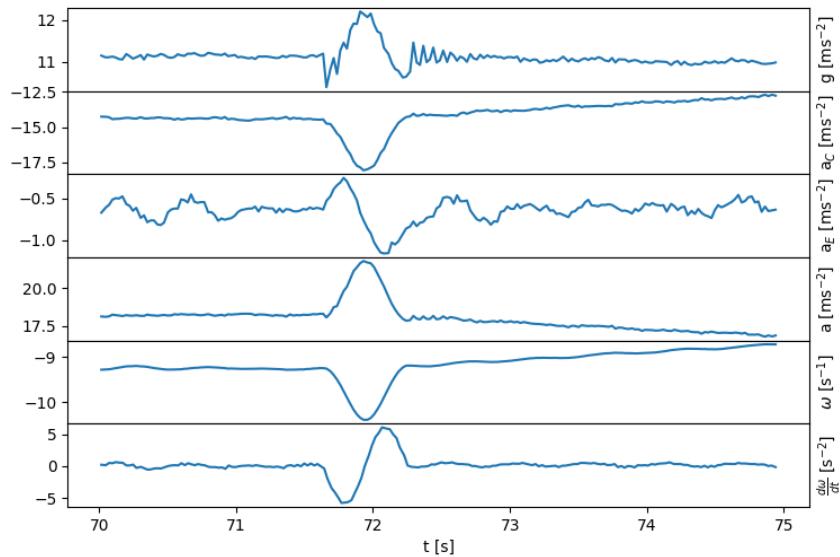
What we measured, then, is not r , but $r (1 + \alpha^2)$.

The Euler's acceleration is expected to be tangent to the trajectory, i.e. parallel to the y' -axis. Indeed, a plot of $a_{y'} = a_E$ as a function of $\frac{d\omega}{dt}$, shown below for $|\frac{d\omega}{dt}| \geq 2 \text{ s}^{-2}$, makes it clear that there is a linear relationship between the acceleration and the angular acceleration.

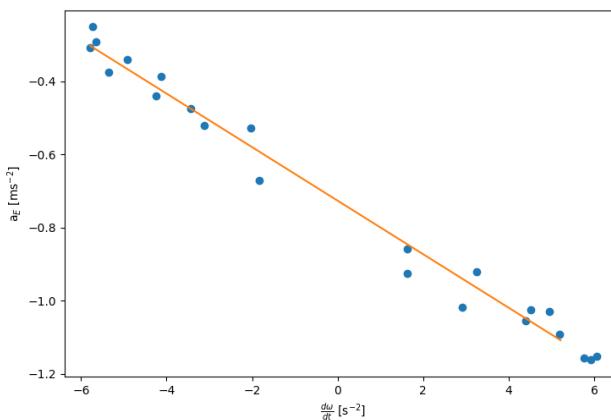


Data distribute along two parallel lines because, during the experiment, the angular velocity changes sign few times.

The larger angular acceleration is experienced by the device around $t \simeq 70$, 180 and 220 s. Let's look at data zooming in the region $70 < t < 75$ s, after subtracting $a_{z'}$ as obtained above.



This impulsive phase corresponds to the time at which one of the rope jumps on a node or the other useful. During this phase the rope has a shudder that reflects into both the acceleration components and the angular rotation vector. The same happens clearly visible in the plot of a_E that in fact is expected to be $a_E \propto \frac{d\omega}{dt}$ (in this case the change in ω happens only in its magnitude and the angle between the angular acceleration and \mathbf{r} is practically $\frac{\pi}{2}$). A plot of a_E versus $\frac{d\omega}{dt}$ is then expected to yield a linear relationship, as shown below.



Data are fitted with a straight line, whose absolute value of the slope is the radius of the trajectory $r_E = 7.3 \pm 0.2$ cm. The intercept $q = -0.727 \pm 0.009$ m/s² is,

as above, due to the fact that the device is not perfectly aligned. This explains also the difference between r_E and r . A similar analysis can be conducted for $t \simeq 180$ s and $t \simeq 220$ s. Averaging the results we obtain $\langle r_E \rangle = 8 \pm 2$ cm, well consistent with the result obtained from the centripetal acceleration.

Summary

Any accelerometer always measure accelerations in its own reference frame. If it is found in a non-inertial reference frame it experiences fictitious forces.

An accelerometer at rest always measures the gravity acceleration, since gravity affects the distance between the proof masses kept separated by an elastic device.

The most general expression of the acceleration of a system includes the linear acceleration of the reference frame, the centripetal acceleration, the Coriolis term and the Euler's acceleration.

The acceleration measured in a freely falling reference frame is null. This is the foundation of General Relativity.

In fact, a cheap accelerometer can be affected by systematic biases and the measured acceleration can be not null while falling. Sometimes, the acceleration measured during free fall may not be constant, due to the fact that the device rotates around one or more axis.

Centrifugal force is as real as other forces, because it can be measured. It is said to be fictitious because there is no source for it for the observer, while the centripetal acceleration (its counterpart in an inertial reference frame), is the result of an interaction.

The cross product does not commute.

Experiments in rotating frames can be easily done

using an accelerometer and a gyroscope together.

During these experiments it may be needed to correct data to take into account the non null readings of an accelerometer at rest.

As usual, linearising the relationship between two quantities simplifies the way to find unknown parameters.

Measurements allow to identify the point where the accelerometer is placed inside the device. The distance from the axis of rotation defines a circumference along which the sensor is placed. To determine in which point of the latter it is, we exploit the acceleration data.

Gyroscopes exploit the Coriolis' force exerted on a device in a non-inertial reference frame. Proof masses are forced to move such that, when the device rotates, they experience the Coriolis' force. In tuning fork gyroscopes, the proof masses are the tines that vibrates at known frequency, while the force is measured from the distance between the proof masses and fixed plates connected by an elastic medium to them.

Accelerometers can be used to experiment with accelerations not usually explicitly treated in physics courses, like the Euler's acceleration. It is enough to measure accelerations in a frame that rotates at non-constant velocity.

Euler's acceleration is easy to understand, however figuring out its components from plots collected

during the experiments is extremely instructive.

Often, results from experiments are consistent with approximations done during the analysis. Refining the analysis, however, may lead to better results or, at least, to a better understanding of possible systematics.

Filming the experiment may turn to be useful in spotting features that can be hard to explain without any clue.

Phyphox

PHYPHOX allows collecting data from several sen-

sors at the same time using custom experiments.

Python

To compute a cross product numerically, better use the determinant method.

$$\mathbf{a} \wedge \mathbf{b} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

The normalisation of a gaussian is given by its integral $N\Delta\theta$, divided by $\sqrt{2\pi}\sigma$.

Chapter 14

Dynamics of rigid bodies

Newtonian mechanics is often mistaken to be coincident with the dynamics of point-like particles. Real life objects are certainly not point-like, however, in most cases, the physics of point-like particles is enough to describe the observations that can be done on them. This is not always true. Sometimes, the behaviour of non-point-like objects (physicists call them **rigid bodies**) exhibit effects that cannot be described by the physics of point-like ones. Often, these effects are every surprising and largely counterintuitive. This chapter is devoted to the study of some properties of rigid bodies, from which we derive a formal definition of what a rigid body is and debunk beliefs arising from the naive application of Newton's Laws.

1. A cylinder rolling along an incline

A cylinder along an incline is subject ~~to its weight and~~ to the normal force exerted by the plane. Naively, then, ~~one would believe~~ that the cylinder must fall along the plane with an acceleration ~~of a rigid body being~~ inclination. If such a description was correct, given that ~~downward~~ is the speed of the cylinder, ω its angular velocity and r its radius ~~we can only write~~ considered

~~equal to that of a~~

~~point-like particle.~~

$$\frac{d\omega}{dt} = \frac{g \sin \theta}{r} \text{ Making an experiment, } \quad (14.1)$$

~~even if only a bit more~~

Let's then set up an experiment to verify if the theory is correct or not. In this section we illustrate the measurement of the angular acceleration $\frac{d\omega}{dt}$ using a smartphone (see next section for details). The measurement can be done using Arduino, too, as described in Section 3.

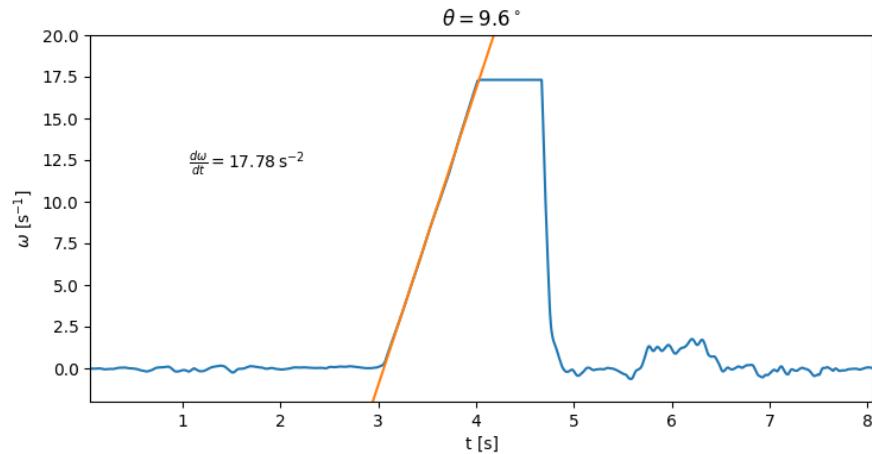
The experiment is done using a cardboard poster tube, whose diameter fits well with many smartphones' width (see Fig. 14.1).

The inclined plane can be done, for example, using an ironing board raised on one end. Angles are measured using a smartphone. PHYPHOX provides the angle of the device with respect to each axis with the inclination tool, that exploits the ratio between the components of the gravity acceleration measured by the accelerometer. In order to make the experiment we exploit a gyroscope,

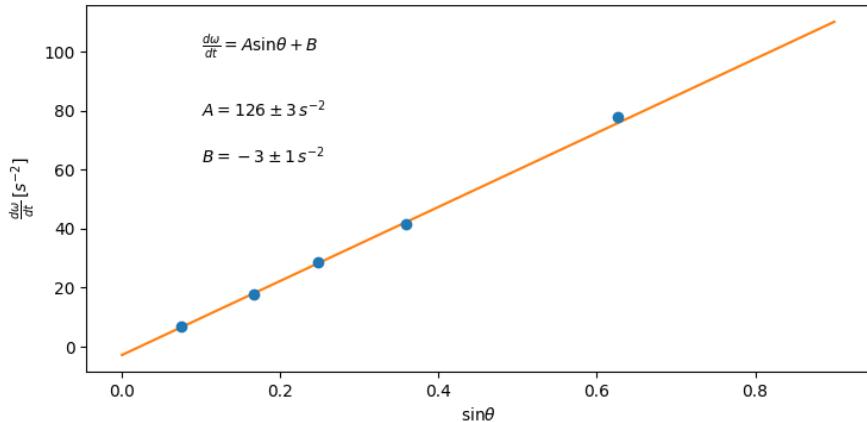


Figure 14.1: Preparation of an experiment on a rolling cylinder. A smartphone is carefully inserted in a poster tube and kept in position with some kitchen paper.

measuring the angular velocity around the cylinder's axis, with the cylinder rolling down starting from a steady state. An example of these measurements is shown below.



A linear fit to experimental data is shown as an orange line. From the slope of the line one gets $\frac{d\omega}{dt}$. Data between 4.5 and 5 s appear to be constant because the maximum range of the instrument has been reached (and the instrument is said to saturate). Repeating the experiment at different angles one can collect and plot $\frac{d\omega}{dt}$ as a function of $\sin \theta$ and fit again with a straight line, as below.



Assuming the same behaviour of a point-like particle, the slope of the line is expected to be

$$A = \frac{g}{r} \simeq 239 \text{ s}^{-2} \quad (14.2)$$

for $r = 4.1 \pm 0.1$ cm, while the intercept B should be compatible with zero. Indeed, we obtained $B = -3 \pm 1 \text{ s}^{-2}$ that is null within three standard deviations interval. However, $A = 126 \pm 3 \text{ s}^{-2}$ is significantly different from our expectations.

After checking that there were no systematic errors in our experiment, we end up with the conclusion that the theory was wrong. In fact, even if no real system is point-like, many behaves as such and we often fall in this *trap*. Indeed, treating an object as point-like is an approximation that depends on the degrees of freedom of the object. The number of the degrees of freedom is the number of independent parameters needed to specify the stationary state of a system. For a point-like particle, the stationary state is defined by its position and it has three degrees of freedom. Such a number can be reduced if the particle is constrained: a particle moving along a straight line has only one degree of freedom, while if it moves on a plane it has two degrees of freedom.

For a rigid body, the position is not sufficient to completely specify its state. Its orientation with respect to the reference frame is important, too. For example, the axis of a cylinder can be oriented in an arbitrary way for which we need extra degrees of freedom (the coordinates of the unit vector of the cylinder axis). In some cases, non rigid systems may have internal degrees of freedom. In a spring, for example, the length is an internal degree of freedom.

Often, the comparison of the relative size of the body r with respect to the system characteristic size R is taken as a criterion to consider the point-like approximation as valid. For example, planets are often considered as point-like

Treating an object as point-like is equivalent to neglect any degree of freedom in excess with respect to the three translational ones. The size and shape of the object does not matter: planets and stars can often be treated as point-like, even if they are giant, while a coin has the shape of a disk even if it behave much like a point-like object in many cases.

even if their diameter r can be as large as thousands of km. The reason for that is ascribed to the fact that their distances R from the Sun are orders of magnitude larger. In fact, the reason for which the point-like approximation holds is that their rotational degrees of freedom are negligible with respect to their positional ones (moreover, they are almost independent on the latter, too). A ball rolling down an incline never behaves as a point-like particle, irrespective of the ratio between its radius and the length of the path. On the contrary, the point-like approximation for a box sliding along an incline is valid, irrespective of the size of the box, provided it does not rotate along any axis. The need to take into account non translational degrees of freedom depends on what we are interested in. For example, if we are interested in the time needed for a spring to fall from a given height, we might ignore its internal degree of freedom, unless the height is comparable with its length. In the latter case, the time can be much shorter, since its lower end can touch the floor much before its center of mass does (however the time spent by any point of the spring while moving is equal to that of a point-like particle). Generally speaking, the point-like particle approximation is valid when either the non translational degrees of freedom are negligible or absent, or when they are decoupled from the translational ones.

Internal forces The reason for which the point-like dynamics is not valid in the experiment maintaining the shape of an object must be taken into account to predict the dynamics of an object. In many textbooks they are ruled out from equations because they sum up to zero, however that does not mean that they are absent: it only have consequences on the dynamics of the objects' centre of mass. Their effects is modeled throughout the moment of inertia.

Applying Newton's laws taking into account internal forces, a symmetric body like a cylinder, a sphere or a disk of mass M rolling down a plane, rotates around its axis with an angular velocity ω increasing with time as

$$\frac{d\omega}{dt} = \frac{Mgr}{Mr^2 + I} \sin \theta, \quad (14.3)$$

r being its radius and I its **moment of inertia**. I depends on the mass distribution of the object and is defined as

$$I = \int_V \rho(\mathbf{x})r^2 dV, \quad (14.4)$$

$\rho(\mathbf{x})$ being the mass density of the body at position \mathbf{x} , r the distance of \mathbf{x} from the axis of rotation, while the integral is taken over the volume of the body V .

For an [Empty toy physics](#) $I = Mr^2$.

[textbooks for a compilation of moments of inertia](#)

It is worth noting that, recalling that $v = \omega r$, the acceleration $\frac{d\omega}{dt}$ obtained from the above equation substituting $I = Mr^2$ is half of the one predicted in the case of a point-like particle. It means that only half of the energy of the system is spent to make it move along the plane: the rest is spent in keeping it rolling. For a point-like object $I = 0$ and we recover the equation of motion predicted in this case. Note that, in the case of the wagon used in Chapter 10, only the moment of inertia of its wheels contributes that, however, is negligible because they are light and small.

From $A = 126 \pm 3 \text{ s}^{-2}$, and knowing the system mass $M = 0.606 \pm 0.001 \text{ kg}$, we can then compute the moment of inertia of our device as

$$I = (9.0 \pm 0.5) \times 10^{-4} \text{ kg m}^2. \quad (14.5)$$

This value can be compared with what can be expected from the theory. The system can be modelled as a cylinder of radius r and mass $M_C = 0.392 \pm 0.001 \text{ kg}$, with a rectangle of mass $m_S = 0.200 \pm 0.001 \text{ kg}$ at its center and filled with a homogeneous mass $m_p = 0.014 \pm 0.001 \text{ kg}$ of paper, such that

$$I_{th} = \left(\frac{m_S}{3} + M_C + \frac{m_p}{2} \right) r^2 = (7.8 \pm 0.4) \times 10^{-4} \text{ kg m}^2. \quad (14.6)$$

Despite the crude approximations, the agreement can be quantified as

$$\frac{|I - I_{th}|}{\sqrt{\sigma^2 + \sigma_{th}^2}} = \frac{90 - 78}{\sqrt{25 + 16}} \simeq 1.9. \quad (14.7)$$

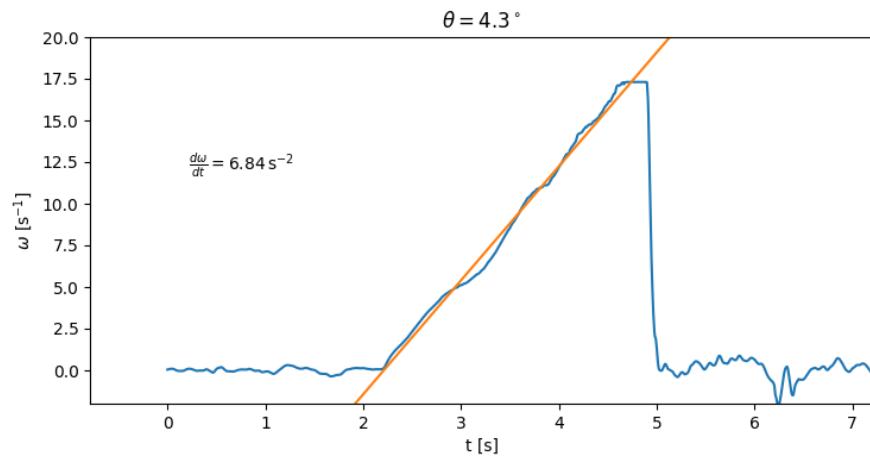
It must be noted that one of the main contribution to the uncertainty comes from the angle measurement, whose uncertainty can be relatively large because the inclined plane often bends under its own weight and the angle is not constant along its length. This contribution is not included in the fit, in which we take into account only the uncertainty on $\frac{d\omega}{dt}$.

In principle one can take it into account modifying the denominator in the χ^2 definition from σ_y^2 to $\sigma_x^2 + \sigma_y^2$.

Alternatives to the poster tube are, e.g., that of a packing tape or a PET bottle cut on the side of its neck. The latter is less stable because it is easier to distort, and loose its shape.

A non symmetrical system can be easily spotted placing the tube on a horizontal plane. If it is symmetric it should not oscillate around an equilibrium position, while if the distribution of the masses is not symmetric, the heavier part tend to stay below the center and the system easily rotate spontaneously to reach the equilibrium position. It is very difficult to reach a perfectly balanced system,

however, if ω grows fast enough (i.e. the angle is large enough) this problem is strongly mitigated. Below we show a measurement of the angular velocity as a function of time in the case of a poorly balanced system rolling down on a plane inclined at an angle $\theta < 5^\circ$.



The oscillations around the mean line is a clear sign of lack of symmetry.

Measurements can be repeated changing the inertia of the system adding, e.g., two identical weights on each side of the phone (e.g. two batteries). If the mass of one of the weight is m , the inertia of the body increases approximately of $2mh$, h being the distance between the axis of the cylinder and the center of mass of the weight.

2. Using a smartphone's gyroscope remotely

Many smartphones have a three-axis gyroscope on board that can be exploited to measure the angular velocity around its three axis. This device is used in gaming applications, as well as in virtual and augmented reality ones, for which the device must know if and how fast it is rotated.

With **PHYPHOX** provides access to the raw data of the gyroscopes. Only the angular velocity around the y -axis is relevant for this measurement. It is important to insert the device in the tube such that it lies along one of its diameters and stay stable during the fall. The paper surrounding it helps keeping it in position. In particular, we find it is also useful to manipulate the device during insertion. In fact, when the smartphone is inside the tube, it is impossible to operate on its touchscreen. Cleverly enough, **PHYPHOX** allows to connect to the phone using a Wi-Fi signal. Clicking on the menu icon on the top right in the experiment, a menu appears. Selecting the ALLOW REMOTE ACCESS checkbox makes it possible to control

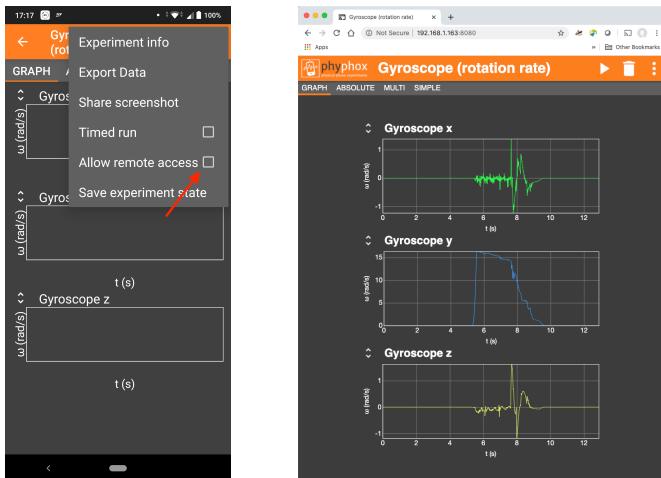


Figure 14.2: To allow remote access to PHYPHOX click on the check box indicated by the red arrow on the left picture. On the right: the display of the smartphone as seen on a web browser.

the App remotely using just a web browser, provided that it is on the same network of the smartphone. It is enough to connect to the website shown on the bottom part of the display when the option is activated to see a copy of the phone display security

reasons, they do not. This way one can control the start and stop of the experiment, as well as download data to the local device. In order for the browser to connect to PHYPHOX, the App must be active. Manipulating the smartphone using the PHYPHOX. In these paper while inserting it into the tube avoids activating functions inadvertently. cases you may want to

The experiment is then very fast: once the connection has been set up, start data taking, leave the tube rolling down the plane and catch it when it is close to the end to avoid damaging your device because of abrupt stops. Repeat the measurements few times to evaluate the uncertainties and spot systematic errors, then download data, then change the angle and make another set of measurements.

Remember to include useful information, such as the angle at which data were taken, in the data file name.

3. Arduino Gyroscopes and I2C communications

The same experiment may be repeated using Arduino. Cost-effective gyroscopes are available on the market, often in conjunction with accelerometers (so-called IMU's (Inertial Measurement Unit)).

These devices cannot work as analog devices, because they must provide a plethora of information. Moreover, often, the resolution required is not sufficient to be obtained with Arduino's analog to digital converters.

An IMU (Inertial Measurement Unit) is a device embedding both a gyroscope and an accelerometer to trace the state of a moving object.

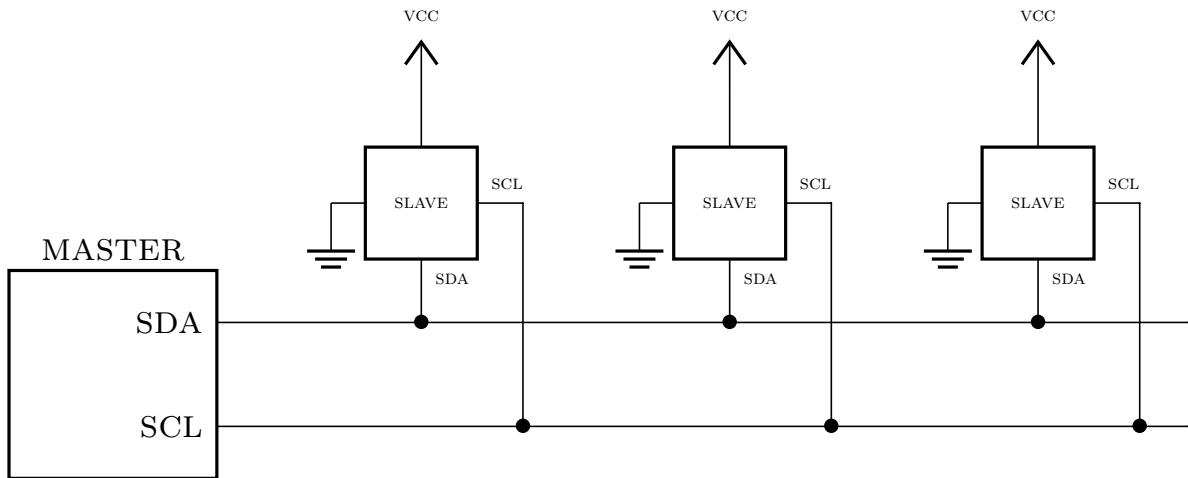


Figure 14.3: In an I₂C network two wires are used for communication between a master and many slaves. The SCL line distributes a clock signal to synchronise data exchanged on the SDA line.

In these cases, sensors are equipped with a device able to exchange data with another device via a serial digital protocol. In other words, data are digitised on board, then sent as a sequence of bits 0 and 1 represented as voltage pulses over a serial line. Polling the line and getting the height of the pulses allows the receiving device to collect data in the digital form. In most cases the protocol allows communications in both senses, such that an external device can also send information (usually to configure it) to the sensor.

I₂C is a **master/slave**One of the most popular protocols is I₂C (standing for Inter Integrated Circuit), **bidirectional**supported by Arduino. The I₂C protocol, illustrated in Fig. 14.3, allows many **synchronised**devices on the same line to communicate with a master, thanks to two electrical **communication**lines SDA and SCL to which all the devices are attached. Any I₂C compatible **protocol supported by**devices have then at least four leads: two for power supply, to be attached, **Arduino**. **Data travels**respectively, to the GND and to the 3.3V or 5V Arduino pins, and two for **com-** **over the SDA line**munication. Each Arduino flavour has its own specification about the position **synchronised by pulses**of the SDA and SCL lines. On an Arduino UNO they coincide respectively, with **on the SCL line**analog pins A4 and A5 and with the two pins close to the AREF one, on the side of digital pins. Data travels over the *Serial DAta* (SDA) line synchronised by clock pulses appearing over the SCL (*Serial CLock*) line.

Each device on the lines can act both as a **master** or as a **slave**. Usually Arduino always behaves as a master and sensors as slaves.

Is master/slave politically incorrect?

While we were writing this textbook, the “Black Lives Matter” movement was raising its voice to

defend the rights of black people and, in general, to fight against social inequalities. Needless to say, we fully support their claims, however other claims stemmed on that for which we have not the same opinion.

In particular, in information technology a relatively old debate revamped about the opportunity of using terms like master/slave or blacklist/whitelist because of their racial connotation.

First of all, those connotations has nothing or little to do with black people. Blacks were only the last slaves in western countries. In the past, it was plenty of slaves that were not coloured. Ancient romans, for example, were used to have slaves that were as white as them, even if often, but not always, belonging to different ethnical groups. Similarly, the term blacklist probably origins from the “black book”: a list of people who committed crimes dating back to XIV century [Merriam-Webster, 2020] compiled by King Henry VIII (the adjective “black” being referred to the color of its cover).

Moreover, in particular for the master/slave pair, its usage in information technology is appropriate just for its negative connotation: a slave, in fact, is a device that cannot take decisions by itself and totally depends on the master, without which it cannot (it is not free to) operate. Adopting a metaphor or alternative terms does not illustrate as clearly its principles of operation. For example, one can use director/executor, primary/secondary, main/replica pairs, etc. While in many cases they are appropriate (e.g., a master copy of a software repository can be called main, while its slave copies may be called replica), they are not as appropriate in the case we are talking about. In particular, a director can be certainly thought as someone who gives orders, however executors that are not slaves are entitled to refuse them or execute them differently with respect to what the executor expects.

Often, using words with negative meaning is more appropriate with respect to other politically correct choices and does not reflect the attitude of the speaker in different frameworks. Hiding the negative meaning of a word does not mean that the concept it represents disappears. On the contrary, carefully choosing a proper name may be of some help to make people reflect on the profound meaning of the words used. This is why, in this textbook, we prefer to continue using the traditional wording, believing that its usage will increase the awareness of their inopportunity in different contexts.

As usual, one can easily find the libraries needed to operate any sensor with Arduino, either using the Arduino IDE with its own library manager or just looking for the source code on software repositories like GITHUB. And, as usual, one can choose among multiple implementations by multiple authors. The choice depends on criteria such as portability, ease of use, efficiency, etc.. Libraries are meant to simplify communication management with a specific device, however it is worth knowing, at least, the principles of native protocols: one day or another you may need to write your own library or to modify the behaviour of a publicly available one.

In this section we provide an example that serves as a starting point for the realisation of experiments based on other I₂C-based devices. Most sensors

Most sensors can be operated using a library provided by its manufacturer or a third-party. Libraries are useful, but knowing the native protocols boosts your reputation.

for Arduino have their own identifier depending on the manufacturer: however many of them are designed around a common architecture. As such, the sensor produced and distributed by a company can often be operated using the libraries for a different sensor based on the same electronics. Reading the documentation for the specific model usually provides such an information. The measurements made in this chapter can be done using IMU's based on the **MPU6050** chip [**MPU6050**].

I₂C based slaves can often, I₂C-based device can be configured such that they can provide the best possible resolution depending on the range to which they are sensitive. As that, for example, they illustrated in the previous chapters, sensors often measure quantities that are in fact related to the one for which they are designed. A gyroscope, for example, as possible or as fast measures in fact electrical currents that depends on the capacitance of the sensor as possible. Many that, in turn, depends on its angular velocity. The digitised current I_i measured parameters, listed ion axis i is always a dimensionless integer number between 0 and $2^N - 1$, N being the number of bits in the ADC on board ($N = 16$ for the **MPU6050**). can be chosen by the This number is then converted into an angular velocity ω_i expressed as a floating userpoint number by multiplying it for a calibration constant C , such that

$$\omega_i = CI_i. \quad (14.8)$$

Setting properly the The current I_i , in turn, depends on the sensitivity of the device. It is in fact full scale range to possible that the device responds differently to the same stimuli: if we need to which a device is measure only very low angular velocity we can configure the gyroscope such that sensitive improves its the maximum digitised current $2^N - 1$ is drawn when ω is, say, $250^\circ/\text{s} \approx 4.36 \text{ Hz}$, precision. such that the resolution of the device is

$$\Delta\omega = \frac{\omega}{2^N - 1} = \frac{250}{2^{16} - 1} \approx 0.004^\circ/\text{s}. \quad (14.9)$$

On the other hand, if we need to measure angular velocities up to $2000^\circ/\text{s} \approx 34.9 \text{ Hz}$, with the same number of bits we can achieve a resolution of

$$\Delta\omega = \frac{\omega}{2^N - 1} = \frac{2000}{2^{16} - 1} \approx 0.03^\circ/\text{s}. \quad (14.10)$$

Once properly configured, I₂C device are ready for providing measurements upon request. The master asks the slave to provide them and the slave responds transmitting as many bytes as requested to host $2^N - 1$ bits. The library running on the master transform the integer provided by the slaves into the desired quantity applying correction factors, if needed, and multiplying by the calibration constants.

The documentation provided by the chip's manufacturer always reports a table (register map) with the meaning of so called **registers**. A register is a group of 8 bits internal to the chip, each with a specific function. The master in the I₂C

network either write or read from the registers, addressing the slave providing Every I₂C device its address: a unique identifier, usually provided as a two-digit hexadecimal number.

address. The address is set at the factory and the same address is shared by all the devices based on the same chip. To write on a register (e.g. to configure the device's range), the master addresses the slave using its identifier, transmits the corresponding register's index to it followed by its content, then closes the communication. For example, for the MPU6050, the unique identifier is provided by the manufacturer as 0x68 (the prefix 0x tells the reader that the following digits must be interpreted as hexadecimal ones, such that the address expressed in base-10 is $6 \times 16 + 8 = 104$), while the index of the register containing the gyroscope configuration is 0x1B. According to the specifications, at the gyroscope configuration register's bits 3 and 4 contains a two binary digit code representing the range of the device. To each range corresponds a different calibration constant. The code can be any of the four possible codes shown in the following table.

the data sheet of the
MPU6050 chip from
[the Internet](#)

full scale range (°/s)	hexadecimal code	binary code	sensitivity (s/°)
±250	0x00	00	131
±500	0x01	01	65.5
±1 000	0x02	10	32.8
±2 000	0x03	11	16.4

In order to configure the gyroscope to measure angular velocities up to 500 °/s, then, the master must transmit over the serial data line the following bytes: 0x1B, 0x08 (the binary representation of the last number is in fact 0000 1000 and remember the first bit index is 0 and corresponds to the rightmost one).

To read from one or more registers, the master transmits the index of the first register to read followed by the number of bytes to obtain. The slave whose address has been specified during transmission, then, transmits as many bytes as required containing the content of the corresponding adjacent registers. For example, gyroscope measurements are stored in registers 67 to 72 (0x43 to 0x48). The angular velocity on each axis is represented by a 16-bit long number: the first two registers contain the MSB (most significant byte) and the LSB (least significant byte) of the *x*-component, the next two those of the *y*-component and the last two the bytes corresponding to the *z*-component. Then, if the master just need the *y*-component of the angular velocity, it transmits the first register 0x45 followed by 0x02. The slave responds transmitting two bytes (*B*₁ and *B*₀) starting from register 0x45. The reading is then

To obtain data from the slaves, the master asks them to provide *N* adjacent bytes starting at a given location address in the register map.

$$I_y = B_1 \times 256 + B_0 \quad (14.11)$$

The corresponding value of ω_y can be obtained as

$$\omega_y = \frac{I_y}{65.5}. \quad (14.12)$$

In two's complement representation, binary numbers starting with a 1 are negative.

It is worth noting that, according to the device's data sheet, I_y is a signed integer represented as a two's complement number such that it represents integer numbers from $-32\,768$ ($0x8000$) to $+32\,767$ ($0x7FFF$). The calibration constant is then given by the inverse of the sensitivity. This way, when I_y is contains $0x8000$, $I_y = -32\,768$ and $\omega_y = -500$.

Multiplexing

The fact that all the individual sensors using the same I₂C chip shares the same address would make it impossible to use more than one device on the same network. On the other hand it can be needed to have, e.g., two or more IMU's on the same I₂C line. In these cases the master can address each of them using different techniques. Some manufacturer provides the possibility to change the address of teh slave, modifying the content of a register. Manifestly, the change must be done attaching only one slave to the master. Only after the address has been changed, different devices can be addressed individually.

Another possibility consists in using an extra line, often called AD0. These devices have two addresses. When AD0 is set to LOW, the device responds if it is addressed using the default address (0x68 for the MPU6050), while if AD0 is HIGH it responds if addressed with an alternative address (0x69 for the MPU6050). In an experiment in which there many IMU's of the same type one can connect the AD0 lead of each device to a different Arduino digital pin kept HIGH. When the master has to talk to one of it, it is enough to put the corresponding digital pin to LOW before starting the dialogue. Only the selected device will respond.

4. The Arduino Wire library

Libraries usually provide classes from which the user can instantiate objects representing the device. One of the various libraries available for the MPU6050 IMU is (not surprisingly) MPU6050. After its inclusion, the user can instantiate an object of class MPU6050, configure it in the setup() and get data in the loop() as easy as follows.

```
#include <MPU6050.h>

MPU6050 mpu;

void setup() {
    Serial.begin(9600);
    Wire.begin();
```

```

mpu.initialize();
mpu.setFullScaleGyroRange(MPU6050_GYRO_FS_500);
Serial.println("IMU ready");
}

void loop() {
    int16_t ax, ay, az;
    int16_t gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    Serial.print("a = ");
    Serial.print(sqrt(ax*ax + ay*ay + az*az));
    Serial.print(" w = ");
    Serial.print(sqrt(gx*gx + gy*gy + gz*gz));
}

```

In this sketch, `mpu` is the name assigned to the object representing the IMU. A good library defines Before using it, it must be initialised using its `initialize()` method. The class to represent the scale range of the gyroscope can be set using `setFullScaleGyroRange()`. device intended to be to which a parameter is passed in the form of an integer represented by the managed. The class constant, defined in `MPU6050.h`, `MPU6050_GYRO_FS_500`. Six integer variables provides methods to ables, each 16-bits long (`int16_t`) are defined in the `loop()` and set using configure it and access the `getMotion6()` method of `MPU6050`. Those variables contains the comits data. ponents of the acceleration and angular velocities vectors in the form of two's complement integer numbers. In order to obtain their values in SI units they must be multiplied by the corresponding calibration constants.

In the Arduino language, derived from the C-language, a method cannot return more than one value. In order to set two or more variables using a method it is necessary to give the method the address of the variables whose content has to be modified. If we passed the variable names as parameters in the method, the latter gets their values. It may, of course, change it during its execution, however that change does not reflect into a change in the content of the original variable. Remember that a variable is a data container: passing it as a parameter does not imply that the content of that container changes. If a function working method must be able to modify the content of a variable must know its address in the memory. In that case it can modify its content overwriting the content of the corresponding memory locations. For this reason, the `getMotion6()` method, aiming at setting the values of its parameters, requires the addresses of the variables to be altered, the latter being returned by the `&` operator. In the above example we just send over the serial line the modulus of both acceleration and angular velocity registered by the IMU.

The `Wire.h` inclusion is needed to add the functionalities of the Arduino library, aiming at providing access to the I²C communication protocol. With respect to that of the parameter.

The Wire library
allows native
communication with
any I₂C device.

Registers are set
addressing the device,
then writing the
register index and its
value on the SDA line.

munication is initialised thanks to the method

```
Wire.begin();
```

To configure the gyroscope configuration register, libraries implement the native communication protocol as follows.

```
Wire.beginTransmission(0x68);
Wire.write(0x1B);
Wire.write(0x08);
Wire.endTransmission();
```

The first line sets the address of the device to which data are addressed (0x68). The next two lines write as many bytes on the serial data line. Only the addressed slave collect and interpret them accordingly, setting the gyroscope configuration register to 0x08.

getMotion6() must read six bytes starting from register 0x43 to obtain gyroscope data. Acceleration ~~data are stored in~~ ^{Don't} six registers from 0x3B to 0x40. Setting the angular velocity ~~components can be~~ ^{the slave and asking} accomplished as follows, using native methods.

```
Wire.beginTransmission(0x68);
Wire.write(0x43);
Wire.endTransmission();
Wire.requestFrom(0x68, 6);
int Ix = Wire.read() * 256 + Wire.read();
int Iy = Wire.read() * 256 + Wire.read();
int Iz = Wire.read() * 256 + Wire.read();
```

The first group of three statements addresses register 0x43 of the device identified by 0x68. The requestFrom() method, then, asks the device whose identifier is 0x68 to transmit six bytes. Those are read sequentially and returned by each Wire.read() statement.

A ~~common~~ alternative to compute the value of Ix from the two bytes is

```
int Ix = (Wire.read() << 0x08) | Wire.read();
```

exploiting the **shift operator** `<<`. The operator shift the left operand towards ~~specified as their right~~ ^{many places as} left by the number of bits specified as its right operand. There is also its right shift operator `>>`. The **bitwise or** operator `|` performs the logical OR between all the bits of the operand on its left and those of the operand at its right. Ix being a 16-bits integer, if the I₂C slave provides 0x31 and 0x03 as a result of a measurement (12547), after the first reading Ix reads

(note how simple is to pass from hexadecimal to binary notation: it is enough to write down the bits separately for each hexadecimal digit). The application of the shift operator modify the content of `Ix` into

$$0011\ 0001\ 0000\ 0000 \quad (14.14)$$

shifting the digits on the left by eight places. The second `Wire.read()` returns `0x03`, used in the logical OR operation such that, remembering that `0 or 1 = 1`,

$$\begin{array}{r} 0011\ 0001\ 0000\ 0000 \text{ or} \\ 0000\ 0011 = \\ \hline 0011\ 0001\ 0000\ 0011 \end{array} \quad (14.15)$$

corresponding to `0x3103` that in turn corresponds to decimal 12547. The **bitwise AND** operator exists, too, represented as `&`.

Knowing the low level protocol described above allows you to write the code to communicate with every I²C compatible device, provided you have access to its documentation, usually distributed on Internet.

5. Using an SD card

To perform an experiment like the one described in this chapter, manifestly, Arduino cannot be attached to a computer with a USB cable. As a consequence, sending data over the serial line is useless. Data can instead be collected attaching a mass storage device to Arduino, such as an SD card. This is done using a tiny SD card reader that uses the SPI (Serial Peripheral Interface) communication protocol to exchange data with the processor. The SPI protocol requires much less power with respect to the I²C, operating on shorter distances. While the latter can be a serious limitation for sensors, it is not for a mass storage device.

The SPI protocol uses four lines to make it possible the communication between a master (Arduino) and a slave (the SD card reader).

1. **MISO** (Master In Slave Out), also called **DO** (Data Output) brings data from slaves to the master. It must be connected to pin 12 on Arduino UNO (the pinout may be different on other Arduino platforms).
2. **MOSI** (Master Out Slave In), also called **DI** (Data Input) brings data from the master to the slaves. The corresponding Arduino pin is 11.
3. **SCK** (Serial Clock) distributes a common clock signal to the whole network of masters and slaves, via the Arduino pin 13.
4. **CS** (Chip Select) or **SS** (Slave Select), using pin 10 on Arduino UNO, by which the master can disable communication with slaves.

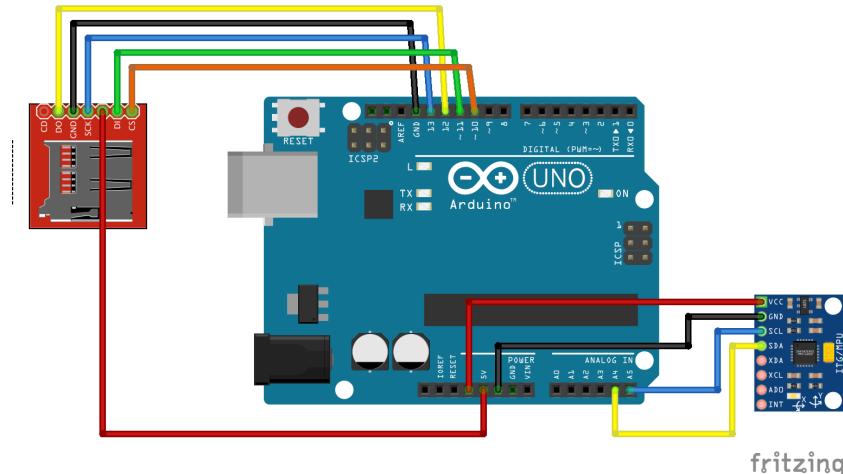


Figure 14.4: Wiring for an experiment adopting an MPU6050-based IMU (the blue device on the right) together with an SPI SD card reader (the red one on the left).

As a result, any SPI compatible device has at least six leads: two for feeding (GND and 5V) and four to participate in the communication.

The ~~CS~~ ~~SPI~~ ~~is used to select~~ the target of an SPI command on a network where many devices ~~are present~~ If, for example, two or more SPI compatible device share the ~~same~~ ~~SPI~~ ~~bus~~ all data travelling over the MOSI and MISO lines are seen by ~~any~~ ~~of~~ ~~them~~ ~~in~~ ~~order~~ to make all devices ~~deaf~~ to these data, but one, it is ~~enough~~ ~~to keep~~ ~~all~~ the CS lines in the HIGH state. Only the device whose CS lead is ~~kept in~~ ~~the~~ LOW state intercepts and interprets the data. By default, pin 10 on an Arduino UNO is automatically set to LOW before any transaction and to HIGH soon after. Another digital pin must be used and properly configured if a second SPI device is present in an experiment.

Fig. 14.4 shows how to connect both an IMU based on an I²C-compatible device and an SD card reader using the SPI protocol. Wiring is complex: following conventions such as those about the cables' colours helps in avoiding mistakes. The MPU6050 can be supplied with a voltage between 3 V and 5 V and we can use the 3.3V Arduino pin for it, while the 5V pin is reserved for the card reader. The power can be supplied by a common 9V battery attached to the external power black connector on the bottom left part of the board in the figure. Care must be taken to properly align the surface of the IMU with respect to the cylinder axis.

6. Using SD cards to store data

The SPI protocol is relatively easy to use on Arduino that natively support reading and writing from and to peripherals. On the other hand, the need to use it as such is actually very limited to few devices. Here, we just illustrate

how to write data on a file on the SD card, being it extremely useful in physics data acquisition. Few details about the protocol are given in the next section.

After including the SD.h library, a file on the SD card can be represented by an object of the class File, as in

```
File dataFile;
```

The file must be opened, once the reader has been properly initialised with SD.begin(). To open a file in write mode the variable representing it must be assigned with the result of the **open()** method of SD, usually in the **setup()** function:

```
void setup() {
    SD.begin();
    dataFile = SD.open("data.csv", FILE_WRITE);
}
```

The first parameter is the file name. Mode FILE_WRITE prepare dataFile to accept data to be written on the SD card. If the file does not exist, it creates it, otherwise new data are appended to existing ones. The FILE_READ prepare the file for reading.

Sending data to a file works as sending data to the serial line. To write characters on a file we can use either the **print()** or the **println()** method, analogous to those for Serial, as in

```
dataFile.print("omega_y = ");
dataFile.println(Iy);
```

Sometimes, reading back the file it may happen that the last expected rows of a file are missing. This happens because writing is not done synchronously for efficiency reasons. Writing on an SD card is expensive from the point of view of the time needed. The systems, then, try to keep as many information as possible in a buffer in the volatile memory and empty the buffer only when needed, actually writing its content on disk. To force the buffer to be emptied, the user can use the **flush()** method.

```
dataFile.flush()
```

The need of reading data from files stored on an SD card is less common in data acquisition applications. However, it may be useful, e.g., to read a configuration stored on it before starting a new run.

Once opened for reading, the size of a file in units of bytes can be obtained using the **File.size()** method as in

```
Serial.print("Data file size = ");
Serial.println(dataFile.size());
```

Reading a file consists in obtaining a sequence of bytes from the current position of a pointer to the file and copying them into a large enough memory location represented by a variable in the sketch. The position of the pointer is set to the beginning of the file at opening time and it advances as reading proceeds.

When the file is no longer needed, it can be closed using

```
dataFile.close()
```

causing the memory buffer to be flushed and securing all the data on the mass storage.

Using `print()` and `println()` methods, data are written on files as if they were ~~writing to a screen~~, i.e. in its decimal representation. What is written in the file ~~in other words~~, are the characters representing the content of the variables ~~or the constants~~ passed as arguments. For example,

```
corresponding numbers
float x = 1024;
dataFile.println(x);
```

~~Arduino needs to convert the sequence of digits into the proper representation (int or float for example).~~

results in finding the characters 1024 in a line of the file. However, 1024 is represented in the Arduino memory as a floating point number: while the string 1024 needs five bytes to be represented (one extra byte is the terminating one, whose ASCII code is 0), a floating point number is represented using four bytes using the IEEE-754 standard.

It is not difficult to realise that its hexadecimal representation in the memory is 0x44800000. In fact $1024 = 1 \times 2^{10}$. Its mantissa 1 is omitted in the IEEE-754 representation and the last 24 bits are all zero. The exponent, instead, is represented in the excess-127 notation, i.e. as $127 + 10 = 137$, whose binary representation is 10001001. Taking into account that the first bit represents the sign (0 in this case), we obtain the full representation as above.

~~The above considerations~~ are relevant when reading back data from a file that, unfortunately, is not as simple as writing (fortunately enough, on the contrary, simple ~~it is that~~ data acquisition applications only seldom require to read data from files). The File class has a method `read()` accepting two arguments: a memory location, represented as the address of a character, and the number of adjacent bytes to be read. Suppose, for example, that we want to read back the "omega_y = " characters such that we can put them into a string of, at least, 11 characters:

```
void loop() {
    char string[11];
    dataFile.read(&string, 11);
}
```

will read 11 characters starting from the current pointer in the file and copy them into as many adjacent memory locations starting at the one corresponding to the first character of string `string`, whose address is returned by `&string`. In fact, a string is just an array of characters and the `char` type represents a character variable, i.e. a 1-byte long variable containing an ASCII code. With such a piece of code, element `string[0]` contains `o`, `string[1]` contains `m`, and so on, up to `string[10]` that contains a blank (note that array indices run from 0 to $N - 1$, N being the size of the array).

The next characters in the file will be those needed to represent the content of the variable `Iy` in our example. What we can read, in this case, are the characters representing the number in its decimal notation, not the value of that number. It is not possible to put the value read this way into a floating point variable: in fact, its decimal representation may need from 1 to many characters to be written, while a floating point variable always occupy four bytes in the IEEE-754 standard. To convert a string into the content of a floating point variable, if possible, the method `toFloat()` of a `String` class exists. Its illustration is beyond the scope of this textbook and we refer to the documentation available on the Arduino website. On the other hand, in this course we are interested in writing data in the form of CSV files to be read on a computer.

7. The native SPI protocol

Even if beyond our scopes, it is worth mentioning the native SPI methods, in the special case in which we need at least two devices on the bus. The SPI protocol does not need to address a specific device: the interested one is addressed by putting its CS line `LOW`. Similarly to I²C, instead, each device defines a set of registers that can be either read or written by the master. The communication is always done transferring a sequence of bits encoding both the address of the register and its content. Suppose, then, to have two devices connected to Arduino: the CS line of device *A* is connected to pin 9, while device *B* is connected to pin 10. A command to the device is represented by, e.g., two bytes. Both pins must be configured to be output pins and both are disabled with:

```
void setup() {
    ...
    pinMode(9, OUTPUT);
    pinMode(10, OUTPUT);
    digitalWrite(9, HIGH);
    digitalWrite(10, HIGH);
}
```

(ellipsis represent other statements for the configuration that may be needed). In order to address device *A*, pin 9 must be set LOW, data are transferred on the communication lines, then pin 9 is set back to HIGH:

```
digitalWrite(9, LOW);
recv1 = SPI.transfer(byte1);
recv2 = SPI.transfer(byte2);
digitalWrite(9, HIGH);
```

When `SPI.transfer(byte1)` is executed, the corresponding sequence of bits is written sequentially on the MOSI line. The device whose CS line is LOW interprets it and returns, on the MISO line, a result, if any. The returned value is put in the `recv1` variable.

Summary

It happens frequently that the acceleration of a rigid body rolling down an incline is mistakenly considered equal to that of a point-like particle. That is a mistake, as it can be easily verified making an experiment. Part of the energy gained descending the incline is in fact converted in rotational kinetic energy and is no longer available to increase the velocity of the centre of mass.

Treating an object as point-like is equivalent to neglect any degree of freedom in excess with respect to the three translational ones. The size and shape of the object does not matter: planets and stars can often be treated as point-like, even if they are giant, while a coin has the shape of a disk even if it behave much like a point-like object in many cases.

The angular velocity of a cylinder rolling down an incline linearly increase with time: its acceleration must then be constant.

Internal forces maintaining the shape of an object must be taken into account to predict the dynamics of an object. In many textbooks they are ruled out from equations because they sum up to zero, however that does not mean that they are absent: it only have consequences on the dynamics of the objects' centre of mass. Their effects is modeled throughout the moment of inertia.

Phyphox

Angles can be measured using the PHYPHOX's inclinometer.

With PHYPHOX we can operate the phone remotely using a Wi-Fi connection. In particular, we can start, stop and configure experiments with gyroscopes.

Many public networks may be configured such that, for security reasons, they do not allow a browser to connect with PHYPHOX. In these cases you may want to use your device as an hotspot, connecting your computer to the network managed by the phone itself.

Arduino An IMU (Inertial Measurement Unit) is a device embedding both a gyroscope and an accelerometer to trace the state of a moving object.

I²C is a master/slave, bidirectional, synchronised communication protocol supported by Arduino. Data travels over the SDA line, synchronised by pulses on the SCL line.

Most sensors can be operated using a library provided by its manufacturer or a third-party. Libraries are useful, but knowing the native protocols boosts your reputation.

I²C based slaves can be configured such that, for example, they provide data as precise as possible or as fast as possible. Many parameters, listed in the documentation, can be chosen by the user.

Setting properly the full scale range to which a device is sensitive improves its precision.

Configuration and data are stored in an internal memory of the slave, whose locations are called

registers. Each register is 8-bits long and its meaning depends on its location.

Every I²C device have a unique 16-bits address. The address is set at the factory and the same address is shared by all the devices based on the same chip.

To obtain data from the slaves, the master asks them to provide N adjacent bytes starting at a given location address in the register map.

Libraries simplify the management of external devices. On the other hand they are not as general. A good library defines a class to represent the device intended to be managed. The class provides methods to configure it and access its data.

In the Arduino programming language, arguments are passed to functions by value, i.e. their content is assigned to formal parameters of the function working as ordinary variables. A change in the parameter, however, does not reflect to a change in the content of the variable passed as an argument, because its location in the memory is different with respect to that of the parameter.

The Wire library allows native communication with any I²C device. Registers are set addressing the device, then writing the register index and its value on the SDA line. Data from registers are requested addressing the slave and asking for N adjacent bytes starting from a given register.

Shift operators << and >> shifts the bits of their left operand by as many places as specified as their right operand.

| and & are, respectively, the bitwise OR- and AND-operators. They apply the logical operation bit by bit to operands.

In many cases collecting data from Arduino using a computer connected via a USB cable is not possible. In these cases data can be logged on an SD card.

The SPI communication bus is composed of four lines: MISO (Master In Slave Out), MOSI (Master Out Slave In), SCK (Serial Clock) and CS (Chip Select).

Using a file to log data persistently is easy with an SD card reader and SPI. Just open the file in write mode and write on it using methods similar to those used to send data over the serial line. Characters can be written on a file using the same `print()` and `println()` methods defined in `Serial`. Flushing the memory buffer to actually dump data on files is achieved by the `flush()` method.

Writing variables containing numbers to files, the decimal representation of the corresponding numbers are recorded. Reading them back with Arduino needs to convert the sequence of digits into the proper representation (`int` or `float`, for example).

To write or read SPI registers one or more bytes have to be transferred over the MOSI line. The bytes encode the addressed register, the command to be executed and its parameters, if any. The slave executes the command and returns a byte on the MISO line as a result.

Bibliography

- [1] The Merriam–Webster Dictionary, online edition 2020. www.merriam-webster.com
- [2] InvenSense Inc., “MPU-6000 and MPU-6050 Register Map and Descriptions”, 2020. RM-MPU-6000A-00 rel. 03/09/2012. www.invensense.com.

Chapter 15

Wave mechanics

The physics of waves is of capital importance. It was used in the past to demonstrate that light propagates as a wave, while nowadays it is exploited to catch gravitational waves: one of the most elusive phenomena ever. Waves play a central role in electromagnetism, as well as in modern quantum physics, where radiation and matter both behave like waves in certain conditions. To understand quantum mechanics it is then mandatory to master the physics of waves. In this chapter we propose few experiments to be done using sound waves.

1. Making waves

Acoustic waves can be produced in a variety of ways. In Python, in particular, this may be accomplished importing the `simpleaudio` module.

Sound is produced by computers driving loudspeakers with currents whose amplitude is encoded, as a function of time, as a sequence of integers, in turn represented by sequences of n bits. The sequence has several important characteristics. The **sampling rate** f_S is the number of samples in one second. Audio CD's, for example, are sampled at 44.1 kHz. In other words, each second of sound is represented as a list of f_S integers. Another important characteristic is the **resolution**, often called **bit depth**. It represents the number n of bits used to represent each sample. Common values are $n = 16$, $n = 20$ and $n = 24$. Finally, a given sound can be reproduced on different **channels**. A channel is a physical device through which data are sent from the source to the speaker. Each channel can transport different information. Standard audio signals can have one (mono) or two (stereo) independent channels. Professional systems may have more channels.

For the purpose of generating plain audio signals to play with waves, we need stereo sinusoidal signals sampled at a reasonable rate as, e.g., $f_S = 44.1$ kHz, with a relatively small depth $n = 16$.

To represent a stereo signal whose duration is t s, we need $N = t \times f_S$, n -bits integers per channel. In Python this requires a bidimensional list (one per channel), each composed of N elements, each containing the samples of the

Sounds can be produced and manipulated using the `simpleaudio` module, in Python.

Aud
cha
san
num
sec
dep
(th
use
and
cha
ste

signal, represented as an integer proportional to

$$y(t) = \sin(2\pi ft), \quad (15.1)$$

f being the frequency of the sound (audible signals are approximately in an interval $20 \leq f \leq 20000$ Hz). For a depth $n = 16$, each sample must be a 16 bit integer, whose values go from $-2^{15} = -32768$ to $2^{15} - 1 = 32767$.

In the previous chapters any sequence of data is called a list, even when they are not. In fact, few of them are **arrays**. There are subtle, but important differences between lists and arrays, in Python. While lists can be heterogeneous, i.e. can contain objects of different types, arrays cannot. From the practical point of view, the main difference between the two sequences is the result of the application of the * operator. Applied between a list and an integer constant N , the operator returns an extended list, whose elements are the same of the list operand repeated N times. The result of

```
t = [1, 2, 3]
print(t)
```

is

```
[1, 2, 3]
```

The execution of the following code

```
to a list and an integer
t2 = t * 3
print(t2)
```

produces

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Applied to arrays, the * operator returns an array whose elements are each multiplied by the constant, e.g.

```
array and a constant
a = array([1, 2, 3])
print(a)
b = a * 3
print(b)
```

produces the following output:

```
[1 2 3]
[3 6 9]
```

Note how the creation of an array implies the usage of parenthesis, and that the output is slightly different in the two cases. It is worth noting that multiplying

an array by a non integer number or dividing it by a constant are perfectly defined operations, while for a list both return a runtime error. As a matter of fact, numpy's `arange()` and `zeros()` return an array.

The samples for the left channel at $f = 440$ Hz of 1 s duration can then be obtained as follows as an array.

```
import numpy as np

f = 440
fs = 44100
seconds = 1

note = np.zeros((int(seconds * fs), 2))
t = np.arange(0, seconds, seconds/len(note))
note[:,0] = ((2**15-1)*np.sin(2 * np.pi * f * t))
```

A multidimensional list (or an array) in Python can be created as a list of lists (an array of arrays). For example, for a bidimensional list:

```
aList = [[0, 1, 2, 3], [4, 5, 6, 7]]
```

Element `aList[1]` represents the second element of the list, i.e. `[4, 5, 6, 7]`. Its third element (6) can then be obtained as `aList[1][2]`. An array of N elements initialised to 0 can be obtained from numpy's `numpy.zero(N, m)`, that returns an array of N arrays, each composed by m null elements.

The `t` array contains the times at which we have to compute the samples. The length of `note` is N (each element's length of `note` is 2, but `note` is an array of N arrays of length 2 each). If, then, we need to play t seconds, we need a sample every $\frac{t}{N}$ seconds. The values of the left channel of the audio sample is assigned computing

$$y(t, 0) = (2^{15} - 1) \sin(2\pi ft) \quad (15.2)$$

such that the highest value attained by it corresponds to the maximum representable value of a 16-bits number. Note that `np.sin()` returns an array whose size is the same of its argument. It is assigned to the first element of each sublist of which `note[]` is made. Similarly, one can assign the values to the right channel using

```
note[:,1] = note[:, 0]
```

For a mono signal, the right (or left) channel must be set to zero, i.e.

```
note[:,1] = [0] * len(t)
```

Note that, in this case, [0] is a list with a single element. Multiplied by the constant `len(t)`, it returns a list of length `len(t)` full of 0 (it is equivalent to `np.zeros(len(t))`). When the list is assigned to an array it is automatically converted (the operation, of course, can only be possible if all the elements of the list belong to the same type, as in this case).

The `simpleaudio` package provides tools to interact with loudspeakers and other peripherals intended to play audio signals.

```
import simpleaudio as sa
po = sa.play_buffer(note.astype(np.int16), 2,
                     2, fs)
po.wait_done()
```

Audio data, in the form of a bidimensional array of n -bits integer, can be played using `play_buffer()`. After invoking `play_buffer()`, the program continues to immediately. To suspend its execution until the audio is played, the play object must contain numbers represented as 16-bit integers. The default type for its elements is `float64`, i.e. each sample is represented as a floating point number in double precision (using 64 bits instead of 32, of which 11 for the exponent and 52 for the mantissa). In order to convert them in the expected type, the `astype()` method is adopted, to which the wanted type is passed as `np.int16` representing a 16-bit integer as defined in numpy.

2. Command line options

It is convenient having the possibility to change the characteristics of the sound produced by the script described above without changing the code. To do that we can profit of the possibility to pass the values of parameters from the commands line. Launching the script from the command line consists in typing its name in a shell (the environment a user find in a terminal), such as

```
./play.py
```

The command will play a sinusoidal tone with $f = 440$ Hz for 1 s. To play a tone with $f = 8200$ Hz for 3 s, it would be nice to write

```
./play.py -f 8200 -d 3
```

or, more explicitly

```
./play.py --frequency 8200 --duration 3
```

Here f and d are called *options*, while frequency and duration are *arguments*. Long options require two dashes to be specified. Different options may be combined into a single option if they do not require arguments such that ls -l -a is equivalent to ls -la (ls is a UNIX command used to list all the files in the current directory). The getopt module is intended for this. The list of possible options is given in getopt.getopt(). The latter accepts a list of strings consisting in the list of options and arguments given with the command, if any (in the example this would be --frequency 8200 --duration 3). The list is followed by a string made of characters representing valid options (f and d), each followed by a semicolon if it requires an argument (e.g., -f or -d). Optionally, a list of long options can be given in the form of a list of strings terminated by an equal sign if they require an argument. The first parameter of getopt.getopt() can be extracted from sys.argv provided by the sys module, containing the name of the script as its first element (sys.argv[0]) and the list of arguments as the following ones, such as the wanted list is represented by sys.argv[1:], as in the script below.

```
import sys
import getopt

f = 440
seconds = 3

opts, args = getopt.getopt(sys.argv[1:], 'hf:d:',
                           ['help', 'frequency=',
                            'duration='])

for opt, arg in opts:
    if opt == '-h':
        help()
        exit(0)
    elif opt in ("-f", "--frequency"):
        f = int(arg)
    elif opt in ("-d", "--duration"):
        seconds = float(arg)
```

In this example we added the option h (whose corresponding long option is `help`) intended to provide a help message to the user. The message is printed by the function `help()`, not shown. `getopt.getopt()` returns a list of pairs representing the options and their arguments, and a list of arguments passed to the command, if any (args, that in this case is empty). To understand the difference between options and arguments, consider the tail UNIX command, intended to display the tail of a file, i.e. its last rows.

tail audioscope.csv prints the last ten rows of the file audioscope.csv

The first object returned by getopt we are appending data to it continuously), we can monitor its content using tail -f audioscope.csv: it displays the last ten rows of the file when it pairs. Each pair is read the first time, but it does not exit. It keeps reading the file such that made of an option and new line appended to it will be displayed immediately. In this case -f is its corresponding option. Options may have arguments, too (not to be confused with value. The second argument of the command). For example, tail -n 20 audioscope. object is a list csv display the last 20 lines of the file. In this case 20 is an argument for the arguments given in option -n.

command line.

If written in Python, tail would start as

```
opts, args = getopt.getopt(sys.argv[1:], 'n:f')
```

When called without arguments nor options, opts and args are both equal to [], i.e. they are empty lists. If the command is invoked as tail audioscope .csv, opts = [], while args = ['audioscope.csv']. In the case of tail -f audioscope.csv, opts is a list of pairs [('-f', '')]: a list with one element consisting in a tuple of two strings, the second one being empty. Finally, upon tail -n 20 audioscope.csv, opts contains [('-n', '20')].

Iterating over opts returns each pair sequentially, decomposed into opt and arg . For each iterator we check if opt is contained in a tuple and, if yes, we reassign the values of variables ~~accordingly~~. Note how the construct

```
opt in ("--f", "--frequency")  
      in its right operand.
```

returns true if the operand on the left (opt) is contained in the tuple on the right of the in operator.

3. Properties of a wave

Waves may have a variety of shapes, however the Fourier's Theorem states that a periodic function $f(x)$ may be expressed as a series of sine or cosine terms, at least if $f(x)$ is integrable in a finite interval of amplitude L . Terms in the Fourier series have specific amplitudes known as Fourier coefficients. In formulas

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} a_n \cos\left(\frac{2\pi nx}{L}\right) + b_n \sin\left(\frac{2\pi nx}{L}\right) \quad (15.3)$$

The Fourier coefficients a_i and b_i are computed as follows.

$$\begin{aligned} a_n &= \frac{2}{L} \int_L f(x) \cos\left(\frac{2\pi nx}{L}\right) \\ b_n &= \frac{2}{L} \int_L f(x) \sin\left(\frac{2\pi nx}{L}\right) \end{aligned} \quad (15.4)$$

The Fourier's Theorem guarantees that the properties of any periodic function are the same of that of sinusoidal, or harmonic, waves. For simplicity, then, we can limit ourselves to study only the latter, since they will share their behaviour with any other wave, being the latter a sum of harmonic terms. A harmonic wave can always be written as

$$f(x, t) = A \cos\left(2\pi \left(\frac{x}{\lambda} \pm \frac{t}{T}\right) + \phi\right), \quad (15.5)$$

where λ is called its wavelength, T the period and ϕ the phase. A is called the amplitude and represents the maximum absolute value of $f(x, t)$ and has the same dimensions. For example, waves on the surface of the sea are characterised by their height with respect to the average sea level and A is measured in m . For sound waves A may represent the local air pressure.

The inverse of the period is the frequency $f = \frac{1}{T}$. The reason why we write the wave that way is that sine and cosine both take an angle as an argument and an angle is always a multiple of 2π . Sine and cosine differ only by the relative phase. The multiple must be dimensionless: since it may depend on x and on t (respectively, a length and a time), the latter must be divided by a parameter having the same nature.

The wavelength represents the distance between two successive points having the same $f(x)$ and the same $f'(x)$. The period is the interval after which the wave reaches the same $f(t)$ and $f'(t)$. These two quantities are linked by the relationship $\lambda = cT = \frac{c}{f}$, where c is called the speed of the wave and represents, in fact, the speed at which the wave propagates in space. The latter depends on the medium within which the wave propagates.

Almost any periodic function can be written as a series of trigonometric functions with appropriate frequency and amplitude, according to the Fourier's Theorem. The latter gives the recipes to compute the amplitudes for each frequency.

It is customary to define the angular frequency defined as $\omega = 2\pi f$ and the wave number $k = \frac{2\pi}{\lambda}$. They are useful to simplify the wave function that becomes

$$f(x, t) = A \cos(kx \pm \omega t + \phi). \quad (15.6)$$

The relationships between these quantities are easy to remember using dimensional arguments. For example, since c is a speed it must be a length divided by a time, i.e. $c = \frac{\lambda}{T}$, while, since f is the inverse of a time, $c = \lambda f$.

The properties of a wave can be explored using PHYPHOX that shows $f(0, t)$, $x = 0$ corresponding to the smartphone's microphone position.

To understand the characteristics of a wave, use the PHYPHOX audioscope that show a graphical display of $f(t)$ (in x corresponding to the microphone position) in arbitrary units. A microphone works as a parallel plate capacitor: the system measure the current driven by it when the distance between the two conductors changes. The capacitance of a system of two conductors is in fact defined as

$$C = \frac{Q}{\Delta V} = \epsilon \frac{S}{h} \quad (15.7)$$

where Q is the electric charge accumulated on one of them and ΔV the voltage between the plates. S is the surface of the plates and h their distance. ϵ is a constant that depends on the insulating material between the plates. Taking the derivative of both members:

$$\frac{dQ}{dt} = -\epsilon \frac{S}{h^2} \frac{dh}{dt} \Delta V. \quad (15.8)$$

$\frac{dQ}{dt}$ is, by definition, the current extracted or injected in the capacitor that then depends on the change of h . You can imagine a microphone as a parallel plate capacitor in which the two plates are connected by an insulating string. The pressure exerted on one plate by air compresses or lengthen the spring and h changes accordingly. For microphones it is not needed to know the local air pressure in the proper units, then they usually return a number proportional to it.

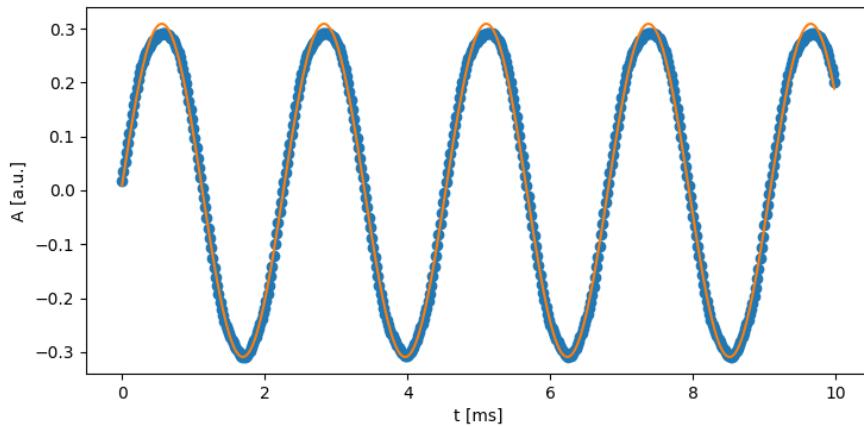
To properly calibrate the audioscope we need a reference sound whose intensity is known. The intensity I of a source is proportional to A^2 and measures the power emitted from it (i.e. the energy per unit time). Sound levels are usually measured in decibel (dB). Manifestly, the power intercepted by a microphone depends on its area and the intensity measured is then given in units of Wm^{-2} . Given $I_0 = 10^{-12} Wm^{-2}$, a sound intensity I in dB is defined as

$$I(\text{dB}) = 10 \log_{10} \left(\frac{I}{I_0} \right) \text{dB}. \quad (15.9)$$

I represents the energy transported by a wave that traverse a unit surface in a unit time. I_0 , then, is the intensity of a sound wave that, in 1 s, carries an

energy of 10^{-12} J traversing a surface of 1 m^2 . In sound waves the energy is the one carried by the air particles.

With the audioscope we record the sound produced by the script described in previous sections, bringing it to the microphone of the smartphone using earphones or loudspeakers. As usual PHYPHOX exports data in columns. In the case of the audioscope there are two columns in the file: the time t and the amplitude $y(t)$. A plot of $y(t)$ as a function of t when $f = 440$ Hz yields



The orange line is the result of a fit with eq. (15.6) for $x = 0$. The waveform is a bit distorted, due to the limitations of the hardware, nevertheless the fit finds a value of $f = 440.0 \pm 0.1$ Hz. The initial parameters for the fit were estimated taking the maximum of $f(t)$ as A and the average distance between successive peaks as T .

Despite the fits appears to be good, model is wrong (it does not take into account distortions introduced by the hardware of both the

source and the receiver). An objective quality of the fit cannot be evaluated and uncertainties do not have a precise statistical meaning.

This is a clear example about the importance of having a good model for data. In fact, we assumed that the correct model of our data was eq. (15.6), but in fact it is not. The waveform is a bit distorted and the χ^2 of the fit cannot be too close to 1. The residuals used to estimate the uncertainties measure the discrepancies between the model and the data and has little to do with the uncertainty on the period, which is a measure of how different the distances between successive peaks are.

In this case, taking the average and the standard deviation of the four differences that can be computed from the five peaks, we find $T = 2.270\,919\,352\,98 \pm 0.000\,000\,000\,02$ ms corresponding to $f = 440.350\,291\,916 \pm 0.000\,000\,004$ Hz, apparently orders of magnitudes better than the fit. The standard deviation of the samples is in fact 8 nHz and the uncertainty on the average is $\frac{\sigma}{\sqrt{N}}$, with $N = 4$. In fact, these numbers seems unreasonably good: a resolution of few nHz seems out of reach for a smartphone.

In fact, we completely neglected the resolution of the instrument. The latter can be estimated taking the duration of the signal (10 ms) divided by the number of samples (480) such that $\Delta_t = \frac{10}{480} \simeq 0.020$ ms. The uncertainty on the time of an event is then

$$\sigma_t = \frac{\Delta_t}{\sqrt{12}} \simeq 0.006 \text{ ms}. \quad (15.10)$$

The total uncertainty is the sum in quadrature between the two, but the square root of the sample variance is negligible, hence $\sigma_t = 0.006$ ms and

$$T = 2.271 \pm 0.006 \text{ ms} \quad (15.11)$$

is the best estimate, leading to

$$f = \frac{1}{T} = 440 \pm 1 \text{ Hz} \quad (15.12)$$

for the frequency. Despite the modelling of the signal was not perfect, the final results are comparable.

4. The Student's t-distribution

In the previous section the standard deviation s of the samples was estimated from four measurements of the period. s is itself the result of a measurement and is affected by an uncertainty that decreases with the number of samples. To assess the significance of a measurement, the ratio are distributed over an

interval with a PDF known as the $S = \frac{x - \mu}{\sigma}$, (15.13)

where μ is the expected value of x and σ the uncertainty, is computed and the corresponding confidence interval is obtained evaluating the integral of a gaussian. When σ is unknown, the significance can only be assessed using an estimation of it:

$$S = \frac{x - \mu}{s}. \quad (15.14)$$

When the number N of measurements is large, $s \simeq \sigma$ and there are no significant differences. However, if N is small the difference above does not follow a gaussian distribution, but the Student's t-distribution. Such a curious name derives from the fact that it was derived and published by William Gosset (1876–1937) using the pseudonym “a student” (According to the legend, the use of the pseudonym became necessary because of the policies of the company where Gossett was employed, which prevented its employees from making their name public).

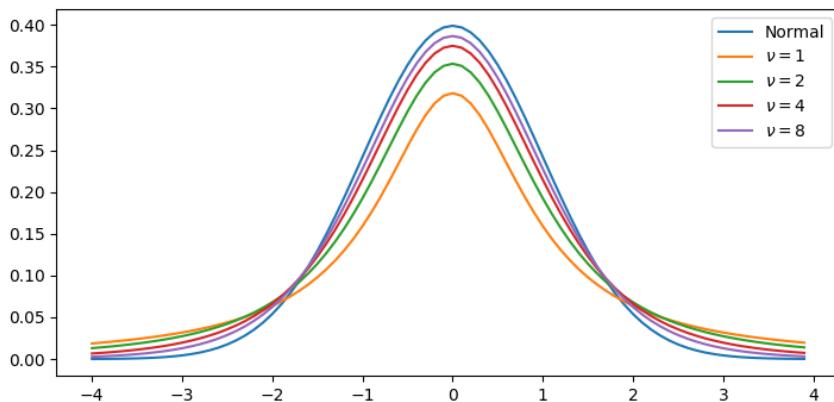
The probability distribution function of the Student's t-distribution is not straightforward (and this is its derivation) and is

complicated, but it

resembles closely a gaussian. Its width PDF_t(x) = $\frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$, where $\nu = N - 1$ is the number of degrees of freedom and $\Gamma(x)$ is the Euler's Gamma function. Its shape is shown below for various ν , compared to the one of the normal distribution.

as predicted by the

Law of large numbers.



The Student's t-distribution is wider than the normal one. Its maximum is lower, while it is higher on the tails. As predicted by the Law of large numbers, the distribution approaches the normal one as ν increases. When the average of a set of measurements differs by one standard deviation from the expected value, we assume that the probability that $|S| \leq 1$ is about 68 %. However, when the standard deviation is estimated from few measurements, the probability must be estimated taking into account that S is distributed as a Student's t and the Student's t.

$$P(|S| \leq 1) = \int_{-1}^1 \text{PDF}_t(x) dx = \int_{-\infty}^1 \text{PDF}_t(x) dx - \int_{-\infty}^{-1} \text{PDF}_t(x) dx \simeq 61 \% \quad (15.16)$$

for $\nu = 3$. Such a confidence interval can be obtained in Python as simply as

```
from scipy.stats import t
cI = t.cdf(1, 3) - t.cdf(-1, 3)
print('Confidence interval: {}'.format(cI))
```

The first parameter of the `t()` function is x , while the second is ν .

5. Interference

Interference ~~co~~^{inter} happens when two or more waves sum in the same place at the same time. Consider two equal in-phase wave sources at position $x = 0$ and $x = L$. two or more waves point $0 \leq x \leq L$, the two waves interfere such that the resulting wave is point. The resulting

wave can have an amplitude ranging

from the difference of their amplitudes to the sum of them.

$$y(x) = A \cos(kx) + A \cos(k(x - L)) \quad (15.17)$$

$$y(x) = 2A \cos\left(\frac{kL}{2}\right) \cos\left(k\left(x - \frac{L}{2}\right)\right) \quad (15.18)$$

(the sum of the cosines of two angles is transformed into the product of the cosines of their half-sum and half-difference). The resulting wave is null when

$$k\left(x - \frac{L}{2}\right) = (2n + 1)\frac{\pi}{2}. \quad (15.19)$$

The position ~~Remembering~~ that $k = \frac{2\pi}{\lambda}$, the condition for null amplitude is the amplitude of the

sum of two equal waves is minimum can

$$x = \frac{L}{2} + (2n + 1)\frac{\lambda}{4}. \quad (15.20)$$

be easily predicted.

In an experiment we put the two speakers of an earphone $L = 40$ cm apart, one in front of the other, playing a sinusoidal tone with $f = 3430$ Hz and $\lambda = \frac{c}{f} = 0.1$ m = 10 cm. We then move a smartphone along the line connecting them at various positions x_i and record the sound using the PHYPHOX's audioscope. Since the sound intensity diminishes as d^{-2} , where d is the distance from the source, in order for the two waves to sum with more or less the same amplitude, the detector must be put close to the middle point $x \approx 20$. For $15 < x < 30$ cm, we expect to observe a minimum for $x = 17.5, 22.5, 27.5$ cm.

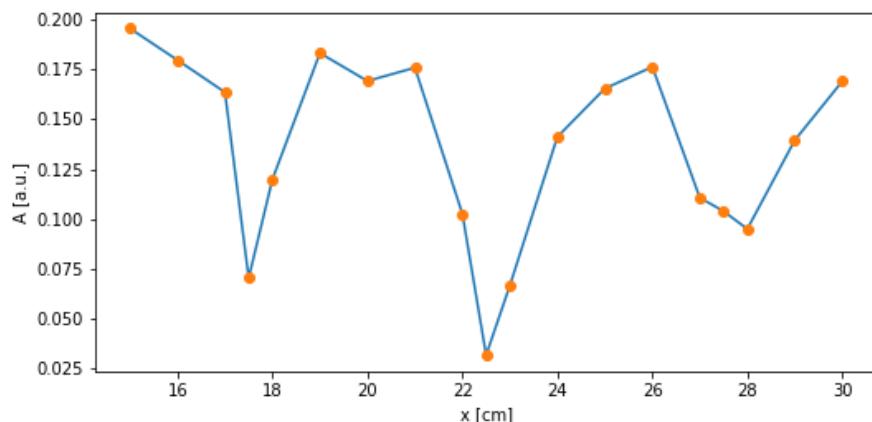
There are different ways to obtain the amplitude. As in Section 3, one may fit the waveform with the amplitude as a free parameter. For this experiment we do not need much precision, then we can evaluate

$$A \approx \frac{\max_i y_i - \min_i y_i}{2}. \quad (15.21)$$

In order to estimate the uncertainty σ_A we can identify several pairs of a maximum and a minimum and evaluate as many A_i , such that A is given by the

average $\langle A_i \rangle$ and σ_A is by their standard deviation (not divided by \sqrt{N} because we are interested in the spread of the values of A_i). The standard deviation of the waveform are expected to be symmetric to maxima with respect to the samples of a waveform. The amplitude can even be estimated just as the average of the maxima recorded in the waveform, while the uncertainty is given by their standard deviation. Given the fact, finding the amplitude is even simpler: as shown in the next section, the standard deviation of the amplitude data is proportional to A : $\sigma_y = \frac{A}{\sqrt{2\omega}}$. Given that A is measured in arbitrary units, taking just the standard deviation of the measurements is already a measure of the amplitude of the signal.

A plot of the sound intensity as a function of x is shown below.



In home made experiments on waves involving sound there are plenty of effects that he elements. Sound propagates in air and media, it is due to echos and intensity diminishes with distance from source. Despite these limitations, many measurements can be fully done.

Minima are, in fact, where they are expected. A fit to the result cannot be too good, given the limitations of the setup. Indeed, apart the limited sensitivity of a smartphone's microphone and the finite dynamic range of the earphones, during the experiment, the smartphone's microphone captures all the environmental noise. Moreover, signal's echos are not negligible, nor the amplitude of the two waves are the same (one of the sources is closer to the microphone). The sound propagates partially in the material of the surface on which the speakers and the smartphone are placed. Nevertheless, an unweighted fit to $|y(x)| + C$ brings relatively good results. Leaving λ and L as free parameters we obtained $\lambda = 10.1 \pm 0.3$ cm and $L = 40.3 \pm 0.2$ cm, perfectly consistent with expectations.

6. Finding the distribution of a random variable

Let x a continuous random variable whose PDF is $f(x)$. By definition, $f(x)$ is such that the probability of x to lie in the interval $[x, x + dx]$ is $f(x)dx$. If $y = f(x)$, then y is a random variable, too, and the probability for y to be in $[y, y + dy]$ is $f(y)dy$, $f(y)$ being the PDF of y . Clearly,

$$f(y)dy = f(x)dx \quad (15.22)$$

because, when $x \in [x, x + dx]$, $y \in [y, y + dy]$, such that

$$f(y) = f(x) \frac{dx}{dy}. \quad (15.23)$$

While $f(x)$ and $f(y)$ are, by definition, positively defined, the derivative $\frac{dx}{dy}$ may not be such. To guarantee that, we write

$$f(y) = f(x) \left| \frac{dx}{dy} \right|, \quad (15.24)$$

authorised by the fact that $dx > 0$ and $dy > 0$, too, in the above computations.

Suppose, then, that we measure $y = A \cos(\omega t)$, with A and ω constants. We sample t uniformly in the interval $0 \leq t \leq t_M$, such that its PDF is uniform and defined by

$$f(t) = \begin{cases} 0 & \text{for } t < 0 \\ c = \frac{1}{t_M} & \text{for } 0 \leq t \leq t_M \\ 0 & \text{for } t > t_M \end{cases} \quad (15.25)$$

To find the PDF of y we need its inverse:

$$t = \frac{1}{\omega} \cos^{-1} \frac{y}{A} \quad (15.26)$$

from which we obtain

$$f(y) = \frac{1}{At_M \omega \sqrt{1 - \frac{y^2}{A^2}}} \quad (15.27)$$

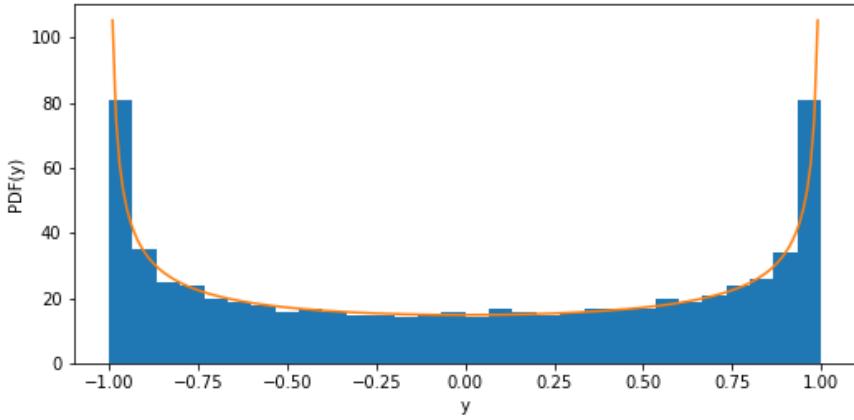
In the interval $[0, t_M]$, the possible values of y are in $[-A, A]$, for $t_M \gg T$, and the appropriate normalisation factor can be found imposing that

$$\int_{-A}^{+A} f(y)dy = \frac{1}{t_M \omega} \sin^{-1} \left(\frac{y}{A} \right) \Big|_{-A}^{+A} = \frac{\pi}{t_M \omega}. \quad (15.28)$$

Finally, we can write

$$f(y) = \frac{1}{A\pi\omega\sqrt{1 - \frac{y^2}{A^2}}} \quad (15.29)$$

A plot of $f(y)$ is shown below, together with an histogram obtained from data generated in an interval $t \in [0, 70]$ in arbitrary units, and $A = \omega = 1$



The histogram is symmetric around $y = 0$, then the mean of the distribution of A is null, while its variance is

$$\sigma_y^2 = \int_{-A}^{+A} \frac{y^2}{A\pi\omega\sqrt{1 - \frac{y^2}{A^2}}} dy = \frac{A^2}{2\omega}. \quad (15.30)$$

7. Beats

Beats happen when two waves of different wavelength interfere in the same point in space. Given two waves with frequency f_1 and f_2 , their sum can be written as

Beats are produced by interference of waves with different frequencies.

$$y(t) = A \cos(2\pi f_1 t) + A \cos(2\pi f_2 t + \phi). \quad (15.31)$$

The sum can be rewritten as a product using prosthaphaeresis formula:

$$y(t) = 2A \cos\left(\frac{f_1 + f_2}{2}t + \frac{\phi}{2}\right) \cos\left(\frac{f_1 - f_2}{2}t - \frac{\phi}{2}\right). \quad (15.32)$$

The latter can be thought as a wave with a frequency equal to the average frequency between the two and an amplitude

$$A(t) = 2A \cos \left(2\pi \frac{f_1 - f_2}{2} t - \frac{\phi}{2} \right). \quad (15.33)$$

The resulting wave has a frequency equal to the average frequency between the interfering ones and an amplitude modulated with a frequency equal to half the difference of the original frequencies.

If $f_1 \simeq f_2 = f$, the resulting wave is a wave with the same frequency of the incident ones with an amplitude modulated with a frequency $\frac{f_1 - f_2}{2}$. For example, if $f_1 = 440$ Hz and $f_2 = 442$ Hz, the corresponding tones are practically indistinguishable and the resulting sound has a frequency of $f = 441$ Hz, that is again barely distinguishable from the incident ones. However, its amplitude varies from 0 to $2A$ with a frequency of 1 s.

We can easily hear beats as an intensity modulated sound whose modulation frequency is exactly $\frac{f_1 - f_2}{2}$. It is enough to modify the script developed in Sect. 15 such that we play a sound of frequency f_1 on the right channel and a sound of frequency f_2 on the left one. The waveform for beats can be appreciated for higher frequencies. For example, for $f_2 = 2$ kHz, $f_1 = 2.27$ kHz, PHYPHOX's audioscope shows the following plot



The waveform of beats is characterised by a constant frequency and a modulated amplitude. In the case of sound waves, if the frequency of the modulation is enough short, it can be heard.

From the plot it is easy to obtain the period of the sound wave of about

$$T = \frac{2}{f_1 + f_2} \simeq 0.5 \text{ ms} \quad (15.34)$$

and the beats' period

$$T = \frac{2}{f_1 - f_2} \simeq 7.5 \text{ ms}, \quad (15.35)$$

as expected.

8. Taking audio data with Arduino

Audio data can be captured with any microphone, available for Arduino, too. Microphones can be seen as analog sensors. They have three leads: two for feeding them and a third one for the signal, represented as a voltage proportional to the sound intensity. Sometimes they have a fourth lead used as a digital pin, whose state is raised when the recorded sound exceeds a given, often adjustable, threshold.

In order to record the waveform of a signal, we need to sample it at highest possible frequency, spending as less time as possible in elaborating data. That can be achieved just repeatedly digitising the analog input connected to the microphone and storing data in a data structure like an array, as in

```
void loop() {
    unsigned long t0 = micros()
    for (int i = 0; i < N; i++) {
        wave[i] = analogRead(A0);
    }
    unsigned long t1 = micros()
    float dt = (float)(t1-t0)/N;
    float t = 0.;
    for (int i = 0; i < N; i++) {
        Serial.print(t);
        Serial.print(", ");
        Serial.println(wave[i]);
        t += dt;
    }
}
```

First of all we obtain the current time t_0 , then we digitise N samples of the sound, storing them in the elements of array `wave`; after reading the time t_1 at the end of data acquisition, assuming that the time needed to get each sample is constant, the time elapsed from one sample to the next is

$$dt = \frac{t_1 - t_0}{N}. \quad (15.36)$$

Note the casting operator (`float`) put in front of $t_1 - t_0$. It is needed because the numerator of the fraction is an integer as the denominator. The result, then, would be an integer, too, unless we cast one of them into a floating point number. Only at the end of the data acquisition cycle we send the samples over the serial line, together with the corresponding times.

N can be as large as possible. On Arduino UNO, where memory is limited, up to

about 500 samples can be stored. Consequently, the range of useful frequencies is limited to around 1000–2000 Hz (it depends both on the memory size and the clock frequency of the board).

Sending characters There are techniques, beyond the scopes of this textbook, exploiting so-called over a serial line **interrupts** that allow better resolution and performance. It is worth noting time consuming and that, at first sight, it might appear better to just send digitised data over the must be avoided serial line, as in during sampling

```
void loop() {
    Serial.print(micros());
    Serial.print(", ");
    Serial.println(analogRead(A0));
}
```

In fact, sending data over the serial line is time consuming and results are disappointing, even removing the first two lines with the information about time.

9. Dimensional analysis

Dimensional analysis if Dimensional analysis is an extremely useful tool to spot mistakes made while a powerful tool to manipulating the equations representing physics laws. Given that each member check the correctness of an equation represents a measurable quantity, both of them must have the of a physics law. same physics dimensions. If they do not, the equation is certainly wrong. The Unfortunately, its contrary is not always true: not all dimensionally correct equations are right. importance is often Still, a check of the dimensions of a physics quantity is always welcome and underestimated. often saves lot of time.

Dimensional analysis is even more powerful. Sometimes it can be used to write a relationship between two or more physics quantities ignoring most of the physics behind the scene. Here, we exploit it to determine the relationship between the speed of sound in air and the properties of the fluid. If v is not constant, it may depend on the properties of the medium through which sound propagates. These are its pressure p , its temperature T , its density ρ . The mass of the air cannot be a parameter, because it depends on the volume we choose to consider and v cannot depend on a choice of us. For the same reason, the volume V is not an option. Suppose, then, that $v = f(p, T, \rho)$ and observe that the ideal gas Law establish a relationship between p and V as

$$pV = nRT \quad (15.37)$$

where n is the amount of gas measured in units of moles and $R \approx 8.314 \text{ J mol}^{-1}\text{K}^{-1}$ a universal constant called the molar gas constant. $f(p, T, \rho)$ can only be a

combination of the three quantities and a constant C like

$$v = f(p, T, \rho) = v_0 + Cp^n T^m \rho^k. \quad (15.38)$$

Here v_0 represents the speed of sound in air when $T = 0^\circ\text{C}$ ($p = 0$ and $\rho = 0$ being unphysical). From dimensional analysis it follows that

$$[v] = [LT^{-1}] = [p^n T^m \rho^k]. \quad (15.39)$$

Pressure is a force per unit area:

$$[p] = \left[\frac{F}{A} \right] = \left[\frac{ma}{A} \right] = \left[\frac{MLT^{-2}}{L^2} \right] = [ML^3 T^{-2}]. \quad (15.40)$$

Density is mass per unit volume:

$$[\rho] = \left[\frac{m}{V} \right] = \left[\frac{M}{L^3} \right] = [ML^{-3}]. \quad (15.41)$$

Combining all together, and ignoring v_0 that already has the right dimensions, we obtain

$$[v] = [LT^{-1}] = [M^n L^{3n} T^{-2n} K^m M^k L^{-3k}] = [M^{n+k} L^{3n-3k} T^{-2n} K^m] \quad (15.42)$$

(we used K as a symbol for temperature, as not to confuse it with time). Comparing the dimensions of the right member with those on the left, we conclude that $n + k = m = 0$, hence $n = -k$ and $3n - 3k = 6n$. From the exponent of T we get $n = \frac{1}{2}$ and

$$v = v_0 + Cp^{\frac{1}{2}} \rho^{-\frac{1}{2}} = v_0 + C \sqrt{\frac{p}{\rho}} = v_0 + C \sqrt{\frac{nRT}{\rho V}}. \quad (15.43)$$

Since $n = \frac{m}{M}$, where $m = \rho V$ is the mass of the gas and M its molar mass (i.e. the mass of a mole of a substance, that depends on the gas mixture composition),

$$v = v_0 + C \sqrt{\frac{RT}{M}} = v_0 + c\sqrt{T}, \quad (15.44)$$

with $c = C \sqrt{\frac{R}{M}}$. In other words, the speed of sound in a gas increases as the square root of its temperature and decreases with the mass M of the molecules of which it is composed, still as \sqrt{M} . Indeed, you may have seen videos in which a person speaks with a much higher tone after inhaling helium. This is because

The speed of sound in a fluid grows as the square root of the temperature of the fluid.

helium is much lighter than nitrogen, of which air is made for about 80 %, then the speed of sound in helium is higher. The frequency f of the voice is related to its wavelength λ by

$$f = \frac{v}{\lambda} \quad (15.45)$$

and the higher is v , the higher is f .

Ultrasonic signals are sound waves at high frequency, then they propagate with the speed of sound.

In order to test the validity of our result and to measure M we need to setup an experiment in which we measure v as a function of T . Using Arduino this is a relatively easy task. We use an ultrasonic sensor aiming its beam of ultrasound waves against a wall at a known distance d and measure the time they need to come back, once reflected:

$$t = \frac{2d}{v} \quad (15.46)$$

from which we obtain v . The temperature can be measured by means of a temperature sensor. Both sensors can be placed into a box such that d is its length. Before starting the experiment we heat the air inside using a hair dryer, then close the box and let it to thermalise, measuring both v and T at the same time.

Linearisation always The aim of this experiment is to verify that $v \propto \sqrt{T}$, equivalent to proving that simplify the analysis of $v^2 \propto T$. The best thing to do is then to make a plot of v^2 as a function of T and verify that data distribute along a straight line. Alternatively, one can check between physics that a distribution of $\frac{v^2}{T}$ is gaussian with a width that is consistent with those quantities expected by the ratio of two quantities.

Impress your classmates

The GPS system relies on signals emitted by a constellation of satellites that continuously send data on Earth containing their coordinates with respect to a reference frame centred on Earth and the exact time at which the message left the satellite: (x_i, y_i, z_i, t_i) . The time is provided by an extremely precise atomic clock on board. The needed precision is so high that the time of each clock must be corrected for general relativity effects. In practice, time runs differently depending on the strength of the gravitational field in which the clock is immersed. Clocks on board orbit at a height of about 20 000 km above our heads, where the gravitational field is lower by a factor

$$\frac{h^2}{r_\odot^2} \simeq \left(\frac{26\,000}{6\,000} \right)^2 \simeq 19 \quad (15.47)$$

with respect to those on Earth, whose radius is $r_\odot \simeq 6\,000$ km. The correction factor must be

such that when a time t elapses on Earth, onboard the time is $t' \neq t$ and

$$t' = (1 - C) t \quad (15.48)$$

where C must be dimensionless and equal to zero where the gravitational field is null. The minus sign comes from the fact that in the presence of a gravitational field, the clock is accelerated and, if it was free, it increased its speed; in special relativity, clocks moving with respect to fixed ones run slower. C can only depend on the Earth's gravitational potential that is

$$\mathcal{G} = G \frac{M}{r} \quad (15.49)$$

$G \simeq 6.6 \times 10^{-11}$ being the Newton's constant in SI units and M the Earth mass. \mathcal{G} has the dimensions of the square of a speed [$L^2 T^{-2}$]. In order to make a dimensionless quantity, it must be divided by a quantity with the same dimensions. There is no other quantity from which C may reasonably depend upon, however, there is a universal constant, the speed of light c , that has the right dimensions. The ratio between the gravitational potential $G \frac{M}{r}$ and c^2 is dimensionless, hence

$$C = C(r) = \alpha G \frac{M}{rc^2} \quad (15.50)$$

with α dimensionless. It turns out that the right correction factor computed from general relativity is in fact that with $\alpha = 1$. Rather impressive, in fact. Dimensional analysis can provide results that only few persons on Earth are able to compute exactly, based on one of the most technically difficult physics theory.

A clock on the Earth's surface runs at a pace that is $(1 - C(r_\oplus))$ with respect to a clock at infinite distance, while GPS clocks run such that the correction is $(1 - C(r))$. The difference between the two clocks paces amounts to (neglecting the effects of special relativity, due to the high speed of GPS satellites, that account for a similar correction)

$$\Delta C = G \frac{M}{c^2} \left(\frac{1}{r} - \frac{1}{r_\oplus} \right) = 6.6 \times 10^{-11} \frac{6 \times 10^{24}}{9 \times 10^{16}} \left(\frac{1}{6 \times 10^6} - \frac{1}{26 \times 10^6} \right) \simeq 0.6 \times 10^{-9} \quad (15.51)$$

In other words, for each second elapsed on Earth, the clocks onboard mark *only* 0.9999999994 s. It may appear to be a negligible correction, but it is a systematic correction that accumulates with time. After one year the difference amounts to about 18 ms: during this time light travels for about 5 300 km, then the satellites appeared to be farther.

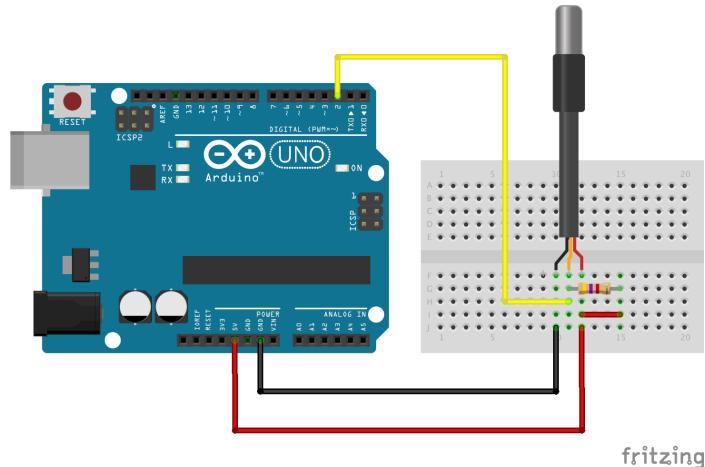


Figure 15.1: Connecting a 1-wire temperature probe to Arduino requires the usage of a *pullup* resistor of $4.7\text{ k}\Omega$ between the signal and the VCC leads.

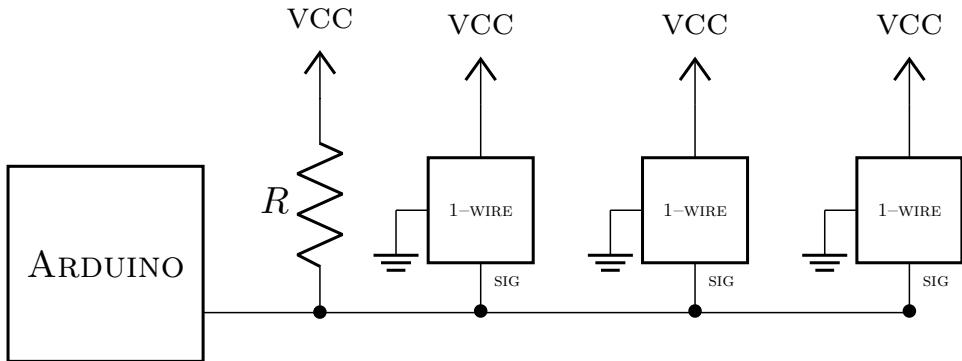
10. Temperature measurements with Arduino

For the experiments described in Section 9 we can use any analog temperature sensor that is enough to connect its GND lead to the pin with the same name on Arduino, its Digital lead with the Arduino 5V pin and measure the voltage connecting the third lead to one of the Arduino analog pins. The sensor data sheet contains the conversion factor from voltage to temperature.

protocol. In this

protocol data between master and slaves are exchanged on the same electrical line. In this case we illustrate the usage of another kind of sensor, based on a digital communication protocol known as 1-wire. Most waterproof sensors are based on this technology and it is useful to know about it. As the name says, the protocol uses just one wire to communicate from Arduino and the device and vice versa. 1-wire sensors have then three leads: two needed to power it, while the third is used to receive and send data over a serial digital line and must be connected to a digital pin. On this line data are represented as a sequence of voltage pulses that can either be 0 V or 5 V with respect to the ground. For this reason, the signal lead must be kept at 5 V when idle. To do that a resistor (called **pullup resistor**) can be used to connect the signal pin to the VCC lead: its suggested value is $4.7\text{ k}\Omega$.

Connections are illustrated in Fig. 15.1. Besides the features of the 1-wire protocol there is the ability to address more than one device attached to the same line as shown below together with a pullup resistor.



Here three 1-wire devices are attached to the same line connecting them to the same Arduino pin and to the pullup resistor R . Each device has its own power inputs to be connected, respectively, to VCC and GND.

Choosing the right resistor

Resistors are electrical circuit elements whose function is to regulate the circulating current. The higher the resistance, the less the current. The resistance is measured in Ohm (symbol Ω). In normal applications, the resistance of a conductor wire is completely negligible and we can usually assume that it is worth zero, but on the market there are resistors that can have electrical resistance values between a few Ω and millions of Ω ($M\Omega$, pronounced *megaohm*). The resistance value is represented by a series of coloured stripes on the body of the circuit element. Each color corresponds to a number according to the following table.

0	1	2	3	4	5	6	7	8	9

On each resistor there are usually four strips: three coloured as shown in the table; the fourth indicates the accuracy of the indicated value (**tolerance**) which is 5 % if gold and 10 % if silver. Let's suppose we have a resistance with the sequence red-red-brown-silver stripes. The resistance value is obtained by writing the numbers corresponding to the first two colours (22) and multiplying them by a power of ten corresponding to the third color (1). The resistance is then $22 \times 10^1 = 22 \times 10 = 220 \Omega$, which is a very common value. Given the tolerance of 10 %, the actual value of the resistance has an uncertainty of about 22Ω . A resistor of $4.7 \pm 0.2 \text{ k}\Omega$ is marked as yellow-purple-red-gold.

In the following we briefly explain how pullup resistors work. You can skip directly to the next section if you are not interested to the details of the instrument, taking the above prescription as a recipe. However, soon or later you need

The ex
pullup
irrelev
large e
the cu
from t
but lo
interfe
circuit
and sl

to study this topic: a bit of previous knowledge about it, though qualitative, helps in understanding it.

The output of a digital device is often driven by a transistor. In this application a transistor can be regarded as an electronic switch. Transistors have three leads: the emitter, the base and the collector. Normally no current flows from the emitter to the collector, unless a small current is injected into the base, that acts as a trigger for the switch. In order for the current to flow from the emitter to the collector, these two leads must be connected, respectively, to some positive voltage (e.g. 5 V) and to ground (at 0 V), by means of a resistor: a device through which a current flows according to the Ohm's Law

$$I = \frac{V}{R}. \quad (15.52)$$

When the resistance is measured in Ohms, the current is given in Ampère (symbol A). Referring to the figure on the side, where the device is represented by the dotted rectangle and the last stage of the communication circuit is shown as a transistor, if no current is injected in the base (labelled B) the transistor works as an open switch and the SIGNAL pin is at $V_{CC} = 5$ V. Upon injecting a small current in the base, a current $I = \frac{5}{4700} \simeq 1$ mA, flows from V_{CC} to the ground, to which the collector (labelled C) is attached. In this conditions, the voltage on the SIGNAL pin is 0 V. $R = 4.7$ kΩ is the pullup resistor, whose value is not extremely important. Its value must be large enough to keep the current flowing in the circuit small. However, if R is too large, resistors connected to the SIGNAL pin (in our case resistors on the Arduino board internally connected to the digital pin) may dominate and current flows from outside the circuit (from Arduino) to the ground. Usually, values from 1 kΩ to 10 kΩ are good enough.

11. The 1-wire protocol

In this protocol a single master device (Arduino) controls one or more slaves (the sensors). Addressing a single slave on a line is made possible assigning to each device a unique 64-bit address. Eight of these bits are reserved to identify the family of the device (devices of the same type share the same family code).

The device usually completes its task (e.g. reading the temperature) upon receiving a specific command represented by an 8-bit code. The command is represented as a sequence of pulses over the communication line, also called the bus. Data are stored into an internal memory (called scratchpad) that can be polled by the master to get them. Polling consists in issuing the reading command and reading back sequences of pulses representing the binary coded data of all the slaves.

Before communicating with slaves, the master must acquire their addresses over the 1-wire line.

They are picked up only by the addressed slaves.

This is done issuing a search command on the bus, to which each device responds providing its own address. For the temperature sensor based on the DS18B20 chip, for example, the search command is 0xF0 (corresponding to decimal 240). Measuring and obtaining temperature consists in the following steps.

1. Address the device sending the proper address over the bus.
2. Issue the start conversion command, that corresponds to obtain the temperature as a voltage and convert it into a number in the proper units. According to the device data sheet this is represented by 0x44 corresponding to the decimal 68.
3. Wait until the conversion is completed.
4. Address again the device.
5. Send the read command (0xBE corresponding to decimal 189).
6. Read as many sequences of 8 bits (bytes) as needed. This depends on the device: for the DS18B20 each data is represented by default as a 12-bit integer, consequently a minimum of two bytes must be read from the scratchpad.

The protocol is implemented under the default library `OneWire.h`. The library is seldom used as such. Most often, the user install libraries dedicated to his/her specific device and use them, being much more user friendly. It is, however, a good idea to see how such libraries are implemented, to be able to do similar things in the future.

To this purpose let's consider the waterproof temperature sensor based on the DS18B20 chip for which we install the library provided by Adafruit, an active developer of both hardware and software renowned also to be certified as a Minority and Woman-owned Business Enterprise (M/WBE). The needed library `DallasTemperature.h` can be obtained via the library manager of the Arduino IDE. A very basic example follows.

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define BUS 2

OneWire oneWire(BUS);
DallasTemperature tSensor(&oneWire);

void setup(void) {
    Serial.begin(9600);
    tSensor.begin();
}

void loop(void)
```

As us
be use
detail:
and w
progra
functi
what
slaves
action
data c
own n
know

```
{  
    tSensor.requestTemperatures();  
    float T = tSensor.getTempCByIndex(0);  
    Serial.println(T);  
}
```

After including `OneWire.h` and `DallasTemperature.h`, we instantiate `oneWire` as an object of the class `OneWire`. It takes a parameter representing the pin to which the 1-wire bus is attached. `tSensor` is instead an object of the class `DallasTemperature` requiring the address of the `oneWire` object (obtained using the operator `&`) to communicate.

The `begin()` method of the latter is used to establish the communication protocol. With this method, the object obtains the address of the sensor on the bus. This is accomplished as follows, as one can spot looking at the source code of the library. First of all the master (Arduino) writes `0x70` on the bus using the `write(0x70)` method of the `OneWire` library. All the slaves on the bus send back the 64^{th} bit (b_{64}^i) of their address followed by its complement (\bar{b}_{64}^i). The bus is designed such that, when more than one slave is connected, the serial line is found at a logic level corresponding to the logic AND of all the bits put on it by the slaves (let's call it b_{64}). The master gets it and respond with b_{64} such that all the slaves whose 64^{th} bit is different from b_{64} stops responding, while the other continue the dialogue sending b_{63}^i and \bar{b}_{63}^i . Again, the master respond with b_{63} and the process continues until all the devices were discovered on the bus. It is useful to mimic this process assuming three devices on the bus. For simplicity we pretend that the address is of four bits instead of 64: $A = 0001$, $B = 1010$ and $C = 1101$. The discovery protocol works as follows (refer to Fig. 15.2 and 15.3 depicting the first two cycles).

1. The master writes `0x70` on the bus.
2. All the three devices put their least significant bit on the line, respectively: 1, 0 and 1. The result is that the line is at logic level 0.
3. They then put the complement of their least significant bit on the line, respectively: 0, 1 and 0. The result is again that the line is at logic level 0.
4. The master put 0 on the line (the result of the AND in step 2) and store it as $R = 0$.
5. Slaves with 1 as the least significant bit stop (A and C).
6. The remaining slave (B) puts a 1 on the bus, followed by a 0.
7. The master gets the 1 and responds with it, storing the latter next to R at its left. R becomes $R = 10$.
8. The slave puts 0 followed by 1 on the bus.
9. The master gets the first 0 and sends it back, storing the 0 in R that becomes $R = 010$.
10. The slave puts 1 followed by 0 on the bus.
11. The master attach the 1 to R ($R = 1010$) and restart the process (in the

example the address if four bit long and R is full). In the next cycle, B will not respond.

12. A and C put both 1 on the bus that is found at logic level 1 by the master. Then put the complement of their last bit on the line whose logic state is then 0.
13. The master reads the first 1, such that $R = 1$, and respond with it.
14. Both devices have a 1 in their last digit and both puts their third bit on the line, whose logic level becomes 0. When they transmit the complements of their third bit the bus is found at 1.
15. The master attaches the first bit received (0) to the left of R ($R = 01$) and sends back 0 to the slaves.
16. Both slaves respond again, having both a 0 in the last transmitted bit. The response for the two slaves is 0 and 1, respectively, whose logic AND is 0. The next bit is the AND between 1 and 0, i.e. 0.
17. The master gets the first 0, attach it to the left of R ($R = 001$) and sends back 0 to the slaves.
18. Since C transmitted 1, it stops responding. Only A responds with 0, followed by its complement 1.
19. The master attach the first bit (0) to the left of R obtaining $R = 0001$ and starts a new cycle.
20. In the new cycle only one slave is present, whose address is then collected as is.

Thanks to the family bits, Arduino is able to spot all the temperature sensors on the bus and assign an index i to each of them, starting from $i = 0$, according to when they are discovered.

`tSensor.requestTemperatures()` asks `tSensor` to start the measurement. This consists in only issuing the command 0x44 (convert) to all sensors.

Finally, the `getTempCByIndex()` method returns the temperature in degree Celsius obtained by the sensor whose index is given in parenthesis.

Usually, it is not worth to handle the protocol by yourself and relying on the manufacturer libraries is a better option. In particular, the search algorithm is quite complicated. Nevertheless, knowing the protocol gives you a plus with respect to those that limit themselves to barely copy others' software.

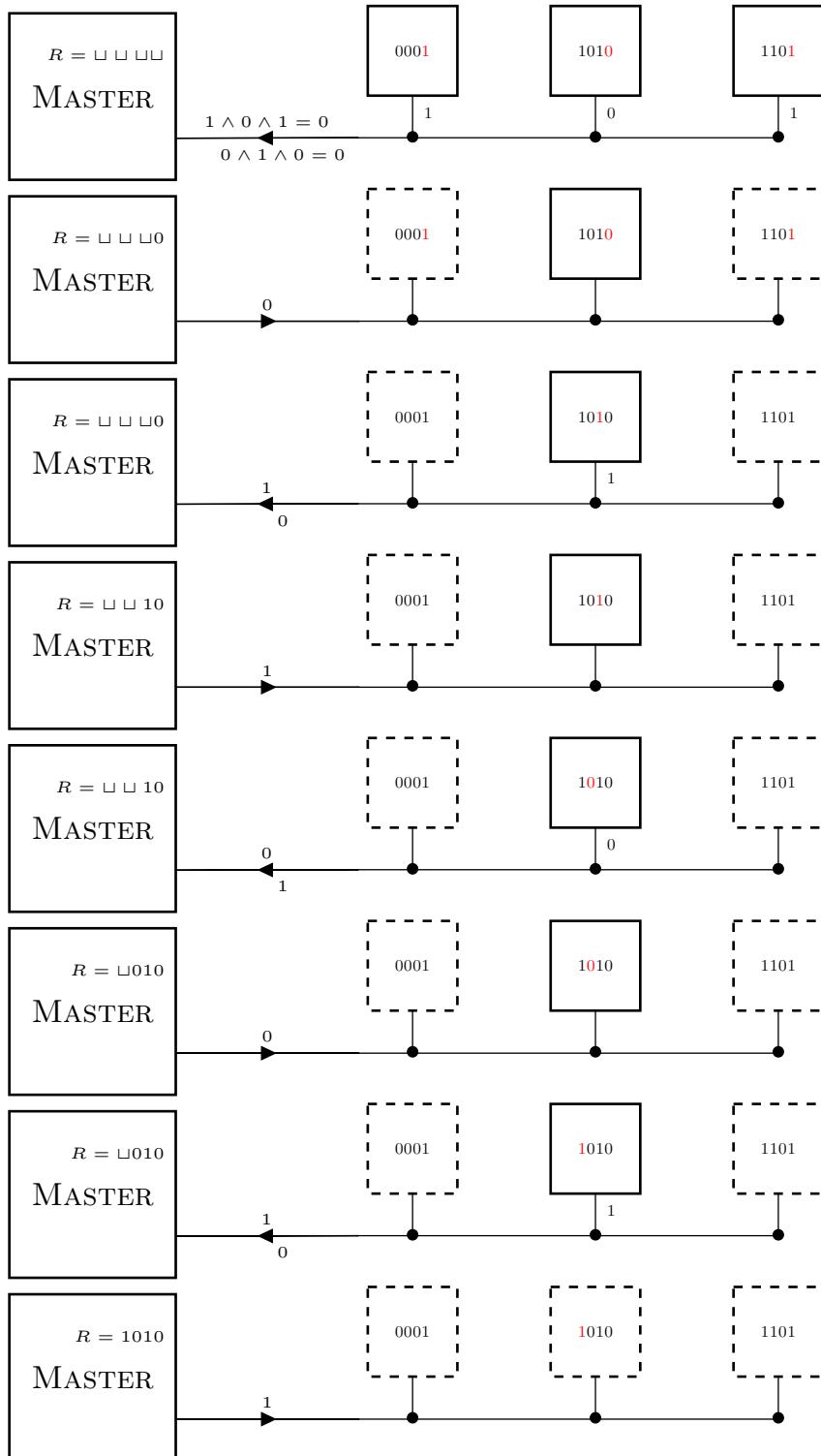


Figure 15.2: First cycle of the search algorithm for 1-wire communication. Slaves put successive bits and their complement on the bus. The bits are read by the master and the first is sent back to slaves. Only slaves with the same bit continue participating.

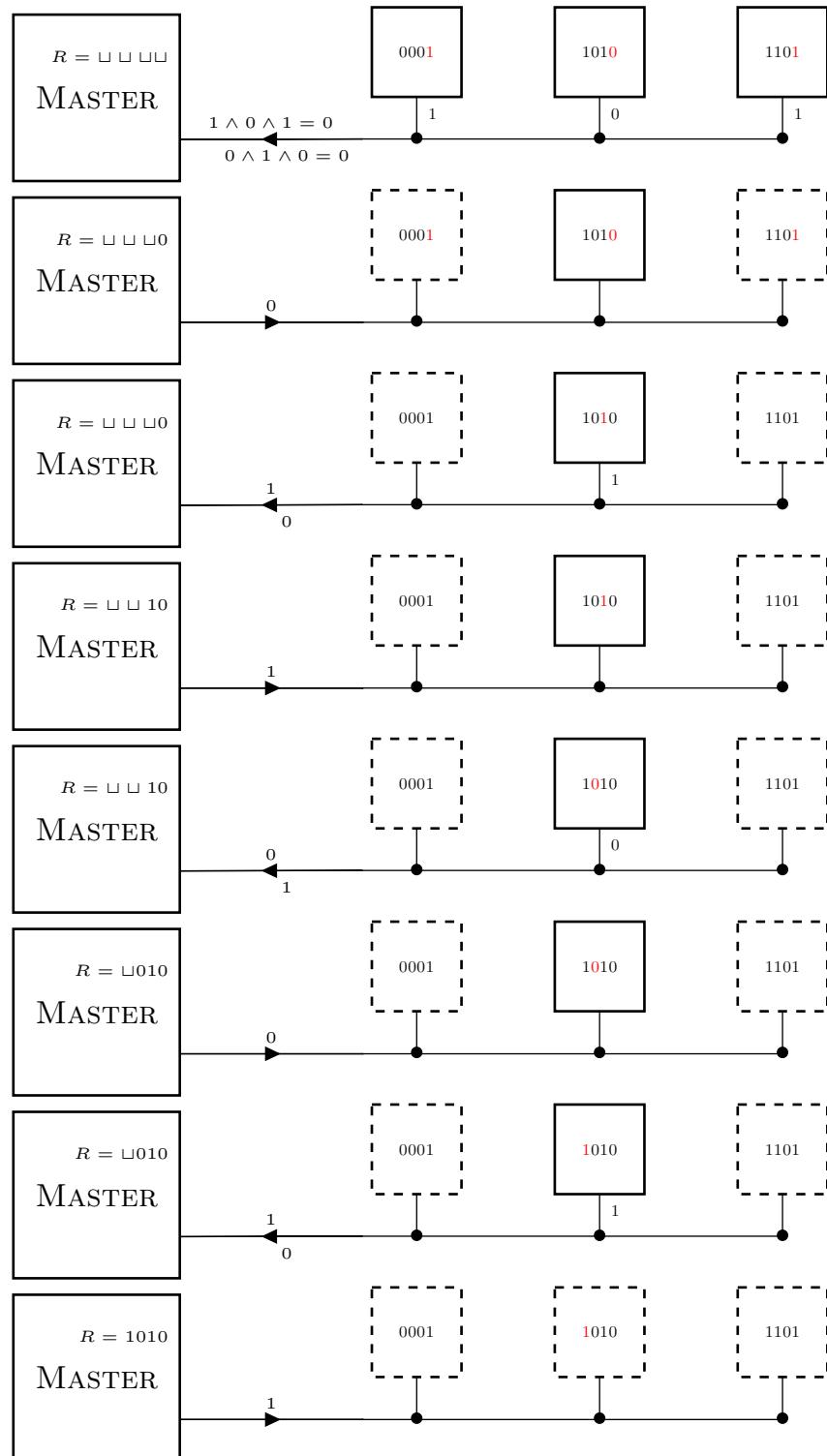


Figure 15.3: Second cycle of the search algorithm for 1-wire communication. Slaves stop participating when they were discovered. In turn, all devices are found.

12. Establishing a correlation

Once data have been collected, the distribution of v^2 as a function of T provides a clue about the possibility that $v^2 = \alpha T + \beta$. A best fit to the data is useful to obtain the values of α and β , while the χ^2 of the fit provides a measure of how valid the hypothesis is (in fact, $P(\chi^2 \geq \chi_0^2)$ is a measure of the probability that the observed fluctuations around the average straight line can be attributed to random fluctuations).

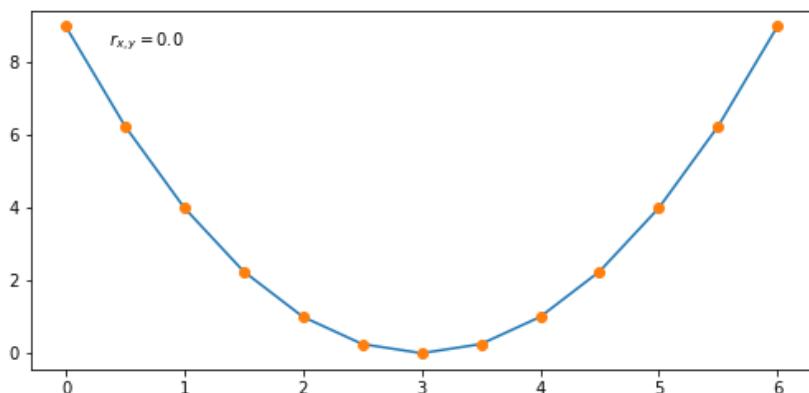
Another possibility to ascertain the existence of a linear relationship between two physical quantities x and y consists in using the **Pearson's correlation coefficient** defined as

$$\text{Pearson's correlation coefficient} \quad r_{x,y} = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y} \quad (15.53)$$

by Karl Pearson (1857–1936), also known as the **Bravais index** (after Auguste Bravais, 1811–1863). By definition, $-1 \leq r_{x,y} \leq +1$. The idea is that when x and y are uncorrelated, their covariance is null, as their $r_{x,y}$. If $r_{x,y}$ is zero, they are fully correlated $\text{Cov}(x, y) = \pm 1$. Two variables appear to be correlated with a strength proportional to $r_{x,y}$. It is useful to observe that such a statement is only valid for linear correlation. In particular, it is very possible that $r_{x,y} = -1$ (then perfectly correlated with x), while $r_{x,y}$ is close to zero. Moreover, if the relationship between x and y is not linear, the result may depend on the range of y .

For example, if $y = (x - 3)^2$ there is a manifest correlation between x and y , as seen in the figure below, despite $r_{x,y} = 0$.

are negatively correlated, the coefficient is -1 when the correlation is the



Moreover, consider Pearson's x , for $x_i \in [0, 2\pi]$ and $x_{i+1} - x_i = 0.5$. The correlation coefficient is not far from -1 , being $r_{x,y} \approx -0.74$; however, when

used to evaluate the degree of correlation of variables, if the expected correlation is not linear.

$x \in [0, 3\pi]$, $r_{x,y} \simeq 0.03$. The correlation coefficient changes also changing the density of the points. For example, for $x_{i+1} - x_i = 1$, $r_{x,y} \simeq 0.05$.

In conclusion, the Pearson's correlation coefficient is a useful tool to evaluate rapidly the degree of correlation of two quantities, if the relationship between them is linear. If not, such a measure is of almost no use.

It can be computed directly as the off-diagonal element of the symmetric matrix R_{xy}

```
R = np.corrcoef(x, y)
cov = R[0][1]
```

whose elements are defined as

$$R_{xy} = \frac{C_{xy}}{\sqrt{C_{xx}C_{yy}}}, \quad (15.54)$$

and C_{xy} are the elements of the 2×2 covariance matrix between x and y . The Pearson's correlation coefficient can also be easily computed as

```
from scipy.stats import pearsonr
corr, pval = pearsonr(x, y)
```

where $pval$ represents the p -value of the coefficient, indicating the probability $P(r > r_0)$ that the observed correlation coefficient r_0 has been obtained just by statistical fluctuation from uncorrelated variables. It is then close to 0 for highly correlated variables and close to 1 in the opposite case.

The
coeff
evalu
pear
the s
pack
value
retur
corre

Summary

Audio files are characterised by their sampling rate (the number of samples per second), their bit depth or resolution (the number of bits used for each sample) and the number of channels (mono or stereo).

The frequency of audible signal is comprised between 20 Hz and 20 kHz.

Almost any periodic function can be written as a series of trigonometric functions with appropriate frequency and amplitude, according to the Fourier's Theorem. The latter gives the recipes to compute the amplitudes for each frequency. Because of that, we do not need to study any possible wave: it is enough to study the dynamics of harmonic ones to predict the behaviour of any other wave. Harmonic waves are characterised by their wavelength, their period and their phase.

Even when the fit to a set of data appears to be good, when the model is wrong, an objective quality of the fit cannot be evaluated and uncertainties do not have a precise statistical meaning.

Whenever evaluating the uncertainties, we need to take into account the limited resolution of the instruments. If it dominates over statistical fluctuations, it must be properly considered.

The uncertainty of a set of measurements is in itself a measurement and, as such, its values are distributed over an interval with a PDF known as the Student's t. It is rather complicated, but it resembles closely a gaussian. Its width is a bit

larger, but the difference reduces as the number of degrees of freedom increases, as predicted by the Law of large numbers.

A function $y(x)$ of a random variable is a random variable. The PDF's $f(y)$ of y is related to that of x , $f(x)$, by $f(x)dx = f(y)dy$.

Interference consists in the superposition of two or more waves in a point. The resulting wave can have an amplitude ranging from the difference of their amplitudes to the sum of them.

The positions at which the amplitude of the sum of two equal waves is minimum can be easily predicted. Minima happen when the phase difference between interfering waves is an odd number of their half wavelength.

The standard deviation of the samples of a waveform is proportional to the amplitude.

In home made experiments on waves involving sound there are plenty of systematic effects that affect the measurements. Sound propagates in air and in solid media, it is subject to echos and its intensity diminishes with the distance from the source. Despite these limitations, many experiments can be successfully done.

Beats are produced by interference of waves with different frequencies. The resulting wave has a frequency equal to the average frequency between the interfering ones and an amplitude modulated with a frequency equal to half the difference of the

original frequencies.

The waveform of beats is characterised by a constant frequency and a modulated amplitude. In the case of sound waves, if the frequency of the modulation is enough short, it can be heard.

Dimensional analysis is a powerful tool to check the correctness of a physics law. Unfortunately, its importance is often underestimated. In fact, dimensional analysis also allows the derivation of relationships between physical quantities. Only constants remain undetermined, but they can be estimated from measurements.

The speed of sound in a fluid grows as the square root of the temperature of the fluid.

Ultrasonic signals are sound waves at high frequency, then they propagate with the speed of sound.

Linearisation always simplifies the analysis of the relationship between physics quantities.

Arduino

A microphone can be imagined as a parallel plate capacitor whose plates are separated by an elastic medium. The sound pressure makes the distance between plates to change. Consequently, an electrical current is drawn from the system.

Microphones for Arduino are analog sensors providing a voltage proportional to the recorded sound intensity.

To collect enough samples, data acquisition must be as fast as possible. Sending characters over a serial line is time consuming and must be avoided during sampling.

The division between two integers gives an integer. To obtain a rational number, one of terms in the division must be cast into a floating point value.

Temperature sensors can either be analog or digital. Digital thermometers often work with the 1-wire protocol. In this protocol data between master and slaves are exchanged on the same electrical

line.

Many 1-wire slaves share the same connection, together with a pullup resistor. The exact value of a pullup resistor is irrelevant. It must be large enough to limit the current drawn from the power supply, but low enough to not interfere with internal circuits of the master and slaves.

In the initialisation phase, slaves communicate their unique identifier to the master bit by bit. A proper algorithm disentangles the signal emitted by each slave and identifies them. Once the master collected the addresses of all the slaves, commands can be sent over the 1-wire line. They are picked up only by the addressed slaves.

Libraries can be used to ignore the details of the protocol and write simple programs. Library functions encapsulate what is needed to ask slaves to perform their actions and return data collected in their own memory (also known as scratchpad).

The Pearson's correlation coefficient, defined as

$$r_{x,y} = \frac{\text{Cov}(x,y)}{\sigma_x \sigma_y}$$

provides a mean to evaluate the degree of linear correlation between two variables x and y . The coefficient is zero for uncorrelated variables. A value +1 indicates the maximum possible positive correlation. If x and y are negatively correlated, the coefficient is -1 when the correlation is the strongest possible.

The Pearson's coefficient must not be used to evaluate the degree of correlation of variables, if the expected correlation is not linear. **Phyphox**

The properties of a wave can be explored using PHYPHOX that shows $f(0, t)$, $x = 0$ corresponding to the smartphone's microphone position.

PHYPHOX provides audio data in arbitrary units. They can be converted into absolute intensities upon microphone calibration, that needs a source whose intensity is known. For most experiments, absolute intensities are useless.

Python

Sounds can be produced and manipulated using the `simpleaudio` module, in Python.

Arrays are a special kind of list, whose elements must belong to the same type.

The `*` operator applied to a list and an integer extends the list. The same operator applied between an array and a constant multiplies each element of the array by the constant.

Multidimensional lists or arrays are represented as list of lists or as array of arrays.

The `simpleaudio` package provides tools to interact with loudspeakers and other peripherals intended to play audio signals. Audio data, in the form of a bidimensional array of n -bits integer, can be played using `play_buffer()`. After invoking `play_audio()` the program continues immediately. To suspend its execution until the audio is played, the `play` object must wait for it.

The `getopt` package allows the usage of options and long options in launching a script on the com-

mand line of a terminal. Options are preceded by a dash (-) and may or not require arguments. They can be combined together if they don't. Long options are preceded by two dashes (--).

`getopt()` returns a list of pairs representing the options with their arguments, and a list of arguments. The first object returned by `getopt()` consists of a list of pairs. Each pair is made of an option and its corresponding value. The second object is a list of arguments given in the command line.

The `in` operator checks if its left operand is contained in its right operand.

Python's `scipy.stats` module has functions to compute the PDF and the corresponding cumulative of many distributions, including the Student's t.

The Pearson's coefficient can be evaluated using `pearsonr`, defined in the `scipy.stats` package. Besides its value, the function returns the corresponding p -value.

Bibliography

- [1] Student, "The probable error of a mean", in Biometrika Vo. 6, No. 1 (1908) pp. 1–25.
- [2] Thomas Young, "On the theory of light and colors", in Philosophical transactions of the Royal Society, v. 92 (1802).

Index

- Ω , 36
- $*$, 66
- $++$, 66
- $+=$, 66
- $//$, 67
- $:$, 47, 57
- $=$, 66
- $\{\}$, 48
- Aachen, 37
- absolute value, 54, 92
- absorption of light, 21
- abstract type, 74
- acceleration, 89, 93
- accelerometer, 37, 93
- acoustic stopwatch, 88
- actuator, 38
- ADC, 43
- aesthetic, 63
- alias, 56
- Almagest, 13
- ampère, 26
- analog sensor, 42
- analog to digital converter, 43
- analogRead(), 44, 46
- Apollo 15, 93
- append(), 72
- approximation, 55
- arbitrary units, 48
- Arduino, 38
- Arduino UNO, 38
- array, 47
- art, 73, 124
- ASCII, 65
- attribute, 57
- augmented reality, 37
- auto-increment, 66
- average, 53
- axiom, 12
- balloon, 88
- Banzi, Massimo, 39
- baud, 45
- Bayes' Theorem, 116, 119
- Bayes, Thomas, 109
- Bayesian, 109
- beauty, 120
- bet, 109
- BGI, 91
- bin, 53, 58, 126
- binary system, 28
- binomial coefficient, 70
- biology, 11
- BIPM, 26
- bolt, 88
- Boole, George, 57
- Boolean expression, 57, 67
- Bureau Gravimetrique International, 91
- Bureau International des Poids et Measures, 26
- byte, 47
- C, 40
- C++, 40
- calibrated instrument, 36

calibration, 36
calibration factor, 67
caliper, 68
camera, 37
cancer, 80
candela, 26
capacitance, 79
capacitor, 28, 78
Celsius, Anders, 25
Celsius, degree, 25
centigrade, 25
CERN, 121
character, 65
characterisation, 16
class, 47, 53, 57, 58, 126
classical mechanics, 52
clock, 86
cloud, 37
coding, 16
colon (:), 57
colon (:), 47
combinations(), 74
combinatorics, 70
comma separated value, 42
comment, 67
common sense, 12
compass, 37
complement, 86
conditional probability, 113
constant, 85
continuous, 51
conversion factor, 23
Copernicus, Nicolaus, 13
cosmic rays, 80
cosmology, 13
COVID–19, 9, 117
Cristin, Jean–Pierre, 25
CSV, 42, 58, 71
Cuartielles, David, 39
current, 78

DAQ, 16
dark matter, 121
data acquisition, 16

data export, 41
DataFrame, 58
decimal system, 28
decode(), 47
delay(), 86
delayMicroseconds(), 86
density, 36
density, 128
derivative, partial, 92
derived units, 29
Dickens, Charles, 123
differential equation, 76
dimensional analysis, 69
dimensionless, 76
Dirac, Paul, 122
direct measurement, 22
discovery, 91
distribution, uniform, 126
double slash (//), 67
Dropbox, 37
duodecimal system, 27, 28
Dupont, connector, 43

echo, 88
econophysics, 11
ecosystem, 38
Einstein, Albert, 13
electrical circuit, 78
electrical resistance, 36
electron, 27
else, 57
emotion, 124
encoding, 65
energy, 27, 78
EOL, 47
epicycle, 13
epistemology, 11
epoch, 48
equation, 14
equator, 26
error, 73
errorbar(), 72
Euler, Leonhard, 123
eV, 27

evidence, 117
excess–127, 65
expected value, 129
experiment, 16
exponent, 65
exponential, 78
exponrntial, 76
exporting data, 41
expression, Boolean, 57

$F = ma$, 14, 123
factorial, 70
Fahrenheit, degree, 25
Fahrenheit, Gabriel, 25
false, 57, 67
false negative, 118
false positive, 118
falsificationism, 11
feather, 93
Feyerabend, Paul, 11
Feynman, Richard, 120
FIFO, 47
file, 47
file mode, 47
finance, 11
flash light, 63
float, 64
floating point, 65
fluid, 93
foot (feet), 27
for, 66, 72
format(), 48
free fall, 83
free software, 39
freedom, 12
French Revolution, 26
frequency, 53, 127
frequentist, 108
friction, 78
Fritzing, 43
function, 45, 73
fundamental unit, 27

galaxy, 13

Galilei, Galileo, 122
Gauss, Carl Friedrich, 129
general relativity, 13
generator, of random number, 111
geometry, 12
GMT, 48
GND, 43
GNU, 39
GNU, 112
God, 125
Google, 25
Google Drive, 37
GPS, 37
graduated instrument, 35
Gran Sasso, 121
gravity acceleration, 91
Guernica, 124
gyroscope, 37

haiku, 123
hammer, 93
 \hbar , 52
heat, 14
Heaviside, Oliver, 123
Hess, Viktor, 80
heuristics, 11
hexadecimal, 65
hexadecimal system, 28
Higgs boson, 121
hist(), 126
hist(), 58
histogram, 53, 126

IDE, 38, 44
IEEE-754, 65
if, 57
illuminance, 41, 53, 67, 74, 85
imperial units, 27
import, 56
import...as, 56
inch, 27
indentation, 46
independency, of events, 113
index of dispersion, 54

index of position, 53
instrument, 35
 calibrated, 36
 graduated, 35
integer, 65
intensity
 of light, 21
intensity, of light, 21
intercept, 70
International System, 26
ISO Latin-1, 65
iterating, 66
 iterating structure, 47, 67, 72
itertools, 74
Ivrea, 39
joint probability, 113
jumper, 43
kelvin, 26
Kepler, Johannes, 12, 13
kilogram, 26
Knuth, Donald, 73
Kuhn, Thomas, 11, 122
label, 58
Lakatos, Imre, 11
lamp, 63
Lamport, Leslie, 73
landscape, 93
Laplace, Pierre-Simon, 108
L^AT_EX, 73
`len()`, 57, 72
library, 46
light absorption, 63
light intensity, 40
light sensor, 37
light year, 27
light, absorption, 21
light, intensity, 21
likelihood, 117
linear congruential method, 112
linearisation, 76, 90
Linux, 45
list comprehension, 72
literature, 123
Im, 41
lockdown, 9
logical statement, 57
`long`, 86, 87
`loop()`, 45
love, 14
lumen, 41
luminous flux, 41
lux, 41
lx, 41
machine code, 44
macOS X, 45
magnetometer, 37
mantissa, 65
marginal probability, 111, 113, 115
Mariotti, Filippo, 12
mass, 93
mathematics, 12, 122
`matplotlib`, 56
Maxwell's equations, 123
Maxwell, James, 123
mean, 53, 59
mean free path, 76
`mean()`, 59
measure, 14
measurement, 12, 14, 16
measurement, direct, 22
median, 53, 59
`median()`, 59
meter, 26
method, 47
metric system, 26
microphone, 37
`micros()`, 85, 86
Microsoft Windows, 45
`millis()`, 86
mode, 54
mode, file, 47
module, Python, 46
mole, 26
moment, of a distribution, 129, 131
moon, 93

multiplication, 66
 music, 13
 NASA, 93
 natural phenomena, 11
 natural unit, 27
 negative number, 87
 negative numbers, 86
 neutrino, faster than light, 121
 newline, 46, 47
 Newton's Laws, 123
 Newton, Isaac, 13, 123
 non-Euclidean geometry, 12
 normal form, 65
 normalisation, 130
 North Pole, 26
 np, 59
 NPV, 118
 NULL character, 65
 numeral, 28
 numeral systems, 27
 numpy, 56, 59
 numpy.random, 126
 object, 47, 57
 Object Oriented Programming, 47
 Ohio, 93
 ohm, 36
 Ohm's Law, 79
 Olivetti, Adriano, 39
 OOP, 47
 open source, 39
 open(), 47
 Opera, 121
 opinion, 12
 oscillator, 75
 pandas, 56, 57
 pandemic, 117
 paradigm shift, 11
 partial derivative, 92
 pd, 58
 pendulum, 78
 period, of a random number generator, 112
 photogate, 85
 PHYPHOX, 37, 40
 physics, 11
 physics law, 14, 68
 physics quantity, 14, 15
 PHYSICS TOOLBOX, 37
 Picasso, Pablo, 124
 Pickwick Papers, 123
 pin, 38
 Pink Floyd, 23
 Planck's constant, 52
 Planck, Max, 52
 planet, 12
 platinum thermometer, 36
 plt, 57
 point-like, 83
 Popper, Karl, 11
 portrait, 93
 positional notation, 27
 post-increment, 66
 posterior probability, 117
 PPV, 118
 prefix, 27, 28
 pressure, 37
 prior probability, 111, 117, 120
 probability, 108
 probability density function, 130
 probability, conditional, 113
 probability, joint, 113
 probability, marginal, 113, 115
 probability, posterior, 117
 probability, prior, 117
 Programma 101, 39
 propagation of uncertainty, 90, 92
 proton, 27
 PRT, 36
 pseudorandom, 112, 125
 PT100, 36
 Ptolemy, 13
 pyplot, 57
 Python, 40
 qualifier, 88
 quantum mechanics, 52, 111, 124

radioactive decay, 79, 111
radiotherapy, 80
Raffaello Sanzio, 124
random number generator, 111
`range()`, 72
RC, 78
`read()`, 57
`read_csv()`, 72
`read_csv`, 58
reading uncertainty, 51
`readline()`, 47
relative uncertainty, 71
reliability, 118
remote access, 42
research program, 11
reset, 45, 67
resistance, 78
resistance, electrical, 36
resolution, 51
retrograde motion, 13
rigour, 14
Rolling Stones, 23
`rstrip()`, 45, 47
ruler, 35
`rwidth`, 127

Sanzio, Raffaello, 124
science, 11
scientific method, 120
scientific notation, 58
scientific revolution, 11
Scott, David, 93
second, 26
seed, 112
semicolon (;), 66
sensor, 38
separable differential equation, 76
sequence, 74
serial, 46
serial monitor, 44, 46
`Serial.begin()`, 44, 45
`Serial.print()`, 46
`Serial.println()`, 44, 46
`serial.Serial()`, 45

`setup()`, 45
`show()`, 59
SI, 26
significant figures, 55
sketch, 44
slope, 70
sound speed, 91
Space Power Facility, 93
special relativity, 69
speed of light, 27
speed of sound, 91
spin, 122
spring, 75
Staacks, Sebastian, 37
standard, 22
standard deviation, 54, 59, 91
state, 83, 84
`stdev()`, 59
stopwatch, 84
stopwatch, acoustic, 88
`String()`, 47
`string`, 46, 65
structure, iterating, 47
subjective probability, 109, 119
submodule, 57
swab, 117
`sys`, 57
`sys.argv`, 57
systematic shift, 91

T, 58, 72
tailor's tape, 86
Taylor's series, 75, 90, 92
technology, 73
temperature, 25, 36, 79
`TeX`, 73
theory, 13
tick mark, 73
time, 46
time dilation, 69
`time()`, 48
time, measuring, 86
timed run, 41, 64, 94
`tolist()`, 58, 72

transistor, 28
transpose, 58, 72
true, 57, 67
True, 47
true negative, 118
true positive, 118
true, value, 107
truth, 12, 13
two's complement, 86
type, 45, 65
typesetting, 73
uncertainty, 15, 73, 90
uncertainty propagation, 90, 92
uncertainty, reading, 51
uniform, 56
uniform, 126
uniform distribution, 126
unit, 15, 22
unit, fundamental, 27
unit, natural, 27
UNO, Arduino, 38
unsigned, 88
unsigned long, 85
USB, 38, 44–46
usb.readline(), 45
UTC, 48
UTF-8, 65
vacuum chamber, 93
values, 58, 72
variable, 64
variance, 54, 67, 129, 131
VCC, 43
velocity, 89
 Vernier scale, 35
virtual reality, 37
while, 47, 67
Wigner, Eugene, 122
Windows, 45
write(), 48
 xlabel(), 58
Yosa Buson, 123
YOUTUBE, 93, 120