UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

etsinf
Escola Tècnica
Superior d'Enginyeria
Informàtica

Computer Science in a Multidisciplinary Environment. Quantum Cryptography and Econophysics.

A Final Year Project by
Gonçal Garcés Díaz-Munío
student of Computer Science and Engineering
at Escola Tècnica Superior d'Enginyeria Informàtica, UPV

as directed by
Dr. Taksu Cheon
(Kochi University of Technology)
and
Dr. Juan Miguel García Gómez
(Universitat Politècnica de València)

I would like to thank Professor Taksu Cheon for his guidance and for his clear explanations and Dr. Juan Miguel García Gómez for his dedication and advice.

全先生、大変お世話になりました。

ABSTRACT

Students of computer science are becoming increasingly specialized, but to work in multidisciplinary teams they need to appreciate different perspectives and methods. This Final Year Project was carried out in the Physics department at Kochi University of Technology (Japan), and it concerns two interdisciplinary fields: quantum cryptography and econophysics. In the field of quantum cryptography, two of the most representative protocols for quantum key distribution, BB84 and B92, were analyzed and compared; their practical implementation was also described. In the field of econophysics, we explored the properties of the statistical family of Lévy distributions, which are applied in financial computing to model the evolution of prices.

RESUMEN

Los estudiantes de Ingeniería Informática se especializan cada día más, pero para participar en equipos multidisciplinares deben ser capaces de apreciar perspectivas y métodos diferentes. Este Proyecto Final de Carrera fue llevado a cabo en el departamento de Física de la Kochi University of Technology (Japón), y se adentra en dos campos interdisciplinares: la criptografía cuántica y la econofísica. En el campo de la criptografía cuántica, se han analizado y comparado dos de los protocolos de distribución cuántica de claves más representativos, el BB84 y el BB92; además se ha descrito su implementación práctica. En el campo de la econofísica, se han explorado las propiedades de la familia estadística de distribuciones de Lévy, las cuales se aplican en computación financiera para modelar la evolución de los precios.

RESUM

Els estudiants d'Enginyeria Informàtica s'especialitzen cada dia més, però per participar en equips multidisciplinaris han de ser capaços d'apreciar perspectives i mètodes diferents. Aquest Projecte de Fi de Carrera va ser desenvolupat en el departament de Física de la Kochi University of Technology (Japó), i recorre dos camps interdisciplinaris: la criptografia quàntica i l'econofísica. En el camp de la criptografia quàntica, s'han analitzat i comparat dos dels protocols de distribució quàntica de claus més representatius, el BB84 i el B92; a més se n'ha descrit la implementació pràctica. En el camp de l'econofísica, s'han explorat les propietats de la família estadística de distribucions de Lévy, les quals s'apliquen en computació financera per modelar l'evolució dels preus.

学際的な計算機科学：量子暗号と経済物理学

Contents

Abstract………………………………………………………………………………………… 2

Contents…………………………………………………………………………………………3

1. Introduction…………………………………………………………………… 4

2. Quantum Cryptography *Quantum key distribution protocols: BB84 and B92*

2.1 Introduction……………………………………………………………….. 6
2.2 Some basic quantum mechanics for quantum cryptography………………… 7
2.3 The BB84 protocol……………………………………………………… 10
2.4 The B92 protocol………………………………………………………….. 13
2.5 Comparison between the BB84 and the B92 protocols………………………. 16
2.6 Application of quantum cryptography………………………………………… 17
2.7 Closing comments………………………………………………………… 18

3. Econophysics *Some statistical properties of the Lévy distribution*

3.1 Introduction……………………………………………………………….. 19
3.2 Random walk. Generation of sets according to specific probability distributions.21
3.3 Additive stability of probability distributions. Rescaling……………………… 25
3.4 Stability of Lévy distributions………………………………………………... 27
3.5 Closing comments…………………………………………………………... 38

4. Concluding comments……………………………………………………... 39

5. Bibliography……..……………………………………………………………... 41

Appendix. **Matlab functions implemented for Chapter 3**…………………………… 43

1. Introduction

In recent times, scientists, researchers and engineers have become increasingly specialized. To reach the high depths of knowledge and expertise which are required of them nowadays, scientists focus their attention on minute areas of study; the current paradigm in our jobs is far from the polymath or Renaissance man of ancient times. However, the most complex problems we face still need the application of different disciplines to tackle them, which creates a necessity for *interdisciplinary collaboration*. Thus today's experts in science and engineering must not only be able to reach achievements within their particular field of study, but must also have the ability to work together with experts from other fields in a *multidisciplinary team* – both in the worlds of academy and business.

Computer scientists and engineers are no exception to this rule – fields such as bioinformatics, cybernetics, information science and quantum computing reside in the intersection between computer science and other disciplines. A student of Computer Science and Engineering will therefore benefit greatly from developing such collaborative skills during the course of its studies. For if he or she wishes to participate in multidisciplinary ventures in the future, he or she must become able to appreciate differing perspectives and methods.

Physics is one of the classical sciences which have provided the basis for computer science: physics constitute the foundation of the hardware in which the mathematical-based apparatus of software is housed. In this Final Year Project, the student has worked as a computer scientist in a physics laboratory, getting himself acquainted with two promising interdisciplinary subjects: quantum cryptography and econophysics.

Quantum cryptography is one of the topics which we have explored in this Final Year Project. We have already mentioned quantum computing as a multidisciplinary field related to computer science, but it is not only one more field in that list: it is among the most important ones in current computer science research. As current computer paradigms reach their physical limits, the application of the theories of *quantum mechanics* to both hardware design and algorithms is opening the way for the future of computer science.

In the chapter on quantum cryptography, we introduce some basic notions of quantum mechanics which are required to understand any quantum protocol. We then proceed to describe, compare and discuss the implementation of BB84 and B92, two protocols which take advantage of the property of quantum indeterminacy to allow for provably secure key distribution. Their provable security makes them ideal candidates to replace the currently prevailing public key cryptographic protocols if they are broken (something that is bound to happen when full-fledged quantum computers become a reality).

Econophysics, the second of the main multidisciplinary fields covered in this Final Year Project, is the application of theories and methods from the field of physics to solve problems in economics. In the last decades, quantitative analysts (or *quants*) have acquired relevance for their work in developing pricing models for investment products. Financial institutions have traditionally recruited these quants from the ranks of mathematics and physics graduates, but a strong background in computer programming or in advanced computational methods such as neural networks or evolutionary computation is becoming increasingly valued for these tasks. Thus computer scientists have also their place in this area, under the new discipline of computational finance; however, they will have to collaborate with physicists and mathematicians, and they will need to understand the methods that physicists and mathematicians use for modelling.

In the chapter on econophysics we study the properties of statistical distributions that are frequently applied for economical and financial modelling, namely the family of Lévy stable distributions, focusing especially on the Gaussian (normal) distribution and the not-so-common Lorentzian (Cauchy-Lorentz) distribution. The Lévy distribution is applied in financial modelling because of its empirical similarity to the returns of securities; changes in prices do not follow a Gaussian distribution, but are rather better modelled by Lévy stable distributions. The background in mathematics and statistics acquired in the degree in Computer Science and Engineering proves to be effective for application in this field.

Quantum Cryptography

Quantum key distribution protocols: BB84 and B92

2.1 Introduction

The theory of quantum mechanics has brought fundamental changes in physics, and science as a whole, since its inception in the early 20th century. Especially since the 1980s, quantum mechanics have been applied also in the field of computation and computer science, to the point that its proponents argue that the future of computation lies in *quantum computation*.

It could be argued that it is just a matter of size. Quantum mechanics can be in some way “ignored” when we deal with the human scale world; we can manage with our common-sense classical physics. But the electronic components inside computers are reaching minute sizes in which quantum mechanics have to be taken into account. Thus a change of paradigm, from classical computers to quantum computers, might be just waiting to take place in the next years [1].

But that is not the main topic of this chapter. What we intend to discuss here is the subfield known as *quantum cryptography*. And once again, the reader might want to know why we need such an exotic thing, why cannot we go on with our classical cryptography like we have done until now. Well, it is precisely because of quantum computing that we will need quantum cryptography.

Let us explain that. Right now, the most popular cryptographic protocols are public key cryptosystems. Public key protocols are not theoretically unbreakable; they rely on the assumption that the calculations needed to break them are hard to solve (meaning, they require too much time to be solved) on classical computers. So, what would happen if those hard problems became easy to solve using quantum computers? That is exactly what might happen when full-fledged quantum computers become a reality. For instance, RSA, the most widely used public key cryptosystem today, relies on the assumption that factoring large numbers is computationally unfeasible on classical computers. However, there is already a quantum computing algorithm, known as Shor’s algorithm, for solving factoring in a fast way (in polynomial time) on quantum computers. That means RSA would become easily breakable by using quantum computers [2].

In that case, we would need some other cryptographic method to encode our private communications. Quantum cryptography gives us just that. Interestingly, the quantum formula for cryptography means going back to private key cryptosystems. The problem with private key cryptosystems, up until now, has been secure distribution of the keys. The key point of quantum cryptography is that it provides a provably secure way to share information over a public channel. Thus we speak of *quantum key distribution* protocols, which exploit quantum indeterminacy to make provably certain that any eavesdropping would be detected. A key shared securely by using quantum cryptography can then be used for secure communication by means of a classical private key cryptosystem.

Quantum computing, therefore, might someday make current public key cryptosystems obsolete, but give us at the same time the first truly, physically secure cryptographic protocols.

In this chapter we will have a look at two of the most significant quantum key distribution protocols. One of them, BB84 [3], is the first of its kind to have been developed, while the second, B92 [4], is a refinement by one of the authors of the first one. We will analyze how they work and what are the differences between them, and then go on to describe the basic requirements to implement them for practical use.

2.2 Some basic quantum mechanics for quantum cryptography

Quantum cryptography is built over basic concepts of *quantum information theory*. To use the properties of quantum mechanics to securely exchange information we need first a way to codify physical information as a *state* of a quantum system.

The basic unit of quantum information is the *qubit*, described by a state vector in a two-level quantum system. A qubit, like a classical bit, can have two possible values: 0 or 1. But, unlike classical bits, it can also be a superposition of both values. The states a qubit may be measured in are known as *basis* states. They are traditionally represented using bra-ket (Dirac) notation; thus, the "0" state is represented as |0> ("*ket* zero"), while the "1" state is represented as |1> ("*ket* one"). As for the physical representation of qubits, any two-level system can be used. For instance, single photons can be used as qubits; photon polarization (horizontal or vertical) will determine the quantum state [5].

In our examples, we will take as a reference the representation of qubits using electrons. In this case, electronic *spin* determines the quantum state. We can imagine the spin as an arrow[1]: the "up" (↑) spin represents the state |0>, while the "down" (↓) spin represents the state |1>.

Quantum cryptography exploits *quantum indeterminacy*, one important and unique property of quantum systems, to achieve security in communications. For our purposes, we can interpret quantum indeterminacy as meaning that observation is not a passive activity in quantum systems; on the contrary, the act of observing (measuring) a quantum system affects its state.

Let us try to explain this property through an example. A first person, whom we will refer to as Alice, can prepare an electron so that it points in a given direction θ. When a second person, Bob, comes and measures in which direction is the electron pointing, the result will depend not only on the angle θ that Alice set on the first place, but also on the angle φ in which Bob conducts the measurement. In fact, the observed direction will be φ or φ+180º, the probability of obtaining each result depending on both φ and the original angle θ; the only measurement that is impossible to achieve is θ+180º. So, Bob can determine the two possible results, but not the probability with which each of them will come out.

We will now introduce some basics about the mathematical notation of quantum states before going on to analyze communications between Alice and Bob in more detail.

Hilbert vectors

Mathematically, quantum states are represented as *vectors in Hilbert space*. The quantum state set up by Alice in angle θ would look like this as a Hilbert vector:[2]

$$|\theta> = \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{pmatrix} = \cos\frac{\theta}{2}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \sin\frac{\theta}{2}\begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ , } 0 \le \theta < 2\pi \qquad \text{(Eq. 1)}$$

As we show, any quantum state can be expressed as a linear combination of the two basis states

$$\text{“up”} = |\uparrow> = |0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and “down”} = |\downarrow> = |\pi> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

[1] The electron has a magnetic moment; we can imagine an arrow pointing through the electron towards its north pole.

[2] *Ket* vectors are represented as column vectors.

The *up-down basis* (↕ basis) is one of the two bases that we will use in our versions of BB84 and B92.

As for the quantum state that Bob is ready to measure in angle φ, its Hilbert vector would look like this:[3]

$$<\varphi| = \begin{pmatrix} \cos\frac{\varphi}{2} & \sin\frac{\varphi}{2} \end{pmatrix} = \cos\frac{\varphi}{2}\begin{pmatrix}1 & 0\end{pmatrix} + \sin\frac{\varphi}{2}\begin{pmatrix}0 & 1\end{pmatrix} \text{ , } 0 \le \varphi < 2\pi \qquad \text{(Eq. 2)}$$

Finally, the probability of Bob finding the final quantum state $<\varphi|$ when measuring the initial quantum state $|\theta>$ would be [1]:[4]

$$P(\varphi \mid \theta) = |<\varphi|\theta>|^2 = |\cos\frac{\varphi}{2}\cos\frac{\theta}{2} + \sin\frac{\varphi}{2}\sin\frac{\theta}{2}|^2 \qquad \text{(Eq. 3)}$$

Thus we have exposed the basic points of the algebra used for quantum measurement.

Just one more concept before going on: as the reader can imagine, the up-down (↕) basis is not the only basis in Hilbert space. There are infinite bases, from which we will only use a second one for our explanations of BB84 and B92: the *right-left basis* (↔ basis). Its two basis states are

$$\text{“right”} = |\rightarrow> = |\pi/2> = \begin{pmatrix} \cos\frac{\pi}{4} \\ \sin\frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \text{ and } \text{“left”} = |\leftarrow> = |3\pi/2> = \begin{pmatrix} \cos\left(-\frac{\pi}{4}\right) \\ \sin\left(-\frac{\pi}{4}\right) \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}.$$

The ↔ basis can be decomposed in terms of the ↕ basis, and vice-versa:

$$\begin{cases} |\rightarrow> = \frac{1}{\sqrt{2}}|\uparrow> + \frac{1}{\sqrt{2}}|\downarrow> \\ |\leftarrow> = \frac{1}{\sqrt{2}}|\uparrow> - \frac{1}{\sqrt{2}}|\downarrow> \end{cases} \qquad \begin{cases} |\uparrow> = \frac{1}{\sqrt{2}}|\rightarrow> + \frac{1}{\sqrt{2}}|\leftarrow> \\ |\downarrow> = \frac{1}{\sqrt{2}}|\rightarrow> - \frac{1}{\sqrt{2}}|\leftarrow> \end{cases}$$

We can now rewrite Eq. 1 in terms of the right-left (↔) basis to check that (just as with the up-down basis) any quantum state can be expressed as a linear combination of the two basis states:

$$|\theta> = \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{pmatrix} = \left(\frac{\cos\frac{\theta}{2} + \sin\frac{\theta}{2}}{\sqrt{2}}\right)\begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} + \left(\frac{\cos\frac{\theta}{2} - \sin\frac{\theta}{2}}{\sqrt{2}}\right)\begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \text{ , } 0 \le \theta < 2\pi \qquad \text{(Eq. 4)}$$

As we will see later, the following convention will be used to represent bit values as quantum states in the up-down basis and the right-left basis:

Basis	**0**	**1**
↕ (*up-down*)	|↑>	|↓>
↔ (*right-left*)	|→>	|←>

[3] *Bra* vectors are represented as row vectors.

[4] The *bracket* $<\varphi|\theta>$ is the inner product of $<\varphi|$ and $|\theta>$ (the state expected by Bob and the state prepared by Alice).

Communication between Alice and Bob

When Alice sends information encoded as quantum states and Bob receives and decodes it as explained, the next table gives a summary of the possible cases that can arise:

Alice			Bob			
Value to send	Encoding basis	Q. state sent	Measuring basis	Q. state read	Read value	Conditional probability
0	↕	|↑>	↕	|↑>	0	100%
			↔	|→>	0	50%
				|←>	1	50%
	↔	|→>	↕	|↑>	0	50%
				|↓>	1	50%
			↔	|→>	0	100%
1	↕	|↓>	↕	|↓>	1	100%
			↔	|→>	0	50%
				|←>	1	50%
	↔	|←>	↕	|↑>	0	50%
				|↓>	1	50%
			↔	|←>	1	100%

For example, let us say Alice wants to send a “0” value. She chooses the ↕ encoding basis, then she encodes “0” as |↑>. Now, the value that is read by Bob depends not only on the qubit sent by Alice, but also on the basis Bob uses to measure it. If Bob measures along the ↕ basis, then the chances are 100% that he will read |↑>, which is decoded as “0” (because P(↑ | ↑) = $| <\uparrow|\uparrow> |^2$ = $|\cos\frac{\pi}{4}\cos\frac{\pi}{4}+\sin\frac{\pi}{4}\sin\frac{\pi}{4}|^2$ = 1, while P(↓ | ↑) = $| <\downarrow|\uparrow> |^2$ = $|\cos\frac{3\pi}{4}\cos\frac{\pi}{4}+\sin\frac{3\pi}{4}\sin\frac{\pi}{4}|^2$ = 0).

On the other hand, if Bob chooses to measure along the ↔ basis, then he will read |→> or |←> randomly, with probabilities split at 50% (as a consequence of the orthogonality of the bases we have chosen: P(→|↑) = $| <\rightarrow|\uparrow> |^2$ = $|\cos\frac{\pi}{2}\cos\frac{\pi}{4}+\sin\frac{\pi}{2}\sin\frac{\pi}{4}|^2$ = 0.5, and P(←|↑) = $| <\leftarrow|\uparrow> |^2$ = $|\cos\frac{0}{2}\cos\frac{\pi}{4}+\sin\frac{0}{2}\sin\frac{\pi}{4}|^2$ = 0.5).

It is important to note in which way quantum indeterminacy is at work here. If Bob does not know which basis was used by Alice for encoding, then he has to choose a basis at random for measuring. In this last example, if he chooses the wrong basis (↔) he will read either |→> or |←>, which means his act of measuring has changed the state of the quantum system; if he later resent the data to a third person, he would be sending the state as he read it, and not as Alice sent it in the first place (the state Alice sent would not exist anymore!).

As we will see in the following sections, this property is what guarantees the security of key distribution using the BB84 and B92 quantum protocols.

2.3 The BB84 protocol

BB84 is the first quantum cryptography protocol, developed by Charles Bennett and Giles Brassard in 1984 [3]. It is a prepare and measure protocol, using discrete variable coding.[5] Its most interesting feature is that it is provably secure. It is usually applied for securely communicating a private key for use in one-time pad encryption (a form of Vernam cipher which provides perfect secrecy).

There are many possible variations of BB84. We will try to describe it in a very simple way.

Alice (the sender) and Bob (the receiver) are connected by a one-way quantum communication channel and a two-way classical communication channel. Neither of these channels needs to be secure.

Alice will use 2 sequences (*a* and *b*) of random bits, while Bob will use 1 sequence of random bits (*b’*).[6]

Alice and Bob have agreed beforehand (through classical communication) on the following convention for encoding and decoding bits into and from quantum states:

Basis	**0**	**1**
0 = ↕	|↑>	|↓>
1 = ↔	|→>	|←>

Alice first encodes *a* as a sequence of quantum states, using *b* to determine the encoding basis for each bit in *a*. She then transmits the quantum states over the quantum channel. Bob decodes every quantum state he receives, choosing every time the basis for decoding according to *b’*, and stores the bits in a new sequence *a’*. Note that, at this point, *b* is only known to Alice, and so Bob has no way of knowing if he is measuring incoming quantum states in the same basis in which they were encoded by Alice – he is doing it randomly.[7] The good thing is, if Eve (an eavesdropper) interfered, she would be in exactly the same situation as Bob, and that is why Alice and Bob would later be able to detect her eavesdropping.

After Bob has received and measured all the quantum states sent by Alice, Alice communicates her encoding sequence (*b*) through the classical channel, and Bob communicates his decoding sequence (*b’*). Alice and Bob compare *b* and *b’*, and they discard those bits in *a* and *a’* for which the encoding and decoding modes did not match.[8] The bits left from *a* and *a’* after the discarding process form the sequences *c* (on Alice’s side) and *c’* (on Bob’s side). On average, 50% of the bits from *a* and *a’* will be kept in *c* and *c’*.

Now, if there was no eavesdropper between Alice and Bob, *c* and *c’* would be exactly the same. Bob decoded each bit of *c’* in the same basis Alice used to encode the ones in *c*, and thus their values would be the same.

[5] *Prepare and measure* protocols exploit the property of quantum indeterminacy in order to detect any eavesdropping on communication, as opposed to *entanglement based* protocols, which exploit the properties of entangled quantum states. As to *discrete variable* coding, it is the first of three families of protocols, the other two being *continuous variable* and *distributed phase reference* coding.

[6] The lengths of *a*, *b* and *b’* will be equal, and chosen before communication based on the number of bits desired for the final shared key (only a part of the bits will be shared even if communication is successful).

[7] As a result, the bits in *a* and *a’* will agree with a probability of 75% (50% from quantum states measured along the right basis + 25% from quantum states measured along the wrong basis but randomly resulting in the right value).

[8] The discarded bits from *a* and *a’* had a 50% probability of being in disagreement (the decoding basis was wrong, but there was still a 50% chance of reading the right value).

On the other hand, if Eve tried to eavesdrop on communications between Alice and Bob, Eve’s measurements would introduce changes in the quantum states that Bob finally received on the other end of the line (*c* and *c’* would then differ).

Eve would have access to both communication channels and would use 1 sequence of random bits, just like Bob. Eve’s *modus operandi* would consist in receiving, measuring, re-encoding and resending every quantum state sent by Alice. But, for each quantum state sent by Alice, Eve would have a 50% chance of decoding it in the wrong basis; then, if Bob measured the intercepted state in the same basis Alice used in the first place he would have a 50% chance of obtaining the same value Alice sent. The probability of Eve’s eavesdropping introducing mismatches between the bits in *c* and *c’*, then, would be 25%.

Thus Alice and Bob, after determining *c* and *c’*, of 2*n* bits each, conduct one further check to make sure no eavesdropping took place. Alice and Bob select and compare *n* bits from their *c* and *c’* sequences. The probability that they find a disagreement introduced by Eve’s actions is $P_d = 1 – (3/4)^n$. Then, to detect an eavesdropper with probability $P_d = 0.99999999$ Alice and Bob need to compare n=72 bits (a figure that has to be taken into account when deciding the length of the original sequence *a* sent by Alice). If they find any disagreement in the comparison, they abort and start over, possibly through a different quantum channel.[9]

If no eavesdropping is detected, then the remaining *n* bits in *c* and *c’* constitute the secret shared key between Alice and Bob (for use in a traditional private key protocol).

Basis	**0**	**1**
0 = ↕	|↑>	|↓>
1 = ↔	|→>	|←>

Let’s have a look at an example of quantum key distribution according to BB84, in the case were no eavesdropping takes place. In this example, a sequence of 15 bits is originally transmitted:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alice															
Original sequence (*a*)	1	1	0	1	1	1	0	1	1	0	0	1	0	0	0
Mode for encoding (*b*)	1	0	1	0	1	1	0	1	0	0	0	1	1	1	0
Transmitted state	|←>	|↓>	|→>	|↓>	|←>	|←>	|↑>	|←>	|↓>	|↑>	|↑>	|←>	|→>	|→>	|↑>
Bob															
Mode for decoding (*b’*)	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1
Observed state	|←>	|←>	|→>	|↓>	|←>	|↓>	|↑>	|↑>	|→>	|←>	|→>	|←>	|→>	|↓>	|←>
Reconstructed sequence (*a’*)	1	1	0	1	1	1	0	0	0	1	0	1	0	1	1

i		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$b_i \neq b_i$’ ?			*				*		*	*	*	*			*	*
Alice	*c*	1		0	1	1		0					1	0		
Bob	*c’*	1		0	1	1		0					1	0		

[9] If Alice and Bob use *information reconciliation* and *privacy amplification*, techniques which were introduced in 1992 [6], they can tolerate a disagreement rate of less than 20% between the *n* bits compared. If the disagreement rate does not reach 20% during the check, then they can apply information reconciliation and privacy amplification to obtain *m* secret shared key bits from the remaining *n* bits in *c* and *c’*.

After Bob has received and measured all the quantum states sent by Alice, Alice communicates her encoding sequence (*b*) through the classical channel. Alice and Bob compare *b* and *b'*, and they discard those bits in *a* and *a'* for which the encoding and decoding modes did not match. As no one interfered in communication over the quantum channel, the resulting sequences *c* and *c'* are exactly the same (1011010). But Alice and Bob cannot be sure of that, so they choose randomly half of the bits in *c* and *c'* to conduct a value comparison. After verifying that all chosen values coincide, they know they can share the rest of the bits in *c* and *c'* as their secret key (for use in a traditional private key protocol).

Let's now analyze what happens when Eve tries to eavesdrop on the communications between Alice and Bob:

Basis	**0**	**1**
0 = ↕	|↑>	|↓>
1 = ↔	|→>	|←>

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alice															
Original sequence (*a*)	1	1	0	1	1	1	0	1	1	0	0	1	0	0	0
Mode for encoding (*b*)	1	0	1	0	1	1	0	1	0	0	0	1	1	1	0
Transmitted state	|←>	|↓>	|→>	|↓>	|←>	|←>	|↑>	|←>	|↓>	|↑>	|↑>	|←>	|→>	|→>	|↑>
Eve															
Mode for decoding	1	0	0	0	0	0	1	1	0	1	1	1	1	0	0
Observed (and resent) state	|←>	|↓>	|↓>	|↓>	|↑>	|↓>	|←>	|←>	|↓>	|←>	|→>	|←>	|→>	|↓>	|↑>
Reconstructed sequence	1	1	1	1	0	1	1	1	1	1	0	1	0	1	0
Bob															
Mode for decoding (*b'*)	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1
Observed state	|←>	|→>	|→>	|↓>	|→>	|↓>	|↑>	|↑>	|←>	|←>	|→>	|←>	|→>	|↓>	|→>
Reconstructed sequence (*a'*)	1	0	0	1	0	1	0	0	1	1	0	1	0	1	0

	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$b_i \neq b_i$' ?			*				*		*	*	*	*			*	*
Alice	*c*	1		0	1	1		0					1	0		
Bob	*c'*	1		0	1	**0**		0					1	0		

Eve receives, measures, re-encodes and resends all the quantum states sent by Alice. This process introduces errors in Bob's measurements.

Just like before, after Bob has received and measured all the quantum states sent by Alice (and tampered by Eve), they compare *b* and *b'* over the public channel and discard all bits in *a* and *a'* for which the encoding and decoding method didn't coincide. They thus obtain the sequences *c* and *c'*, which would have matched 100% if it were not for Eve's interference (in this example 1 error was introduced in Bob´s measurements: c_5' should have been 1, but has been read as 0).

To detect any possible eavesdropping, Alice and Bob randomly select and compare half of the bits from their *c* and *c'* sequences. For long enough shared sequences, they will have an almost 100% probability of finding the disagreements between them caused by Eve, which will evidence that eavesdropping has taken place. That is enough for Alice and Bob to know they have to abort.

2.4 The B92 protocol

The B92 protocol was developed by Bennett in 1992 as a simplification of BB84 [4]. The most important refinement introduced in B92 is that only 2 states are used for encoding and 2 for decoding. Thus an encoding/decoding convention for B92 might look like this:

	0	**1**
Alice (encoding)	|↑>	|←>
Bob (decoding)	|→>	|↓>

As a result, encoding becomes immediate for Alice. She does not need to choose the encoding basis at random for each bit anymore.

Alice will use only 1 sequence (*a*) of random bits. Bob will still use 1 sequence of random bits (*b’*).

Alice and Bob are still connected by a one-way quantum communication channel and a two-way classical communication channel.

Alice first encodes *a* as a sequence of quantum states, as indicated in the previous table. She then transmits the quantum states over the quantum channel. Bob decodes every quantum state he receives, choosing every time the basis for decoding according to *b’*, and stores the bits in a new sequence *a’*. In B92, the following situations can arise during Bob’s measurement of each quantum state received from Alice:

Alice		**Bob**					
Q. state sent	Prob.	Measuring basis	Conditional probability	Q. state read	Conditional probability	Value	Absolute probability
|↑> = 0	50%	0 = ↕	50%	|↑>	100%	*Erasure*	25%
		1 = ↔	50%	|→>	50%	0	12.5%
				|←>	50%	*Erasure*	12.5%
|←> = 1	50%	0 = ↕	50%	|↑>	50%	*Erasure*	12.5%
				|↓>	50%	1	12.5%
		1 = ↔	50%	|←>	100%	*Erasure*	25%

As seen in the table, only in 2 cases Bob accepts the read value: when he measures the quantum states he was expecting (|→> for 0 or |↓> for 1); in these 2 cases he can be sure that the value he has read is exactly the one Alice encoded and sent in the first place. On the other hand, if the quantum states he measures are |↑> or |←>, then he cannot be sure which value did Alice send originally. This last situation is called an *erasure*, and has a 75% absolute probability of happening, which means that in B92, on average, 75% of the original sequence transmitted by Alice will be discarded from the beginning.

Evidently, in B92 Alice and Bob do not need to compare the bases they used for encoding and decoding. Instead, after Bob has received and measured all the quantum states sent by Alice, he uses the classical channel to communicate which bits were non-erasures, and they both discard all other bits (all erasures). The remaining bits form the sequences *c* (on Alice’s side) and *c’* (on Bob’s side). The length of *c* and *c’* will be, on average, 25% of the length of the original transmitted sequence *a*.

If there was no eavesdropper between Alice and Bob, *c* and *c'* would be exactly the same. That being so, they would be able to securely use *c* and *c'* as their secret key (for use in a traditional private key protocol).

However, if Eve tried to eavesdrop on communications between Alice and Bob, Eve's measurements would introduce changes in the quantum states that Bob finally received on the other end of the line (*c* and *c'* would then differ).

Just as in BB84, Eve would have access to both communication channels and would use 1 sequence of random bits, like Bob. Eve would receive, measure, re-encode and resend every quantum state sent by Alice. But, like in BB84, Eve would have no way to know if she is measuring the quantum states sent by Alice along the right basis, and so the quantum states she would resend to Bob would not be the same ones Alice originally encoded. The non-erasures found by Bob would change from the ones he would have found without Eve's interference, and most importantly, in some of the non-erasures detected by Bob he would read a different value from the one originally sent by Alice.

Again, that allows Alice and Bob to detect any eavesdropping. To that end, after determining *c* and *c'*, Alice and Bob conduct one further check: they select and compare *n* bits from their *c* and *c'* sequences. If they find any disagreement in the comparison, they abort and start over, possibly through a different quantum channel.

If no eavesdropping is detected, then the remaining bits in *c* and *c'* constitute the secret shared key between Alice and Bob (for use in a traditional private key protocol).

Let's have a look at an example of quantum key distribution according to B92, in the case were no eavesdropping takes place. In this example, a sequence of 15 bits is originally transmitted:

	0	**1**
Alice	|↑>	|←>
Bob	|→>	|↓>

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alice															
Original sequence (*a*)	1	1	0	1	1	1	0	1	1	0	0	1	0	0	0
Transmitted state	|←>	|←>	|↑>	|←>	|←>	|←>	|↑>	|←>	|←>	|↑>	|↑>	|←>	|↑>	|↑>	|↑>
Bob															
Mode for decoding (*b'*)	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1
Observed state	|←>	|←>	|→>	|↑>	|←>	|↓>	|↑>	|↓>	|←>	|←>	|→>	|←>	|→>	|↑>	|←>
Erasure? (*e*)	*	*		*	*		*		*	*		*		*	*
Reconstructed sequence (*c'*)			0			1		1			0		0		

After Bob has received and measured all the quantum states sent by Alice, he uses the classical channel to communicate which bits were non-erasures, and they both discard all other bits (all erasures). As there was no eavesdropper between Alice and Bob, the remaining sequences *c* (on Alice's side) and *c'* are the same (01100). But Alice and Bob do not know that, so they still have to check that there was no eavesdropping. They randomly select half of the bits in *c* and *c'* to conduct a value comparison over the classical channel. After verifying that all chosen values coincide, they know they can use the rest of the bits in *c* and *c'* as their secret key (for use in a traditional private key protocol).

Let’s now analyze what happens when Eve tries to eavesdrop on the communications between Alice and Bob:

	0	**1**
Alice	|↑>	|←>
Bob	|→>	|↓>

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alice															
Original sequence (*a*)	1	1	0	1	1	1	0	1	1	0	0	1	0	0	0
Transmitted state	|←>	|←>	|↑>	|←>	|←>	|←>	|↑>	|←>	|←>	|↑>	|↑>	|←>	|↑>	|↑>	|↑>
Eve															
Mode for decoding	1	0	0	0	0	0	1	1	0	1	1	1	1	0	0
Observed (and resent) state	|←>	|↑>	|↑>	|↓>	|↑>	|↑>	|←>	|←>	|↓>	|→>	|←>	|←>	|→>	|↑>	|↑>
Erasure?	*	*	*		*	*	*	*			*	*		*	*
Reconstructed sequence				1					1	0			0		
Bob															
Mode for decoding (*b’*)	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1
Observed state	|←>	|←>	|←>	|↓>	|→>	|↑>	|↑>	|↓>	|→>	|→>	|←>	|←>	|→>	|↑>	|→>
Erasure? (*e*)	*	*	*			*	*				*	*		*	
Reconstructed sequence (*c’*)				1	**0**			1	**0**	0			0		0

Eve receives, measures, re-encodes and resends all the quantum states sent by Alice. As in BB84, this process introduces errors in Bob’s measurements.

After Bob has received and measured all the quantum states sent by Alice (and tampered by Eve), he uses the classical channel to communicate which bits were non-erasures. Keeping only those bits, they obtain the sequences *c* (on Alice’s side) and *c’*, which would have matched 100% if it were not for Eve’s interference (in this example Eve’s eavesdropping has introduced 2 errors in Bob’s measurements: c_5’ and c_9’ should have been 1, but have been read as 0).

To detect any possible eavesdropping, Alice and Bob randomly select and compare half of the bits from their *c* and *c’* sequences. For long enough shared sequences, they will have an almost 100% probability of finding the disagreements between them caused by Eve, which will evidence that eavesdropping has taken place. That is enough for Alice and Bob to know they have to abort.

2.5 Comparison between the BB84 and the B92 protocols

The B92 protocol very much follows the method established in BB84 (indeed, BB92 was conceived by one of the creators of BB84). Both are prepare-and-measure quantum key distribution protocols, which work following similar steps, and which achieve the same result: provable security, through the fact that any eavesdropping is inevitably detected.

As such, both protocols are susceptible of being implemented and used for secure key distribution. Both are effective in that respect.

B92 can be considered a refinement of BB84 in the fact that encoding the initial random bit sequence *a* into qubits becomes immediate for Alice, using only 2 states, and less initial data is required (Alice only needs 1 random bit sequence in B92, as opposed to 2 in BB84). Also, B92 avoids the step in which Alice and Bob compare the bases they used for encoding and decoding (*b* and *b'*); in B92, Bob can communicate directly to Alice which bits were erasures and thus can be discarded by both of them to obtain *c* and *c'*. From this point of view, B92 is simpler and more efficient in its steps.

On the other side, if we examine the efficiency of transmission, B92 does not come out as better than BB84. As we already explained, in BB84, on average, 50% of the bits sent initially by Alice are measured along the right basis by Bob, and so kept in *c'*; then, if 50% of those correctly measured bits are compared to check for eavesdropping, the resulting shared key will have an average of 25% of the length of the original *a* sequence transmitted by Alice. In contrast, in BB92, just from the beginning, 75% (on average) of the bits sent by Alice are discarded by Bob as erasures; and then, if 50% of the remaining bits are compared to check for eavesdropping, the resulting shared key will be on average 12.5% of the length of the original *a* sequence transmitted by Alice.

In the following table we can compare how both protocols perform if we want to obtain a 128-bit shared key:

Protocol	Length of ***a, a', b'*** (also *b* in BB84)	Bits **discarded from *a, a'*** (average)	Length of ***c, c'***	Bits **used from *c, c'*** to check for eavesdropping	Length of the **secure key** obtained and shared by Alice and Bob	Transmission efficiency rate
BB84	512 bits	**50%**	256 bits	50%	128 bits	**1/4**
B92	1024 bits	**75%**	256 bits	50%	128 bits	**1/8**

As we can see, even if both protocols guarantee the security of key distribution, B92 is more refined in its method, while BB84 is more efficient in the transmission of data. The convenience of choosing one protocol over the other will be determined by the constraints of each particular case.

2.6 Application of quantum cryptography

Most practical implementations of quantum key distribution protocols are based on single photons transmitted over optical fibre [7] (as mentioned in section 2, single photons are a two-level quantum system that can be used to physically represent qubits).

In choosing the source, the detectors and the actual optical fibre to be used, the most important factor is the wavelength to be used; two main possibilities thus arise. The first option is to use commercially available single photon counters, which operate on a wavelength range of around 800 nm; the second one is to use a wavelength compatible with standard telecommunications optical fibres, i.e., 1300 nm or 1550 nm. The choice to operate with existing single photon counters requires the use of special fibres, which would prevent the use of already installed telecommunications networks. On the other hand, the choice of a wavelength suitable for today’s optical fibres requires the development of detectors for 1300 nm or 1550 nm.

When using single photons to encode classical information as qubits, different polarization states can be used as the bases for BB84 and B92 [2]. We will take as a reference the 4 states we used to explain the protocols in sections 3 and 4, that is: |→> and |←> (from the right-left ↔ basis); plus |↑> and |↓> (from the up-down ↕ basis). Linear polarization can be used as one of the bases (whose two states are still represented as |↑> and |↓>), while circular polarization can be the second basis (whose two states would be |↻> for |→> and |↺> for |←>). Our conventions for encoding and decoding would then become as follows:

BB84		
Basis	**0**	**1**
0 = Linear polarization	|↑>	|↓>
1 = Circular polarization	|↻>	|↺>

B92		
	0	**1**
Alice (encoding)	|↑>	|↺>
Bob (decoding)	|↻>	|↓>

A diode laser can be used to generate the photons in the chosen wavelength, polarized in one of the 4 possible states. As single photon states are difficult to realize in practice, faint laser pulses are used to produce approximately single photon states (“approximately” single photons will require some extra measures for such an implementation to work in practice). Photons then travel through optical fibre, and they are received by a single-photon polarization analyzer.

Thus we have a quantum channel for the BB84 or B92 protocols. We just need to complement it with a classical channel, such as the Internet, and a trivial implementation of the steps of the protocol as an algorithm to be run in a classical computer, and so we have all we need for a practical implementation of quantum cryptography.

Still one more option has been tried in physical implementations of BB84 and B92: free space transmission of the photons, instead of the use of optical fibre. Free space transmission is restricted to line-of-sight links, but eliminates the need for a fibre-optics infrastructure to be in place. In free space transmission, the choice of wavelength is easy, as the region were good photon detectors exist, around 800 nm, is at the same time the wavelength were absorption is lower, which makes it ideal for transmission [8]. A free-space link, then, is another viable option for transmission; coupled with the rest of the elements explained before, it allows for the implementation of secure quantum cryptography.

2.7 Closing comments

Quantum cryptography (quantum key distribution) protocols are one of the most active and fruitful areas in the field of quantum computing, both in the theoretical and practical sides. Since the introduction of the first ideas on quantum cryptography by Wiesner in the 1960s, many protocols have been described which take advantage of the properties of quantum mechanics to ensure provably secure communication. Not only that, but also many successful experiments have demonstrated the practical applicability of quantum key distribution (QKD) protocols to real situations.

Among QKD protocols, BB84 and B92 are two of the most representative. As we have seen, they are simple, elegant, easily described and understood. We have confirmed how they use the properties of quantum indeterminacy to ensure the security of the key distribution. And we have shown how they work when electronic spin is used to represent qubits, in a way that is extensible to any other quantum state representation. Also on the theoretical side, we have confirmed that, while B92 is in some ways more refined, BB84 still makes a more efficient use of the data that is transmitted.

In the last section, we have described how BB84 and B92 can be implemented through components available today. Single photons transmitted over optical fibre have been chosen to represent quantum states in our description; it is a very fruitful method which continues to be used in a majority of the experiments on QKD with very promising results. For more information on the latest results in quantum cryptography, both from the theoretical and the practical perspectives, see [8] and [9].

Econophysics

Some statistical properties of the Lévy distribution

3.1 Introduction

Econophysics is the application of theories and methods from the field of physics to solve problems in economics. The main area in this interdisciplinary research field is that of *statistical finance*, that is, the application of methods from statistical physics to the study of financial markets, especially in problems including uncertainty or stochastic processes and nonlinear dynamics.

Concepts from the field of physics such as power-law distributions, correlations, scaling, unpredictable time series and random processes can be applied to financial markets. Indeed, the first use of a *power-law distribution* took place in the field of economics, one century ago, when Pareto used the distribution $y \sim x^{-\nu}$ to model the wealth of individuals in a stable economy [10].

The concept of *random walk* was also first applied to economics, specifically to the pricing of options in speculative markets (a very relevant issue today) [11]. The Black & Scholes option-pricing model —the milestone in option-pricing theory— was only published decades later, and it still needs correction in its application. Thus, the problem of which stochastic process describes the changes in the logarithm of prices in a financial market is still an open one.

The problem of the distribution of price changes has been object of research since the 1950s. Bachelier originally proposed a Gaussian distribution model for price changes, which was replaced by a log-normal distribution model (geometric Brownian motion). The latter model, however, only provides a first approximation of what is observed in real data. Therefore, various alternative models have been proposed, the most revolutionary of which being Mandelbrot’s hypothesis that price changes follow a *Lévy stable distribution* [12]. Distributions of this kind, however, are difficult to use for modelling.

Regarding time series of asset prices, it is widely accepted that they are unpredictable. Thus, *stochastic processes* (which represent the evolution of random variables over time) are usually applied for the description of price dynamics. While it cannot be ignored that unpredictable time series can come from deterministic nonlinear systems, and so financial markets might follow *chaotic dynamics*, most research is being conducted assuming that price dynamics are *stochastic processes*.

Financial markets exhibit several of the properties that characterize *complex systems*. Nowadays it is possible to develop models and to test their accuracy and predictive power using data available from large databases. Recently, methods from the field of physics are increasingly used to analyze economic systems. This research activity is complementary to the traditional approaches of finance and mathematical finance. A new emphasis is put on the *empirical analysis* of economic data, bringing to the subject the background of theory and method of statistical physics (which include applicable concepts such as scaling, universality, disordered frustrated systems and self-organized systems).

Among the most important areas in econophysics research, one concerns the complete statistical characterization of the stochastic process of price changes of a financial asset. A second area concerns the development of a theoretical model that is able to encompass all the essential features of real financial markets. But this new discipline is being applied in many other areas of economics research.

The Lévy distribution in econophysics

As we already mentioned, the first use of a *power-law distribution* took place in the field of economics, when Pareto modelled the wealth of individuals in a stable economy by using the distribution $y \sim x^{-v}$, where y is the number of people having income x or greater than x, and v is the *Pareto exponent* (which Pareto estimated to be 1.5). But power-law distributions, characterised by their long tails, are in some way counterintuitive, because they lack any characteristic scale. Only the recent emergence of new paradigms has brought to their application for this kind of statistical modelling.

Thus, power-law distributions have found their way into statistical finance. To model the distribution of price changes, Bachelier originally proposed a Gaussian distribution model for price changes, which was replaced by a log-normal distribution model (geometric Brownian motion). However, these models (not based on power-law distributions) only provide a first approximation of what is observed in real data; specifically, empirical evidence shows that the tails of measured distributions are fatter than expected for a geometric Brownian motion. That is why recent models are based on Mandelbrot's hypothesis that price changes follow a *Lévy stable distribution*. Lévy stable processes are stochastic processes obeying a generalized central limit theorem. In Lévy processes, the sum of independent identically distributed stochastic processes $S_n \equiv \Sigma^n{}_{i=1}\, x^i$ characterized by a probability density function with power-law tails $P(x) \sim x^{-(1+\alpha)}$ will converge to a Lévy stochastic process of index α when n tends to infinity. That means the distribution of a Lévy stable process is a power-law distribution for large values of the stochastic variable x.

Different values for the index α give us some special cases. The Lévy distribution with $\alpha = 2$ is the *Gaussian (normal) distribution*. When $\alpha = 1$, we obtain the long-tailed *Lorentzian (Cauchy-Lorentz) distribution*. Among the different Lévy distributions in the range $1 <= \alpha <= 2$, only the Gaussian distribution ($\alpha = 2$) has finite variance. All the rest ($\alpha < 2$) have infinite variance (stochastic processes with infinite variance are especially difficult to use).

One of the most important features of the Gaussian distribution is its being an *attractor* in terms of the Central Limit Theorem; that is to say, stochastic processes tend to a Gaussian distribution. But non-Gaussian stable distributions are also attractors. All Lévy distributions, including the Lorentzian distribution, have an associated limit theorem by which a sum of independent random variables can converge to them [13].

So, if the distribution of price changes cannot fit a Gaussian distribution, because empirical distributions feature fatter tails than the Gaussian distribution, then more fitting models can be formulated based on other Lévy distributions.

In the following pages we will study some properties of these statistical distributions that are frequently applied for economical and financial modelling, namely the family of Lévy stable distributions. We will focus especially on the Gaussian (normal) distribution, the only one with finite variance, and on the Lorentzian (Cauchy-Lorentz) distribution, a notorious representative of stable distributions with infinite variance.

3.2 Random walk. Generation of sets according to specific probability distributions.

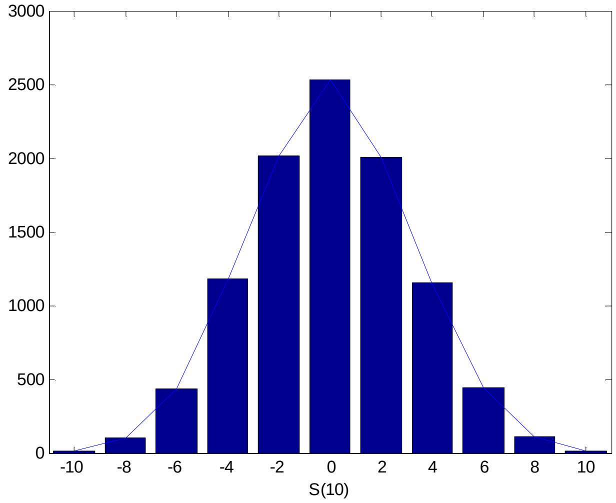
The concept of *random walk* was first applied to economics, to model the pricing of options in speculative markets. The evolution of prices in an ideal *efficient market* can be modelled as a stochastic process [1]. Under the efficient market hypothesis, the evolution of prices behaves approximately like uncorrelated random walks [10].

While real markets are not ideally efficient, the use of an idealized system (as is usual in physics) is instrumental to develop theories and models and to perform empirical tests. The validity of the results obtained in this way must always be analyzed taking into account that the idealized efficient market only approximates the real market.

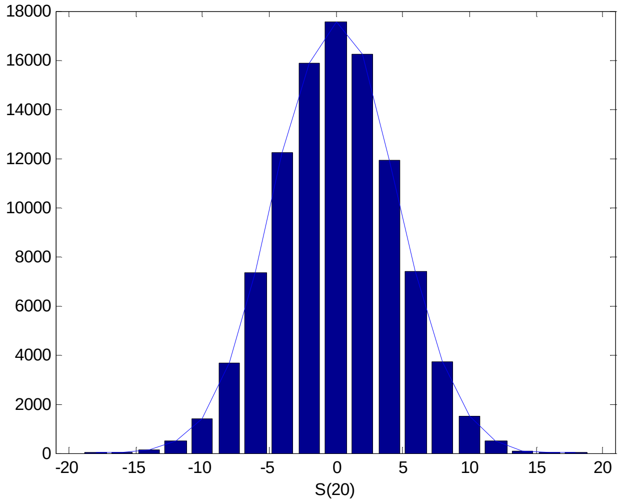
To better understand random walks, a Matlab program was created to generate (with graphic output) a distribution function P(S_n) where S_n = x_1 + x_2 + … + x_n and each x_i is a two-valued uniform random number x_i = s or –s in equal probability (n:integer and s:positive real are two inputs).

As expected in random walks, when the number of steps $n \to \infty$, the random walk tends to follow the Gaussian distribution.

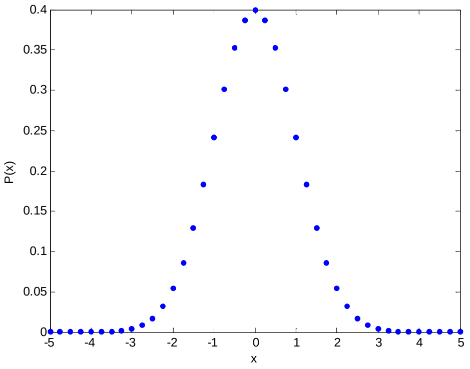
```
>histog2(n=10, s=1, iter=10000)    %P(S_n) distribution function generator
```



```
>histog2(n=20, s=1, iter=100000)
```



(follows
Gaussian distribution)



[1] Specifically, stochastic processes applied to the evolution of prices are of the kind known as *martingales* (in which the conditional expected value of an observation at some time *t*, given all the observations up to some earlier time *s*, is equal to the observation at that earlier time *s*). For more information on martingales in econophysics, see [10] and [11].

Generation of data sets according to a specific probability distribution function

Three programs where created to generate random numbers according to specific distribution functions f(x): P(x) = f(x). The three distribution functions are: Gaussian, Lorentzian, and Lévy. Gaussian and Lorentzian distributions can then be generated through their specific programs, or through the Lévy distribution program (setting α at values 1 or 2). The Lévy program is shown as an example.

The method used for random number generation according to a given distribution function was the *random deletion method*.

For a distribution F(x) in the range $x \in [a, b[$,

- *Choose a real number *c* such that *c* >= F(x), $x \in [a, b[$
- *Prepare a candidate random number *x* from a uniform distribution in the range [a, b[; P(x) = 1-(b/a)
- *Prepare a uniform random number *u* in the range [0, *c*[
- *If *u* <= F(x), accept *x*, otherwise reject.

The set of accepted *x* follows the distribution P(x) = F(x).

```
function x=g_gen(xmin,xmax)
% Gaussian random number generator
```

$$PDF : \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2} \right)$$

```
function x=lor_gen(gamma,xmin,xmax)
% Lorentzian random number generator
```

$$PDF : \frac{\gamma}{\pi} \frac{1}{x^2 + \gamma^2}$$

```
function x=levy_gen(a,xmin,xmax)
% Lévy random number generator
% Usage: x=levy_gen(alpha,xmin,xmax)
% default: alpha=1, xmin=-5, xmax=5

    if nargin < 1, a=1; end
    if nargin < 2, xmin=-5; xmax=5; end

    C = 0.4; %C >= F(x) for every x in [a,b)

    accept = 0; %0=not accepted
    while accept == 0
        x = rand*(xmax-xmin)+xmin;
        u = rand*C;

        y = (1/pi) * quad(@pp,0,10); %PDF

        if u <= y
            accept = 1;
        end
    end

    function y = pp(q) %nested function (used for quad)
        y = exp(-(q.^a)).*cos(q.*x);
    end
end
```

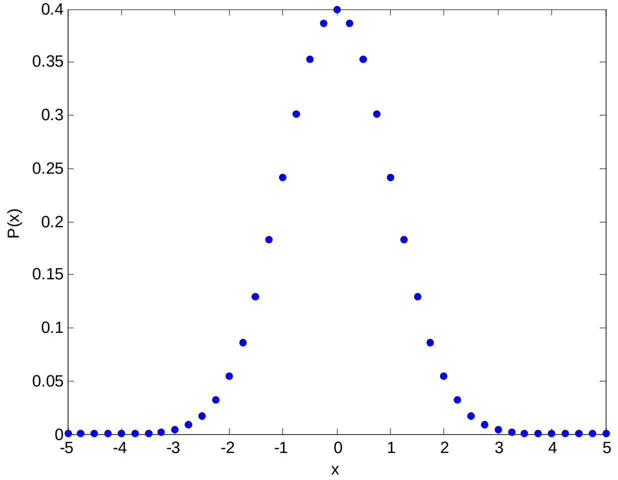
$$PDF : p^{(\gamma)}(x) = \frac{1}{\pi} \int_0^\infty \varphi_\alpha(q) \cos(qx) dq ,$$

$$\varphi_\alpha(q) = e^{-\gamma |q|^\alpha}$$

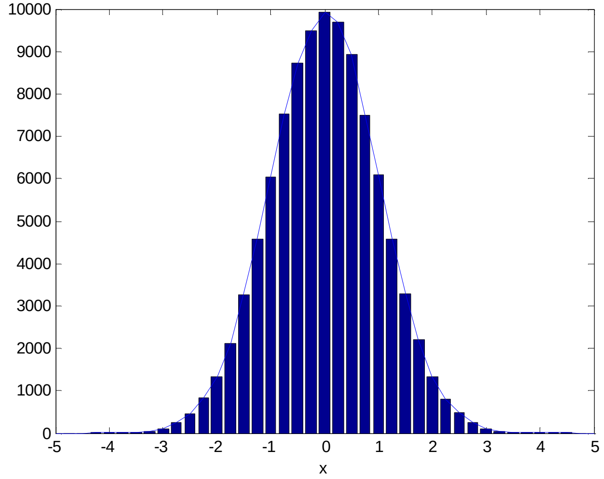
In the following graphs, sets of random data generated using these three programs (Gaussian, Lorentzian and Lévy) are graphically compared to the theoretical graph drawn directly from each probability distribution function (PDF). Note the increasingly fat tails of Lévy distributions when we follow the reduction of parameter α from α = 2 (Gaussian) to α = 1 (Lorentzian).

Gaussian distribution:

`>g_dist` *[Probability Distribution Function]*



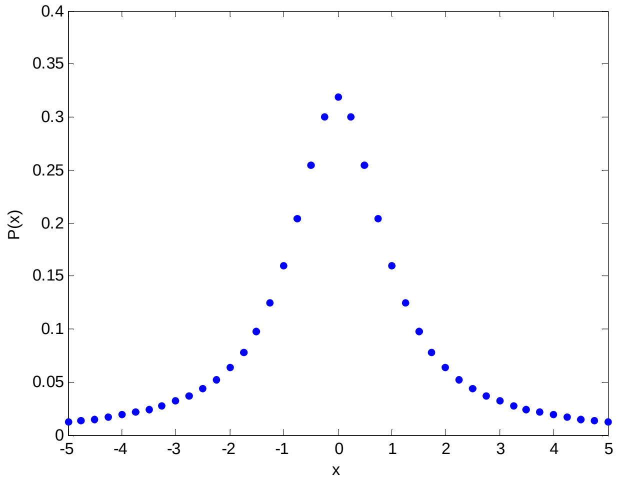
`>g_histog(iter=100000)` *[Generated random data]*



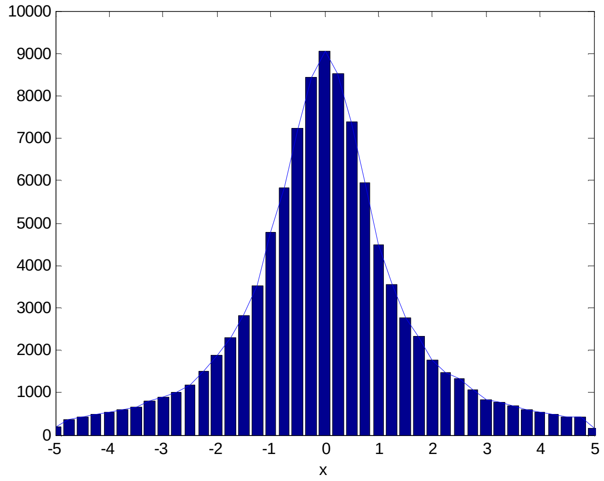
$$PDF:\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$$

Lorentzian distribution (γ=1):

`>lor_dist` *[Probability Distribution Function]*



`>lor_histog(iter=100000)` *[Generated random data]*



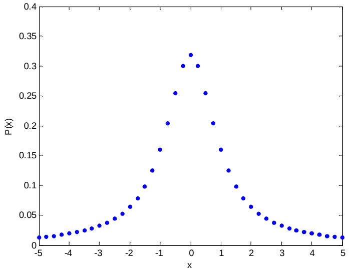
$$PDF:\frac{\gamma}{\pi}\frac{1}{x^2+\gamma^2}$$

Lévy distribution:

[Probability Distribution Function]

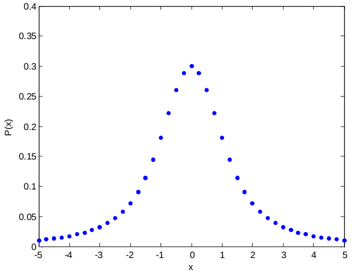
$$PDF: p^{(\gamma)}(x) = \frac{1}{\pi}\int_0^{\infty} \varphi_\alpha(q)\cos(qx)dq\,,\ \varphi_\alpha(q) = e^{-\gamma|q|^\alpha}$$

```
>levy_dist(alpha=1)
```

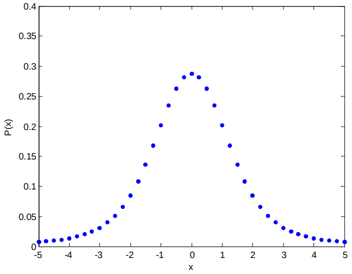


(Lorentzian)

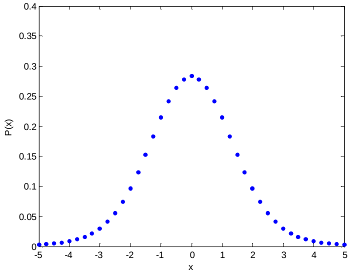
```
>levy_dist(alpha=1.2)
```



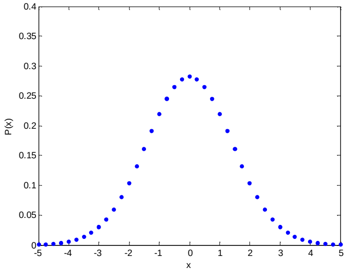
```
>levy_dist(alpha=1.5)
```



```
>levy_dist(alpha=1.8)
```



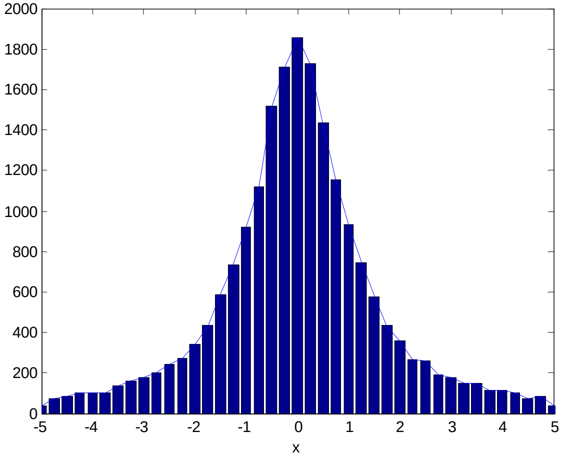
```
>levy_dist(alpha=2)
```



(Gaussian)

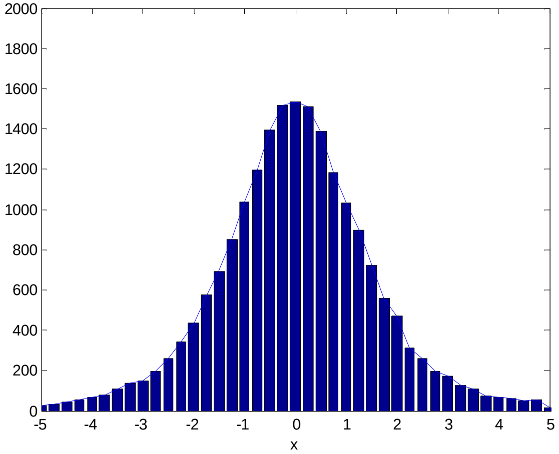
[Generated random data]

```
>levy_histog(alpha=1, iter=20000)
```

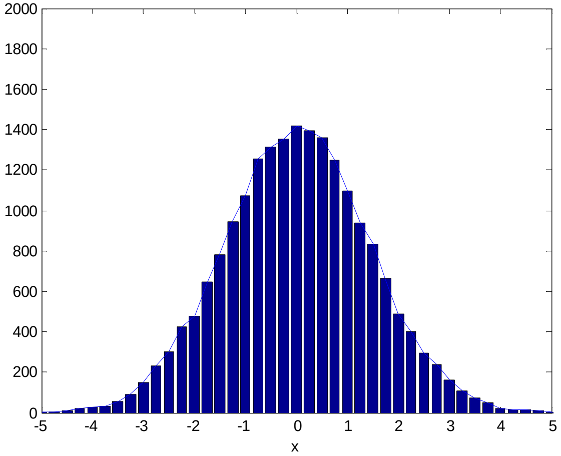


(Lorentzian)

```
>levy_histog(alpha=1.5, iter=20000)
```



```
>levy_histog(alpha=2, iter=20000)
```



(Gaussian)

3.3 Additive stability of probability distributions. Rescaling.

All Lévy distributions are stable. The most important property of stable distributions for econophysics is that all of them are *attractors* for sums of independent and identically distributed random variables. We have designed a set of Matlab programs to study graphically the stability properties of Lévy distributions.

To begin with, we examine the additive stability of the Gaussian distribution (the stability of other Lévy distributions will be studied in the following section). We define additive stability in the following terms:

> *Additive stability*: Suppose two random numbers {x} and {y} have identical distributions P(x) = f(x); P(y) = f(y). When the sum of the elements z = x + y have essentially the same distribution under f after proper *rescaling,* P(z) = A f(Bz) for some A and B (constants), the distribution f is called stable against addition of elements.

Rescaling is necessary to compare the distribution of the original sets and the added set and see if they are indeed the same. Thus we have implemented functions to rescale sets of data P(S_n) to P(T_n), where mean=0 and variance=1.

For Gaussian distribution: rescaling with the <u>square average</u> method

```
a = 1 / sqrt( mean(X.^2) - mean(X)^2 );
b = -mean(X) * a;

X = a*X + b;
```

For Lorentzian and Lévy (non-Gaussian) distributions: (in which `var(X) = ∞`)

a) rescaling with the <u>average of absolute value</u> method

```
    a = 1 / mean( abs(X) );
    X = a*X;
```

b) rescaling with the <u>half-width</u> methd

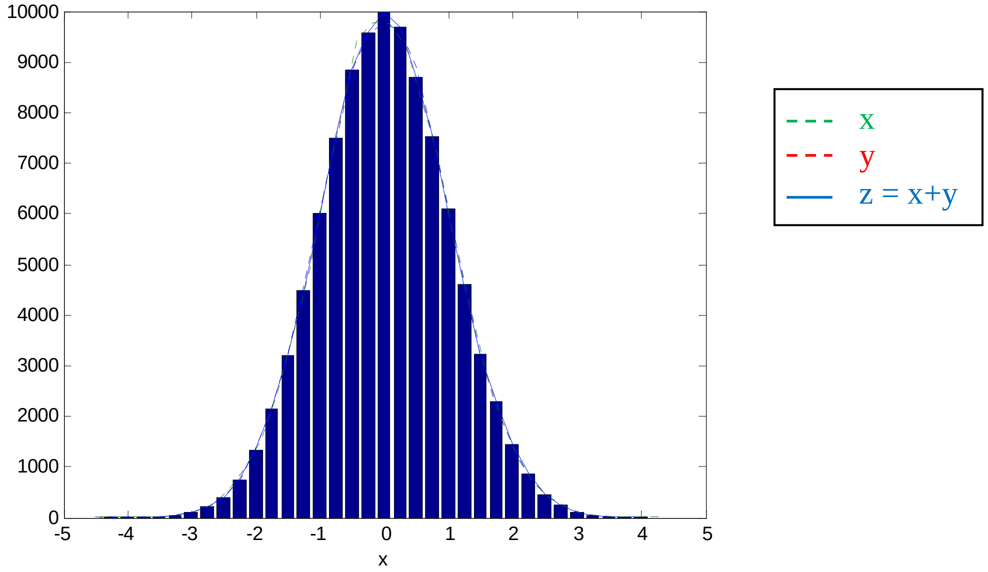
```
    a = 1 / Dx;   % P(Dx / 2) = 1/2 * P(0)
    X = a*X;
```

As only the Gaussian distribution has finite variance, many of the rescaling techniques that can be applied in its case cannot be applied to other Lévy distributions (including the Lorentzian distribution). We will come back to this point in the next section, where we will explore other Lévy distributions.

Now, incorporating rescaling into our program, we can observe the additive stability of the Gaussian distribution.

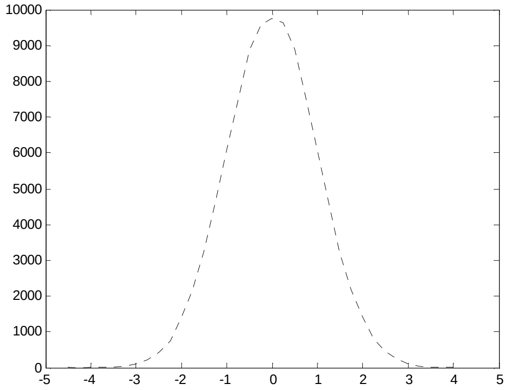
Gaussian distribution:

```
>g_stable(iter=100000)  % Gaussian additive stability checker
                        % (distributions rescaled with square average)
```



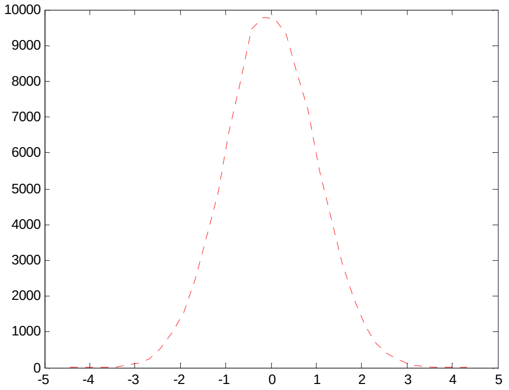
x

```
>plot( X(1,:), X(2,:) );
```



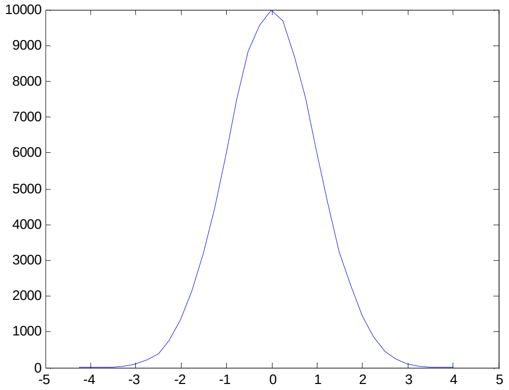
y

```
>plot( Y(1,:), Y(2,:) );
```



z = x + y

```
>plot( Z(1,:), Z(2,:) );
```



➔As expected, the numerical results confirm that the Gaussian distribution is additively stable. After rescaling, the added set $\{z\}$ fits the Gaussian distribution, just as the two original ones $\{x\}$ and $\{y\}$.

3.4 Stability of Lévy distributions

The Lévy distribution is studied in this section, according to its Probability Distribution Function:

$$p^{(\gamma)}(x)=\frac{1}{\pi}\int_0^\infty \varphi_\alpha(q)\cos(qx)dq\,,\qquad \varphi_\alpha(q)=e^{-\gamma|q|^\alpha}$$

The parameter γ is fixed at γ=1, while different values are assigned to α in the range 1 <= α <= 2 (α=1 gives a Lorentzian distribution; α=2 gives a Gaussian distribution).

As we explained in the previous section, *rescaling* is necessary to graphically compare the distributions of different sets of data. Therefore, some rescaling functions where implemented in the first place, to be employed in later programs to check for stability in different Lévy distributions.

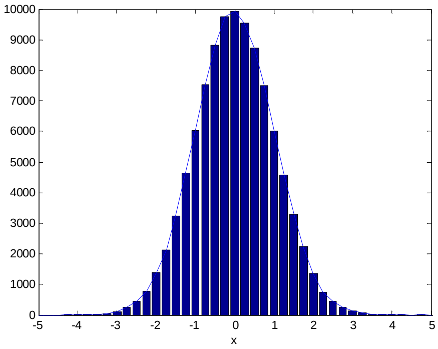
Initially, from a set of random numbers {x_i} (i=1..n), we planned to produce a rescaled set of random numbers {X_i} (i=1..n) with zero average and unit standard deviation (Av[X]=0, Var[X]=1) by a linear transformation X_i = a*x_i + b. That is what the square average method does in the case of the Gaussian distribution, which has *finite variance*. But the rest of the Lévy distributions have *infinite variance*, and so different approaches are required. Let us have a look at the different rescaling methods we implemented and some graphical examples of their results.

Gaussian distribution (Lévy distribution with α=2):

Rescaling can be achieved using square average:

```
a = 1 / sqrt( mean(x.^2) - mean(x)^2 );
b = -mean(x) * a;

X = a*x + b;
```

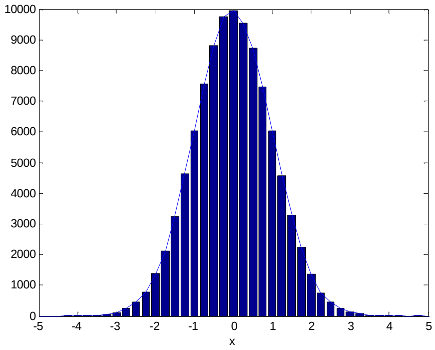
```
>g_histog2(iter=100000)
%Gaussian distribution function generator
```



mean(X) = 6.51e-4 ; var(X) = 1.0020

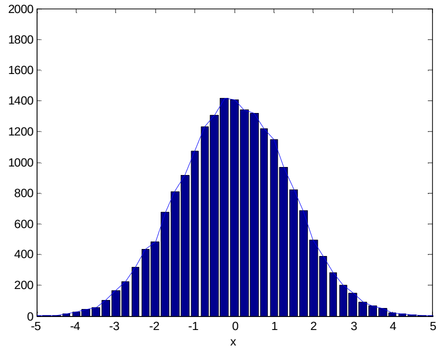
→

```
>rescale_g
%square average rescaler
```



mean(X) = 3.15e-17 ; var(X) = 1.0000

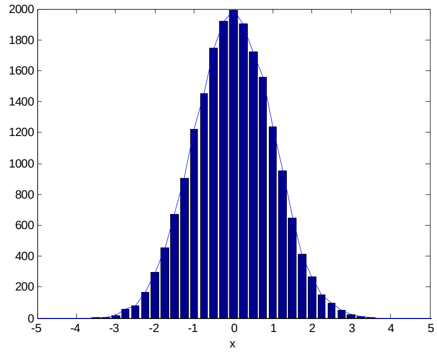
```
>levy_histog2(alpha=2, iter=20000)
%Lévy distribution function generator
```



mean(X) = -4.70e-3 ; var(X) = 1.9807

→

```
>rescale_g
%square average rescaler
```



mean(X) = -5.37e-17 ; var(X) = 1.0001

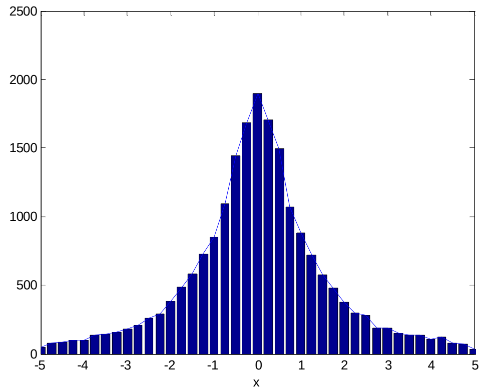
Other Lévy distributions, including Lorentzian distribution (1 <= α < 2):

As `var(x) = ∞`, the square average method cannot be applied for rescaling in this case. Two alternatives have been tested for rescaling these distributions in order to compare them:

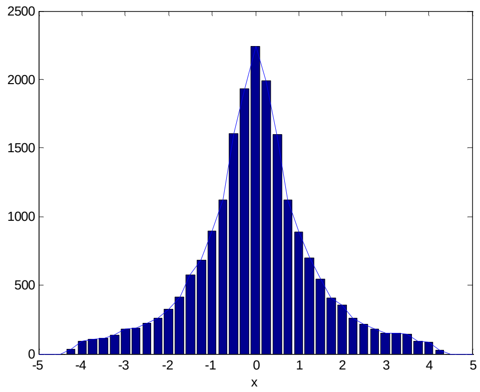
a) rescaling with <u>average of absolute value</u>

```
a = 1 / mean( abs(x) );
X = a*x;
```

```
>levy_histog2(alpha=2, iter=20000)
%Lévy distribution function generator
```



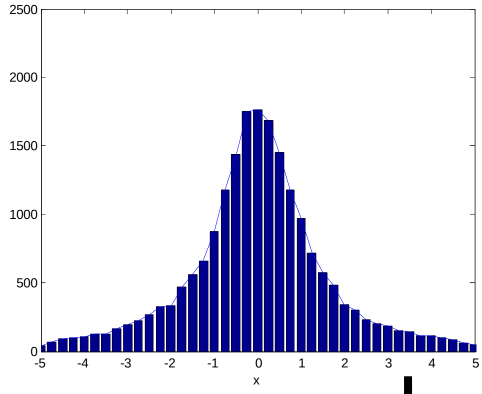
```
>rescale_abs
%avg. of abs. value rescaler
```



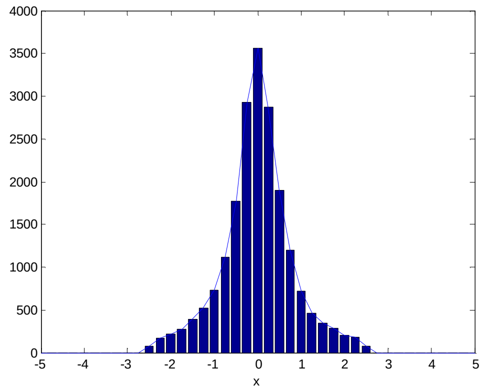
b) rescaling with <u>half-width</u>

```
a = 1 / Dx;        % P(Dx / 2) = 1/2 * P(0)
X = a*x;
```

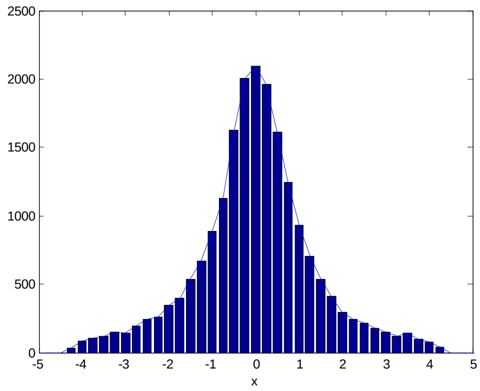
```
>levy_histog2(alpha=2, iter=20000)
%Lévy distribution function generator
```



```
>rescale_hw
%half-width rescaler
```



```
>rescale_abs
%avg. of abs. value rescaler
```



➨The incorporation of rescaling in our programs will let us graphically compare the distribution of different sets of data, to observe if they conform to the same distribution.

Gaussian distributed sets are easily rescalable. As the variance is defined in the Gaussian distribution, it is the only Lévy distribution which can be rescaled using the square average method. The other two rescaling methods will also work for Gaussian distributed data sets.

The rest of the Lévy distributions ($1 <= \alpha < 2$), including the Lorentzian distribution ($\alpha = 1$) cannot be rescaled using the square average method, because their variance is not defined ($\mathrm{var}[X] = \infty$). Different methods have been thus tested for rescaling these distributions: the average of absolute value method and the half-width method.

Additive stability of Lévy distributions

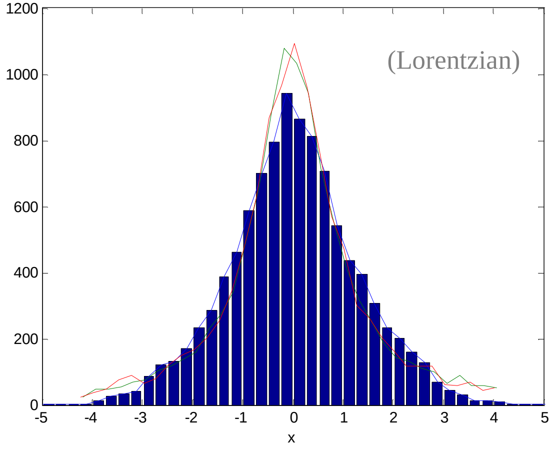
In the previous section we showed the property of additive stability in the Gaussian distribution. We will now try to observe graphically the property of additive stability in Lévy distributions in the range $1 <= \alpha < 2$. We will compare results with the different rescaling methods we have already introduced.

From the definition of additive stability given in the previous section, our programs start with two sets of random numbers {x_i} and {y_i} whose distribution is given by the Lévy distribution with identical shape parameter α; P(x) = Lev(x, α), P(y) = Lev(y, α). Then graphical results are given to check that the set of random numbers {z_i} produced by the addition of the elements of {x_i} and {y_i}, namely z_i = x_i + y_i, is also Lévy distributed with the same α, namely P(z) = Lev(z, α), after rescaling. We observe the results obtained with 5 values of α between α=1 (Lorentzian) and α=2 (Gaussian).

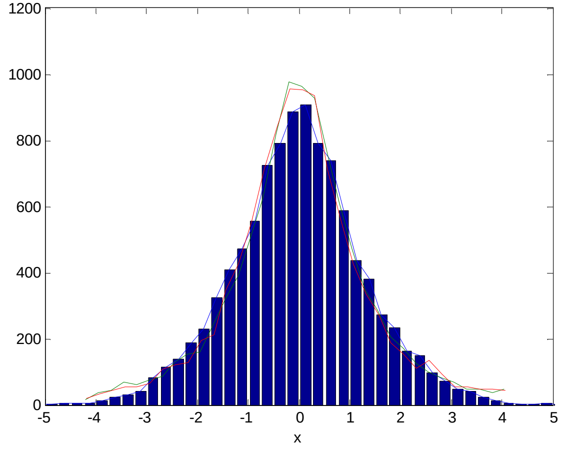
a) Rescaling with average of absolute value:

```
a = 1 / mean( abs(X) );
X = a*X;
```

```
>levy_stable2(alpha=1, iter=10000)
% Lévy additive stability checker
```

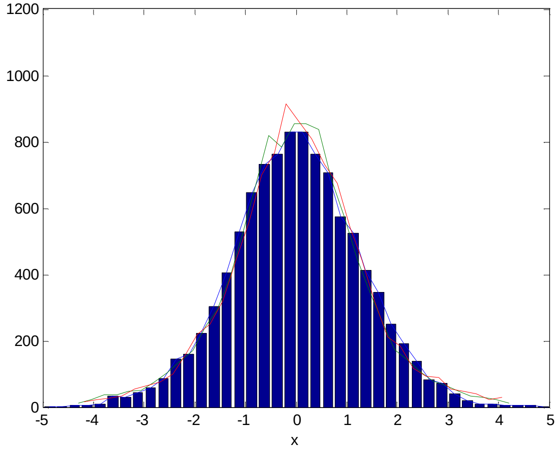


```
>levy_stable2(alpha=1.2, iter=10000)
```

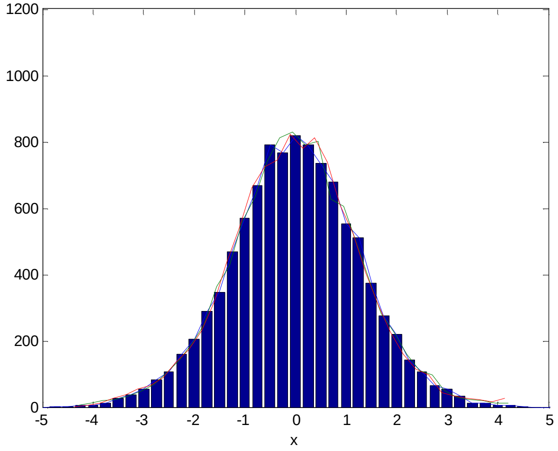


- - - x
- - - y
— z = x+y

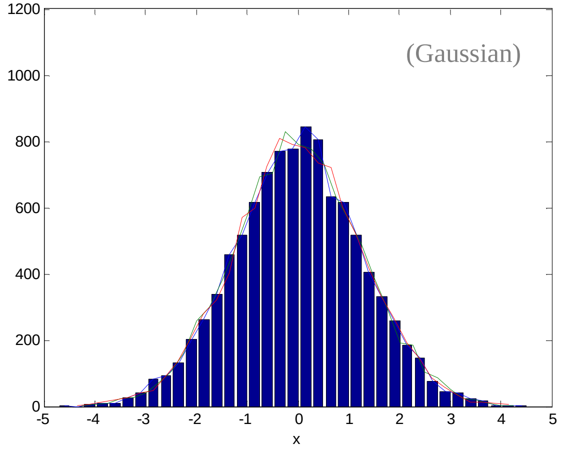
```
>levy_stable2(alpha=1.5, iter=10000)
```



```
>levy_stable2(alpha=1.8, iter=10000)
```



```
>levy_stable2(alpha=2, iter=10000)
```

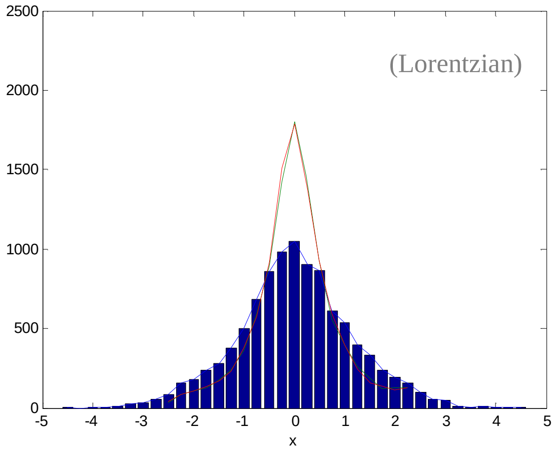


➔The added set {z} keeps the distribution of the original ones {x} and {y} after rescaling with the average of absolute value method. The fit is especially apparent for distributions in the range 1.5 < α <= 2, that is, for the Gaussian distribution and other Lévy distributions close to it.

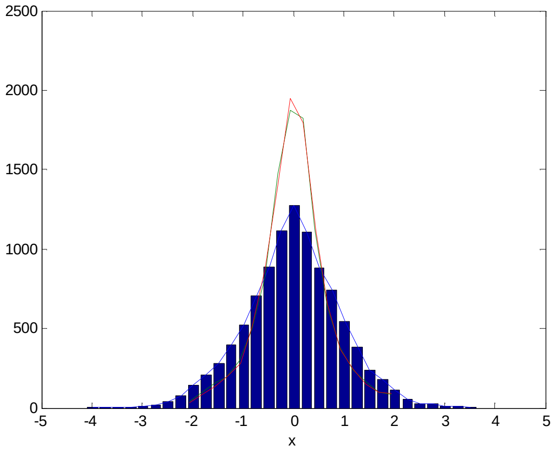
b) Rescaling with half-width:

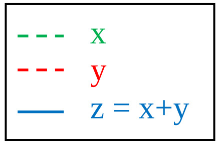
```
a = 1 / Dx;     % P(Dx / 2) = 1/2 * P(0)
X = a*X;
```

```
>levy_stable3(alpha=1, iter=10000)
% Lévy additive stability checker
```

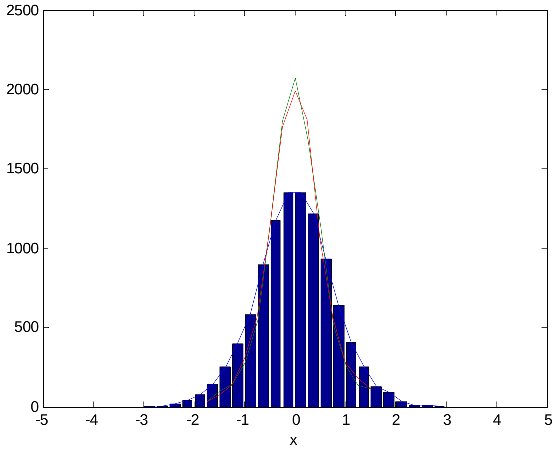


```
>levy_stable3(alpha=1.2, iter=10000)
```

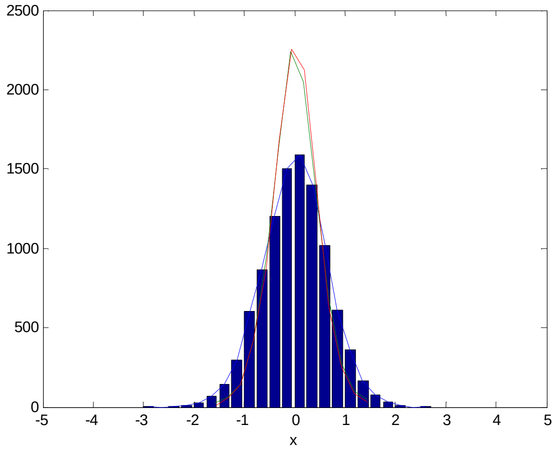




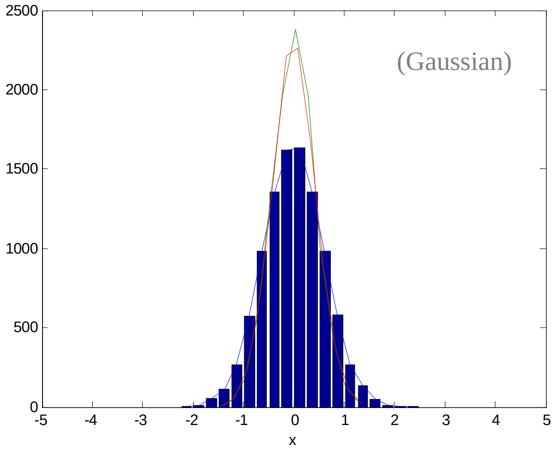
```
>levy_stable3(alpha=1.5, iter=10000)
```



```
>levy_stable3(alpha=1.8, iter=10000)
```



```
>levy_stable3(alpha=2, iter=10000)
```



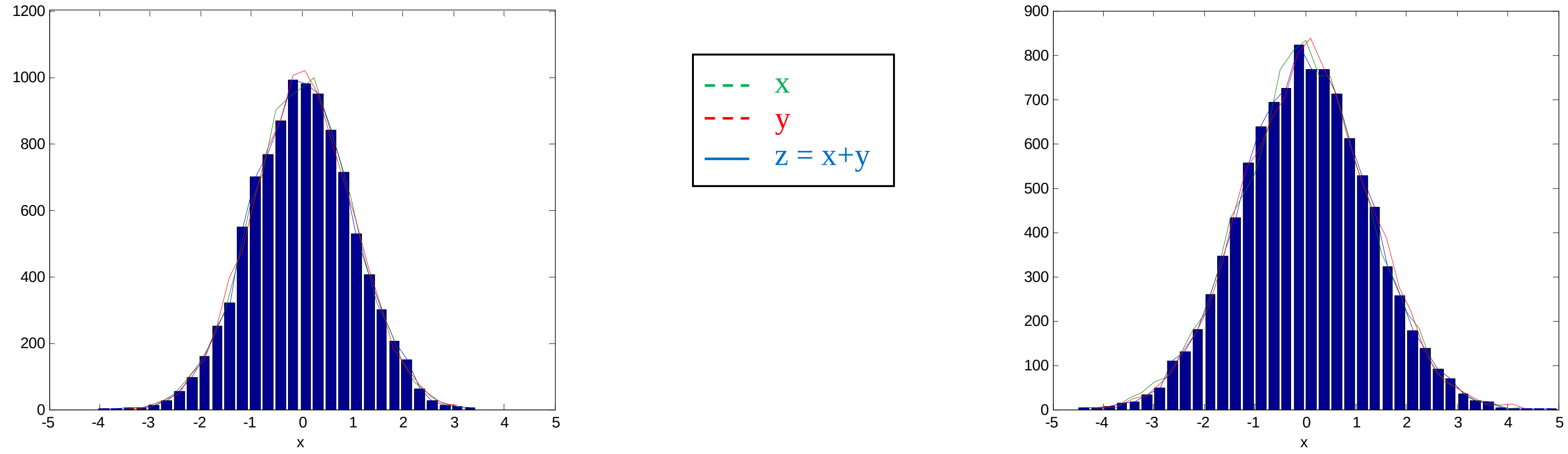
➔The additive stability of the distributions cannot be appreciated when using the half-width method for rescaling. The average of absolute value method is more suitable for this task.

Gaussian distribution (Lévy distribution with α=2):

```
>levy_stable(alpha=2, iter=10000)   % Lévy additive stability checker   >levy_stable2(alpha=2, iter=10000)
```

*Rescaling with <u>square average</u>

*Rescaling with <u>average of absolute value</u>

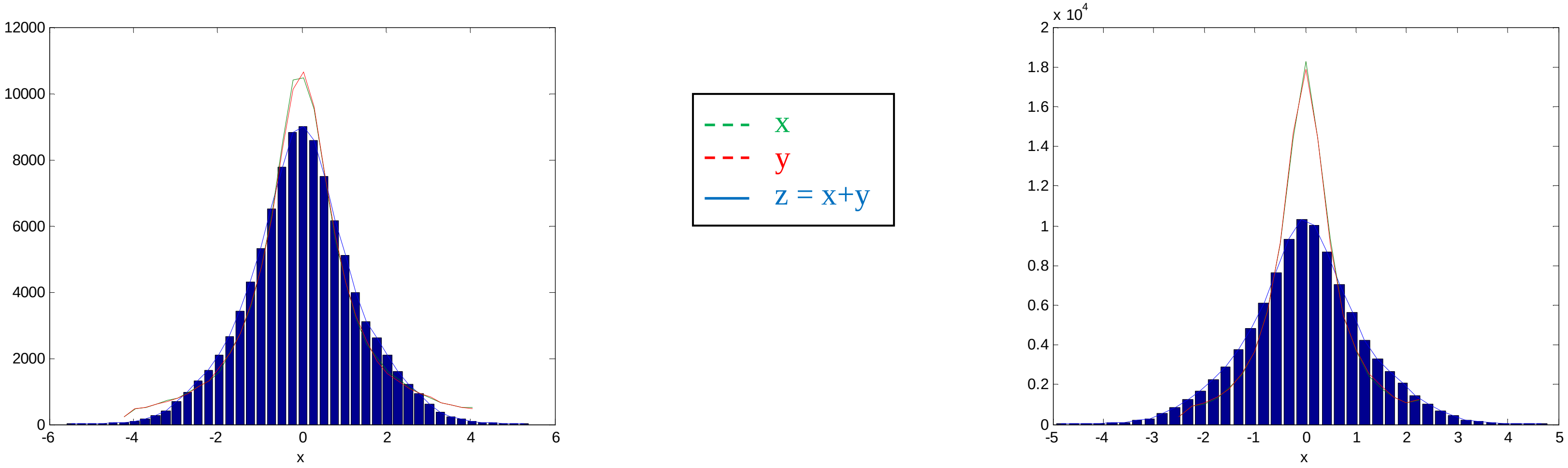


Lorentzian distribution (α=1):

```
>lor_stable(iter=100000)      % Lorentzian additive stability checker   >lor_stable2(iter=100000)
```

a) Rescaling with <u>average of absolute value</u>

b) Rescaling with <u>half-width</u>:



➔Let us now take a closer look at the Gaussian (α=2) and Lorentzian (α=1) distributions.

Random sets of numbers according to the Gaussian distribution can be generated using the Probability Distribution Function (PDF) for Gaussian distributions or the Probability Distribution Function for Lévy distributions with α=2. The resulting distribution is the same in both cases, but the random number generation is much quicker when using the specific Gaussian PDF, which is simpler than the integration-containing Lévy PDF. Quicker random number generation allows us to generate big data sets for tests in a very short time.

In the previous section, the additive stability of Gaussian distributions was checked by generating random numbers using the PDF for Gaussian distributions, using the function `g_stable`. The original sets ($\{x\}$ and $\{y\}$) and the added set $\{z\}$ where rescaled using the square average method (which is valid only for the Gaussian distribution).

In this section, the initial $\{x\}$ and $\{y\}$ sets have been generated using the Lévy PDF with α=2 (through the function `levy_stable`), and all sets have been rescaled using the square average method and the average of absolute value method. The results have been positive in all cases: the additive stability of the Gaussian distribution is always apparent.

As for the Lorentzian distribution, tests have been made using both the specific Lorentzian PDF (`lor_stable` function) and the Lévy PDF with α=1 (`levy_stable` function). Both methods for random number generation yield the expected results, but again the specific Lorentzian PDF is much quicker in execution than the Lévy PDF.

Lorentzian distributed data sets have been rescaled using two different methods: the average of absolute value method and the half-width method. While the average of absolute value method allows us to observe the additive stability of the Lorentzian distribution, the half-width method does not make it apparent.

Divisive rescaling

The simplest stochastic models of trade or wealth exchange involve random divisions and additions of random numbers. Therefore, as our last experiment, we have checked the behaviour of Lévy distributed sets when subject to random divisions.

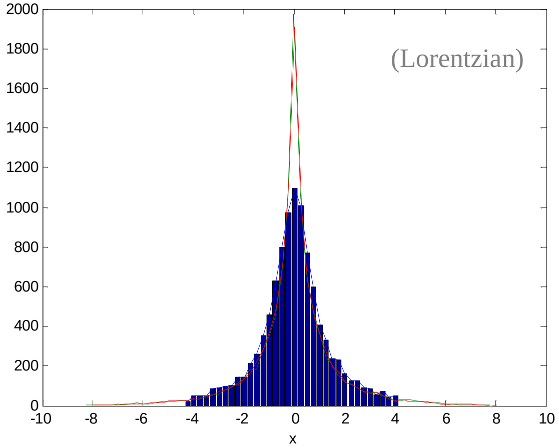
A set of Matlab programs have been implemented which start from a set of random numbers {z_i} whose distribution is given by the Lévy distribution with shape parameter *α*; P(z) = Lev(z, *α*). Graphical results are generated to observe which distribution will be obtained for {x'_i} and {y'_i} whose elements are produced by random division of z_i into two: x'_i = R*z_i , y'_i = (1-R)*z_i, with R drawn from uniform random numbers in the range [0,1[. The experiment was repeated for 5 values of *α* between *α*=1 (Lorentzian distribution) and *α*=2 (Gaussian distribution).

The following fragment of code shows the way in which the set {z} is randomly divided into {x} and {y}.

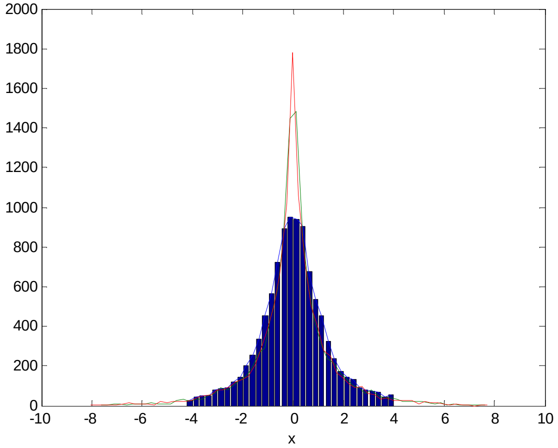
```
function [Xres Yres Zres]=levy_stable_div(a,iter,xmin,xmax)
%Levy divisive stability checker
%Usage: [Xres Yres Zres]=levy_stable_div(alpha,iter,xmin,xmax)
%default alpha=1,iter=1000,xmin=-5,xmax=5

[...]
      for i=1:iter
            R = rand(1); %uniformly distributed random number

            Z(1,i) = levy_gen(a,xmin,xmax);

            X(1,i) = R * Z(1,i);
            Y(1,i) = (1-R) * Z(1,i);
      end
[...]
```

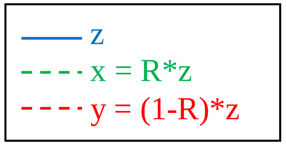
Rescaling with <u>average of absolute value</u>:

```
>levy_stable_div2(alpha=1, iter=10000)
% Lévy divisive stability checker
```

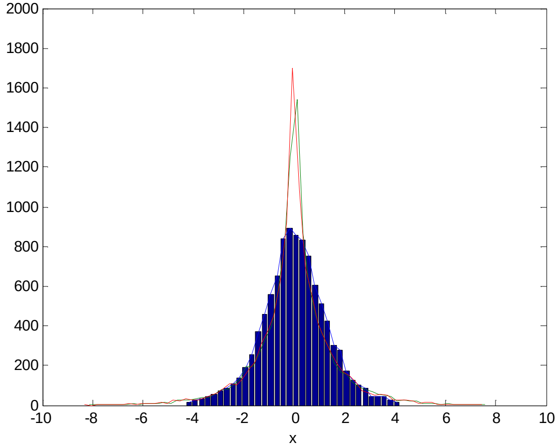


```
>levy_stable_div2(alpha=1.2, iter=10000)
```

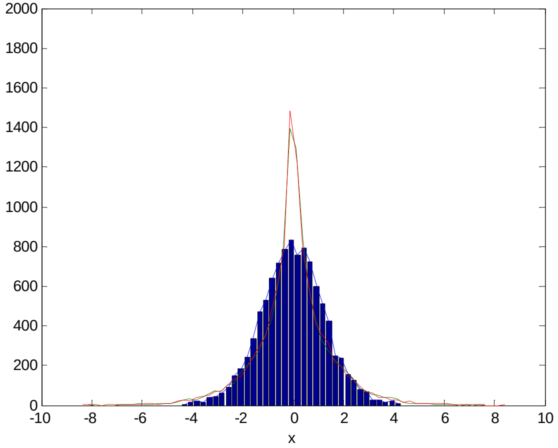




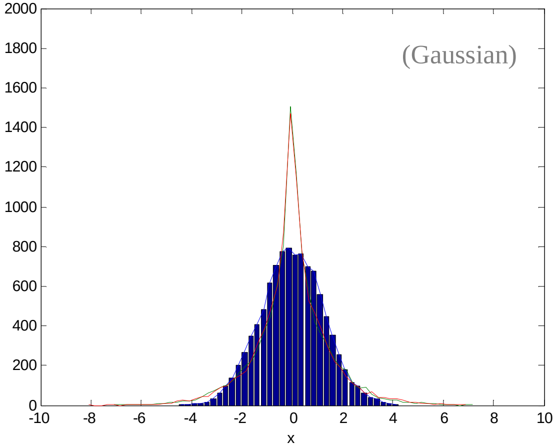
```
>levy_stable_div2(alpha=1.5, iter=10000)
```



```
>levy_stable_div2(alpha=1.8, iter=10000)
```



```
>levy_stable_div2(alpha=2, iter=10000)
```

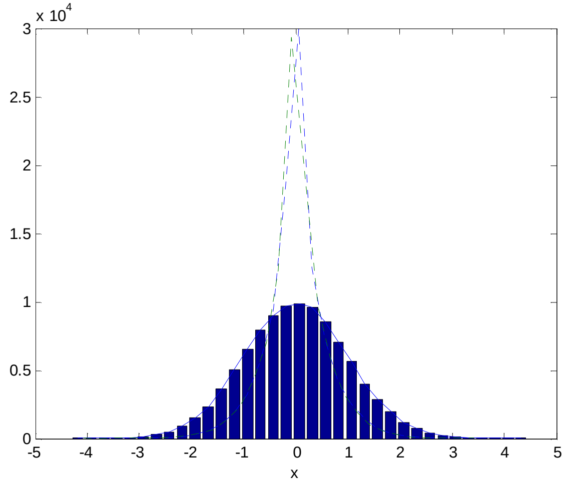


➔The distribution of the initial set {z} and those of the divided sets {x} and {y} do not fit well after rescaling. No divisive stability was appreciated in this test for any value of the Lévy distribution. Even so, the fit seems to be better for Lévy distributions in the range 1 <= α <= 1.2, that is, for the Lorentzian distribution and other Lévy distributions close to it. (No better results were obtained when the test was repeated using rescaling with half-width).

Gaussian distribution (Lévy distribution with α=2):

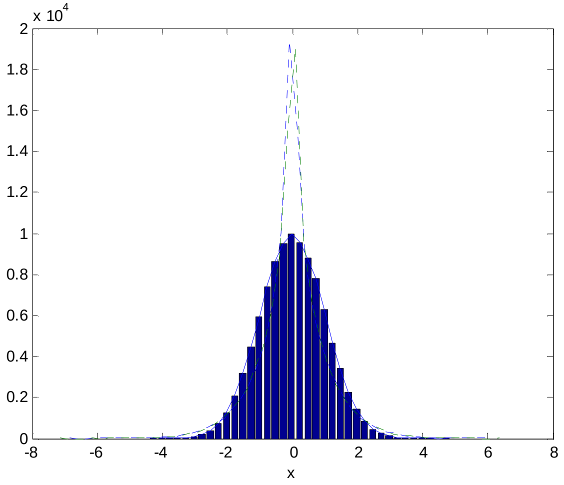
Without rescaling:

```
>g_stable_div2(iter=100000)
```



With rescaling (<u>square average</u> method):

```
>g_stable_div2(iter=100000)
```



—— z

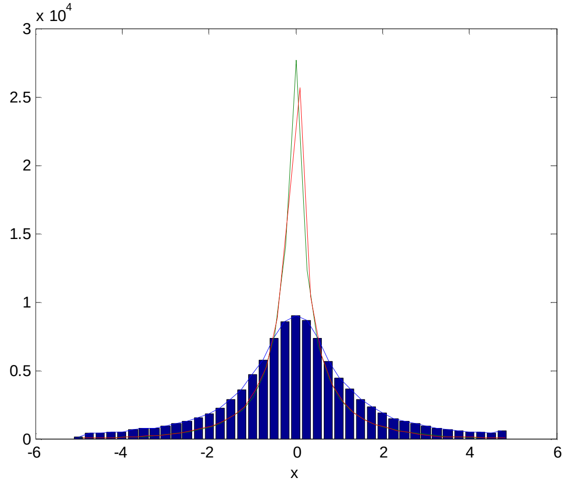
- - - - x = R*z

- - - - y = (1-R)*z

Lorentzian distribution (α=1):

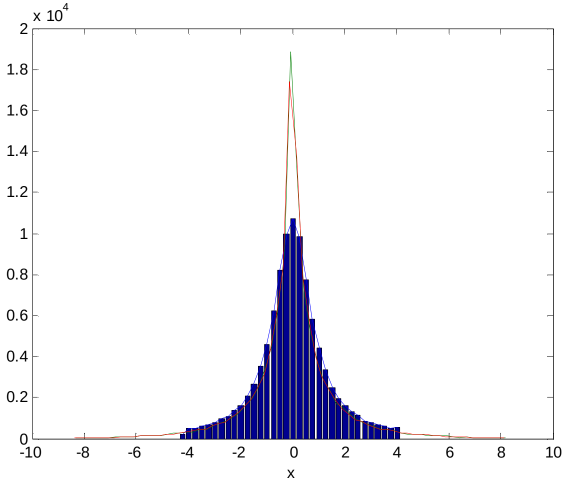
Without rescaling:

```
>lor_stable_div2(iter=100000)
```



With rescaling (<u>average of absolute value</u> method):

```
>lor_stable_div2(iter=100000)
```



—— z

- - - - x = R*z

- - - - y = (1-R)*z

➔This time, we generate Gaussian distributed data directly with the Gaussian PDF, and we rescale using the square average method. The result, as expected, is the same: no divisive stability is observed.

Lastly, we generate Lorentzian distributed data directly with the Lorentzian PDF, and we rescale using the average of absolute value method. Once again, no divisive stability is observed, but the fit still seems to be better than in the Gaussian case.

3.5 Closing comments

While natural random data tends to follow the Gaussian distribution, social and economical phenomena like the distribution of wealth [14] and the evolution of prices [11] fit different Lévy stable distributions, which are characterized by power-law tails (long tails). Lévy distributions have some qualities which make them especially suitable for modelling economic and financial data: they are stable, and therefore attractors with their associated limit theorems, and among stable distributions they are unique in having an analytically expressible probability density function.

In this chapter we have explored the statistical methods used in econophysics, while examining some basic properties of different Lévy distributions. First, a set of programs was implemented to generate random numbers according to Lévy distribution functions. With its graphical results we observed the different Lévy shapes, from the Gaussian distribution to the long-tailed Lorentzian distribution. We then proceeded to implement a set of rescaling programs based on three different methods (square average, average of absolute value and half-width methods). Lastly, the three rescaling methods were tested while observing the stability properties of Lévy distributions. A set of programs was developed to graphically show the stability of added Lévy distributions, and another one to check the behaviour of Lévy distributions when subject to random divisions (such as the ones needed for simple stochastic models of trade or wealth exchange).

From our implementation of different rescaling methods, we have observed that the square average method can only be used for the Gaussian distribution (which has a finite variance). The other two methods can be used for all Lévy distributions, but only the average of absolute value method has given us good results in examining the stability of Lévy distributions.

With regard to the analyzed properties, we have seen graphically how Lévy distributions are increasingly fat-tailed when going from Gaussian (α = 2) to Lorentzian (α = 1). We have also represented graphically the stability of added Lévy distributions. Finally, we could not observe divisive stability in Lévy distributions applying our rescaling methods, that is, random divisions applied to a specific Lévy distribution did not result in the same Lévy distribution.

As for the actual implementation of our programs and functions, Matlab was the chosen language. Matlab, as a language for numerical computing and mathematics, carries a wide range of pre-implemented mathematical functions, and it allows also for easy plotting of functions and data, all of which simplifies the task of statistical programming. Still, there is one disadvantage to the use of pre-implemented functions: the programmer cannot tweak those functions for efficiency. We ran into this problem when we had to calculate the integral in the probability density function of the Lévy distribution; calculations would go on for minutes before we obtained the results. Efficiency was not a requirement for these exercises; however, in cases where efficiency is the goal, Matlab can call functions written in C, so quicker C functions could be used instead of the Matlab equivalents. The Matlab functions we implemented are attached to this project for reference and use.

To sum up, models based on fat-tailed Lévy distributions are needed to overcome the limitations of the widespread Black-Scholes model for option-pricing (based on geometric Brownian motion, and thus on the Gaussian distribution), which is useful as an approximation, but disagrees significantly with real markets as observed in empirical data.

But Lévy distributions still have a downside: they have infinite variance, which makes them difficult to apply for practical modelling. This problem can be circumvented through the use of truncated Lévy distributions, as proposed in the works of Mantegna & Stanley [10] and Vasconcelos [11].

The Lévy distribution has been applied by Mantegna & Stanley to model the distribution of changes in the S&P 500 stock index of the US Stock Exchange [11]. Many other applications of Lévy processes in finance have been performed, such as in [15], [16] and [17].

4. Concluding comments

Quantum cryptography and econophysics are two multidisciplinary fields in which computer science interacts with physics, even if the nature of the interaction is different in each case. On the one hand, in quantum physics (and quantum computation in general), physical theory is applied to computer science, to find new ways for computers to evolve. Conversely, in econophysics (and computational finance), it is computer science that can provide methods and perspectives that can be applied, with the framework of physical statistics (statistical finance), to solve problems in the area of economics.

Most theoretical results of quantum computation are still far from becoming a practical reality. At the moment, the most important practical results are being obtained in the subfield of quantum cryptography. Quantum cryptography has been implemented in many occasions, even over long distances, and there are already several companies which offer commercial quantum cryptography systems.

In our chapter on quantum cryptography, we have described the two main quantum key distribution protocols, BB84 and B92. They both use the properties of quantum indeterminacy to ensure the security of the key distribution. We have shown how they work using electronic spin to represent qubits, in a way that is extensible to any other quantum state representation. And we have compared both protocols, to see how B92 is in some ways more refined, while B84 is more efficient in the use of transmitted data.

We have finally described the implementation of quantum cryptography protocols. In this case we have used single photons to represent quantum states, which is one of the most popular practical implementations, as it allows for the transmission of secure data over optical fibre, and for free space transmission as well. In this way, through the use of components available today, quantum cryptography is becoming a practical and even commercial reality.

As for econophysics, the development of new and better fitting economical models has become a field of extreme importance in banking and in the world of finance; computer scientists are members of teams for statistical finance, along with physicists and mathematicians. Statistical finance, quantitative finance and computational finance, all of them are in some way interchangeable, and combinable; approaches from different disciplines, all external to traditional economics, which bring solutions from new perspectives.

In our chapter on econophysics, we have explored the statistical methods used in statistical finance, while examining some basic properties of different Lévy distributions. Matlab programs were implemented all thorugh the chapter to test and show the shapes and behaviours of different Lévy distributions, from Gaussian to Lorentzian. Three different rescaling methods were tested so as to observe the stability properties of Lévy distributions. A set of programs was developed to graphically show the stability of added Lévy distributions, and another one to check the behaviour of Lévy distributions when subject to random divisions (such as the ones needed for simple stochastic models of trade or wealth exchange).

Computer scientists and engineers apply their skills in many multidisciplinary fields. Different specialized profiles have a place in modern research and industry. Quantum computation is a field which will be acquiring more and more importance in the next years. Computer scientists with a good understanding of the physical foundations involved will participate in the development of the new technologies that will become more of a practical reality in the medium term and which might change the very foundations of computing. Quantum cryptography can be considered the first practical reality of quantum computation; it is already there for us to experiment and take advantage of it. On the other hand, computer scientists with a good grasp of mathematics, statistics and advanced computational methods such as neural networks and evolutionary computation can partake in research in the world of finance, in the disciplines of econophysics, statistical finance and computational finance.

5. Bibliography

Quantum cryptography

[1] Cheon, T. *Quantum Information for Quantum Cats*. Kochi: Kochi University of Technology, **2006**. Online: <http://www.kochi-tech.ac.jp/~cheon/q-inf/q-inf00_e.html> [last visit: 8 Nov 2009].

[2] Nielsen, M.; Chuang, I. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, **2000**.

[3] Bennett, C. H.; Brassard, G. "Quantum Cryptography: Public Key Distribution and Coin Tossing". In: *Proceedings of IEEE International Conference on Computers Systems and Signal Processing*, Bangalore, December **1984**. PP. 175-179. Online: <http://www.research.ibm.com/people/b/bennetc/bennettc198469790513.pdf> [last visit: 10 Nov 2009].

[4] Bennett, C. H. "Quantum Cryptography Using any Two Nonorthogonal States". In: *Physical Review Letters*, May **1992**. Vol.68, pp.3121-3124. Online: <http://www.research.ibm.com/people/b/bennetc/qc2nos.pdf> [last visit: 10 Nov 2009].

[5] Lomonaco, S. J. *A Quick Glance at Quantum Cryptography*. Baltimore: University of Maryland, **1998**. Online at *arXiv.org*: quant-ph/9811056v1.

[6] Bennett, C. H.; Brassard, G.; Smolin, J.; et al. "Experimental Quantum Cryptography". In: *Journal of Cryptology*, 1992. Vol.5, no.1, pp.3-28. Online: <http://cs.uccs.edu/~cs691/crypto/BBBSS92.pdf> [last visit: 6 Dec 2009].

[7] Papadakos, P. N. *Quantum Information Theory and Applications to Quantum Cryptography*. London: University of London, **2001**. Online at *arXiv.org*: quant-ph/0201057v1.

[8] Gisin, N.; Ribordy, G.; Tittel, W.; Zbinden, H. "Quantum cryptography". In: *Reviews of Modern Physics*, March **2002**. Vol. 74, pp. 145-195. Online at *arXiv.org*: quant-ph/0101098.

[9] Scarani, V.; Bechmann-Pasquinucci, H.; et al. "The Security of Practical Quantum Key Distribution". In: *Reviews of Modern Physics*, September **2009**. Vol. 81, pp. 1301-1351. Online at *arXiv.org*: arXiv:0802.4155.

➢ Bennett, C. H.; Brassard, G.; Mermin, N. D. "Quantum Cryptography without Bell's Theorem". In: *Physical Review Letters*, May **1992**. Vol. 68, pp. 557-559. Online: <http://www.research.ibm.com/people/b/bennetc/BBM92.pdf> [last visit: 10 Nov 2009].

Econophysics

[10] MANTEGNA, R.; STANLEY, H. E. *An Introduction to Econophysics*. Cambridge: Cambridge University Press, **2000**.

[11] VASCONCELOS, G. L. “A Guided Walk Down Wall Street: An Introduction to Econophysics”. In: *Brazilian Journal of Physics*, September **2004**. Vol. 34, nº 03B, pp. 1039-1065. Online at *arXiv.org*: cond-mat/0408143v1.

[12] MANDELBROT, B. B. “The Variation of Certain Speculative Prices”. In: *Journal of Business*, October **1963**. Vol. 36, pp. 394-419.

[13] BERTRAM, W. K. *Modelling asset dynamics via an empirical investigation of Australian Stock Exchange data*. Sydney: University of Sydney, **2005**. Online: <http://ses.library.usyd.edu.au/bitstream/2123/1593/1/Modelling%20Asset.pdf> [last visit: 10 Dec 2009].

[14] YAKOVENKO, V. M. “Econophysics, Statistical Mechanics Approach to”. In: *Encyclopedia of Complexity and Systems Science*. Springer, **1999**. Online at *arXiv.org*: arXiv:0709.3662v4.

[15] MIRANDA, L. C; RIERA, R. “Truncated Lévy walks and an emerging market economic index”. In: *Physica A*, August **2001**. Vol. 297, pp. 509-520.

[16] SCHOUTENS, W. *Lévy Processes in Finance: Pricing Financial Derivatives*. Wiley, **2003**.

[17] MATACZ, A. “Financial Modeling and Option Theory With The Truncated Levy Process”. In: *Int. J. of Theoretical and Applied Finance*, January **2000**. Vol. 3, pp. 143-160. Online at *arXiv.org*: cond-mat/9710197v1.

➢ APPLEBAUM, D. "Lévy Processes – From Probability to Finance and Quantum Groups". In: *Notices of the American Mathematical Society*, December **2004**. Vol. 51, pp. 1336-1347. Online: <http://www.ams.org/notices/200411/fea-applebaum.pdf> [last visit: 10 Dec 2009].

Appendix. Matlab functions implemented for Chapter 3.

3.2 Random walk. Generation of sets according to specific probability distributions.

```
function X=essu(n,s)
% S_n (uniform random) generator
% default n=10, s=1

function X=histog(n,s,iter)
%P(S_n) distribution function generator
%[manually calculated histogram]
%Usage: X=histog(n,s,# of iterations)
%default s=1, iter=1000

function X=histog2(n,s,iter)
%P(S_n) distribution function generator
%[uses Matlab hist function]
%Usage: X=histog2(n,s,# of iterations)
%default s=1, iter=1000

function Z=g_dist(n,s)
%Gaussian distribution function generator
%default n=5, s=1

function x=g_gen(xmin,xmax)
%Gaussian random number generator
%default xmin=-5, xmax=5

function X=g_histog(iter,xmin,xmax)
%Gaussian distribution function generator
%default iter=10000,xmin=-5,xmax=5

function Z=lor_dist(gamma,xmin,xmax)
%Lorentzian distribution function generator
%default gamma=1, xmin=-5, xmax=5

function x=lor_gen(gamma,xmin,xmax)
%Lorentzian random number generator
%default gamma=1, xmin=-5, xmax=5

function X=lor_histog(iter,gamma,xmin,xmax)
%Lorentzian distribution function generator
%default iter=10000,gamma=1,xmin=-5,xmax=5

function Z=levy_dist(a,xmin,xmax)
%Levy distribution function generator
%Usage: X=levy_dist(alpha,xmin,xmax)
%default alpha=1, xmin=-5, xmax=5

function x=levy_gen(a,xmin,xmax)
%Levy random number generator
%Usage: x=levy_gen(alpha,xmin,xmax)
%default alpha=1, xmin=-5, xmax=5

function X=levy_histog(a,iter,xmin,xmax)
%Levy distribution function generator
%Usage: X=levy_histog(alpha,iter,xmin,xmax)
%default alpha=1, iter=1000, xmin=-5, xmax=5
```

3.3 Additive stability of probability distributions. Rescaling.

```
function [Xres Yres Zres]=g_stable(iter,xmin,xmax)
%Gaussian additive stability checker
%[uses rescaling with square average]
%default iter=10000,xmin=-5,xmax=5
```

3.4 Stability of Lévy distributions

```
function X=g_histog2(iter,xmin,xmax)
%Gaussian distribution function generator
%this version returns the original numbers, not the histogram
%(later used for rescaling)
%default iter=10000,xmin=-5,xmax=5

function X=lor_histog2(iter,gamma,xmin,xmax)
%Lorentzian distribution function generator
%this version returns the original numbers, not the histogram
%(later used for rescaling)
%default iter=10000,gamma=1,xmin=-5,xmax=5

function X=levy_histog2(a,iter,xmin,xmax)
%Levy distribution function generator
%this version returns the original numbers, not the histogram
%(later used for rescaling)
%Usage: X=levy_histog2(alpha,iter,xmin,xmax)
%default alpha=1, iter=1000, xmin=-5, xmax=5

function Xres=rescale_g(X)
%Gaussian distribution rescaler
%[rescaling with square average]

function Xres=rescale_abs(X)
%distribution rescaler: avg. of absolute value
%[rescaling with avg. of absolute value]

function Xres=rescale_hw(X,a)
%distribution rescaler: half-width
%for Levy distribution
%[rescaling with half-width]
%Usage: Xres=rescale_hw(X,alpha)

function Dx=levy_dist_hw(a,xmin,xmax)
%Levy half-width calculator
%Usage: Dx=levy_dist_hw(alpha,xmin,xmax)
%default alpha=1, xmin=-5, xmax=5

function [Xres Yres Zres]=levy_stable2(a,iter,xmin,xmax)
%Levy additive stability checker
%[rescaling with mean(|x|)]
%default a=1,iter=1000,xmin=-5,xmax=5

function [Xres Yres Zres]=levy_stable3(a,iter,xmin,xmax)
%Levy additive stability checker
%[rescaling with half-width]
%default a=1,iter=1000,xmin=-5,xmax=5
```

```
function [Xres Yres Zres]=lor_stable(iter,gamma,xmin,xmax)
%Lorentzian additive stability checker
%[rescaling with mean(|x|)]
%default iter=10000,gamma=1,xmin=-5,xmax=5

function [Xres Yres Zres]=lor_stable2(iter,gamma,xmin,xmax)
%Lorentzian additive stability checker
%[rescaling with half-width]
%default iter=10000,gamma=1,xmin=-5,xmax=5

function [Xres Yres Zres]=levy_stable_div2(a,iter,xmin,xmax)
%Levy divisive stability checker
%[rescaling with mean(|x|)]
%default a=1,iter=1000,xmin=-5,xmax=5

function [Xres Yres Zres]=levy_stable_div3(a,iter,xmin,xmax)
%Levy divisive stability checker
%[rescaling with half-width]
%default a=1,iter=1000,xmin=-5,xmax=5

function [Xres Yres Zres]=g_stable_div2(iter,xmin,xmax)
%Gaussian divisive stability checker
%[uses rescaling with square average]
%default iter=10000,xmin=-5,xmax=5

function [Xres Yres Zres]=lor_stable_div2(iter,gamma,xmin,xmax)
%Lorentzian divisive stability checker
%[rescaling with mean(|x|)]
%default iter=10000,gamma=1,xmin=-5,xmax=5
```

Others

```
function Xres=histogx(X,xmin,xmax)
%shows the histogram for a set of numbers X
%[uses Matlab hist function]
%default xmin=-5,xmax=5
```