Entropy density with an application to time series

Dany Yatim

Supervisor : Alexis Penot

Master 2 Economic and Finance, Lyon 2, 2016/2017

Abstract. The use of entropy in economic and finance stands as an ambassador of the "econophysics" ground, and this work wishes to extend the comprehension of its ability to capture a particular view on time series. Returns behavior remains difficult to describe, while the widely used normal density to model returns appears to be a biased hypothesis on the empirical ground. This work considers a particular kind of density, a one that places inference instead of deduction as an alternative way of modeling returns, the entropy density. It is shown that it outperforms the normal density in the context of time series models such as ARMA model on several empirical series of returns. Its use is extended to Value at Risk and Conditionnal Value at Risk simulations in order to observe its ability to fits returns' distribution. This second application gives some interesting results, as it succeeds in modeling the leptokurtic form of the empirical distributions, while it fails to capture extreme returns. This work that goes from a theorical to an empirical ground wishes to contribute to a barely fresh and new way of representing the behavior of returns, while it nudges to further works toward this conceptual view.

I would like to thank my advisor, Alexis Penot, for the time he took to supervise this work, and for his curiosity toward the subject of this work.

I would like to extend my gratitude to the kindness and understanding of Mr. Jouneau and Mr. Eyquem.

My gratitude goes also to Edmée for her invaluable support and the time she took to tear her hair out while correcting this work.

1 Introduction

The essence of entropy resides in its ability to take in account every amount of information that we do have about any quantity. It allows to know how close we are from having all the informations needed to predict any phenomena or, how close do we stand from a complete hazard. Being used in information theory and thermodynamics, its use in economics remains marginal. The word "econophysics" that arised with entropy is far from being a classical topic. However, it has at least the benefit of looking at topics such as the study of time series with different angles. One of these interesting (and that is only the opinion of this work) angle is the principle of maximum entropy : subject to some informations, the best probability distribution to assume of a variable regarding our state of information is the one that provides the maximum entropy. As it will be seen later, this view herited from statistical mechanics and expounded by Edwin Thompson Jaynes in 1957 appears to be a useful principle and quite adapted to time series on the economic/financial ground. When we choose a probability distribution for random variables, such as the widely used normal density, we may be driven toward assuming more informations that we really have. The density of maximum entropy does not stand in front of a normal density but rather next to it : if we do have enough informations that permit the hypothesis of exactly a normal density, the density of maximum entropy (E.D.) will take the form of the famous Gauss bell. But what if a more leptokurtic density function, and/or an asymmetric one, fits better our data ? This is what E.D. allows to do : a particular (and somehow unique) density function for our data, regarding some testable informations that we have and use to construct the E.D. .

We define $H(x)$ as the entropy of a random variable $x \in A$ as

$$H(x) = -\int_A p(x) ln(p(x)) dx$$

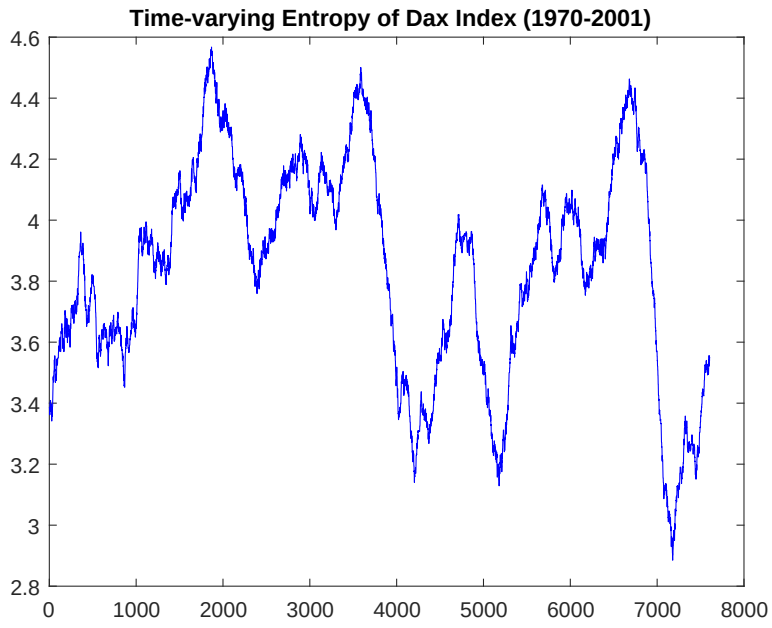
Entropy could be defined as a measure of uncertainty. As it is seen just before, it takes in account the probability of a random variable. Let us assume that we have a message to code in a binary language. If we use a two caracters alphabet with equal frequency, and words of only one caracter, we have 50% chances to have one of the two possible words : we stand in a complete random state with an entropy of one, and

a message that will look like 01101010.... In the opposite, if we are 100% sure that we will be using only one character, entropy falls to 0 : no incertainty remains, either we have 00000... or 11111.... If we use a more complete alphabet of 27 caracters, still with equal frequencies, we end up with $log(27) = 4.75$ bits per caracters (with log base 2, $2.^{4.75} = 27$). The main difference between the two alphabets is the amount of uncertainty about an outcome, and this is what we will keep in mind during this work.

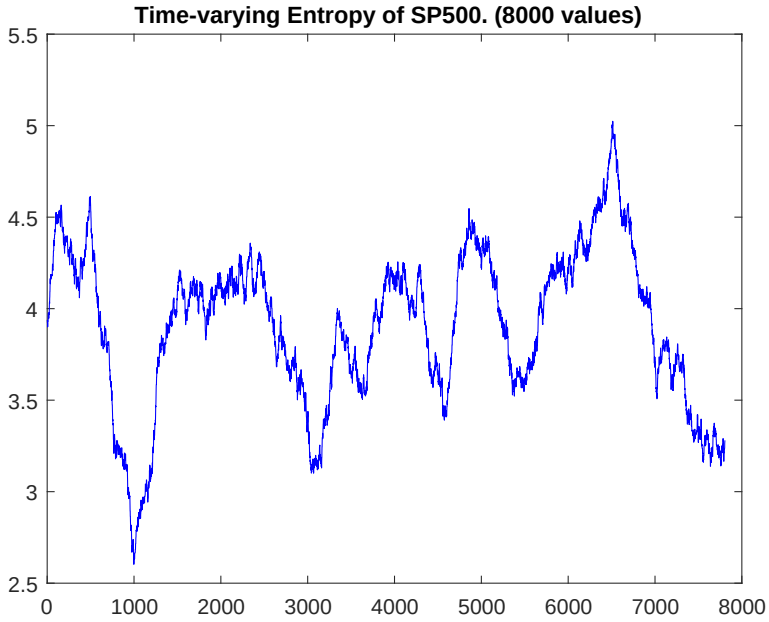
In an economic or financial context, we may ask the following question : how many possible states do we have for a stock's return ? Now, if one would express each of these states (i.e. possible values of return) in a binary language, how many bits would he or she use on average ? That is what entropy answers. Entropy could be helpful to observe windows of time during which uncertainty becomes smaller and thus informative of some particular and more deterministic movements of the stock price, or, of some more hazardous behavior.

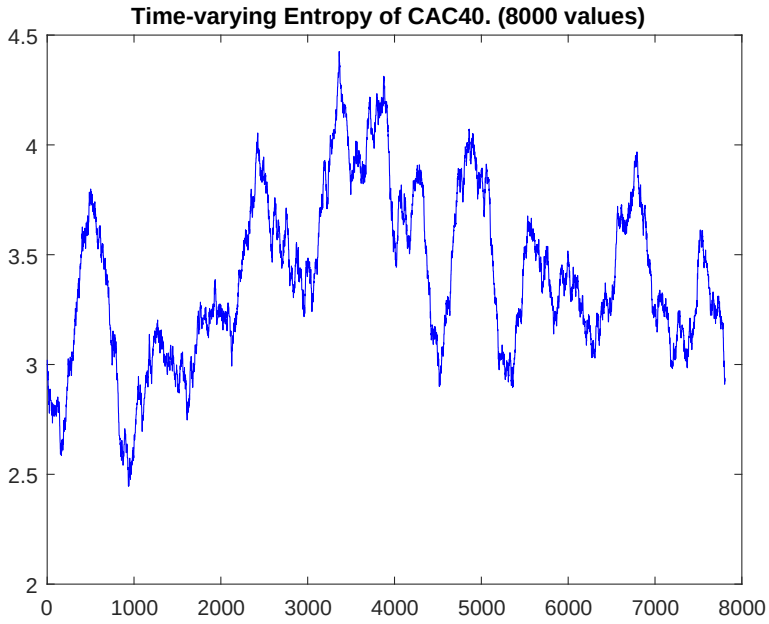
1.1 Entropy of a time serie

On figure 1, we can observe the time-varying entropy of the dax index's return between 1970 and 2001 (empirical frequencies are used, daily returns for every data used in this work). Each point is the entropy of 300 rolling values. We define a fall of entropy as a loss of uncertainty about the return's value, in a sense that the frequencies are distribued among a smaller group of values. The first fall of entropy that begins at the end of 1971 (500th value) corresponds to the period of the first oil crisis with the lowest point (1000th value) that covers the entropy of the October 1973/December 1974 period. The second one, right after the 2500th value corresponds to the second oil crisis of 1979-1981 followed by the Latin American debt crisis. The fall between the 4000th and the 5000th value emphasise the crash of 1987, followed by another falling entropy period right after the 5000th value, which is the 1989-1991 period (junk-bond crisis of 1989 and the internet bubble). Finally, the last fall turn out to be during the asian financial crisis of 1997 and the russian crisis of 1998.



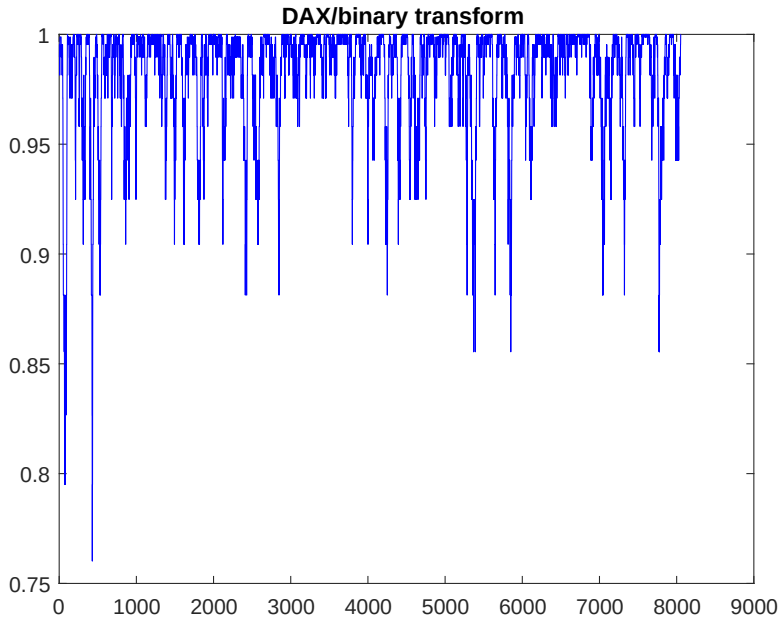
The next two figures are the time-varying entropy of the SP500 index and the CAC40 index. Not surprisingly, we meet again the same patterns. However, we observe different ranges of entropy's fall depending on the period. The fall of entropy due to the oil crisis of 1973 is bigger for the CAC40 and SP500 : a loss of two in the entropy scale (from 4.5 to 2.5), whereas it is "only" of one for the Dax. The improvement of entropy after the asian crisis could be noticed for the two European index, but not for the American one.





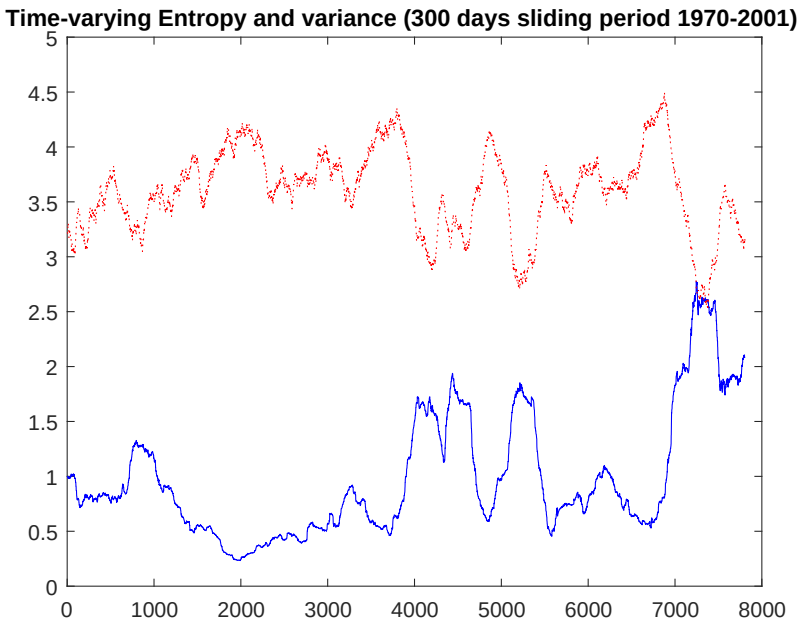
Even if the aim of this work is not to contest in a radical way the normal assumption of returns, entropy could be helpful to observe divergence of the empirical returns with the normal density. In order to have a more meaningful view, the next plots are

the ones of binary transforms of the previous time series. A positive (negative) return is equal to 1 (0). We recall that a $(1/2, 1/2)$ probability function has an entropy of one, a completely uninformative probability distribution. This kind of transform allows to see how close (or far) we stand from a 50% chance of having a negative(or positive) return, which is what a normal density assumes.



As it is seen upward, every fall of entropy emphasis periods where the 50% rule of having negative(or positive) returns is broken, in favor of one of the two cases (we recall that it is impossible here to say if we stand on one side or another).

A non-instinctive result is that the entropy is, globally but not exclusively, a negative function of the variance. This is due to the fact that the variance expresses a distance from the mean, whereas the entropy expresses the multiplicity of a return's/price's possible states. If, during a high volatility period, a return concentrates its values far from the mean, we expect a high variance consequently, but a low entropy. In the opposite, a variance of 0 goes with an entropy of 0, which never happens in a financial context. As is it clear in the next figure, every high volatility period is symmetric to a low entropy period.


Time-varying Entropy and variance (300 days sliding period 1970-2001)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1000
2000
3000
4000
5000
6000
7000
8000

2 The maximum entropy density

The idea is to find a probability density that allows a maximum entropy, on the basis of partial informations. For this purpose we use the first four moments of a random variable, namely the mean, standard error, skewness and kurtosis. Once these moments are known, the aim is to solve the following program :

$$\text{Max}H(x)$$

subject to

$$\int_A p(x)dx = 1 \tag{1}$$

$$\int_A xp(x)dx = E(x)$$

$$\int_A x^2p(x)dx = \sigma^2(x)$$

$$\int_A x^3p(x)dx = S(x)$$

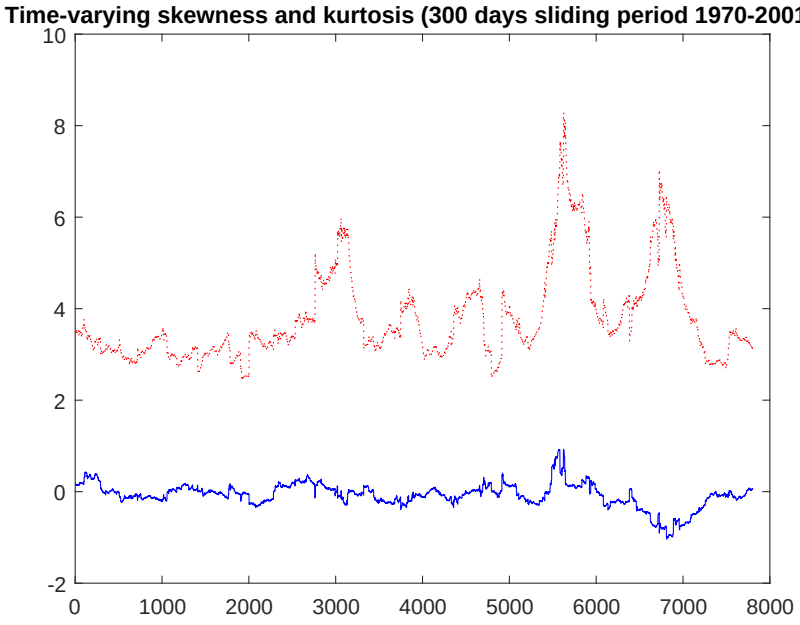
$$\int_A x^4p(x)dx = K(x)$$

with $p(x) = 0\ \forall x \notin$ A.

In other words, we seek to maximize the function f defined as :

$$f(x) = -\int_A p(x)ln(p(x))dx + \lambda_0[\int_A p(x)dx - 1] + \lambda_1[\int_A xp(x)dx - E(x)] + \lambda_2[\int_A x^2p(x)dx - \sigma^2(x)]$$
$$+ \lambda_3[\int_A x^3p(x)dx - S(x)] + \lambda_4[\int_A x^4p(x)dx - K(x)]$$

As it is shown in the next figure, the skewness and kurtosis of the dax's return are clearly not stable through time. A normally distributed variable would have a skewness of 0 (dax's skewness is not that far from this value, but not stable either) and a kurtosis of 3 (which here goes through highly instable periods). The aim of the entropy density is hence to use these informations.



If we consider $exp(f)$, and take its derivative when $df = 0$ w.r.t $\int p(x)$:

$$df = 0 \implies p(x) = exp(-1 + \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)$$

This result has been shown by Jaynes in 1957.

Injecting $p(x)$ into (1) :

$$\int_A exp(-1 + \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)dx = 1$$

by taking $\lambda^* = 1 - \lambda_0$

$$\int_A exp(\lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)dx / exp(\lambda^*) = 1$$

$$\implies \int_A exp(\lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)dx = exp(\lambda^*)$$

We recall that $p(x) = exp(-\lambda^* + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)$, and finally we get :

$$p(x) = \frac{exp(\lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)}{\int_A exp(\lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4)dx} \tag{2}$$

$P(x)$ appears to take the form of a Boltzmann distribution, hence the denominator of (2) is a potential function in this context, i.e. a special case of normalizing constant.

However, as shown by Rockinger and Jondeau(2002), it is useful to consider the following optimization program that is still equivalent to (2). By taking $\lambda^{**} = \lambda^* - \sum_i \lambda_i(b_i)$ and thus $p(x) = exp(-\lambda^{**} + \sum_i \lambda_i(x^i - b_1))$ we obtain :

$$exp(\lambda^{**}) = \int_A exp(\sum_i \lambda_i(x^i - b_1))dx \tag{3}$$

so that

$$p(x) = \frac{exp(\sum_i \lambda_i(x^i - b_1))}{\int_A exp(\sum_i \lambda_i(x^i - b_1))dx} \tag{4}$$

The Minimization of $Z(\lambda_i) \propto \int_A exp(\sum_i \lambda_i(x^i - b_1))dx$ will affect value for the λ_i that will be reinjected in $p(x)$ in order to get the entropy density. Before going further into details, we underline some useful results about $Z(_i)$. We define the derivatives of the critical points of Z :

$$grad(Z) = 0 \implies \frac{dZ}{d\lambda_i} = \int_A (x^i - b_i)exp(\sum_i \lambda_i(x^i - b_1))dx = \int_A (x_i - b_i)p(x)dx = 0$$

Therefore, the minimization of Z will bring a solution to maximize the entropy under the constraints defined before.

Moreover, the Hessian matrix of Z takes the form of a variance-covariance matrix, symmetric and positive by definition. $H(Z(x))$ is full of rank so that $H(Z(x))^{-1}$ exists :

$$H(Z) = \frac{d^2 Z}{d\lambda_i d\lambda_j} = \int_A (x_i - b_i)(x_j - b_j)p(x)dx$$

A matrix full of rank ensures that either a unique solution exists, or an infinity of those. Here, H is of rank $N = dim(\lambda) = 4$ i.e. the number of unknown parameters λ_i.

3 Methodology and computation

3.1 Numerical approximations

At this stage, we face two numerical issues : computing the integral $Z(x)$ and find its minimum.

In order to get rid of the integral form of this function, several approximation methods could be helpful. Here, we make use of the Gaussian quadrature. This type of approximation transforms an integral into a sum in the following way :

$$\int_{-1}^{1} y(l_i)dl = \sum y(l_i)w(l_i) \tag{5}$$

where l are the quadrature points and w the associated weights. One would choose the degree of approximation n, i.e. the number of points and associated weights($card(l) = n$), and compute the approximation in [-1,1]. When a random variable x is defined on a set [a,b], y becomes a function of a transformed variable l, defined as $x = 1/2(l(a-b)+(a+b))$. This linear transformation will make possible the computation of

$$\int_{-b}^{a} y(x_i)dl = \sum y(x_i)w(x_i) \tag{6}$$

Once done, we redefine the function $Z(x)$:

$$Z(\lambda_i) = \sum_j exp(\sum_i \lambda_i(x_j^i - b_1))w_j \tag{7}$$

for $j = 1$ to n, $i = 1$ to 4.

In order to find a minimum for the function, it is useful here to consider the Newton Raphson algorithm, which works this way :

The Taylor expansion of order 1 of $f(x)$ gives

$$f(x_1) = f(x_0) + f'(x_0)(x_1 - x_0) \tag{8}$$

By taking its derivative, and solving for $f'(x_n) = 0$

$$x_n = x_{n-1} - f'(x_{n-1})/f''(x_{n-1})$$

, or

$$x_n = x_{n-1} - grad(x_{n-1})/H(x_{n-1})$$

We got the solution if the algorithm is asked to stop once the gradient of x_{n-1} is small enough ($< 1.e-5$ for example), i.e. when $minf(x)$ is reached.

This algorithm does not provide a global solution. Hence a starting point near the solution is needed. For this purpose, either one should guess a good initial value, or would plot the function. Here, the function Z is impossible to draw, so we choose as a starting value the vector $(0, 0, 0, 0)$. We recall that the aim is to reduce at the maximum the weight of the constraints in order to get the maximum entropy under the laters. If a solution exists, it should be in the neighborhood of this starting vector.

3.2 Existence of a solution

Jondeau al. have been looking for a domain of skewness and kurtosis where the entropy density admits a solution (figure below). However, while using non-time varying skewness and kurtosis, every empricial values of these parameters in this work are in the authorized domain. This might be helpful if one would achieve a time-varying parameters entropy density.

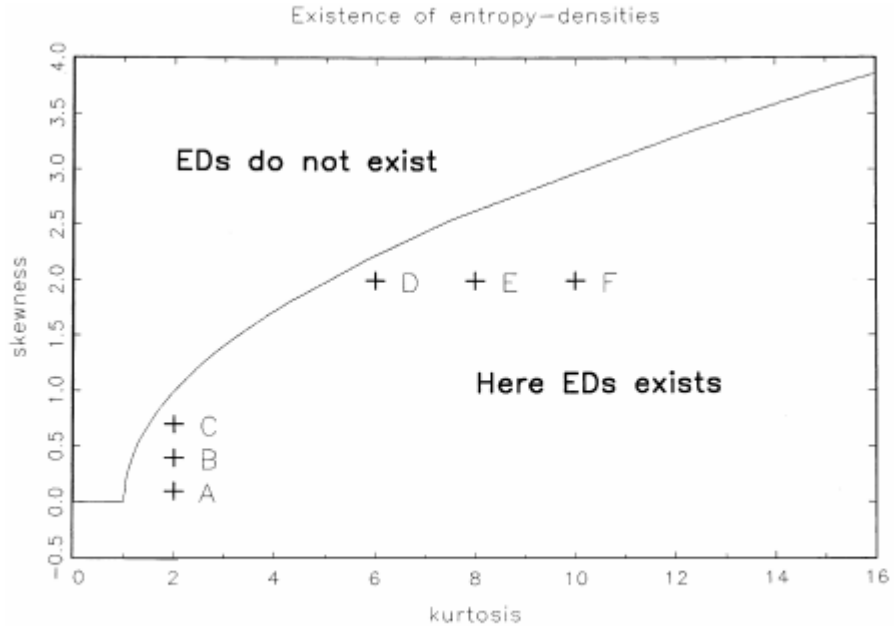


Fig. 1. Here we represent the frontier delimiting the skewness and kurtosis domain where entropy densities exist. This graph represents only the upper half of the authorized domain.

3.3 Algorithm

This algorithm is designed to find the value of the λ vector. It makes use of one sub-code provided by the matlab community (legzo) . This code is available in Annex one.

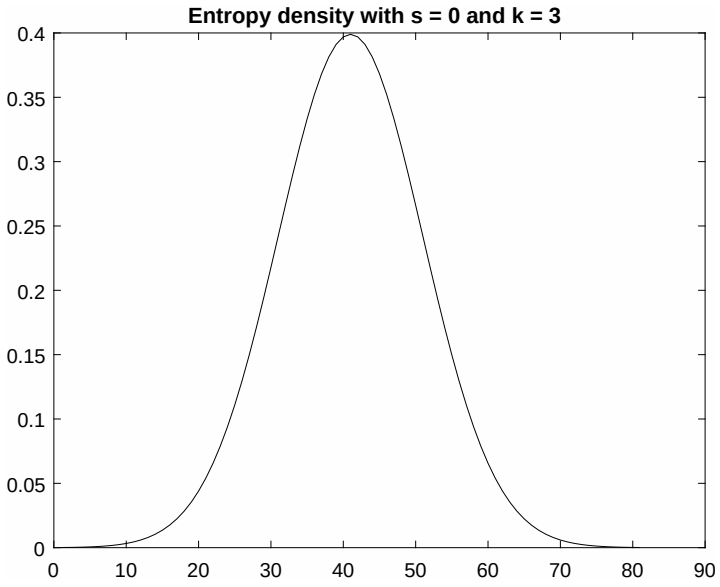
```
%usage for s = -0.28 and k = 4.68
>>x = [-5:0.1:5]
>> y = EDproba2(x,-0.28,4.68)

function p = EDproba2(v,s,k)
format shortG;
Xinit = zeros(4,1); %initial points for the fmincon function

options = optimoptions('fmincon',...
    'Algorithm','sqp-legacy','Display','iter','ConstraintTolerance',1e-30);
```

```
%the minimization function
[xopt,fval,exitflag,output,lambda,grad,hessian] =...
    fmincon(@fct,Xinit,[],[],[],[],[],[],[],options);

function [f] = fct(Xinit)

N = 4;  % select number of variables

n=40; %order of Gauss-Legendre quadrature
[z,w]=legzo(n, -6,6); %function that returns
%weights and points of G-L quadrature

Mean = 0;
Variance = 1;
Skewness= s;
Kurtosis=  k;

A = z;
B= zeros(n,4);

B(:,1)= A';
B(:,2)= A'.^2;
B(:,3)= A'.^3;
B(:,4)= A'.^4;


for i = 1:n
C(i,1)= Mean;
C(i,2)= Variance;
C(i,3)= Skewness;
C(i,4)= Kurtosis;
end
```

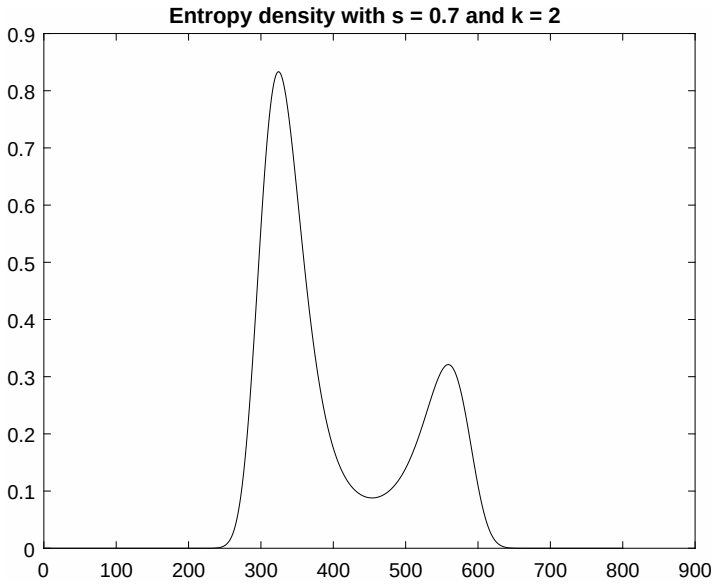
```
D = B - C;
u = (D*Xinit);

f = sum((w)*exp(u)); %the function to minimize

end

%Compute the probability of v

p = exp((xopt(1)*(v-Mean) + xopt(2)*((v.^2) - Variance)...
 + xopt(3)*((v.^3) - Skewness)...
    +xopt(4)*((v.^4) - Kurtosis))) ...
  /fval;

end
```

3.3.1 Graphics

For particular values of the constraints, the entropy density is the reduced centered normal distribution.



For instance, some particular values of the constraints gives bimodals distributions.



4 Garch(p,q) process and entropy density

Jondeau and Rockinger tried the use of an E.D. instead of different densities in a generalized autoregressive conditionnal heteroscedasticity model with time varying skewness and kurtosis. The work presented here is clearly inspired from this previous work. In order to propose a continuity to their work through AR and ARMA estimations, the first step was to try a GARCH estimation using entropy densities as to understand their work. Hence, the following tables show a partial replication of their work (using time series that have not been used in their work and non time-varying skewness and kurtosis).

4.1 Model and results

We make use here of a Garch(1,1) model : $\sigma._t^2 = a + by._{t-1}^2 + \sigma._{t-1}^2$, with $y_t = e_t\sigma_t$ and e following an entropy density of mean 0, variance 1, and empirical skewness and kurtosis.

The algorithm used is similar to the one of classical maximum log-likelihood estimation algorithm, with an Log-likelihood function defined as $log(p(x)) = ln(ed(x)) - ln(\sigma_t)$, with the ending term that comes from the fact that $e_t = y_t/\sigma_t$.

In both cases, we observe that the E.D. performs better than the normal one.

Normal	parameter	param	s.e	t-student	p-value	N. of observations	Value of likelihood
	1	0.0153	0.0047	3.2792	0.0000	8105.0000	-10065.2392
	2	0.0896	0.0171	5.2349	0.0000		
	3	0.8970	0.0174	51.4933	0.0000		
Entropy	parameter	beta	stderr	t-student	p-value	N. of observations	Value of likelihood
	1	0.0326	0.0039	8.4179	0.0000	8105.0000	-9468,13
	2	0.1207	0.0073	16.6331	0.0000		
	3	0.8793	0.0079	111.7103	0.0000		

Table 1: CAC40 1971-2001

Normal	parameter	param	s.e	t-student	p-value	N. of observations	Value of likelihood
	1	0.0155	0.0023	6.7812	0.0000	8105.0000	-9766.9633
	2	0.1002	0.0080	12.5441	0.0000		
	3	0.8856	0.0089	99.7774	0.0000		
Entropy	parameter	beta	stderr	t-student	p-value	N. of observations	Value of likelihood
	1	0.0539	0.0056	9.7072	0.0000	8105.0000	-9150,9
	2	0.1714	0.0110	15.6139	0.0000		
	3	0.8286	0.0111	74.6926	0.0000		

Table 2: DAX 1971-2001

5 ARMA(p,q) process and entropy density

5.1 Maximum likelihood estimation

Here we try to estimate an ARMA process using the entropy density. While it is more common to estimate this type of process assuming than the error terms follow a normal density, we define an alternative ARMA process with errors distribued according to an entropy density. The idea is to use the four first moments of some returns, estimate their entropy density with respect to these moments, and to estimate the ARMA process using the newly estimated density.

Of course, the available packages used to study time series use gaussian error for a huge majority. Some allow for other distributions (t-student mainly), and none used the entropy density. For this work's purpose, an algorithm has to be implemented. We explain the algorithm used to compute the ARMA(p,q) process.

We wish to estimate the following model :

$$x_t = \theta_1 x_{t-1} + ... + \theta_p x_{t-p} + \alpha_1 \epsilon_{t-1} + ... + \alpha_p \epsilon_{t-p}$$

with ϵ the innovation terms that follows an entropy density of mean zero and variance one.

First, the model is represented in a state space form :

$$x_t = \theta_t x_{t-1} + \alpha_t \epsilon_t$$

$$y_t = z_t x_t = z_t(\theta_t x_{t-1} + \alpha_t \epsilon_t)$$

then a Kalman filter recursion is started with $E(x_t) = 0$. The aim is to estimate a *filtered* (corrected) value of $E(x_t|x_{t-1})$ that would be $E(x_t|e_t)$.Say $T = 3$ and we want to estimate$E(x_2|e_2)$. The model is cut in two parts :

the predictive part :

$$E(x_2|x_1) = \theta_2 E(x_1|e_1)$$

$$cov(x_2, x_2|x_1) = \theta_2^2 cov(x_1, x_1|\epsilon_1) + \alpha_2^2$$

and the updating part :

we start with an observation y_2 that defines the innovation

$$\epsilon_2 = y_2 - z_2 E(x_2|x_1)$$

$$var(\epsilon_2) = (z_2^2) cov(x_2, x_2|x_1)$$

we compute the Kalman gain, that will be added to $E(x_2|x_1)$

$$K_2 = cov(x_2, x_2|x_1) \times z_2 \times var(\epsilon_2)^{-1}$$

and finally estimate a new version (update) of $E(x_2|x_1)$ and $Var(x_2|x_1)$

$$E(x_2|e_2) = E(x_2|x_1) + K_2\epsilon_1$$

$$cov(x_2, x_2|\epsilon_2) = cov(x_2, x_2|x_1) - K_2 \times z_2 \times cov(x_2, x_2|x_1)$$

Thus, with an initial value of $E(x_1|x_0) = E(x_t) = 0$, and an initial $cov(x_1, x_1|x_0)$ value we could estimate a filtered version of x_t.

Once the ARMA process is defined, a log-likelihood function is defined as

$$logf(y_t|y_{t-1}; \theta) = \sum_{t=1}^{T} (\lambda_1 e_t + \lambda_2(e_t^2 - 1) - \lambda_3(e_t^3 - Skewness) + \lambda_4(e_t^4 - Kurtosis) - log(Z(\lambda))$$

6 Data and results

The main code to estimate ARMA coefficients(written by Constantino Hevia. August 2008 and modified for this work) can be found in Annex 1. The data used are filtered : extreme values are dropped from the series (between 15 and 30 values depending on the series), thus we keep a range of $(-5 : 5)$. In a second step, values are centered and reduced (the SP500's values in the descriptive statistics are obtained after the division by the std.error).

	Cac40	Dax	SP500
Median	-0.039379	-0.030529	-0.031556
Minimum	-4.736	-4.8752	-5.4747
Maximum	4.9235	4.7743	5.4565
Skewness	-0.15852	-0.1218	0.11741
Kurtosis	5.2148	4.6224	5.5183
t(sk)	-5.826	-4.4764	4.3149
t(ku)	40.6993	29.8131	46.2754
J.B	1690.3735	908.8578	2160.0269

For each series, we proceed in two ways : first, the density of the whole time series is used (1970-2001) to estimate ARMA or AR model depending on what is more accurate. Secondly, smaller windows of time are used (500 values) with two entropy densities : the one constructed with the skewness and kurtosis of the whole series (first step), and another one using the smaller window's skewness and kurtosis. The goal is to observe which of the global entropy density or the adaptive density performs better. For each step, we compare the estimations to the ones from the normal density log-likelihood.

The first three estimations show that using the entropy density gives more accurate results. We observe higher log-likelihood value for the three estimations. The last three estimations show that an adaptive density seems to perform better than the global density, that in turn performs better than the normal one except for the SP500. This case could inform about an over-fitting problem. The results of the first model's

	Entropy	Normal
AR(1)	0.89063	-0.779502
s.e	0.13569	0.11371
t-stat	6.5635	-6.85516
AR(2)	0.063768	-0.3052
s.e	0.10178	0.0502742
t-stat	0.62655	-6.07071
MA(1)	-0.83603	0.929012
s.e	0.33335	0.113098
t-stat	-2.508	8.21424
MA(2)	-0.22182	0.39954
s.e	0.0439	0.0486645
t-stat	-5.0529	8.21008
Log-Lik	-11172	-11400

Table 3: CAC40 1971-2001

estimations are surprising, the AR coefficients turn positive with the entropy density while negative with the normal density, as the MA coefficients that turn out to be positive (negative) with the E.D. (normal density). Moreover, the AR(2) coefficient of the E.D. estimation is not significant anymore.

	Entropy	Normal
AR	-0.22349	-0.270935
s.e	0.046238	0.119431
t-stat	-4.8334	-2.26854
MA	0.28711	0.335984
s.e	0.046702	0.11658
t-stat	6.1476	2.88201
Log-Lik	-11286	-11466

Table 4: DAX 1971-2001

	Entropy	Normal
AR(1)	0.094875	0.0976517
s.e	0.0099118	0.0087428
t-stat	9.5719	11.1694
AR(2)	-0.016541	-0.0170378
s.e	0.0091852	0.00877645
t-stat	-1.8008	-1.9413
Log-Lik	-11274	-11460

Table 5: SP500 1971-2001

	Entropy Adaptive	Normal	Entropy
AR(1)	0.074663	0.146843	0.078268
s.e	0.03385	0.0257281	0.03549
t-stat	2.2057	5.7075	2.2054
Log-Lik	-678.1	-703.62	-708.93

Table 6: SP500 500 values, s = 0.74044, k = 8.3975

	Entropy Adaptive	Normal	Entropy
AR(1)	0.16571	0.184778	0.15874
s.e	0.042307	0.0389167	0.041381
t-stat	3.9169	4.74803	3.8362
Log-Lik	-619.34	-656.6	-639.32

Table 7: CAC40 500 values, s = -0.69248, k = 7.4604

	Entropy Adaptive	Normal	Entropy
AR(1)	0.14157	0.122866	0.14573
s.e	0.047203	0.0385955	0.04761
t-stat	2.9992	3.18343	3.0609
Log-Lik	-564.21	-601.48	-583.29

Table 8: DAX 500 values, s =0.3359 , k = 6.6838

7 Value at risk/Expected Shortfall

We hope to extend the use of the entropy density to Value at risk and Expected Shortfall estimations. These values may help to appreciate the goodness of fit of the entropy density, conditional on the four moments of a return serie.

We define the Value at Risk m% as $P(r > Var) = m\%$ and the expected shortfall as $E(r|x < Var)$. For this purpose, VaR and ES are extracted from the SP500 return serie between 1970 and 2001. Then, 10000 random variables are generated according to a normal density then to the entropy density estimated with the first four moments of the SP500's returns.

As with the ARMA and AR estimations, a global entropy density and an adaptive one are used through different windows of time (500, 2000 and 6000 values). To this purpose, we used the non-filtered data sets of the SP500.

In order to generate random number with a given entropy distribution, we are in need of a particular algorithm that works the following way :

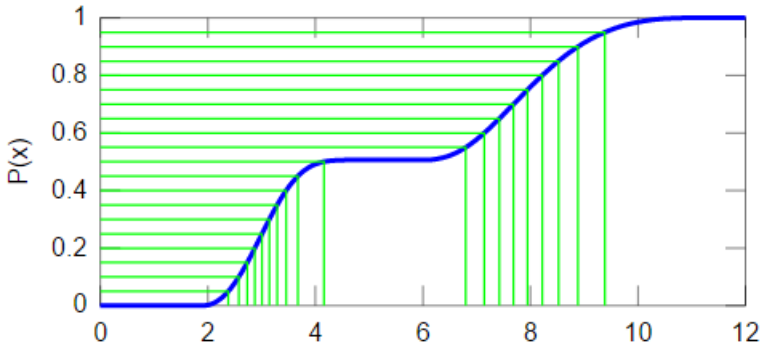
- we compute the probabilities of $x \in [a; b]$ according to a given E.D. The goal is to have enough probabilities to achieve a right smoothing of the density in the second step.

- we smooth the density with an interpolation. Let us say we computed $x \in [-4; 4]$ with a step of 0.1 between each x. The goal is to obtain $p(x)$ with $x \in [-4; 4]$ and a smaller step between each point, 0.01 for example.

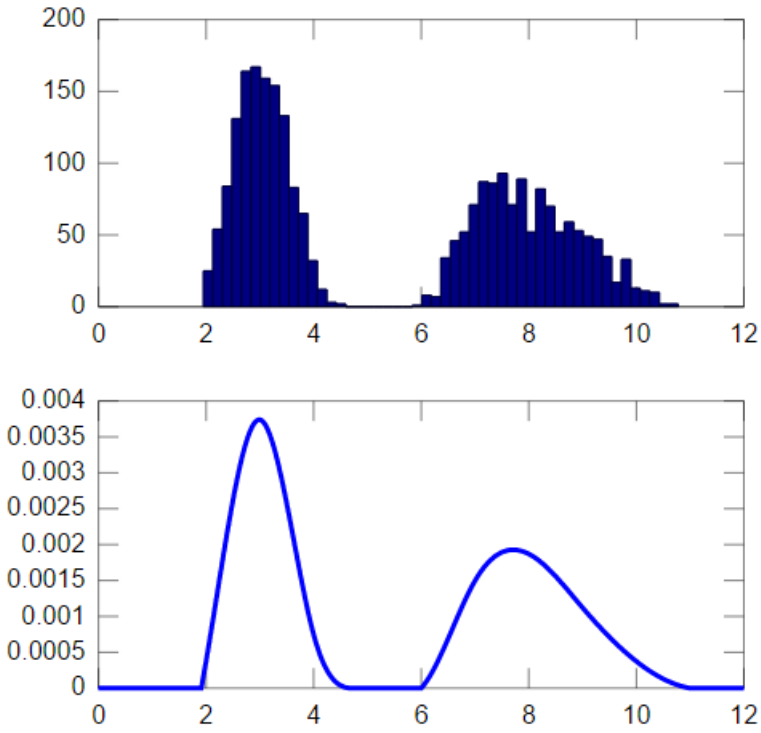
- with the integral of the obtained density, we can construct the cumulative distribution function.

- we generate random numbers between 0 and 1 (uniformly distributed) that we project on the domain of x through the cumulative distribution function. The more the C.D.F. is increasing, the more we get values on the x region. The figure below

illustrates this step.



To see the result, we plot an histogram of the projected value. We can see below the histogram with the original density function that goes with the previous C.D.F



The next three tables are obtained using an unique entropy density computed with the skewness (-1.7192) and the kurtosis (45.9414) of the whole data set (8104 values). The extreme values have been incorporated.

Concerning VaR value at 90% and 95% confidence interval, the entropy density outperforms the Gaussian one for all windows of time used here. Especially, it succeeds in taking in account the asymmetric caracteristic of the returns, as it could be seen for the VaR 95% values, except for the windows of time that contains 2000 values, whose empirical VaR 95% does not make appear a notable asymmetry. E.D. is also more precise to estimate C-VaR 90%, except for the right tail value of the 500 lenght window. One may think that the whole time window's density overfits the density of a smaller value of time. We recall that this issue appears to be one of the remained question : should we consider the moments of a particular set of return or the ones from a bigger period ? This would need an hypothesis that states a convergence toward a particular density, which is a conclusion impossible to generalize here, even if the previous study of AR-ARMA models showed that computing each time a new density may be a good direction to take. E.D. never outperforms the normal density in the C-Var 95% context. The concentration of the density around the mean reduces its ability to replicate extreme values. By the way, the computation of the E.D. with its constraint may be improved through other forms of constraints that may help to avoid this issue (constraints of inequality, higher moments,etc).

SP500 (500 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.1288	1.0843	-1.2787	1.2798	-1.2241	1.2304
VaR 95%	-1.5544	1.5809	-1.6405	1.6348	-1.5478	1.6015
C-VaR 90%	-1.697	1.9213	-1.7597	1.7491	-1.6582	1.7241
C-Var 95%	-2.0832	2.5617	-2.0585	2.0604	-1.9259	2.0441

SP500 (2000 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.0753	1.0328	-1.2787	1.2798	-1.2241	1.2304
VaR 95%	-1.4643	1.4604	-1.6405	1.6348	-1.5478	1.6015
C-VaR 90%	-1.6254	1.7003	-1.7597	1.7491	-1.6582	1.7241
C-Var 95%	-1.9957	2.1724	-2.0585	2.0604	-1.9259	2.0441

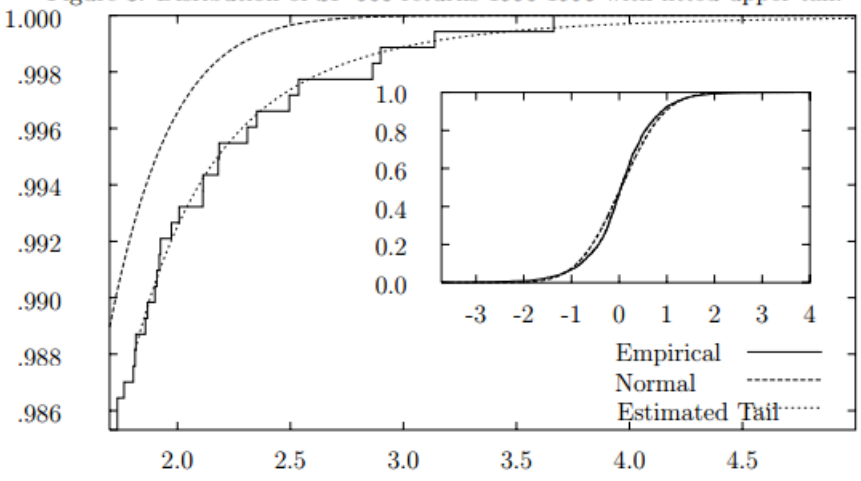
SP500 (6000 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.0219	1.0772	-1.2787	1.2798	-1.2241	1.2304
VaR 95%	-1.4212	1.4844	-1.6405	1.6348	-1.5478	1.6015
C-VaR 90%	-1.6491	1.6948	-1.7597	1.7491	-1.6582	1.7241
C-Var 95%	-2.1007	2.1304	-2.0585	2.0604	-1.9259	2.0441

For the next three tables, we make use of several densities : the ones computed with the windows lengths skewness and kurtosis. For the first and third tables, the previous observations made for the first three tables could be assessed again, except that we observe slightly better estimation of C-Var 95%. However, the 2000 values window of time, outperforms the normal density in estimation C-Var 95% while it is less precise in estimating right tail Var 95% and C-Var 90%. The density computed here may overestimate the asymmetry of the empirical values. If we compare the two kind of entropy density, adaptive E.D. appears to be more precise here (than the E.D. used before) for the VaR 90% (left and right tails) and VaR 95% (left tails) for the first and third table, as for the C-VaR 90%(right tails) and C-VaR 95% (right tails). The density computed for the second table seems to overfit the true distribution of the returns, except for the C-Var 95% (right tails), where it outperforms the normal density.

Furthermore, as stated by Jon Danielsson and Casper G. de Vries in their paper *Value–at–Risk and Extreme Returns*, a major issue is the discreteness of extreme returns with this kind of simulation. Here, the sampling distribution is dense, reducing the space between each values, hence the interval that could separate 90% of the values and extreme ones is close, whereas it is not the case for empirical values.

The next figure shows the empirical cumulative distribution function for extreme value. One should notice that the empirical c.d.f. is not smooth and reflects this interval problem between adjacent returns that biases the simulation's results.

Figure 3: Disribution of SP-500 returns 1990-1996 with fitted upper tail.



In conclusion, the use of an adaptive density may be helpfull to improve returns distributions, but one might be careful to the overfitting case that could occur as for the second table and to methodology issues.

SP500(500 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.1288	1.0843	-1.2787	1.2798	-1.2263	1.2313
VaR 95%	-1.5544	1.5809	-1.6405	1.6348	-1.5445	1.6149
C-VaR 90%	-1.697	1.9213	-1.7597	1.7491	-1.64	1 .7555
C-Var 95%	-2.0832	2.5617	-2.0585	2.0604	-1.9075	2.1038

Table 9: Adaptive density (500 values) with skewness = 0.74044 and kurtosis = 8.3975

SP500(2000 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.0753	1.0328	-1.2787	1.2798	-1.2457	1.2648
VaR 95%	-1.4643	1.4604	-1.6405	1.6348	-1.5756	1.6628
C-VaR 90%	-1.6254	1.7003	-1.7597	1.7491	-1.671	1.7964
C-Var 95%	-1.9957	2.1724	-2.0585	2.0604	-1.9467	2.1467

Table 10: Adaptive density (2000 values) with skewness = 0.29119 and kurtosis = 5.7101

SP500(6000 values)	Empirical		Normal		Entropy	
	LT	RT	LT	RT	LT	RT
VaR 90%	-1.0219	1.0772	-1.2787	1.2798	-1.1841	1.2278
VaR 95%	-1.4212	1.4844	-1.6405	1.6348	-1.5074	1.611
C-VaR 90%	-1.6491	1.6948	-1.7597	1.7491	-1.597	1.7269
C-Var 95%	-2.1007	2.1304	-2.0585	2.0604	-1.8696	2.054

Table 11: Adaptive density (6000 values) with skewness = -2.2182 and kurtosis = 59.212

8 Conclusion

The analysis of time series with the entropy density has given good results, according to this work. It may be a further step to a better understanding, if not, a better estimation of the studied returns in opposition with the normal density context. Moreover, it is clear here that we make use of inference more than deduction, and this may bring this work into some epistemological ground easily, even if it is not the purpose here. The entropy density helps to achieve better AR/ARMA estimations, while it illustrates in a better way some particularity of the returns' distributions, as it is seen in the computation of VaR and C-Var. However, its use could be long to implement. It is in need of some specially designed algorithms that could not been found in the widely used statistical or mathematical softwares's included packages. This is why its comprehension goes through a three dimensions work : the one of the domain on which we apply it and its potential links with the notion of entropy, a mathematical one, and finally and algorithmic one.

Bibliography

Michael Rockinger, Eric Jondeau,2001, *Entropy densities with an application to autoregressive conditional skewness and kurtosis*, Journal of Econometrics 106 (2002) 119–142

Andreia Dionisio,Rui Menezes and Diana A. Mendes, 2007, *Entropy and Uncertainty Analysis in Financial Markets*,University of Evora, Center of Business Studies

Georges A. Darbellaya,Diethelm Wuertzc, 2000, *The entropy as a tool for analysing statistical dependences in financial time series*,Physica A 287 (2000) 429-439

E. T. Jaynes, 1991, *HOW SHOULD WE USE ENTROPY IN ECONOMICS? (Some half-baked ideas in need of criticism)*, St. John's College Cambridge CB2 1TP, England

E. T. Jaynes, 1996, *MACROSCOPIC PREDICTION*,Washington University

Christopher A. Zapart, *Forecasting with Entropy*,The Institute of Statistical Mathematics (Japan)

Abdullah Z. Sheikh, Hongtao Qiao, *Non-normality of Market Returns*, J.P. Morgan Asset Management—Strategic Investment Advisory Group

Matthew Richardson,Tom Smith, *A Test for Multivariate Normality in Stock Returns* , The Journal of Business, Vol. 66, No. 2 (Apr., 1993), pp. 295-321.

Todd Young and Martin J. Mohlenkamp, *Introduction to Numerical Methods and Matlab Programming for Engineers*,Ohio University

In Jae Myung, *Tutorial on maximum likelihood estimation*, Department of Psychology, Ohio State University

A. Galantai, *the theory of Newton's Method*, Institute of Mathematics, The University of Miskolc, 3515 Miskolc-Egyetemvaros, Hungary

Rongxi Zhou, Ru Cai and Guanqun Tong, *Applications of Entropy in Finance: A Review*,School of Economics and Management, Beijing University of Chemical Technology

P.A. Bromiley, N.A. Thacker and E. Bouhova-Thacker, 2010, *Shannon Entropy, Renyi Entropy, and Information*, School of Cancer and Imaging Sciences, The University of Manchester

Jon Danielsson,Casper G. de Vries, 2000, *Value–at–Risk and Extreme Returns*,London School of Economics

Constantino Hevia, 2008, *Maximum Likelihood Estimation of an ARMA(p,q) Model*, The World Bank

Dan Simon, 2001, *Kalman Filtering*, Embedded Systems Programming

9 Annex

GenerateRand, Author : Zoltan Fegyver, http://matlabtricks.com/post-44/generate-random-numbers-with-a-given-distribution.

Legzo (Gauss-Legendre quadrature) : Matlab community

AR ARMA code, Author : Constantino Hevia (modified for this work).

```
function [res,resu] = GenerateRand
format shortG;
% probability density function is based on this data
probabilities = [
   %write p(x) for x = -4:0.1:4

];
x = -4:0.1:4;

% do spline interpolation for smoothing
xq = -4: 0.01 :4;
pdf = interp1(x, probabilities', xq, 'spline');

% remove negative elements due to the spline interpolation
pdf(pdf < 0) = 0;

% normalize the function to have an area of 1.0 under it
pdf = pdf / sum(pdf);

cdf = cumsum(pdf);
plot(cdf);
% these two variables holds and describes the CDF
```

```
xq;          % x
cdf;         % P(x)

% remove non-unique elements
[cdf, mask] = unique(cdf);
xq = xq(mask);

% create an array of 2500 random numbers
randomValues = rand(1, 100000);


% inverse interpolation to achieve P(x) -> x projection of the random values
projection = interp1(cdf, xq, randomValues);
disp(projection)
```

```
function [x,w]=legzo(n, a, b)
%       ==========================================================
%       Purpose : Compute the zeros of Legendre polynomial Pn(x)
%                 in the interval [a,b], and the corresponding
%                 weighting coefficients for Gauss-Legendre
%                 integration
%       Input :   n    --- Order of the Legendre polynomial
%                 a    --- Lower boundary (optional)
%                 b    --- Upper boundary (optional)
%       Output:   x(n) --- Zeros of the Legendre polynomial
%                 w(n) --- Corresponding weighting coefficients
%       ==========================================================
if nargin == 1
    a = -1;
    b =  1;
```

```
end;
x = zeros(1, n);
w = zeros(1, n);
m = (n+1)/2;
h = b-a;

for ii=1:m
    z = cos(pi*(ii-.25)/(n+.5)); % Initial estimate.
    z1 = z+1;
    while abs(z-z1)>eps
        p1 = 1;
        p2 = 0;
        for jj = 1:n
            p3 = p2;
            p2 = p1;
            p1 = ((2*jj-1)*z*p2-(jj-1)*p3)/jj; % The Legendre polynomial.
        end
        pp = n*(z*p1-p2)/(z^2-1); % The L.P. derivative.
        z1 = z;
        z = z1-p1/pp;
    end
    x(ii) = z; % Build up the abscissas.
    x(n+1-ii) = -z;
    w(ii) = h/((1-z^2)*(pp^2)); % Build up the weights.
    w(n+1-ii) = w(ii);
end

if a ~= -1 || b ~= 1
    x = (x+1)*(h/2) + a;
end
```

```
function results = EDarma_mle( y, p, q, info )

% ARMA_MLE.M
% results = armae_mle(y,p,q,[info])
% This function computes MLE estimates of the (demeaned) ARMA(p,q) model
%
%  y[t] = phi(1)y[t-1] + ... + phi(p)y[t-p] + e[t] + theta(1)e[t-1] +
%         ... + theta(q)e[t-q]
%
% The algorithm computes the exact likelihood function using the Kalman
% filter. It initializes the ARMA coefficients using a method of Hannan and
% Kavalieris.
%
% INPUTS:
% y      : time series of size T with the observations
% p      : autoregressive order of the ARMA model
% q      : moving average order of the ARMA model
% info   : show optimization info =0 no, /=0 yes. If this input is omitted,
%          it is set to 0.
%
% OUTPUT:
% result: structure with the ARMA estimates
%
% REFERENCES:
% 1) E.J. Hannan and L. Kavalieris. "A Method for Autoregressive-Moving
%       Average Estimation" Biometrika, Vol 71, No2, Aug 1984.
% 2) E.J. Hannan and A.J. McDougall. "Regression Procedures for ARMA
%       Estimation" Journal of the American Statistical Association, Vol
%       83, No 409, June 1988.
% 3) R.H. Jones : Maximum Likelihood Fitting of ARMA Models to
%       Time Series With Missing Observations" Technometrics, Vol 22, No3,
%       August 1980.
```

```
%
% Written by Constantino Hevia. August 2008

if nargin<4; info = 0; end

% Check data:
[m,n] = size(y); if n>1; T = n; y = y'; else T = m; end
r = max(p,q+1);

% Set optimization options:
if info == 0;
    options = optimoptions('fmincon',...
    'Algorithm','sqp-legacy','Display','iter','ConstraintTolerance',1e-30);
else
    options = optimset('Algorithm ','off','Display','iter');
end

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% INITIALIZE COEFFICIENTS USING Hannan and McDougall (1988) -see below:
coefs = initialize_arma;

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% COMPUTE MLE ESTIMATES USING THE INTERNAL FUNCTION loglikelihood AND THE
% OPTIMIZATION ROUTINE fminunc

lb=ones(size(coefs));
lb = -lb.*0.99;
ub=ones(size(coefs));
ub = ub.*0.99;
[coefs1,fval,exitflag,output,lambda,grad,hessian] =...
    fmincon(@log_likelihood,coefs,[],[],[],[],lb,ub,[],options);
```

```
% Estimate standard deviation of ARMA
[ L sigmahat] = log_likelihood( coefs1 );

results.ar      = coefs1(1:p);
results.ma      = coefs1(p+1:p+q);
results.loglik = -L;

%%se and t-student

inv_h=-hessian\eye(size(hessian,1));
vc = -inv_h;
stderr  = sqrt(diag(vc));
disp(stderr);
tstudent= coefs1./stderr;
disp(tstudent);
    % ---------------------------------------------------------------------
    % INTERNAL FUNCTIONS

    function coefs = initialize_arma
        % This function uses Hannan and McDougall (1988) to initialize the
        % coefficients of the ARMA process using a regression approach

        % Step 1: Choose order of autoregression.
        best = 1e+10;
        for h = p : ceil(log(T)^1.5)
            Y = y(h+1:T);
            X = [];
            for j=1:h
                X = [X y(h+1-j : T-j)];
            end
            bet = (X'*X)\X'*Y;
            res = Y - X*bet;
```

```
        sigma2 = (res'*res)/length(res);
        BIC = log(sigma2) + h*log(T)/T;
        if BIC < best
            best = BIC;
            horder=h;
            residuals = res;
        end
    end

    % Step2: Run regression on lagged values and residuals
    nlag = max(p,q);
    Y = y(horder+nlag+1:T);
    X = [];
    for i=1:p
        X = [X y(horder+nlag+1-i:T-i)];
    end
    for i=1:q
        X = [X residuals(nlag+1-i:T-horder-i)];
    end
    bet = (X'*X)\X'*Y;
    coefs(1:p) = bet(1:p);
    coefs(p+1:p+q) = bet(p+1:p+q);
    coefs = coefs';
end



function [ L sigmahat] = log_likelihood( coefs )
    % function [L sigmahat] = log_likelihood(coefs)
    % This function computes the negative of the log-likelihood of an
    % ARMA(p,q) model:
    %
    %  y[t] = phi(1)y[t-1] + ... + phi(p)y[t-p] + ...
    %         + e[t] + theta(1)e[t-1] + ... + theta(q)e[t-q]
```

```
%
% where phi     = coefs(1:p)
%       theta   = coefs(p+1:p+q)

phi   = coefs(1:p);
theta = coefs(p+1:p+q);

% Build matrices of state space representation:
%
%       x[t+1] = A x[t] + R eps[t+1]
%       y[t]   = Z'x[t]
%

A = zeros(r,r); R = zeros(r,1); Z = zeros(r,1);
A(1:p,1)=phi; A(1:r-1,2:r)=eye(r-1);
R(1) = 1; R(2:q+1) = theta;
Z(1) = 1; Q = R*R';

% Initialize state and covariance matrix
xhat_tt  = zeros(r,1);
Sigma_tt = reshape( (eye(r^2)-kron(A,A) ) \ Q(:), r, r );

logsum = 0;
sumsq  = 0;
l = 0;
for i=0:T-1      % Start Kalman Filter Recursion
    xhat_t1t  = A*xhat_tt;
    Sigma_t1t = A*Sigma_tt*A' + Q;
    yhat_t1t  = Z'*xhat_t1t;
    omega     = Z'*Sigma_t1t*Z;
    delta_t1  = Sigma_t1t*Z/omega;
    x      = y(i+1) - yhat_t1t;
    xhat_t1t1 = xhat_t1t + delta_t1*x;
```

```
            Sigma_t1t1= Sigma_t1t - delta_t1*Z'*Sigma_t1t;

            % Add likelihood terms:

            logsum = logsum +  (( -0.015855*x-0.6091*((x.^2)...
             - 1) -0.002542*((x.^3)-0.3359) - 0.010422*((x.^4) - 6.6838)-...
             log( 3.9953)));
            % Update estimates;
            xhat_tt  = xhat_t1t1;
            Sigma_tt = Sigma_t1t1;
        end

        L = -logsum;
        sigmahat = sqrt(sumsq/T);

    end % function log_likelihood
disp(coefs1);

end %function arma_mle
```