LUISS Guido Carli
LIBERA UNIVERSITÀ INTERNAZIONALE DEGLI STUDI SOCIALI

Department of Economics and Finance

Mathematical Finance

Portfolio Optimization using CVaR

Supervisor:

Papi Marco

Student:

Simone Forghieri

170261

2013-14

Abstract

In this thesis we perform the optimization of a selected portfolio by minimizing the measure of risk defined as Conditional Value at Risk (CVaR). The method described is very robust, and allows us to calculate the optimal asset weights while simultaneously minimizing the CVaR and the Value at Risk (VaR).

The future return scenarios used in the portfolio optimization formula are sampled from an estimated distribution, which is the best approximation of the historical distribution of stocks returns. This estimation is conducted on a comparative level by using a Normal distribution, *t*-Location Scale distribution, and Generalized Hyperbolic distribution. By comparing the results of the portfolio optimization using the different distributions, we provide both a graphical and a mathematical proof that the Generalized Hyperbolic distribution delivers the best fit for the real distribution of returns and is the most accurate in minimizing risk and calculating optimal weights.

Table of Contents

1. Introduction

Measures of risk are a crucial part in portfolio optimization, in particular in order to maintain a strict control of risk and expected losses. The numerous publicly known cases of problems in handling risk, from both banks and companies, have raised awareness on the importance of methods and measures to manage portfolio risk. Markowitz was the first to address the portfolio selection problem (H. Markowitz, 1952) as a one-period static setting where maximizing expected return, subject to a constraint on variance. In 2005, the mean-variance problem was solved in a dynamic complete market setting (Bielecki et al., 2005).

The center of research then shifted towards risk measures that focus on the portfolio losses that occur in the tail of the loss distribution, and quantile-based models have become more and popular. One of the most widespread quantile-based risk measures is the Value-at-Risk (VaR). The VaR refers to the worst expected loss at a target horizon, according to a determined confidence level. This value is a quick and easy measure that is frequently used to determine the stop-loss thresholds by traders, or to evaluate risk-adjusted returns by companies (P. Jorion, 1996). The popularity of the VaR was also determined by its inclusion in the Basel II Accords as a primary risk gauge for banks' exposure.

The Value at Risk concept lies however on the assumption that the loss distribution, imagined as a function $z = f(x, y)$ (where x is the decision vector - defined by the current portfolio - and y is the vector of predicted future values), is distributed according to a Normal distribution. This distribution though does not usually occur in reality. In fact, the empirical distribution of the loss function z is mainly characterized by fatter tails than the Normal distribution. This difference between the estimated and the observed distribution leads to biased results, making the VaR fail to

be coherent (P. Artzner et al., 1999). Another shortcoming of the VaR is the fact that the measurement can be done with several legitimate methods, which yield very different results (M. Pritsker, 1997) and make it very inconsistent and unreliable. The measure itself is very limited, as it does not provide any information on the extent of the losses that will be suffered beyond the VaR threshold. It only provides a lower bound for losses, without distinguishing between situations in which losses may be slightly or much higher than the threshold. In addition the VaR calculated with scenarios is a non-convex non-smooth function, with multiple local extrema, which make it a very unsuitable function for optimization models based on minimization.

For the reasons listed above, the VaR has been associated with an alternative measure that aims at quantifying the losses that will be held when they exceed the VaR threshold. This measure is called the Conditional Value at Risk (CVaR), and it is defined as the weighted average of the VaR and of losses strictly exceeding the VaR. Rockafellar and Uryasev were the first to visualize the CVaR concept and develop its minimization formula (R. T. Rockafellar and S. Uryasev, 2000). They demonstrated the effectiveness of CVaR through several case studies, including portfolio optimization and options hedging. The CVaR was then found to have many computational advantages over the VaR, while maintaining consistency with the VaR by yielding the same results in cases where applied to Normal or elliptical distributions (P. Embrechts et al., 2001). In these cases in fact, working with VaR, CVaR, or minimum variance (H. Markowitz, 1952) is equivalent (R. T. Rockafellar and S. Uryasev, 2000). Moreover, the fact that the CVaR function is convex, and its minimization model can be condensed into a simple linear programming formula, make it a widely used and studied area of research and development. On the other side the VaR becomes a rather complex model when applied to more detailed

distributions, making it unsuitable for environments such as the financial world, where computational speed is a necessary condition for the applicability of a model. For this reason, an efficient algorithm for VaR optimization in high-dimensional settings is not yet available, despite the great efforts in research (J. V. Andersen and D. Sornette, 1999; S. Basak and A. Shapiro, 2001; A. A. Gaivoronski and G. Pflug, 2000; C. Gourieroux et al., 2000; H. Grootweld and W. G. Hallerbach, 2000; R. Kast et al., 1998; A. Puelz, 1999; D. Tasche, 1999).

Meanwhile, the CVaR has been studied as both a minimization problem with an expected return constraint, and as a maximization of expected return with the CVaR constraint (P. Krokhmal et al., 2002). Strategies for investigating the efficient frontier between CVaR and return were considered as well. Moreover the concept was applied to credit risk management of a portfolio of bonds (C. Andersson et al., 2000), and extended to the concept of conditional drawdown-at-risk (CDaR) in the optimization of portfolios with drawdown constraints (Checklov et al. - Press). Today's currents in the field flow towards models of CVaR in varying distribution with the simultaneous drop of some of its assumptions. Although the CVaR is not yet a standard in finance, it provides investors with a flexible and strong risk management tool, therefore it will most likely plays a major role in portfolio optimization.

In this thesis we are going to use the original linear programming CVaR optimization model studied by R.T. Rockafellar and S. Uryasev, 2000, focusing on the prediction of future scenarios and their impact on its results.

2. Dataset

The dataset used in this thesis consists in a sample of daily prices from 10 stocks listed in the Nasdaq 100 market index. Prices are taken from the 17/05/2004 to the 16/05/2014, amounting to 2518 observations. The stocks chosen are taken from different sectors and with different capitalizations in order to make the model more generally applicable and effective in varying situations.

For the purpose of this work, stocks are taken from and compared to one single index representing the market, allowing for the prediction of returns using the Normal distribution, the Capital Asset Pricing Model while implementing a fitted t-location scale distribution, and the hyperbolic distribution. Market returns are approximated by the returns of the index in which the stocks are listed; in our case the Nasdaq 100 index was used.

The composition and the characteristics of the sample taken are listed in Table 2.1 and Table 2.2.

	Min	**Max**	**Mean**	**Median**
AAPL	-0.6850	0.1302	0.0012	0.0013
CSCO	-0.1769	0.1480	0.0001	0.0004
MSFT	-0.1246	0.1706	0.0002	0
CMCSA	-0.4179	0.2193	0.0002	0
WFM	-0.6844	0.3166	-0.0003	0.0004
PFE	-0.1182	0.0969	-0.0001	-0.0004
MAR	-0.7029	0.1412	0.0001	0.0006
MNST	-1.4835	0.2323	0.0005	0.0007
SBUX	-0.6792	0.1687	0.0003	-0.0002
FISV	-0.6946	0.1466	0.0002	0.0009

Table 2.1 *– Sample statistics of the dataset, for daily prices.*

	Variance	Standard Deviation	Kurtosis	Skewness
AAPL	0.0007	0.0268	174.9831	-6.7184
CSCO	0.0004	0.0197	13.8786	-0.4002
MSFT	0.0003	0.0171	14.1104	0.0064
CMCSA	0.0004	0.0211	72.5071	-3.0062
WFM	0.001	0.0321	158.0768	-6.9383
PFE	0.0002	0.0152	10.9353	-0.3874
MAR	0.0007	0.0258	222.9072	-8.0153
MNST	0.0023	0.0483	396.2292	-14.1807
SBUX	0.0006	0.0252	215.7112	-7.6391
FISV	0.0004	0.0212	465.5107	-14.0554

Table 2.2 *– More sample statistics of the dataset, for daily prices.*

From these statistics, we can already notice some characteristics of stock returns that can help us predict their distribution. In particular, the mean returns approximate zero, therefore we will expect the cumulative distribution functions to be intersecting the y-axis at values around 50%. The negative values of skewness suggest that the distributions are left-skewed, therefore we expect higher probabilities in the negative returns tail compared to the positive returns tail. Moreover the values of kurtosis are on average very high; therefore we will expect a leptokurtic behavior, meaning higher probability around the mean and fatter tails.

We refer to Appendix 1.1 for the MatLab code related to the calculation of the sample statistics.

3. CVaR Algorithm

In this section we describe the algorithm used to calculate the CVaR and to find the optimal weights by minimizing that value. The portfolio optimization is then solved by using both a general scenario and a more specific one with constraints on expected portfolio return and asset weights.

The first step of the CVaR calculation is to find the matrix of historical returns from the matrix of historical prices. We consider logarithmic returns, which are the preferred method for return calculations in finance (E. Eberlein, 2001), and will make calculations simpler in later stages of the thesis. The general formula for logarithmic returns is the following:

$$r_{\log} = \ln\left(\frac{P_{i+1}}{P_i}\right)$$

Here P_i denotes the initial price of the security, whereas P_{i+1} is the price in the next period. We refer to Appendix 1.2 for the MatLab code related to the calculation of the matrix of logarithmic returns.

We consider the loss function $f(x,y)$, where x is the decision vector (represented by our portfolio), and y is a random vector (representing the future values of the items in the portfolio). Suppose that x belongs to a set of portfolios X that satisfies the given requirements on short selling and expected return, while y represents the uncertainty of future returns. For each x, the loss function $f(x,y)$ can be seen as a random variable characterized by a probability distribution $p(y)$ induced by the

probability distribution of y. This property of the CVaR algorithm is relevant to this work. Since the different models used to derive the probability distribution of returns have an impact on the final results, they will be discussed in the following chapter.

Let the portfolio of assets x be constructed as $x = (x_1, \ldots, x_n)$ where x_j represents our position in instrument j such that:

$$x_j \geq 0 \quad \text{for} \quad j = 1, \ldots, n \quad \text{with} \quad \sum_{j=1}^{n} x_j = 1$$

The random returns is defined as the R^n-valued vector $y = (y_1, \ldots, y_n)$ defined on a given probability space (Ω, F, P), endowed with the σ-algebra of events F and the probability measure P. Here y_j is the future return of instrument j, and it is distributed according to the probability distribution $A \in F \rightarrow P(Y \in A)$, having a continuous density function $p(y)$. As long as $p(y)$ is continuous, also the probability density function of $f(x, y)$ is continuous, allowing for the use of simpler methods for the minimization, see Y. S. Kan and A. I. Kibzun, 1996 and S. Uryasev, 1995.

In order to simplify the presentation, let the portfolio consist of only two assets (Asset 1 and Asset 2). In this case, x is the vector of positions of the two instruments $x = (x_1, x_2)$. Let y be the vector of future returns $y = (y_1, y_2)$. Keep in mind that y includes a certain level of uncertainty, as it expresses either a prediction or expectation of return calculated with future prices. The loss function $f(x, y)$ will be equal to the sum of the product between the weight and the relative return:

$$f(x, y) = x_1 y_1 + x_2 y_2 .$$

Since, in our case, where more assets are included we have:

$$f(x,y) = -[x_1 y_1 + \ldots + x_2 y_2] \, .$$

Let us denote with $\Psi(x,\alpha)$ the probability that $f(x,y)$ does not exceed the threshold α, that is:

$$\Psi(x,\alpha) = \int_{f(x,y)\le\alpha} p(y)\,dy \, ,$$

Where the integral is over R^n. By fixing x, $\Psi(x,\alpha)$ is a function of α that represents the cumulative distribution function for the loss associated with x. This function is fundamental for the definition of VaR and CVaR and, as stated above, we assume that it is continuous with respect to α.

When considering a general case with probability level β, then we could see α as the function $\alpha(x,\beta)$ expressing the percentile of the loss distribution with confidence level β: by definition the VaR. In other words, the VaR is defined as the lowest value such that $\Psi(x,\alpha(x,\beta)) = \beta$:

$$VaR_\beta = \alpha_\beta(x) = \min\left\{\alpha \in R : \Psi(x,\alpha) \ge \beta\right\} .$$

We refer to Appendix 1.3 for the MatLab code related to the calculation of the VaR. If $f(x,y)$ exceeds the VaR (with threshold α) then the expected loss, defined as CVaR (denoted by $\Phi_\beta(x)$), is expressed by:

$$\Phi_\beta(x) = \frac{1}{(1-\beta)} \int_{f(x,y)>\alpha(x,\beta)} f(x,y)p(y)dy .$$

The presence of the VaR function in the CVaR formula makes the model complicated and difficult to handle in the minimization; therefore the approach used in this work handles with a simpler function for the CVaR expression:

$$F_\beta(x,\alpha) = \alpha + \frac{1}{(1-\beta)} \int_{f(x,y)>\alpha} (f(x,y)-\alpha)p(y)dy .$$

This function behaves exactly like the original CVaR function, as it is convex with respect to α, and the VaR is a minimum point of $F_\beta(x,\alpha)$ with respect to α. Moreover, minimizing $F_\beta(x,\alpha)$, with respect to α, yields the CVaR. In short, the results are summarized in Theorem 1 of R. T. Rockafellar and S. Uryasev, 2000:

$$\Phi_\beta(x) = F_\beta(x,\alpha(x,\beta)) = \min_\alpha F_\beta(x,\alpha) .$$

This statement can be proven by taking the derivative of the function $F_\beta(x,\alpha)$ with respect to α. The derivative equals $1+(1-\beta)^{-1}(\Psi(x,\alpha)-1)$. By setting it equal to zero and solving for $\Psi(x,\alpha)$, we obtain that $\Psi(x,\alpha)=\beta$, which was the original VaR equation. This means that by minimizing the function $F_\beta(x,\alpha)$ with respect to both x and α, we simultaneously optimize the CVaR and calculate the corresponding VaR.

We solve the problem using a linear programming approach. We consider a portfolio

composed of a finite number of assets, each with a finite sequence of historical data. As seen above, the calculations of the CVaR and VaR require future values for each asset, therefore by assuming that the future returns will follow a specific distribution $p(y)$, a finite set of scenarios y_j with $j = 1,...,J$ can be inferred from this distribution. If the assumptions above are respected, the function $F_\beta(x,\alpha)$ can be approximated with the function:

$$\tilde{F}_\beta(x,\alpha) = \alpha + \nu \sum_{j=1}^{J} (f(x,y_j) - a)^+ ,$$

where $\nu = \frac{1}{((1-\beta)J)}$ and $[t]^+ = t$ if $t > 0$, while $[t]^+ = 0$ if $t \le 0$.

The proof is available in R. T. Rockafellar and S. Uryasev, 2000.

In this case, since the function $f(x,y)$ is linear with respect to x, the problem can be solved with linear programming techniques. We first replace the term $(f(x,y_j) - \alpha)^+$ with auxiliary variables z_j in the function $\tilde{F}(x,\alpha)$, with the constraints: $z_j \ge f(x,y_j) - \alpha$ and $z_j \ge 0$. Then the minimization of the function $\tilde{F}(x,\alpha)$ is equivalent to solving the LP problem:

$$\min \ \alpha + \nu \sum_{j=1}^{J} z_j \qquad \text{subject to: } x \in X,\ z_j \ge f(x,y_j) - \alpha,\ z_j \ge 0 .$$

This procedure yields the minimal CVaR (proof available in F. Andersson, H. Mausser, D. Rosen, and S. Uryasev, 2000).

In this thesis we extend the constraints to expected portfolio return, and upper-lower

bound for the stock weights. In order to implement the minimization, we construct matrices of constraint A, b, Aeq and beq.

To create A, first consider a general column vector of stock weights, where each weight is calculated as $\frac{1}{n}$, where n is the total number of assets, so that each stock has the same weight in the portfolio and the sum of the weights equals 1. Then we set in cell $n+1$ the VaR of the portfolio; the reasons for this will be explained later. Subsequently we construct the matrix A, which is expressed as follows:

$$A=\begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_n & 0 \\ -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & & \vdots & 0 \\ \vdots & & -1 & 0 & 0 \\ 0 & \cdots & 0 & -1 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & \vdots & 0 \\ \vdots & & 1 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

Where the first row represent the set of stock average returns $\mu_1,\mu_2,\cdots,\mu_n$, which are followed by the negative of an $n\times n$ identity matrix and an $n\times n$ identity matrix below. The two identity matrices represent the coefficients of weights in order to apply the upper bound and lower bound for the portfolio weights.

Then we construct the column vector b, which is equal to:

$$b = \begin{pmatrix} -Er \\ -LB \\ \vdots \\ \vdots \\ -LB \\ UB \\ \vdots \\ \vdots \\ UB \end{pmatrix}.$$

The first component represents the negative of the expected portfolio return Er, which allows for the constraint on the expected portfolio return. Then it is followed by a column vector with length n where each cell equals the negative of the lower bound LB for the weights; below another n column vector that contains the upper bound UB. Consequently we find the column vector Aeq and beq composed as:

$$Aeq = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad beq = (1).$$

Where Aeq is an $n \times 1$ column vector of ones with an added zero cell below, and beq is equal to 1.

After defining these matrices, we can minimize the function $\alpha + \nu \sum_{j=1}^{J} z_j$ with the following constraints:

$$A \cdot w \leq b \quad \text{and} \quad Aeq \cdot w = beq.$$

The solution is the CVaR, and a vector w of optimal weights.

In case the objective is to only minimize the CVaR without setting any limit on upper or lower bound for weights or on expected return, the computations remain unchanged, with the only difference that the matrix A and b are not used as constraints in the minimization.

We refer to Appendix 1.4 for the MatLab code on the CVaR algorithm and calculation of optimal weights, with and without the constraints.

4. Generating Future Return Scenarios

In this section we use the Maximum-Likelihood Estimation method in order to estimate future scenarios for the stocks' returns. This method is generally defined in the first part, and then applied to the Normal, t-location scale, and generalized hyperbolic distributions.

4.1 Maximum-Likelihood Estimation (MLE)

The Maximum-Likelihood Estimation (MLE) is a method that estimates the parameters of a statistical model. Starting from the set of observed data, the MLE is a method that, given the statistical model of interest, finds the parameters of that distribution that maximize the likelihood function (J.W. Harris and H. Stocker, 1998). The likelihood function $L(\theta \mid x)$ is a function of the parameter values, given the observed data, which equals the probability of the observed data, given those parameter values:

$$L(\theta \mid x) = P(x \mid \theta) .$$

Suppose we have an independent and identically distributed sample of returns $y_1, y_2, y_3, \cdots, y_n$. The probability density function of the historical returns $f_0(y)$ is unknown, but we assume that it belongs to a certain class of parametric distributions, denotes as $f_0(y \mid \theta)$ such that $f_0(y) = f_0(y \mid \theta_0)$. Where θ_0 is the vector of true parameters.

The MLE aims at finding a vector of parameters $\hat{\theta}$ that is a good approximation of

the true vector of θ_0. The first step is to find the joint density function of the independent and identically distributed sample:

$$f(y_1, y_2, \cdots, y_n \mid \theta) = f(y_1 \mid \theta) \times f(y_2 \mid \theta) \times \cdots \times f(y_n \mid \theta).$$

Then we use the joint density function to calculate the likelihood by considering a function of the parameters θ with given $y_1, y_2, y_3, \cdots, y_n$:

$$L(\theta \mid y_1, y_2, \cdots, y_n) = f(y_1, y_2, \cdots, y_n \mid \theta) = \prod_{i=1}^{n} f(y_i \mid \theta).$$

To simplify the calculations in this thesis we use a modification of the likelihood function that is obtained by taking the logarithm of both sides in the previous equation. The new function is called the log-likelihood:

$$\log L(\theta \mid y_1, y_2, \cdots, y_n) = \log \prod_{i=1}^{n} f(y_i \mid \theta).$$

Since the logarithm is a monotonically increasing function, then it reaches the maximum point at the same point of the original function. This means that in likelihood maximization problems, the likelihood function can be substituted with the log-likelihood without altering the results. The property of the logarithm allows us to express the logarithm of a product as the sum of logarithms:

$$\log L(\theta \mid y_1, y_2, \cdots, y_n) = \log \prod_{i=1}^{n} f(y_i \mid \theta) = \sum_{i=1}^{n} \log f(y_i \mid \theta)$$

.

This feature makes the calculations much easier, since the maximization problems often require to take the derivative, and taking the derivative of a sum is always easier than taking the derivative of a product.

The MLE is therefore a key method for our purposes, since it provides us with the opportunity to generalize a set of historical stocks returns data into a given distribution, that will subsequently be used as a basis to sample random returns coherent with that distribution, which represent the predicted returns.

Historical data of asset returns follow complex distributions, and in order to provide better fit for the historical observations, three different models for predicting the future returns are considered: The Normal distribution, the *t*-Location Scale distribution and the Generalized Hyperbolic distribution.

4.2 Normal Distribution

The Normal or Gaussian distribution is a very common distribution in natural and social sciences. Its central role in statistics comes from the Central Limit Theorem, which states that under some basic assumptions, the mean of a large pool of independent random variables is normally distributed. As a fundamental distribution in mathematics, we have applied it to forecasting returns. The general formula for the probability density function $f(x, \mu, \sigma)$ (PDF) of the Normal distribution is:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} .$$

In our case μ is the mean return, and σ is the standard deviation of returns.

The cumulative distribution function $F_N(x)$ (CDF) of a Normal, which represents the probability that a scenario is lower than or equal to the value at which the cumulative distribution function is calculated, can be written as:

$$F_N(x) = P(X \leq x).$$

The CDF can also be expressed as the integral, between $-\infty$ and x, of the probability density function $f(x,\mu,\sigma)$. This means that we can write $F_N(x)$ as:

$$F_N(x) = \int_{-\infty}^{x} f(x,\mu,\sigma) = \int_{-\infty}^{x} \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

or

$$F_N(x) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right].$$

As shown in the second expression, the solution of the integral cannot be represented by elementary functions, therefore we have to include a special function named the error function $erf(x)$:

$$erf(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\, dt\,.$$

We refer to Appendix 1.5 for the MatLab code on the construction of the cumulative

distribution function for the Normal distribution.

For the purpose of this work, we use matrix notation in order to construct the model and implement it in the MatLab environment. We started with a matrix of logarithmic returns as follows:

$$\begin{pmatrix} y_1 \\ . \\ . \\ . \\ y_n \end{pmatrix} \approx \begin{pmatrix} y_1^1 & . & . & . & y_1^n \\ . & . & & . & . \\ . & & . & & . \\ . & . & & . & . \\ y_n^1 & . & . & . & y_n^n \end{pmatrix}.$$

The first necessary assumption is that the returns for each stock are normally distributed in the form:

$$\begin{pmatrix} y_1 \\ . \\ . \\ . \\ y_n \end{pmatrix} \approx \begin{matrix} N(\mu_1, \sigma_1^2) \\ . \\ . \\ . \\ N(\mu_2, \sigma_2^2) \end{matrix}.$$

The future values of stock i can then be predicted by summing the average return μ_i and a factor N that accounts for the standard deviation. To comply with the model, the factor N has to be normally distributed. Therefore the general formula is:

$$y_i^j = \mu_i + N \qquad \text{where} \qquad N \sim N(\mu, \sigma^2).$$

To obtain the factor N we first decomposed the variance-covariance matrix into

product of matrix Σ and its transpose Σ^T, using the Cholesky decomposition as follows:

$$\begin{pmatrix} \sigma_1^2 & . & . & . & \sigma_{1n} \\ . & . & & . & . \\ . & & . & & . \\ . & . & & . & . \\ \sigma_{1n} & . & . & . & \sigma_n^2 \end{pmatrix} = \Sigma\Sigma^T .$$

Then we multiply the matrix Σ for a vector of independent standard Normal random variables z. Hence, to generate the scenario j for the asset returns, we use the following representation:

$$\begin{pmatrix} y_1^j \\ . \\ . \\ . \\ y_n^j \end{pmatrix} = \begin{pmatrix} \mu_1 \\ . \\ . \\ . \\ \mu_n \end{pmatrix} + \Sigma \begin{pmatrix} z_1^j \\ . \\ . \\ . \\ z_n^j \end{pmatrix},$$

Where z_j denotes a sample of n random numbers from the standard Normal distribution. We refer to Appendix 1.6 for the MatLab Code on the generation of return scenarios using the Normal distribution.

By doing simple statistics it is however easy to realize that the Normal distribution cannot perfectly approximate the historical distribution of returns (J. Y. Campbell, A. W. Lo, and A. C. MacKinlay, 1997). In fact the values of skewness and kurtosis for the Normal distribution are both equal to 0, while the real values shown in Table 1.2 are on average very far from 0. Moreover in Figure 4.2.1 the distribution of market

returns of the Nasdaq 100 index it is plotted against the fitted Normal distribution. The graphical representation stresses the impossibility of the Normal distribution to fit higher values around the mean and the fatter tails, therefore a different distribution should be applied.

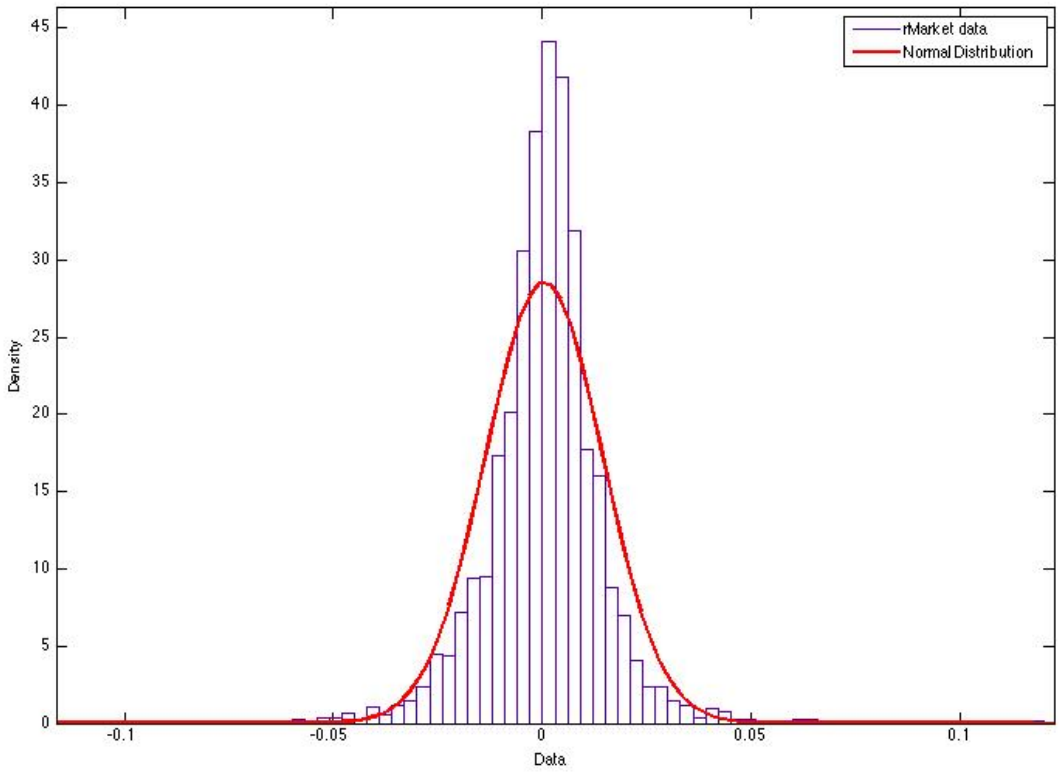

Figure 4.2.1 *– Plot of the Nasdaq 100 returns against the fitted Normal distribution.*

4.3 *T*-Location Scale Distribution

The *t*-location scale is a more complex distribution than the Gaussian, since it includes in addition to the expected return μ and standard deviation σ, a parameter ν expressing the degrees of freedom. This parameter determines the shape of the distribution around the mean and the tails. For high values of ν, we have higher probabilities around the mean, while for lower ones, we have higher probabilities around the tails. Moreover, when the degrees of freedom go to infinity, the *t*-location

scale is exactly equal to the Normal distribution. Thanks to the listed properties, the *t*-location scale distribution is preferred in cases of leptokurtosis, as expressed by the distribution of returns in our case (H. Rinne, 2010).

We denote the probability density function as $T(x,\mu,\sigma,\nu)$, expressed by the formula:

$$T(x,\mu,\sigma,\nu)=\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sigma\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left[\frac{\nu+\left(\frac{x-\mu}{\sigma}\right)^2}{\nu}\right]^{-\left(\frac{\nu+1}{2}\right)},$$

Where $\Gamma(n)$ is the Gamma function. In case n belongs to the set of integers, then the Gamma function is simply a variation of the factorial function, where the argument is $n-1$. Therefore it is summarized in the expression:

$$\Gamma(n)=(n-1)!$$

However, if n is a complex number with a positive real part, then it is expressed as the integral:

$$\Gamma(n)=\int_0^\infty x^{n-1}e^{-x}\,dx\,.$$

Since the t-location scale includes the gamma function, the cumulative distribution function (CDF) cannot be explicitly found by taking the integral of the probability density function above, but it can be approximated by analyzing the probability density function (PDF). The *t*-location scale distribution is in fact a broad expression

that includes the Student *t*-distribution. Therefore if we consider a random variable x distributed according to the *t*-location scale distribution with parameters μ, σ, and ν; then $\frac{x-\mu}{\sigma}$ is distributed according to a Student *t*-distribution with ν degrees of freedom, that is:

$$T(y,\mu,\sigma,\nu)=\frac{1}{\sigma}PDF_{tStudent}\left(\frac{y-\mu}{\sigma}\right).$$

This property allows us to approximate the *CDF* of the t-location scale by using the *CDF* of the Student *t*-distribution with argument $\frac{x-\mu}{\sigma}$. In short:

$$\int_{-\infty}^{x}T(y,\mu,\sigma,\nu)dy=\frac{1}{\sigma}\int_{-\infty}^{x}PDF_{tStudent}\left(\frac{y-\mu}{\sigma}\right)dy=\int_{-\infty}^{\frac{x-\mu}{\sigma}}PDF_{tStudent}(z)dz$$

Or equivalently, we can express it as a relationship between the cumulative distribution functions:

$$F_{tStud}\left(\frac{y-\mu}{\sigma}\right)=F_{tLoc}(y)$$

Here F stands for the CDF.

We refer to Appendix 1.7 for the MatLab code related to the calculation of the cumulative distribution function for the t-location scale distribution.

To predict the future values of the stocks returns, we have to analyze the probability density function formula of the *t*-Location scale distribution. All the variables in the equation are known, apart from the number of degrees of freedom ν, which has to be

estimated. This estimation complicates the procedure, making the model slower and unsuitable for the purpose of this work. Moreover, another limitation is brought by the presence of the Gamma function $\Gamma(n)$. The argument n in the gamma function is in our case a linear combination of the degrees of freedom ν, and since the degrees of freedom can only be greater than or equal to zero, we can isolate the gamma function in the first quadrant of the Cartesian plane when plotted in a graph with ν on the x-axis and $\Gamma(\nu)$ on the y-axis (as in Figure 4.3.1).

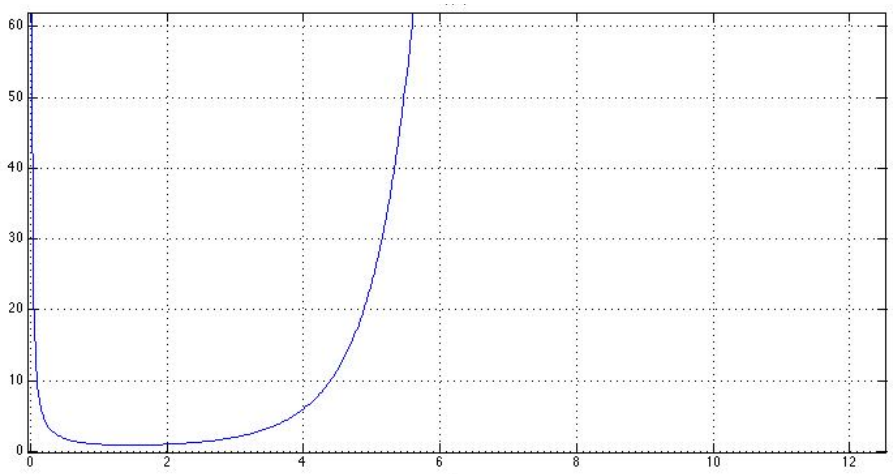

Figure 4.3.1 *– Plot of the Gamma function* $\Gamma(n)$ *on the first quadrant of the Cartesian plane.*

The chart shows the values of the gamma function plotted over the degrees of freedom. We can see that the value of the function quickly goes to infinity as the degrees of freedom rise, making the *t*-location scale equal to the Normal distribution. This downside, added to the limitations on the definiteness of the kurtosis, make this approach unsuitable (from the numerical and computational storage viewpoint) for a multivariate version of the scalar *t*-location scale distribution.

We have therefore constructed a new model based on a classical financial theory: the Capital Asset Pricing Model (CAPM) (W.F. Sharpe, 1964). The CAPM tries to combine concepts from portfolio valuation and market equilibrium in order to construct a formula for the pricing of assets based on their risk, and therefore provide a tool to measure and to price risk. We consider a model based on the direct relationship between risk and return, where the risk is considered as the sum of systematic and unsystematic risk. The CAPM however assumes that the unsystematic risk, which can be diversified by using the correlations between assets in a portfolio, is not relevant; while systematic risk is the only risk present in a well-diversified portfolio and is measured with β. The CAPM formula is:

$$E(R_i) = R_f + \beta_i[E(R_m) - R_f] + \varepsilon_i$$

Where $E(R_i)$ is the expected return on security i = 1, 2, … , n, which is equal to the sum of the risk-free rate R_f, plus the product of the market sensitivity β_i times the equity risk premium $E(R_m) - R_f$, all summed to an error term ε_i. In our case, the market is taken to be the index Nasdaq 100, while the risk-free rate corresponds the short-term U.S. Government bonds, as they are highly rated and in line with the market for the stocks chosen. The β_i coefficient, which represents the sensitivity of security i returns to the market returns, has obtained by a simple linear regression between the historical returns of security i and the market returns. For every asset return, the slope coefficient of the line obtained is described by β_i. Appendix 1.8 includes the MatLab code used for the computation of β_i.

The error term ε_i is instead a predicted value of the error term. To remain in line with the assumption on the *t*-location scale distribution of returns, we assume that the error terms $\{\varepsilon_i\}_i$ are independent random variables on the same probability space following:

$$\varepsilon_i = \sigma_i T_{tStudent}(\nu_i),$$

Where $T_{tStudent}(\nu_i)$ describes a *t*-Student distributed random variable with ν_i degrees of freedom. They are also assumed to be independent of the market index return. The value is estimated by fitting a *t*-location scale to the distribution of historical residuals. The value of the error used to forecast scenarios for the expected returns of security *i* can be generated with a random process following the *t*-Location Scale distribution with estimated parameters. We refer to Appendix 1.9 for the MatLab code related to the calculation of the error ε_i.

The expected value of the returns of the market is calculated with the same method as the error term. With all the variables needed, we can simply follow the CAPM formula and calculate the expected returns of security i following a *t*-location scale distribution.

We refer to Appendix 1.10 for the MatLab code to predict scenarios coherent with the t-Locations Scale distribution using the CAPM model.

One of the major issues related to the use of the CAPM model is the correlation between the predicted returns and the error term. In fact it would mean that the model, with the current structure and variables is not explaining the behavior of returns in a complete way. Part of the influence of returns would in fact be captured by the error term itself. For this reason we tested the model evaluating the covariances and

correlations between the error term ε_i of stock *i* with the respective predicted returns. The covarainces are listed in Table 4.3.1, while the correlations are listed in Table 4.3.2.

Covariance between ε_i and predicted returns	***1.0e-04**
AAPL	0.0054
CSCO	-0.0676
MSFT	-0.0146
CMCSA	0.0155
WFM	0.0489
PFE	0.0556
MAR	-0.0232
MNST	0.1105
SBUX	0.0555
FISV	0.0010

Table 4.3.1 *– Covariances between the error term and predicted returns.*

Correlation between ε_i and predicted returns	
AAPL	0.0010
CSCO	-0.0242
MSFT	-0.0070
CMCSA	0.0049
WFM	0.0067
PFE	-0.0125
MAR	0.0142
MNST	0.0066
SBUX	0.0118
FISV	0.0003

Table 4.3.2 *– Correlations between the error term and the predicted returns.*

As shown in Table 4.3.1, the results obtained exhibit almost null covariances, displaying that the model is constructed in an effective way. Moreover, the correlations in Table 4.3.2 are quite low, meaning that the model is not biased by a relationship between the error terms and the predicted returns. We refer to Appendix 1.11 for the MatLab code on the calculations of the covariance and correlation vector.

Moreover, an important test is on the independence of the error terms, defined in statistics as: autocorrelation. This property has been tested by applying the Durbin-Watson test. This test provides a value $p \in [0,1]$, such that if the value is close to 0 we reject the null hypothesis of no autocorrelation, in case it is close to 1 we have a proof of no autocorrelation. The test results are listed in Table 4.3.3.

Correlation between ε_i and predicted returns	
AAPL	0.0139
CSCO	0.4598
MSFT	0.3235
CMCSA	0.4902
WFM	0.7301
PFE	0.0711
MAR	0.1362
MNST	0.1395
SBUX	0.1926
FISV	0.1808

Table 4.3.3 *– Results of the Durbin-Watson test.*

The results of the test are relatively low, showing that the error terms exhibit clear signs of autocorrelation. Even if the estimation is robust, this property may negatively affect our results.

We also realize that the CAPM imposes some strict assumptions, which in some cases may have to be relaxed. Therefore we computed the estimation of returns through the decomposition of the *t*-location scale into a Student-*t* distribution (as explained above), which allowed us to avoid using the CAPM. Appendix 1.12 contains the MatLab code for the calculation of returns as explained above.

Below, in Figure 4.3.2, we represent the distribution of market returns (Nasdaq 100) against the fitted *t*-location scale distribution.

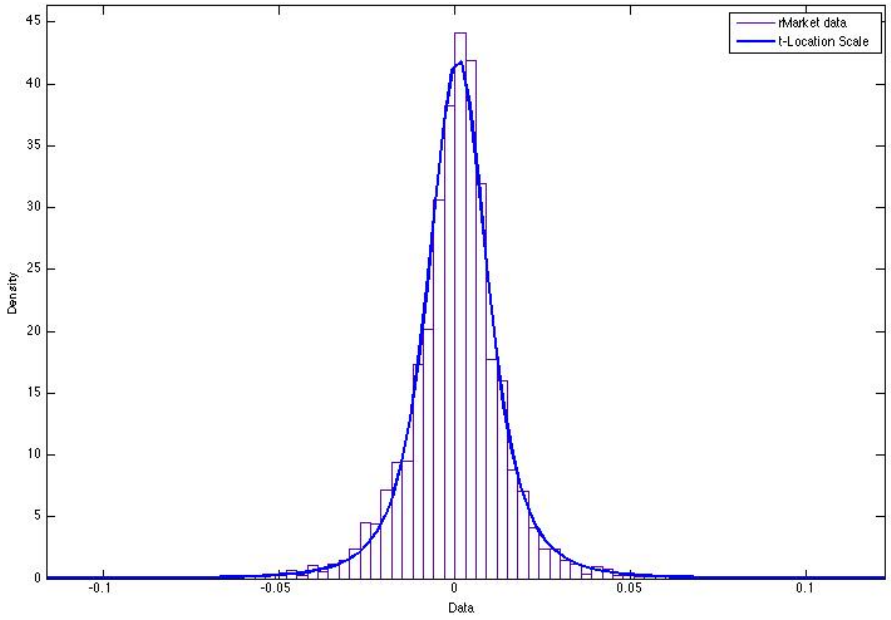

Figure 4.3.2 *– Plot of the market returns distribution against the t-location scale distribution.*

Figure 4.3.2 shows that the *t*-location scale fit represents a much more precise approximation of the real data. In terms of sample statistics, the values of skewness and kurtosis for the *t*-location scale are obtained by considering the skewness of the Student-t distribution. The skewness is 0 for degrees of freedom $\nu > 3$; otherwise it is

undefined. In this statistic the *t*-location scale does not provide a better approximation than the Normal distribution, therefore it does not capture the asymmetry of the peak in the dataset. The kurtosis instead equals $\frac{6}{\nu - 4}$ for $\nu > 4$, ∞ for $4 \geq \nu > 2$, and for all other values of ν it is undefined. The plot of the value of the kurtosis over the different degrees of freedom for the case where $\nu > 4$ are represented in Figure 4.3.3.

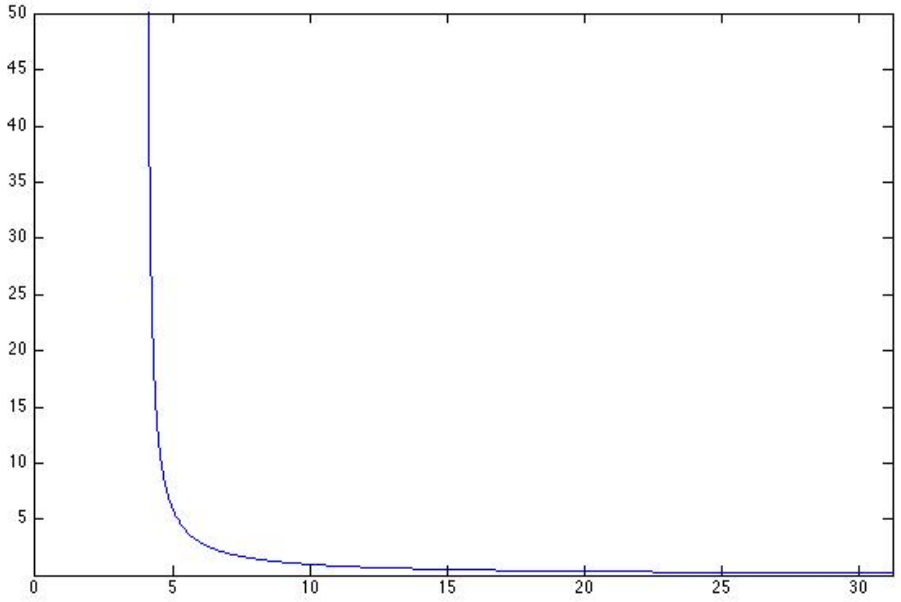

Figure 4.3.3 *– Plot of the kurtosis value against the number of degrees of freedom, for the case* $\nu > 4$.

As we can see the from Figure 4.3.3 the values of kurtosis in the specific interval of degrees of freedom, range from infinity to zero, providing a much better approximation for the real values.

The sample statistics discussed above demonstrate our hypothesis that the *t*-location scale better approximates the real distribution in the peak around the mean, thanks to the possibility to have positive kurtosis.

In Figure 4.3.4 the comparison between the Normal distribution and the *t*-location scale is shown, emphasizing the advantage of using the *t*-location scale.

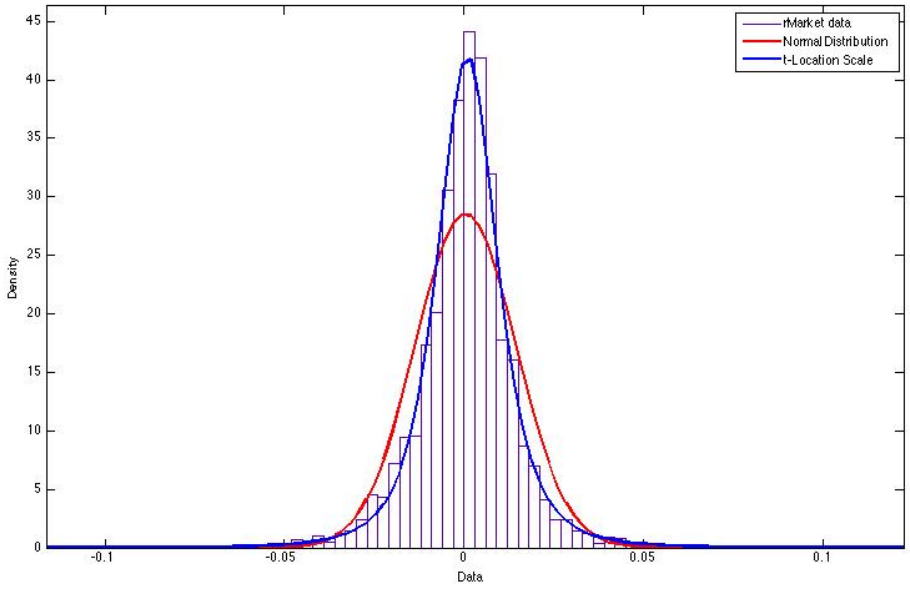

Figure 4.3.4 *– Plot of the market returns against both the fitted Normal distribution (in red) and the fitted t-location scale (in blue).*

Unfortunately, even if the *t*-location scale follows the empirical distribution of real data much more closely and accurately than the Normal distribution, some characteristics of the empirical distribution are still not well approximated. In fact, by comparing the indices of skewness of the *t*-location scale against the real values, we observe that the results are very different, meaning that the probability values around the tails are not well estimated; therefore a different distribution should be considered.

4.4 Generalized Hyperbolic Distribution

The Generalized Hyperbolic distribution (GH) was first introduced in 1977 by Barndorff-Nielsen to represent a mathematical model for the movement of sand dunes

(O. Barndorff-Nielsen, 1977). This distribution is a very broad form, which can itself represent various distributions, one of which is the *t*-location scale. Moreover its characteristic semi-heavy tails allow it to model samples such as financial market returns, where the probability in the tails is not well captured by the Normal distribution or *t*-location scale (K. Prause, 1999).

The probability density function of the generalized hyperbolic distribution is defined by the expression:

$$d_{GH(\lambda,\alpha,\beta,\delta,\mu)}(x) = a(\lambda,\alpha,\beta,\delta,\mu)(\delta^2 + (x-\mu)^2)^{\left(\frac{2\lambda-1}{4}\right)} e^{\beta(x-\mu)} \times K_{\lambda-\frac{1}{2}}(\alpha\sqrt{\delta^2 + (x-\mu)^2}),$$

Where α is a shape factor whose value has to be greater than 0. The skewness is determined by the absolute value of β which has to satisfies the condition $0 \le |\beta| < \alpha$. The parameter $\mu \in R$ is a location parameter. $\lambda \in R$ characterizes the specific subclasses, and is the main influencer of the tail sizes; while $\delta > 0$ is a scaling factor. Moreover the norming constant $a(\lambda,\alpha,\beta,\delta,\mu)$ takes the following form:

$$a(\lambda,\alpha,\beta,\delta,\mu) = \frac{(\alpha^2 - \beta^2)^{\frac{\lambda}{2}}}{\sqrt{2\pi}\alpha^{\left(\lambda-\frac{1}{2}\right)}\delta^{\lambda}K_{\lambda}(\delta\sqrt{\alpha^2-\beta^2})}.$$

The function K_{λ} represents the third kind modified Bessel function with index λ, which in practice is a linear combination of the first kind and second kind modified Bessel functions (M. Abramowitz and I. A. Stegun, 1964).

The generalized hyperbolic distribution strictly depends on the Generalized Inverse Gaussian Distribution (GIG), in fact its sample statistics contain values derived from it. By taking $\delta\sqrt{\alpha^2+\beta^2}=\zeta$, then the expected value, or mean of the GH is equal to:

$$E(GH)=\mu+\frac{\beta\delta^2}{\zeta}\frac{K_{\lambda+1}(\zeta)}{K_{\lambda}(\zeta)},$$

or

$$E(GH)=\mu+\beta E(GIG).$$

While the variance of the GH is:

$$VAR(GH)=\frac{\delta^2}{\zeta}\frac{K_{\lambda+1}(\zeta)}{K_{\lambda}(\zeta)}+\beta^2\frac{\delta^4}{\zeta^2}\left(\frac{K_{\lambda+2}(\zeta)}{K_{\lambda}(\zeta)}-\frac{K_{\lambda+1}^2(\zeta)}{K_{\lambda}^2(\zeta)}\right),$$

or, equivalently

$$VAR(GH)=E(GIG)+\beta^2 Var(GIG).$$

In order to predict scenarios from a *GH* distribution that accounts for the correlation between assets, we accounted for the multivariate model by applying the CAPM (as for the *t*-location scale). To then approximate the single asset returns, we estimate the market returns with the two approximation techniques described below.

The first approximation (Option 1) method is based on the construction of a strictly increasing function $h(x;u)=F_{GH}(x)-u$ that determines the difference between the cumulative distribution function of the portfolio *GH* distribution and a given random number u uniformly distributed between 0 and 1. Then we find a numerical approximation $x(u)$ for the unique solution of $h(x;u)=0$, in order to obtain the value

a random number extracted by the *GH* distribution. The function can be expressed as:

$$h(x;u) = \int_{-\infty}^{x} d_{GH(\lambda,\alpha,\beta,\delta,\mu)}(x)dx - u.$$

The integral yields the *GH* cumulative distribution function. Appendix 1.13 presents the MatLab code for the computation of the cumulative distribution function for the *GH* distribution.

The second approximation (Option 2) method consists in the application of the Newton-Raphson algorithm to approximate the (unique) root of a strictly monotonic and differentiable function through an iterative process. The iterative procedure follows the scheme:

$$\begin{cases} x_1, \quad given, \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)} \end{cases}.$$

In our case $f(x) = h(x;u) = F_{GH}(x_u) - u$, and the derivative is simply the *GH* probability density function. Hence, the iterative process becomes:

$$\begin{cases} x_1, \quad given, \\ x_{k+1} = x_k - \dfrac{F_{GH}(x_k) - u}{d_{GH(\lambda,\alpha,\beta,\delta,\mu)}(x_k)} \end{cases}.$$

Here x_1 represents a starting point for the iterative process, which is given by the user and is an approximation of the root of the function that is known in advance. This is the first point where the iteration starts, so the closer it is to the real value, the faster the algorithm will be in reaching the solution. Figure 4.4.1 below provides a graphical explanation of the functioning of the Newton-Raphson method.

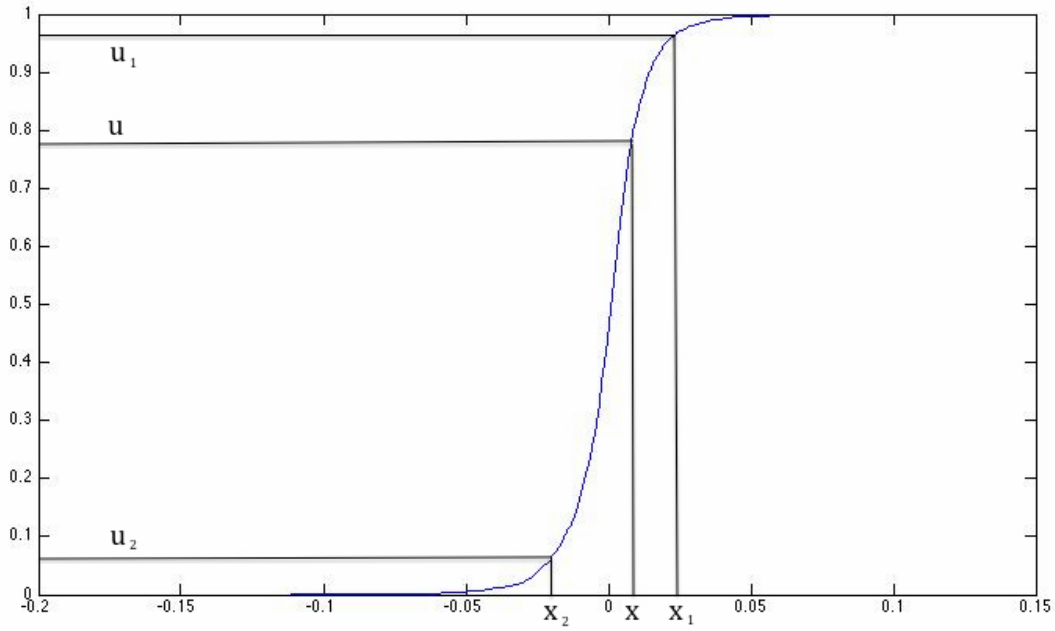

Figure 4.4.1 *– Approximation using the Newton-Raphson method.*

Since the density function is strictly positive, then suppose we are referring to the CDF in Figure 4.4.1, and assume we start the Newton-Raphson iteration at a point x_1 greater than the solution x. In this case the value of $F_{GH}(x_u) - u$ is equal to $u_1 - u$, which is a positive number since $u_1 > u$. Therefore we are subtracting a positive quantity from x_1, meaning that the next iteration will start from a point $\hat{x}_2$ such that $x < \hat{x}_2 < x_1$. If the iteration goes on, the value of the quantity subtracted decreases until it reaches 0, which is exactly the solution x.

Suppose instead we start from a value x_2 lower than x. In this case the value of $F_{GH}(x_u) - u$ is negative since $u > u_2$, hence we are subtracting a negative quantity (adding) to x_2. Again if the iteration goes on, the value of the quantity added decreases until it reaches 0, which is exactly the solution x.

We refer to Appendix 1.14 for the MatLab code on the generation of scenarios from a Generalized Hyperbolic distribution using the two procedures (Option 1 and Option 2) discussed above. Moreover we refer to Appendix 1.15 for the Matlab code on the generation of stocks returns based on the two methods discussed above, using the CAPM.

The results of the Hyperbolic Distribution are much closer to the real values, especially for the values in the tails. This distribution provides a good fit for the real data, which will allow for more accurate predictions and estimations of the CVaR.

In Figure 4.4.2, the Generalized Hyperbolic distribution is plotted against the distribution of returns of the market (Nasdaq 100).

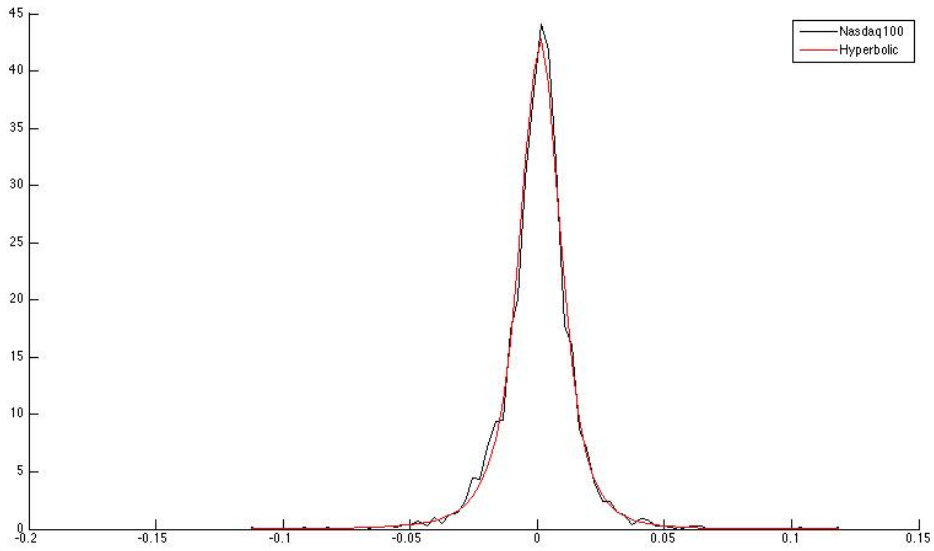

Figure 4.4.2 *– Plot of the fitted GH distribution against the market return distribution*

5. Goodness of Fit

To assess whether the above considered distributions represent a good approximation of real data, and in order to measure their effectiveness in modeling all the characteristics of the empirical distribution of asset returns, we have compared the empirical distribution with the estimated distribution applying some discrepancy measures.

5.1 Anderson & Darling Test

This statistical test for goodness of the fit is particularly important in our analysis for its intrinsic characteristics (T. W. Anderson and D. A. Darling, 1952). The Anderson & Darling test for any distribution is expressed by:

$$AD = \max_{x \in R} \frac{\left| F_{emp}(x) - F_{est}(x) \right|}{\sqrt{F_{est}(x)(1 - F_{est}(x))}},$$

where $F_{emp}(x)$ is the empirical CDF and $F_{est}(x)$ is the estimated CDF. The effect of the factor $\sqrt{F_{est}(x)(1 - F_{est}(x))}$ at the denominator in AD can be deduced by Figure 5.1.1 below.

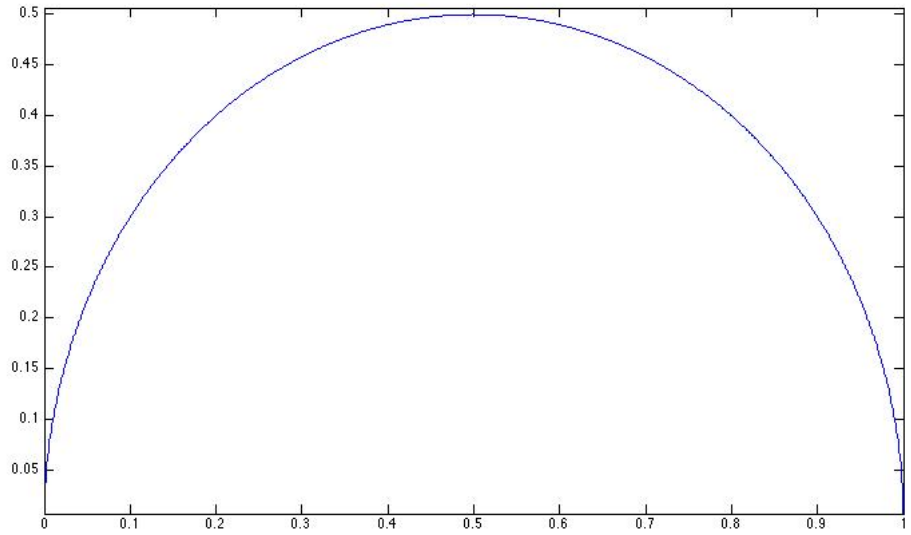

Figure 5.1.1 *– Plot of the* $\sqrt{F_{est}(x)(1-F_{est}(x))}$ *factor at the denominator of the Anderson & Darling test formula, where* $F_{est}(x)$ *is on the x-axis and the value of the factor on the y-axis.*

From Figure 5.1 we can see that the function expressing the total value of the factor (on the y-axis) has a global maximum when the estimated CDF $F_{est}(x)$ (on the x-axis) is at 0.5, meaning that we are at the mean point μ of the distribution. Then as we move towards the tails of the distribution (e.g. when the CDF approaches 0 and 1), the value of the function decreases to zero. This means that the Anderson & Darling test divides the absolute difference $\left|F_{emp}(x)-F_{est}(x)\right|$ for greater values when it is around the mean, and for smaller values when it is around the tails, in this way emphasizing the Anderson & Darling test result on the distribution's tails (S. R. Hurst, E. Platen, and S. T. Rachev, 1995). Thanks to this property, the Anderson & Darling test is a widely used statistical tool to accurately measure the error of distributions that exhibit specifically fat tails (M. A. Stephens, 1974).

We refer to appendix 1.16 for the MatLab code on the application of the Anderson & Darling test.

5.2 Kolmogorov Distance

The Kolmogorov distance is defined as the supremum of the absolute difference between the predicted and the empirical cumulative distribution functions. The Kolmogorov distance is expressed in mathematical terms as:

$$KD = \sup_{x \in R} \left| F_{emp}(x) - F_{est}(x) \right|$$

This function assigns a quantity to the distance between the predicted and empirical CDF, allowing to quantify the goodness of fit.

We refer to Appendix 1.17 for the MatLab code on the Kolmogorov distance test.

5.3 L1 Distance

The L1 distance is another estimator of the goodness of fit, and is defined as the sum of the absolute difference between the predicted and the empirical cumulative distribution functions evaluated at a given set of points $\{x_i\}_i$. Precisely, we have:

$$L1 = \sum_i \left| F_{emp}(x_i) - F_{est}(x_i) \right|.$$

We refer to Appendix 1.18 for the MatLab code on the L1 distance test.

5.4 L2 Distance

The L2 distance, also called the Euclidean distance, is a similar measure to the L1 distance, and is calculated as the square root of the squared deviation between the predicted and the empirical cumulative distribution functions. The L2 distance is expressed by:

$$L2 = \sqrt{\sum_i \left| F_{emp}(x_i) - F_{est}(x_i) \right|^2} .$$

We refer to Appendix 1.19 for the MatLab code on the L2 distance test.

5.5 Estimated Results

We ran each of the distance measures on the market index (Nasdaq 100), since it can be considered a representative proxy for the market and of our portfolio. The results are listed in Table 5.5.1.

	Normal	***t*-Location Scale**	**Generalized Hyperbolic**
Anderson & Darling	7.6099	0.1679	0.0309
Kolmogorov Distance	0.0284	0.0093	0.0081
L1 Distance	29.3731	7.6295	6.5530
L2 Distance	0.6920	0.1846	0.1587

Table 5.5.1 *– Test results for the market index (Nasdaq 100).*

We estimated the returns using the Normal distribution, *t*-location scale and the generalized hyperbolic. The results are very different, and their significance is discussed below.

As expected, the values of the distance measures under the Normal distribution are substantially higher than the values obtained for other distributions, in every test. The fact that returns do not follow a Normal distribution is clear, even from the graphical comparison between the probability density function of historical returns and the fitted Normal distribution in Figure 4.2.1. However, this serves as a measure to quantify the error caused by the use of this distribution as a predictor of future returns. The *t*-location scale scores are much lower than the Normal distribution, meaning that it is a more accurate predictor. However, the values are still slightly higher than the hyperbolic distribution in all tests apart for the Anderson & Darling test, where the t-location scale is still much higher. This result clearly proves our expectations, meaning that the *t*-location scale is a valid estimator for the values around the mean, but a poor predictor when the distribution is around the tails.

The hyperbolic distribution definitely yields satisfactory results, as the error measures are low for each test.

6. Portfolio Optimization

In this section we apply the portfolio optimization algorithm defined in Section 3 to the representative portfolio selected from the Nasdaq 100 index, with the objective to find the optimal asset weights that minimize risk.

The portfolio optimization was applied to our dataset (described in Section 1), and the results using the minimization of CVaR formula are listed in Table 6.1.

	Weight
AAPL	0.0456
CSCO	0.1030
MSFT	0.2589
CMCSA	0.0058
WFM	0.0286
PFE	0.5400
MAR	-0.0675
MNST	0.0156
SBUX	0.0317
FISV	0.0382

Table 6.1 *– List of weights resulting from the minimization of CVaR.*

The value of the CVaR obtained is 0.0311.

To parallel the methods of predictions and evaluate their effect on portfolio optimization, we compared the expected returns of the portfolio following the different distributions with real returns of the month after the stock prices were reported. Our expectations from the test results of Section 5.5 suggest that the best approximation should be provided by the hyperbolic distribution, followed by the *t*-location scale, and lastly by the Normal distribution. The empirical results are listed in Table 6.2.

	Expected Return
Real data	0.0012
Normal	0.0002
***t*-Location Scale**	-0.0035
Hyperbolic	0.0021

Table 6.2 *– Expected returns of the portfolio using different distributions*

The results do not completely match our expectations, in fact the distribution providing the greater distance from the real data is the *t*-location scale, while we expected it to be the Normal distribution. We believe this is due to the autocorrelation that the error terms exhibit, and may yield biased results. Moreover the lack of data for the future prices comparison may negatively affect our results. In fact we had the possibility to take only 19 real return observations after the original dataset, making the computations subject to very high standard deviations. In our case the average standard deviation of the dataset is calculated to be around 0.0091, which is a very high value compared to the values of the expected returns.

However, we consider the results to be effective in explaining the dominance of the hyperbolic distribution in comparison with the other two methods.

7. Conclusions

This thesis has provided an in depth comparison between the Normal distribution, *t*-location scale and generalized hyperbolic distribution in their application to portfolio optimization and evaluation of risk. The distributions' efficacy is measured through the use of various tests, which show a clear predominance in terms of accuracy and precision for the generalized hyperbolic distribution. Particular importance is given to the Anderson & Darling test (Section 5.1), since it emphasizes the error on the tails, which is on of the most particular features that the empirical distribution of stock returns exhibit.

Moreover, by comparing the expected returns resulting from the minimization of CVaR using different distributions, the hyperbolic distribution yields far more precise estimates in the calculation of risk. Even if the dataset expected returns is limited, we believe that the results are representative since the distributions estimated allow for the generalization of our results. Therefore, it is clear that the hyperbolic distribution provides investors with accurate predictions, allowing for lower uncertainty on the measurements and more efficient portfolio optimization.

References

Markowitz, H. (1952). "Portfolio Selection." *The Journal of Finance,* 7.1: 77-91.

Bielecki, T.R., Jin, H., Pliska, S.R. and Zhou, X.Y. (2005). "Continuous-time-mean-variance portfolio selection with bankruptcy prohibition." *Mathematical Finance,* 15: 213-244.

Jorion, P. (1996). "Risk2: Measuring Risk in a Value at Risk." *Financial Analysts Journal* 52 (6): 47.

Artzner, P., Delbaen, F., Eber, J., and Heath, D. (1999). "Coherent measure of risk." *Mathematical Finance*, 9 (3): 203–228.

Pritsker, M. (1997). "Evaluating Value at Risk Methodologies." *Journal of Financial Services,* 12 (2/3): 201-242.

Uryasev, S. (2000). "Conditional Value-at-Risk: Optimization Algorithms and Applications." *Financial Engineering News*, 14.

Rockafellar, R. T. and Uryasev, S. (2000). "Optimization of Conditional Value-at-Risk." *Journal of Risk*, 2: 21–41.

Rockafellar, R. T. and Uryasev, S. (2001). "Conditional Value-at-Risk for General Loss Distributions." *University of Florida*.

Embrechts, P., Frey, R., Furrer, H. (2001). "Stochastic Processes in Insurance and Finance." In: Handbook of Statistics, vol. 19 "Stochastic Processes: Theory and Methods", *Elsevier Science*: 365-412.

Andersen, J.V. and Sornette, D. (1999). "Have Your Cake and Eat it Too: Increasing Returns While Lowering Large Risk!" *University of California*.

Basak, S. and Shapiro, A. (2001) "Value-at-risk based risk management: optimal policies and asset prices." *Review of Financial Studies* 14: 371-405.

Gaivoronski, A.A. and Pflug, G. (2004/05). "Value-at-risk in portfolio optimization: Properties and computational approach." *Journal of Risk* 7: 1-31.

Gaivoronski, A.A. and Pflug, G. (2000). "Finding optimal portfolio with constraints on value-at-risk."

Gourieroux, C., Laurent, J.P. and Scaillet, O. (2000). "Sensitivity Analysis of Value at Risk." *Journal of Empirical Finance* 7: 225–245.

Grootweld, H. and Hallerbach, W.G. (2000). "Upgrading VaR from Diagnostic Metric to Decision Variable: A Wise Thing to Do?" *Erasmus Center for Financial Research*.

Kast, R., Luciano, E., and Peccati, L. (1998). "VaR and Optimization: 2nd International Workshop on Preferences and Decisions."

Puelz, A. (1999). "Value-at-Risk Based Portfolio Optimization." *Southern Methodist University*.

Tasche, D. (1999). “Risk Contributions and Performance Measurement.” *University of Technology*.

Krokhmal, P., Palmquist, J. and Uryasev, S. (2002). “Portfolio optimization with conditional Value-at-Risk criterion” *Journal of Risk* 4 (2): 99-14.

Andersson, F., Mausser, H., Rosen, D. and Uryasev, S. (2000). “Credit risk optimization with condition value- at-risk.” *Mathematical Programming*, *Series B*: 99-9.

Checklov, A., Uryasev, S. and Zabarankin, M. (2003). “Portfolio optimization with drawdown constraints.” *University of Florida*.

Eberlein, E. (2001). “Application of generalized hyperbolic Lévy motions to finance.” In: “Lévy Processes: Theory and Applications” O.E. Barndorff-Nielsen, T. Mikosch, and Resnick, S. *Birkhäuser Verlag*: 319–337.

Kan, Y.S. and Kibzun, A.I. (1996). “Stochastic Programming Problems with Probability and Quantile Functions.” *John Wiley & Sons*: 316.

Uryasev, S. (1995). “Derivatives of Probability Functions and Some Applications.” *Annals of Operations Research* vol. 56: 287–311.

Harris, J.W. and Stocker, H. (1998). “Handbook of Mathematics and Computational Science.” *Springer-Verlag*.

Campbell J.Y., Lo A.W., MacKinlay C.A., Adamek P., Viceira L.M. (1997). “The Econometrics of Financial Markets.” *Princeton University Press*.

Rinne, H. (2010). “Location-scale Distributions”. *Justus–Liebig–University*.

Barndorff-Nielsen, O. (1977). “Exponentially decreasing distributions for the logarithm of particle size.” *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences* 353: 401–409.

Abramowitz, M. and Stegun, I.A. (1968). “Handbook of Mathematical Functions.” *Dover Publications*.

Prause, K. (1999). “The Generalized Hyperbolic Model: Estimation, Financial Derivative, and Risk Measures.” *Albert-Ludwigs University*.

Anderson, T. W. and Darling, D. A. (1952). “Asymptotic theory of certain ‘goodness-of-fit’ criteria based on stochastic processes.” *Annals of Mathematical Statistics* 23: 193–212.

Hurst, S.R., Platen, E. and Rachev, S.T. (1995). “A Comparison of Subordinated Asset Pricing Models.” *Australian National University.*

Stephens, M. A. (1974). “EDF Statistics for Goodness of Fit and Some Comparisons.” *Journal of the American Statistical Association* 69: 730–737.

Appendix – MatLab codes

Appendix 1.1

MatLab code on the calculation of sample statistics for the dataset.

```
function [ ExpRet,Variance,StDev,Skewness,Kurtosis,VarCov ] =
SampleStat( rStocks )

%This function generates the sample statistics for a matrix of prices
%Insert: pStocks - matrix of prices, each stock has a column of
prices.

n=0;
m=0;
j=size(rStocks,2);

%Expected Return
for n=1:j
    ExpRet(1,n)=mean(rStocks(:,n));
end

%Variance
for n=1:j
    Variance(1,n)=var(rStocks(:,n));
end

%Standard Deviation
for n=1:j
    StDev(1,n)=std(rStocks(:,n));
end

%Skewness
for n=1:j
    Skewness(1,n)=skewness(rStocks(:,n));
end

%Kurtosis
for n=1:j
    Kurtosis(1,n)=kurtosis(rStocks(:,n));
end

%Variance-Covariance Matrix
VarCov=cov(rStocks);

end
```

Appendix 1.2

MatLab code for the calculation of logarithmic returns.

```
function [ rStocks,rMarket ] = logr( pStocks,pMarket )
```

```
%This function yields the matrix of logarithmic returns from the
matrix of
%prices of stocks and market prices
%  Insert:
%    . pStocks - matrix of prices of stocks. Stocks are listed in
rows,
%      so each column is a column of prices from the latest to the
oldest
%      price
%    . pMarket - column vector of prices of the market index

i=size(pStocks,1);
j=size(pStocks,2);

rStocks=zeros(i-1,j);
rMarket=zeros(i-1,1);

for n=1:j
    for m=1:(i-1)
        rStocks(m,n)=log(pStocks(m,n)/pStocks(m+1,n));
    end
end

for m=1:(i-1)
    rMarket(m,1)=log(pMarket(m,1)/pMarket(m+1,1));
end


end
```

Appendix 1.3

MatLab code on the calculation of the Value at Risk of the portfolio.

```
function [ VaR,portVaR ] = VaRfunction( rStocks,p,w )

%VaRfunction - This function yields a vector with the corresponding
Value
%at Risk of every stock in the portfolio
%   Input:
%   . rStocks - matrix of returns of stocks (columns of returns for
every
%   stock)
%   . p - probability level
%   . w - weight vector (column vector)
%   Output:
%   . VaR - column vector with Value at Risk of stock 1 in cell 1...
%   . portVar - Value at Risk of the portfolio with weights equal to
w
%     if w is null, then the weights are assigned with equal
proportion to
%     every asset.


[i,j]=size(rStocks);

for n=1:j
```

```
    VaR(n,1)=quantile(rStocks(:,n),p);
end

if isempty(w)
    w=[(1/j)*ones(1,j)]';
end
portVaR=quantile(rStocks*w,p);

end
```

Appendix 1.4

MatLab code on the calculation of Conditional Value at Risk and optimal portfolio weights.

```
function [CVaR,w]=CVaRfunction(r, Er, b, UB, LB)

%CVaR - this function finds the Conditional Value at Risk of each
asset in
%a given portfolio.
%   INPUT:
%    . r - matrix of returns
%    . b - probability level (0.9<b<1)
%    . LB - lowest bound for each stock weight (-1<LB<1)
%    . UB - upper bound for each stock weight (-1<UB<1)
%       always  UB>LB
%    . Er - target portfolio return (-1<Er<1)
%   OUTPUT:
%    . CVaR - is the Conditional Value at Risk for the portfolio
%    . w - the optimal weights for each asset

[i, j]=size(r);
w0=[(1/j)*ones(1,j)];
VaR=quantile(r*w0',b);
w0=[w0 VaR];

% objective function
n=1:j;
Rfunction=@(w) w(j+1)+(1/i)*(1/(1-b))*sum(max(-w(n)*r(:,n)'-
w(j+1),0));

if isempty(Er),isempty(UB),isempty(LB)
    % matrices
    Aeq=[ ones(1,j) 0];
    beq=[1];

    % minimize the risk function
    [w,CVaR]=fmincon(Rfunction,w0,[],[],Aeq,beq);

else
    % matrices
    A=[-mean(r) 0;  -eye(j) zeros(j,1); eye(j) zeros(j,1)];
    b=[-Er -LB*ones(1,j) UB*ones(1,j)]';
    Aeq=[ ones(1,j) 0];
    beq=[1];
```

```
    % minimize the risk function considering the constraints on w and
Er
    [w,CVaR]=fmincon(Rfunction,w0,A,b,Aeq,beq,LB,UB,[]);
end


end
```

Appendix 1.5

MatLab code on the computation of the cumulative distribution function for the Normal distribution.

```
function [ Fncap ] = ncdf( rStocks )

%ncdf - Normal Cumulative Distribution Function, yields the matrix
composed
%as such:
%In cell 1,1 of Fn there is the probability that the return is lower
than
%or equal to the return in cell 1,1 of rStocks, if returns follow a
normal
%distribution
%   INPUT:
%    . rStocks - matrix of stocks returns
%   OUTPUT:
%    . Fncap - matrix of cumulative distribution function


[i,j]=size(rStocks);

for n=1:j
    A=fitdist(rStocks(:,n),'Normal');
    for m=1:i
        Fncap(m,n)=normcdf(rStocks(m,n),A.mu,A.sigma);
    end
end


end
```

Appendix 1.6

MatLab code on the prediction of returns using the Normal distribution.

```
function [ Fncap ] = ncdf( rStocks )

%ncdf - Normal Cumulative Distribution Function, yields the matrix
composed
%as such:
%In cell 1,1 of Fn there is the probability that the return is lower
than
```

```
%or equal to the return in cell 1,1 of rStocks, if returns follow a
normal
%distribution
%    INPUT:
%     . rStocks - matrix of stocks returns
%    OUTPUT:
%     . Fncap - matrix of cumulative distribution function


[i,j]=size(rStocks);

for n=1:j
    A=fitdist(rStocks(:,n),'Normal');
    for m=1:i
        Fncap(m,n)=normcdf(rStocks(m,n),A.mu,A.sigma);
    end
end


end
```

Appendix 1.7

MatLab code on the calculation of the cumulative distribution function for the t-location scale distribution.

```
function [ Ftcap ] = tlocCDF( rStocks )

%tloccdf - t-Location Scale Cumulative Distribution Function, yields
the
%matrix composed as such:
%In cell m,n of Ftloc there is the probability that the return is
lower
%than or equal t the return in cell m,n of rStocks, if returns follow
a
%t-Location Scale distribution
%    INPUT:
%     . rStocks - matrix of stocks returns
%    OUTPUT:
%     . Ftcap - matrix of cumulative distribution function


[i,j]=size(rStocks);

for n=1:j
    A=fitdist(rStocks(:,n),'tLocationScale');
    for m=1:i
        Ftcap(m,n)=tcdf((rStocks(m,n)-A.mu)/A.sigma,A.nu);
    end
end

plot(sort(rStocks),sort(Ftcap));


end
```

Appendix 1.8

MatLab code on the calculation of β_i .

```
function [ beta ] = beta( rStocks,rMarket )

%This function generates a column vector that contains the betas of
all
%stocks. eg. The beta of stock 1 is in cell (1,1) while beta stock2
is in cell
%2,1...
%   INPUT:
%    . rStocks - matrix of stocks returns
%    . rMarket - column vector of returns of the market
%   OUTPUT:
%    . beta - column vector of stocks betas

n=0;
j=size(rStocks,2);

for n=1:j
    beta(n,1)=regress(rStocks(:,n),rMarket);
end


end
```

Appendix 1.9

MatLab code on the calculation of the error ε_i .

```
function [e] = err(rStocks,rMarket,beta,Rf)

%err - this function generates the error values for the CAPM formula
from a
%process following an fitted t-Location Scale Distribution.
%   INPUT:
%    . rStocks - matrix of stocks returns
%    . rMarket - matrix of market returns
%    . beta - column vector of stocks betas
%    . Rf - risk free rate
%   OUTPUT:
%    . e - matrix of error terms for each stock

[i,j]=size(rStocks);
e=zeros(i,j);
n=1;

for n=1:j
    e(:,n)=rStocks(:,n)-Rf-beta(n,1)*(rMarket-Rf);
end

end
```

Appendix 1.10

MatLab code on the prediction of scenarios coherent with the t-Locations Scale distribution using the CAPM model.

```
function [ rbarStocks ] = exprStocks(rMarket,beta,Rf,e)

%exprStocks - This function creates the matrix of predicted returns
following the CAPM
%model.
%    INPUT:
%     . rMarket - matrix of market returns
%     . beta - vector of stocks betas
%     . Rf - risk free rete of return
%     . e - error matrix
%    OUTPUT:
%     . rbarStocks - matrix of predicted stocks returns following a
%     t-Location Scale


[i,j]=size(e);
dMarket=fitdist(rMarket,'tLocationScale');

for n=1:j
    dErr=fitdist(e(:,n),'tLocationScale');
    for m=1:i
        rbarStocks(m,n)=Rf+beta(n,1)*(random(dMarket)-
Rf)+random(dErr);
    end
end


end
```

Appendix 1.11

MatLab code on the computation of the vector of covariance between the error terms and the predicted returns.

```
function [ CovER,Cor ] = cov_error_ret( rbarStocks,e )

%cov_error_ret - yields the vector of covariances between the error
term
%and the predicted stocks returns.
%    INPUT:
%     . rbarStocks - matrix of predicted stock retruns
%     . e - matrix of error terms
%    OUTPUT:
%     . CovER - column vector with the covariances between the error
%     and the stocks returns
```

```
%     . Cor – column vector of correlations

[i,j]=size(rbarStocks);

for n=1:j
    A=cov(rbarStocks(:,n)',e(:,n)');
    if n>=2
        CovER=[CovER; A(1,2)];
    else
        CovER=A(1,2);
    end
end

for i=1:j
    Cor(i,1)=corr(rbarStocks(:,i),e(:,i));
end


end
```

Appendix 1.12

MatLab code on the prediction of scenarios coherent with the t-location scale distribution, without using the CAPM.

```
function [ rbarStocks_tloc ] = tlocscale( rStocks )

%tlocscale - function that predicts stocks returns, assuming that
they
%follow a t-Location Scale distribution
%   INPUT:
%     . rStocks - matrix of stocks returns
%   OUTPUT:
%     . rbarStocks - matrix of predicted stocks returns

[i,j]=size(rStocks);

for n=1:j
    A=fitdist(rStocks(:,n),'tLocationScale');
    for m=1:i
        rbarStocks_tloc(m,n)=A.mu+A.sigma*trnd(A.nu);
    end
end


end
```

Appendix 1.13

MatLab code for the generation of the CDF of the *GH* distribution:

```
function [ Fgh ] = ghcdf( rStocks )
```

```
%This function generates the CDF for the hyperbolic distribution.
%    INPUT:
%      . rStocks
%    OUTPUT:
%      . Fgh - CDF

[i,j]=size(rStocks);

% Create the cumulative distribution function for the Hyperbolic
% distribution
for n=1:j
    [p] = gh_density(rStocks(:,n),'hyperbolic');
    for m=1:i
        Fgh(m,n)=feval(@hyperbolic_distribution,rStocks(m,n),-10,p);
        disp(m);
    end

end


end
```

Appendix 1.14

MatLab code on the prediction of scenarios from the generalized hyperbolic distribution using either the first method (Option 1) or the Newton-Raphson method (Option 2).

```
function [X,Err] = hyp_random_gen(N,p,opt)

%hyp_random_gen - generates random scenarios from a generalized
hyperbolic
%distribution with defined parameters.
%    INPUT:
%     . N - number of scenarios that the function has to generate
%     . p - structure varaible containing the parameters mu, alpha,
beta,
%     delta, lambda of the GH distribution
%     . opt - user choice of method 1 or 2
%    OUTPUT:
%     . X - number coherent with the hyperbolic distribution
%     . Err - distance from the real value and the predicted one

u=rand(N,1);

if(opt==1)
    for i=1:1:N
        fun=@(x)(hyperbolic_distribution(x,-10,p)-u(i));
        X(i)=fzero(fun,0);
        Err(i)=hyperbolic_distribution(X(i),-10,p)-u(i);
        disp(i);
    end
else
    for i=1:1:N
        X(i)=0;
```

```
        for j=1:1:100
            X(i)=X(i)-(hyperbolic_distribution(X(i),-10,p)-
u(i))/hyperbolic_density(X(i),p.mu,p.alpha,p.beta,p.delta,p.lambda);
        end
        Err(i)=hyperbolic_distribution(X(i),-10,p)-u(i);
        disp(i);
    end


end
```

Appendix 1.15

MatLab code for the calculation of the predicted stocks returns following a hyperbolic distribution, using the CAPM model:

```
function [ rStocks_hyp ] = rS_hyp(rM_hyp,beta,Rf,e,opt)

%exprStocks - This function creates the matrix of predicted returns
following the CAPM
%model.
%   INPUT:
%    . rM_hyp - matrix of market returns
%    . beta - vector of stocks betas
%    . Rf - risk free rete of return
%    . e - error matrix
%    . opt - option 1 or 2 for the prediction.
%   OUTPUT:
%    . rbarStocks - matrix of predicted stocks returns following a
%    t-Location Scale

[i,j]=size(e);

for n=1:j
    [p_e]=gh_density(e(:,n),'hyperbolic');
    [e_hyp]=hyp_random_gen(i,p_e,opt)';
    for m=1:i
        rStocks_hyp(m,n)=Rf+beta(n,1)*(rM_hyp(m,:)-Rf)+e_hyp(m,:);
    end
end


end
```

Appendix 1.16

MatLab code on the application of the Anderson & Darling test.

```
function [ AD ] = AndersonDarling( Femp,Fcap )

%AndersonDarling - this function gives the result of the Anderson-
Darling
%test
```

```
%    INPUT:
%     . Femp - cumulative distribution matrix of the historical stock
%     returns
%     . Fcap - cumulative distribution matrix of the predicted stock
%     returns
%    OUTPUT:
%     . AD - matrix result of the Anderson-Darling test

i=length(Femp);

for n=1:i
    AD(n)=abs(Femp(n)-Fcap(n))/sqrt(Fcap(n)*(1-Fcap(n)));
end

AD=max(AD);


end
```

Appendix 1.17

MatLab code on the computation of the Kolmogorov distance.

```
function [ KD ] = Kolmogorov( Femp,Fcap )

%Kolmogorov - this function yields the result of the Kolmogorov
distance
%    INPUT:
%     . Femp - cumulative distribution matrix of the historical stocks
%     returns
%     . Fcap - cumulative distribution matrix of the predicted stocks
%     returns
%    OUTPUT:
%     . KD - matrix result of the Kolmogorov test

[i,j]=size(Femp);

for n=1:i
    for m=1:j
        KDm(n,m)=abs(Fcap(n,m)-Femp(n,m));
    end
end

for n=1:j
    KD(1,n)=max(KDm(:,n));
end


end
```

Appendix 1.18

MatLab code on the computation of the L1 distance.

```
function [ L1 ] = L1dist( Femp,Fcap )

%L1dist - this function yields the result of the L1 distance
%   INPUT:
%    . Femp - cumulative distribution matrix of the historical stocks
%    returns
%    . Fcap - cumulative distribution matrix of the predicted stocks
%    returns
%   OUTPUT:
%    . L1 - matrix result of the L1 distance test

[i,j]=size(Femp);

for n=1:j
    L1(1,n)=sum(abs(Fcap(:,n)-Femp(:,n)));
end


end
```

Appendix 1.19

MatLab code on the computation of the L2 distance.

```
function [ L2 ] = L2dist( Femp,Fcap )

%L1dist - this function yields the result of the L1 distance
%   INPUT:
%    . Femp - cumulative distribution matrix of the historical stocks
%    returns
%    . Fcap - cumulative distribution matrix of the predicted stocks
%    returns
%   OUTPUT:
%    . L2 - matrix result of the L2 distance test

[i,j]=size(Femp);

for n=1:j
    L2(1,n)=sqrt(sum(abs(Fcap(:,n)-Femp(:,n)).^(2)));
end


end
```