Portfolio Optimization with Non-Linear Instruments

Mattias Strandberg

Mattias.Strandberg@outlook.com

Umeå University

Department of Physics

June 12, 2017

Master's Thesis in Engineering Physics, 30 hp

Supervisor: Kristofer Eriksson (Kristofer.Eriksson@nordea.com)

Examinator: Martin Rosvall (Martin.Rosvall@umu.se)

Abstract

Investors that prefer not to take unnecessarily excessive risks strive to maximize the expected return based on their accepted risk level. Based on the estimated prospects of the returns, investors can make asset allocation decisions from the trade-off between risk and return. Thus, selecting the optimal portfolio is a forward-looking optimization problem. By simulating risk factors for financial instruments, one can generate estimated prospects for the return. Due to the uncertainty in the returns, one must adopt an appropriate risk measure that quantifies the risk so that a decision on the assets can be made. If the problem assumes to have linear constraints, classical programming techniques can minimize the risk, and thus solve the optimal portfolio problem. However, it is not uncommon for investors to put claims on the number of assets they wish to hold in their portfolio. These claims are typically categorized as nonlinear constraints, and, to solve them, portfolio managers use techniques that are very computational-intensive and time consuming. In many cases, it may be sufficient to search for near-optimal solutions, instead of the truly optimal solution which could reduce the time complexity of the problem. Here we present an evolutionary technique called Differential Evolution, which mimics natural selection and tries to optimize a problem by iteratively improving a candidate solution. We found that the results of the Differential Evolution algorithm resemble the result of the classic programming techniques under linear constraints. The Differential Evolution algorithm has shown to have a decent run time compared with classical programming techniques. It is shown that Differential Evolution is a robust search algorithm that is capable to solve nonlinear constraints and may therefore be useful in selecting optimal portfolios. We anticipate that this thesis form the basis for implementing the Differential Evolution algorithm. In addition, we hope that the selection of parameters will help users to improve the speed without losing accuracy.

Sammanfattning

Investerare som föredrar att inte ta onödigt stora risker strävar efter att maximera den förväntade avkastningen utifrån deras accepterade risknivå. Baserat på de estimerade avkastningsmöjligheterna kan investerare fatta beslut om fördelningen bland tillgångarna utifrån avvägningen mellan risken och avkastningen. Att välja den optimala portföljen är således ett framåtblickande optimeringsproblem. Genom att simulera riskfaktorer för finansiella instrument kan man generera estimerade utsikter för avkastningen. På grund av osäkerheten i avkastningen måste man anta en lämplig riskmått som kvantifierar risken så att man kan fatta beslut om hur tillgångarna ska fördelas. Om problemet antas ha linjära begränsningar kan risken minimeras med hjälp av klassisk programmeringsteknik och därmed löser det optimala portföljproblemet. Det är emellertid inte ovanligt att investerare gör anspråk på antalet tillgångar som de vill behålla i sin portfölj. Dessa anspråk kategoriseras som typiska icke-linjära begränsningar och för att lösa dessa använder portföljförvaltare andra typer av tekniker som är mycket beräkningskrävande och tidsödande. I många fall kan det vara tillräckligt att söka efter nästan optimala lösningar istället för den verkligt optimala lösningen som kan minska problemets komplexitet. Vi presenterar här en evolutionsteknik, kallad Differential Evolution, vars metod efterliknar ett naturligt urval som försöker optimera ett problem genom att iterativt förbättra en kandidatlösning. Vi fann att resultatet av Differential Evolution-algoritmen har liknade lösningar som de klassiska programmeringsteknikerna under linjära begränsningar. Tidsperioden för Differential Evolution algoritmen har visat sig vara anseneliga i jämförelse med de klassiska programmeringsteknikerna. Det visas att Differential Evolution är en robust sökalgoritm som kan lösa icke-linjära begränsningar och kan därför vara användbar för att välja optimala portföljer. Vi förutser att denna avhandling kommer att användas som grund för implementeringen av differentialutvecklingsalgoritmen. Dessutom hoppas vi att valet av parametrar kommer att underlätta för användaren med att förkorta tidsperioden utan att resultaten blir missvisande.

Contents

Table of notations

B: Bond price

β: Probability level

c: Call option

C: Cholesky factor

CR: Crossover ratio

Δ: Change over a period

ξ: Absolute error

ε: Portfolio weights rounding cut-off ratio

f: Probability density function

F: Mutation factor

F_β: Conditional Value-at-Risk

G: Objective function value

K: Strike price

κ: Kurtosis

λ: Risk aversion

N: The number of vectors in the population

σ: Volatility

σ_{imp}: Implied volatility

σ^2: Variance

θ: Characteristic function

p: Put option

P: Portfolio value

ρ: Correlation

r_f: Risk-free interest rate

R: Linear return

r: Log return

$risk$: Arbitrary risk measure

S: Stock price

Σ: Covariance matrix

$T - t$: Time to expiration

Γ: Time duration

ϑ: Skewness

μ: Mean

μ_P: Portfolio return

$\bar{\mu}_p$: Target portfolio return

Var: Variance

VaR: Value-at-Risk

v: Portfolio weight vector

w: Portfolio weight

X_t: Time invariant

z_i. Random number distribution

1 Introduction

Investors commits capital to the financial market with the expectation that they will make profits on future returns. But by participating in the financial market one has to accept to take on risks because there is no such thing as a guaranteed profit in an efficient market. Thereby, there will always be some level of probability that the investor will lose money no matter how small the risk. However, by holding various offsetting financial instruments in a portfolio investors can hedge market movements thus lowering the risk and at the same time be provided with a potential profit on their future returns. It so commonly accepted to consider the optimal portfolio as a trade-off between risk and return.

In 1952 Harry Markowitz published his paper "portfolio selection" in which he presented his concept, known as the modern portfolio theory, how diversification may help investors reduce risk. Markowitz argued that the optimal portfolio should not be determined by an asset's individual return but rather from the interaction between assets. By assuming that the market returns are normally distributed Markowitz illustrated how *diversification* on variance may yield higher returns and pose a lower risk on the investment. Nevertheless, variance can just explain the risk effectively if the first two moments of market return are sufficient to explain its nature which only satisfies for elliptically distributions. For some securities such as stocks variance is an appropriate risk measure but certainly not for derivative contracts such as *options*. Equity options is a type of derivative which gives the holder the right, but no obligation, to buy or sell a specified amount of stock shares at a specific strike price on a specific date determined by the form of the contract. Due to the asymmetric shape in their returns several *measures of risk* has developed in attempt to reflect the risk of options contracts.

Markowitz's and other risk models are still often used by practitioners due to their convex objectives and linear constraints which makes it possible for classical programming techniques to solve the optimal portfolio problem. Nevertheless, it is not uncommon for investor's objectives to have nonlinear constraints which cannot be solved by these techniques since the nature of the problem is non-convex. For these kind of problems the industry uses other conventional techniques to find exact solutions which are very computationally-intensive and time consuming. An alternative would be to search for near-optimal solutions instead which could lead to a reduction in the computational time where a sufficiently good approximate solutions can be obtained by using evolutionary

algorithms. Differential Evolution (DE), introduced by Storn and Price in 1997, is a type of evolutionary algorithm that has proven to be very powerful in finding solutions for non-convex problems. Since its introduction DE has been applied in many engineering fields and has recently also gained attention in the field of finance[1]. The DE algorithm tries to optimize a problem by iteratively improving a candidate solution. By making a few or no assumptions about the problem being optimized DE can search vast spaces of candidate solutions. Unlike the classical programming techniques DE does not rely on the gradient which means that the objective does not need to be differentiable. Hence, DE can solve any convex objective whether it is restricted by linear or nonlinear constraints.

The aim of this thesis is twofold. Selecting the optimal portfolio is a forward-looking optimization problem which means that the solution is space and time dependent. Therefore, we will need to use a method that may explain and project expected future prices. Once the prices have been obtained we are going to build an efficient heuristic search algorithm called the Differential Evolution. We will present three cases and illustrate the effectiveness of DE and show that it can handle any risk measure and any type of constraints. By approximating the returns of a stock portfolio and only consider linear constraints in a long position portfolio we will illustrate the correctness and effectiveness of DE by comparing its solutions and running time against the classical programming techniques. Thereafter options will be included in the portfolio and a projection model of expected returns is used where we will show, in the last case, the robustness of the DE by adding a nonlinear constraint that limits the maximal number of securities that can be held in the portfolio.

1.1 Diversification

We illustrate here how diversification may reduce risk of a portfolio consisting of two asset. Assume that the returns on an asset are normally distributed, where the mean is the expected return and the variance is the risk. By defining variance of the portfolio return as a linear combination of the two assets we get,

$$\mathrm{Var}(R) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + \rho w(1-w)\sigma_1\sigma_2 \,, \tag{1.1}$$

where the portfolio weight, $0 \leq w \leq 1$, is the allocation between the assets, σ is their respective standard deviation, and ρ is the correlation. By rearranging the terms we get that

$$\mathrm{Var}(R) = (w\sigma_1 + (1-w)\sigma_2)^2 - 2(1-\rho)w(1-w)\sigma_1\sigma_2 \ . \quad (1.2)$$

Since the correlation factor has a lower and an upper bound it implies that variance also has a lower and an upper bound. It is easy to show that,

$$(w\sigma_1 - (1-w)\sigma_2)^2 \leq \mathrm{Var}(R) \leq (w\sigma_1 + (1-w)\sigma_2)^2 \ . \quad (1.3)$$

If the assets are perfect negatively correlated, i.e. $\rho = -1$, then variance touches its lower bound. Likewise if the assets are perfect positively correlated, i.e. $\rho = 1$, then variance reaches its upper bound. Note that if $\rho = -1$ then it is in fact possible to arrange the weights so that the portfolio is completely without risk but may still take advantage from the asset's individual expected return. In reality, assets are though not perfectly correlated and so there will always be some risk that cannot be diversified away. This is known as systematic risk and as a consequent the feasible set is a parabola in the risk-return spectrum where all individual assets lie either on or within the parabola. An optimal portfolio is every portfolio that lies on the Pareto front, also known as the efficient frontier, and is the rand of the upper part of the parabola. Because it is only rational that an investor would accept higher risk if they get compensated with a higher return. It is for this reason the optimal portfolio considers as a trade-off between risk and return[2].

1.2 Options

There are mainly two type of equity option contracts. A call option gives the holder the right to buy a specific number of shares of stock at a specific strike price on a specific date. A put option gives the holder the right to sell a specific number of shares of stock at a specific strike price on a specific date. Options differ from other derivative contracts in the way that it gives the holder the right, but not the obligation to exercise the contract. This means that the option holder will only exercise if the option is in-the-money, i.e. have a positive value.

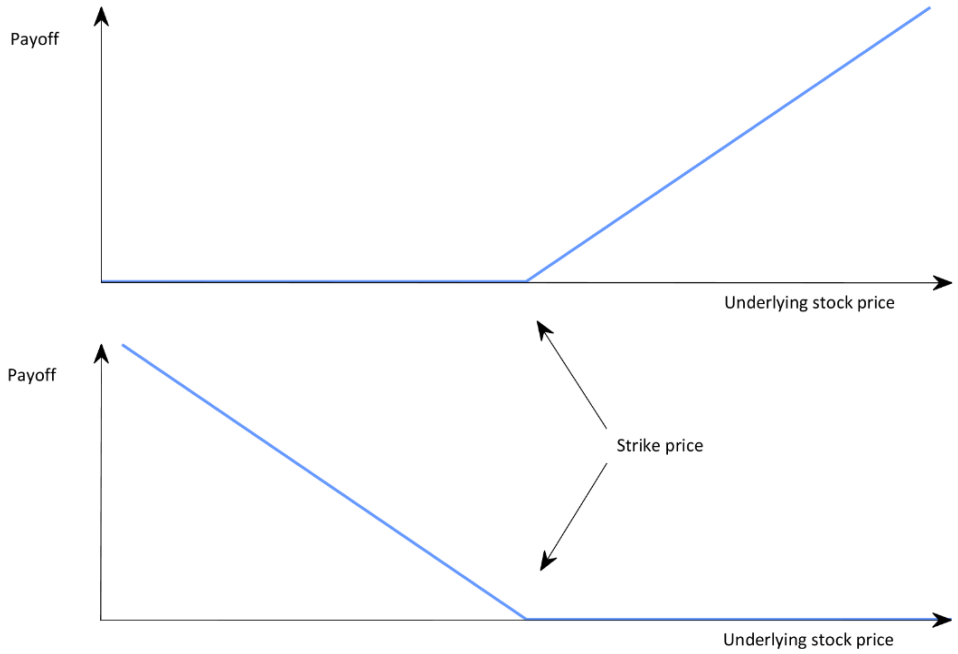

Figure 1: The upper diagram illustrate the payoff for call options and the lower diagram illustrate the payoff for put options.

Mathematically, the payoff on a call option is defined as

$$\max(S_T - K, 0) \,, \tag{1.4}$$

where S_T is the price of the stock at the date T, and K is the strike price at which the share can be sold. Similarly, the payoff on a put option can be described as

$$\max(K - S_T, 0) \,. \tag{1.5}$$

Option writers, i.e. those who sell the options, specify the terms of the contract and charge a premium fee from investors that wants to hold the contract. The terms of the contract specifies the length of the contract and states how the option shall be exercised. Due to the terms of the contract it is difficult to determine what the premium fee should be for an option so that it has a fair price because there are several factors involved with pricing an option, such as:

- The spot price of the stock, S_t
- The strike price, K
- Time to expiration, $T - t$
- The implied volatility of the stock, σ_{imp}
- The risk-free interest rate, r_f

One of the more common type of options is the European option. European options can only be exercised on the expiration date of the contract and because of the terms in the contract it is possible to price these options analytically by a formula. The famous Black-Scholes formula gives the price of a standard European option at any time before the expiry date of the option. Assuming that the stock price is log-normally distributed and follows a geometric Brownian motion with a constant implied volatility the Black-Scholes formula puts a fair price for European styled options[3]. The fair price of a European call option is given by

$$c = S_t\Phi(d_1) - Ke^{-r_f(T-t)}\Phi(d_2)\ , \tag{1.6}$$

where Φ is the cumulative distribution function of a standard normal distribution, and

$$d_1 = \frac{1}{\sigma_{imp}\sqrt{(T-t)}}\left[ln\left(\frac{S_t}{K}\right) + \left(r_f + \frac{\sigma_{imp}^2}{2}\right)(T-t)\right], \tag{1.7}$$

$$d_2 = d_1 - \sigma_{imp}\sqrt{(T-t)}\ . \tag{1.8}$$

Whereas the corresponding European put option is given by

$$p = Ke^{-r_f(T-t)}\Phi(-d_2) - S_t\Phi(-d_1)\ . \tag{1.9}$$

1.3 Measures of Risk

The optimal portfolio is a single-period choice problem. An investor is expected to make allocation decisions at the beginning of a given period (e.g. quarter or a year) based on estimated prospects for the trade-off between the risk and return from a set of assets over the horizon. Once the allocation decision is made the position will stay static over the chosen investing period. Since the investment period is typically long it is important to have a risk measure that accurately reflects the uncertainty of the expected values.

1.3.1 Volatility

Volatility of a portfolio is a measure from historical returns that explains how much the portfolio tends to move. The most popular method of calculating the volatility is by addressing it as the standard deviation of the return, R, that is

$$\sigma = \sqrt{Var(R)}\ . \tag{1.10}$$

If the return is assumed to be independent and identically distributed volatility can truthfully reflect the risk and thus be appropriate to use as a risk measure. Nevertheless, the assumption that market returns are i.i.d. is very strong and may only apply to stocks. It is for that reason that volatility is not an appropriate measure of risk. Besides, risk managers should be more concerned about the left-hand tail of the distribution, corresponding to big losses, instead of accounting for the entire variability.

1.3.2 Value at Risk

The failure of the assumption of i.i.d. became clear in the early 1990s when a number of investment banks plummeted due to the underestimated risk of derivatives, such as options. In response to these events a new risk measure known as Value at Risk (VaR) was developed. VaR focuses on tail losses at a given probability level β from the portfolio's real value profit and loss distribution, ΔP, that is

$$Pr(\Delta P < -VaR_{\beta}) = \beta \ . \tag{1.11}$$

VaR has quickly become the new international standard of measuring financial risk. Even though VaR is a popular risk measure it is not coherent due to the lack of sub-additivity, i.e. VaR of a portfolio with two instruments may be greater than the individual VaRs. This is a major problem since diversification is one of the cornerstones of the modern portfolio theory[4].

1.3.3 Conditional Value at Risk

Another approach of modeling tail losses is by looking at the expected loss that exceeds VaR

$$E[\Delta P | \Delta P < -VaR_{\boldsymbol{\beta}}] \ . \tag{1.12}$$

This approach is known as Conditional Value at Risk (CVaR) and unlike VaR it is coherent and thereof generally accepted as a 'good' risk measure. A risk measure is coherent if two representative portfolios A and B can be described by a function θ that have the following characteristics

- $\theta(A) \leq \theta(B)$ (Monotonicity)
- $\theta(A + B) \leq \theta(A) + \theta(B)$ (Sub-additivity)
- $\theta(\gamma A) = \gamma\theta(A)$ (Homogeneity)
- $\theta(A + z) \leq \theta(A) - z$ (Translation invariance)

In short, these properties can be summarized as following

- Monotonicity: If A weakly stochastically dominates B, meaning that if A always have a better value than B then it should also hold that A is less risky than B.
- Sub-additivity: A position in two different securities should only decrease the portfolio risk (diversification).
- Homogeneity: The position in an asset is proportional to the risk. Doubling the position doubles the risk.
- Translation invariance: Adding an amount of capital reduces the risk by the same amount.

Seeing that CVaR has all of these properties it may be more appropriate to optimize a portfolio using CVaR as the risk measure rather than VaR. If the P&L distribution is normally distributed then CVaR (and VaR) is a scalar to volatility and can thus be determined parametrically. But if the distribution is asymmetrical CVaR needs to be estimated either from historical values or by generating simulated scenarios.

2 Theory

In this section we derive the theory behind the time part of the optimization. To model a random variable over time it is convenient to convert it so that it is time independent. Since the return on a security is dependent on its risk factors (i.e. characteristics or elements that can change and affect the value of the security) over the time horizon invariants of these risk factors need to be found. By converting risk factors into invariants their distributions can be estimated and projected as prospective distributions. With that, the projected probability distribution of the invariants can be converted back into prices of the securities so that the risk and expected return of the portfolio can be estimated. In short, we follow these five steps to optimize a portfolio:

- *Invariants of risk factors*
- *Distribution estimation*
- *Evolution of invariants*
- *Potential return on securities*
- *Estimating portfolio risk and return*

By projecting invariants into future expected returns securities can be allocated in such a way so that the trade-off between risk and return is optimized. However, a simpler method to optimize the portfolio is by approximating the returns. If the portfolio only consist of stocks and the returns are i.i.d. then the expected returns can be scaled proportionally to time and the risk proportionally to *square-root-of-time.*

2.1 Invariants of risk factors

The price of a stock is observed as a discrete random variable that is determined directly by the market and thereby it consists of one single risk factor, i.e. the stock price itself. It is evident by looking at any stock chart that stock prices cannot be invariants since they have trends where the long-term trend usually is positive. However the changes in the return on a stock have similar variation for short time periods. This can be seen by splitting the time series of the return into two halves and creating a scatter plot of each half.

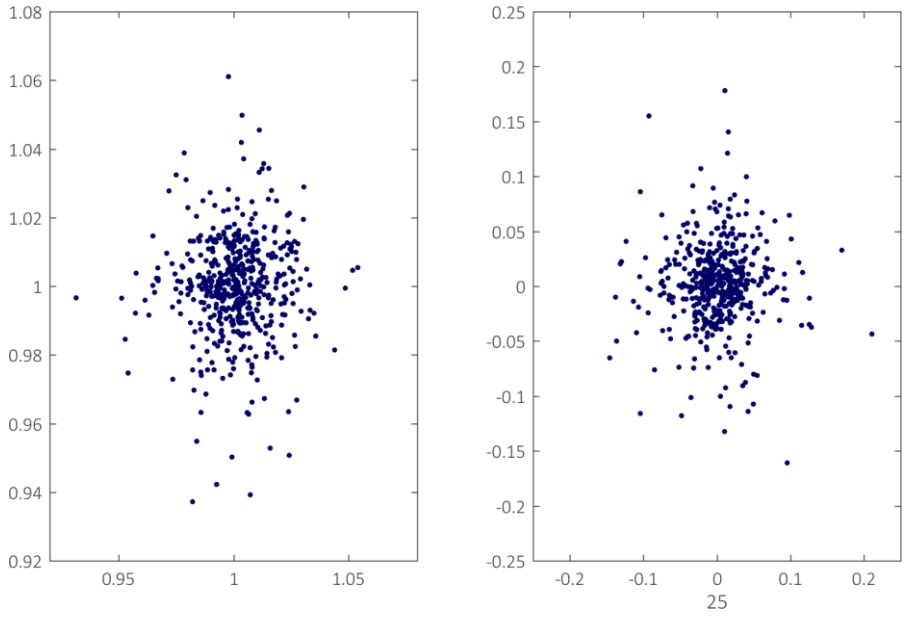

Figure 2: To the left is a scatter plot with lags of linear returns, and to the right is a scatter plot with lags of log returns.

Since there is no visible pattern in figure 2 it concludes that they may be identically distributed. Thus, the return can be regarded as invariants of the stock prices.

It is time that we provide a definition of the return but before we do note that the return can both be defined as *linear return* and as *log return*. Assume that we observe the value on a stock, S_{t-1}, a

period ago, e.g. one week, and at the end of the period a new value is observed. The change of the stock price over a period is defined as

$$\Delta S = S_t - S_{t-1} \text{ ,} \tag{2.1}$$

where the one-period linear return is defined as

$$R_t = \frac{S_t - S_{t-1}}{S_{t-1}} \text{ .} \tag{2.2}$$

The linear return may just as well be defined as a one-period forward looking return. By eq. (2.2) the forward return can be express as

$$R_t = \frac{S_{t+1}}{S_t} - 1 \text{ .} \tag{2.3}$$

The log return is defined by the *discrete compounding factor* and is expressed as

$$1 + R_t = e^{r_t} \text{ .} \tag{2.4}$$

Hence,

$$r_t = \ln\left(\frac{S_{t+1}}{S_t}\right) \text{ .} \tag{2.5}$$

Thus, we have two definitions of the return on a stock. Even though they may both invariants of the stock price it is more convenient to represent the stock market by the log return. We will explain why this is in section 2.4.

The search of invariants for options is though a bit different than from simple stocks since their prices are determined by several variables. Recall that the option price of a European Call option (1.3) can be determined from the underlying stock price if all other factors are constant. In reality both the risk-free interest rate and the implied volatility varies. The change in the risk-free interest rate is usually very small and so we may only need to be concerned of the changes in the underlying stock and the implied volatility. In particular, we consider the at-the-money-forward implied volatility which is the implied percentage volatility of an option whose strike is equal to the forward price of a non-dividend stock at expiry

$$K = S_t e^{r(T-t)} \text{ .} \tag{2.6}$$

The implied volatility is nonetheless not considered to be invariants since there are dependencies within the time series. But the "differences" in the at-the-money-forward implied percentage volatility can be regarded as invariants, that is

$$X_t = \sigma_{imp,t} - \sigma_{imp,t-1} \,. \tag{2.7}$$

We see that by splitting the time series in two halves and creating a scatter plot of each half.

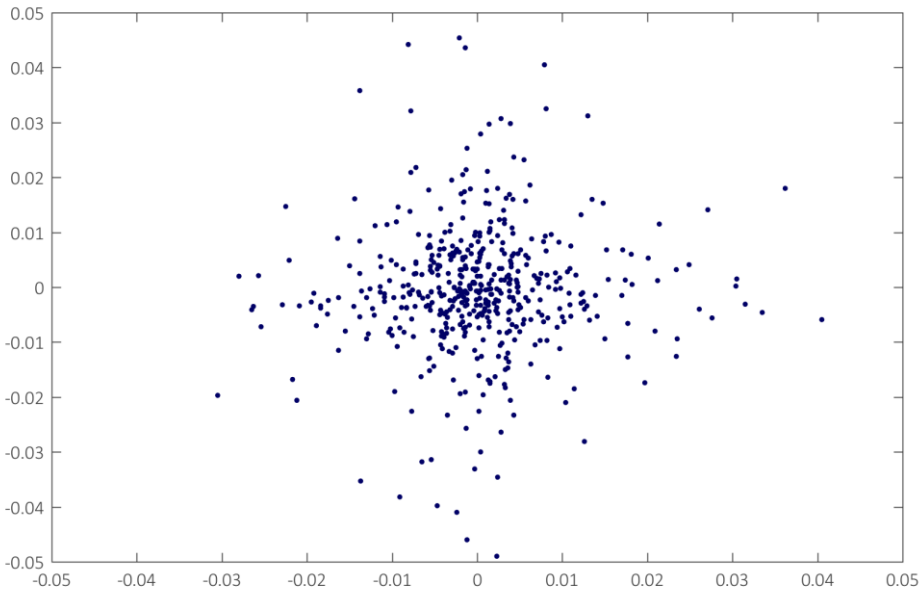

Figure 3: Scatter plot with lags of changes in the implied percentage volatility.

Since there is no visible pattern in figure 3 we may conclude that their "differences" are invariants of the in the at-the-money-forward implied percentage volatility[5].

2.2 Distribution estimation

Any statistical distribution can be described in terms of its moments. For a normal distribution it is sufficient to describe its nature from the first two moments. The first moment is the mean

$$\mu = E[x] \,, \tag{2.8}$$

which estimates the value around which central clustering occurs. If the center is known the "width" of the distribution, known as the variance, can be described as

$$\sigma^2 = E[(x-\mu)^2] \,. \tag{2.9}$$

Market invariants are though usually not normally distributed. Empirical studies suggest that it is not uncommon for market returns to have fat tails, i.e. very large positive or negative observations, which should be very unlikely to occur if they were in fact normally distributed. Therefore higher moments needs to take into account so that their distribution can be explain more accurately. Usually the first four moments are sufficient to explain the nature of their distribution.

The skewness is a non-dimensional quantity characterizes the degree of asymmetry of a distribution around its mean. It is a number that characterizes the shape of the distribution and which is defined as

$$\tau = E[(x-\mu)^3]/\sigma^3. \tag{2.10}$$

The fourth moment is the kurtosis and is also a non-dimensional quantity. It measures the relative peakedness or flatness relative to a normal distribution. The definition of the kurtosis is

$$\kappa = E[(x-\mu)^4]/\sigma^4. \tag{2.11}$$

A standard normal distribution has a kurtosis value of 3. If the kurtosis value is greater than 3, which is usually the case for stock returns, the tails of the distribution is fatter than under normality. These four moments may thus be adequate to explain the market invariants[6].

2.3 Evolution of invariants

The estimated moments, or the characteristics, of the invariants, $\boldsymbol{X}_{\tilde{t}}$, contains all the information on the market for a specific horizon, $\boldsymbol{\tau}$. Indeed, if we have enough of, let's say daily data, we can estimate the risk and the expected return one day forward. However the investment horizon, $\boldsymbol{T}$, is typically longer and so a model is needed to project these invariants into future expected returns so that the risk and the expected return can be estimate at the investment horizon.

To be able to project expected returns from the market invariants we assume that their distribution is jointly normal, that is

$$\boldsymbol{X}_T \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) , \tag{2.12}$$

where $\boldsymbol{\mu}$ is the column mean vector of $\boldsymbol{n}$ elements and $\boldsymbol{\Sigma}$ is the covariance matrix, that is

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{1,1} & \sigma_{2,1} & \cdots & \sigma_{n,1} \\ \sigma_{1,2} & \sigma_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \sigma_{1,n} & \cdots & \cdots & \sigma_{n,n} \end{bmatrix}. \tag{2.13}$$

The characteristic functions of the invariants can be estimated by generating sample distributions using Monte Carlo simulations.

2.3.1 Monte Carlo

The concept behind Monte Carlo is that it calculates the volume of a set by interpreting the volume as a probability. The idea relies on the two fundamental theorems of probability, the central limit theorem, and the law of large numbers. The central limit theorem states that the distribution mean of a large number of independent and identically distributed variables will be approximately normal regardless of the underlying distribution. The law of large numbers states that the sample mean converges to the distribution mean as the sample size increases ensuring that the estimate converges to the correct value. If we assume that the vector probability density function of the invariants $f_{\boldsymbol{X}_\tau}(\boldsymbol{x})$ is integrable over $[-\infty, \infty]$, that is

$$\boldsymbol{\varphi} = \int_{-\infty}^{\infty} f_{\boldsymbol{X}_\tau}(\boldsymbol{x})\, d\boldsymbol{x}, \tag{2.14}$$

then the expected values may be estimated from drawing independent and identically distributed samples $\boldsymbol{z_i}$. By drawing $\boldsymbol{n}$ samples, the expected value of $\boldsymbol{f}$ is

$$\widehat{\boldsymbol{\varphi}}_n = \frac{1}{n}\sum_{i=1}^{n} f_{\boldsymbol{X}_\tau}(\boldsymbol{z}_i)\ . \tag{2.15}$$

According to the strong law of large number,

$$\widehat{\boldsymbol{\varphi}}_n \to \boldsymbol{\varphi} \qquad \text{with probability 1 as} \qquad n \to \infty\ , \tag{2.16}$$

and if $\boldsymbol{f}$ is in fact square integrable, that is

$$\boldsymbol{\sigma}_f^2 = \int_{-\infty}^{\infty} \left(f_{\boldsymbol{X}_\tau}(\boldsymbol{x}) - \boldsymbol{\varphi}\right)^2 d\boldsymbol{x}, \tag{2.17}$$

then the error $\widehat{\boldsymbol{\varphi}}_n - \boldsymbol{\varphi}$ in the estimation is approximately normally distributed. Thus the sample standard deviation of the invariants can be estimated according to

$$\boldsymbol{s}_f = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(f_{\boldsymbol{X}_\tau}(\boldsymbol{z}_i) - \widehat{\boldsymbol{\varphi}}_n\right)^2}\ . \tag{2.18}$$

The convergence rate of the distribution depends on the generated independent and identically distributed samples. The samples can be generated from a pseudo-random number or from a low

discrepancy sequence where the latter is better known as quasi-random sequence and has an important advantage. A pseudo-random numbers is a uniformly distributed random number whereas a low discrepancy sequence is a design that covers the n-dimensional hypercube uniformly. The pattern of its sequence makes it ideal for simulations since it requires fewer samples for the distribution to converge than it does by generating pseudo-random samples[7].

2.3.2 Cornish-Fisher expansion

Recall the assumption we made from eq. (2.12) that the market invariants are jointly normal distributed. Yet, empirical studies suggest that they are not. So how should we handle the higher moments?

Instead of including higher moments in the joint probability distribution they can be imposed on the random number distribution $\boldsymbol{z_i}$. The Cornish-Fisher expansion modifies the quantiles of the probability distribution $\boldsymbol{z_i}$ by including higher moments. By estimating the skewness, $\hat{\vartheta}$, and the kurtosis, $\hat{\kappa}$, for all moments we get from Cornish-Fisher expansion

$$\tilde{\boldsymbol{z}}_i = \boldsymbol{z}_i + \frac{\hat{\vartheta}}{6}\left(\boldsymbol{z}_i^2 - 1\right) + \frac{\hat{\kappa}}{24}\boldsymbol{z}_i\left(\boldsymbol{z}_i^2 - 3\right) - \frac{\hat{\vartheta}^2}{36}\boldsymbol{z}_i\left(2\boldsymbol{z}_i^2 - 5\right). \tag{2.19}$$

By including the first two moments from eq. (2.12) the Cornish-Fisher expansion can be used to generate Monte Carlo samples so that it includes the effects of higher moments. The outcome of the sample $\boldsymbol{x_i}$ can mathematically be formulated as

$$\boldsymbol{x}_i = \hat{\boldsymbol{\mu}} + \boldsymbol{C}\tilde{\boldsymbol{z}}_i, \tag{2.20}$$

Where $\boldsymbol{C}$ is the Cholesky factor of $\boldsymbol{\Sigma}$ that decompose the covariance matrix by the product of its lower triangular matrix and its conjugate transposition, that is

$$\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}'. \tag{2.21}$$

Thus, by generating multiple scenarios from eq. (2.20) the characteristics of the invariants distribution can be determined. Because every outcome is a potential realization and so it means that by redoing the process it is possible to step forward in time. The expected return may thus be determined at the investment horizon by summing up their generated distributions[8].

2.4 Potential return on securities

Now that we are at the investment horizon the distribution of the market return and risk from the distribution needs to be recovered from the investment-horizon invariants. In this section we discuss how market return can be recovered for stocks and options.

2.4.1 Recovering of stock returns

In section 2.1 we saw that linear returns and log returns are invariants of the stock prices where we choose the log returns to be market invariants. We will explain here why we choose the latter by again define the one-period forward looking log return on a stock

$$r_t = \ln\left(\frac{S_{t+1}}{S_t}\right) . \tag{2.22}$$

Since the investment period, h, is typically longer than a single period forward the accumulated effect of the returns needs to be taken into account. The one-period log return can be expanded to an h-period log return and thus defined as

$$r_{ht} = \ln\left(\frac{S_{t+h}}{S_t}\right) = \ln S_{t+h} - \ln S_t . \tag{2.23}$$

Note that,

$$\begin{aligned} \ln S_{t+h} - \ln S_t = \ln S_{t+h} + [-\ln S_{t+h-1} + \ln S_{t+h-1}] + [-\ln S_{t+h-2} + \ln S_{t+h-2}] + \cdots + \\ [-\ln S_{t+1} + \ln S_{t+1}] - \ln S_t \\ = [\ln S_{t+h} - \ln S_{t+h-1}] + [\ln S_{t+h-1} - \ln S_{t+h-2}] + \cdots + \\ [\ln S_{t+1} - \ln S_t] \ . \end{aligned} \tag{2.24}$$

In other words, the h-period return is the sum of the differences in the single period log prices of the stocks which is the same as

$$r_{ht} = \sum_{i=1}^{h} r_{t-i} . \tag{2.25}$$

Hence, it is convenient use the log return to evaluate the return on the stocks at the investment horizon since they are additive across time. The log return can then be turned into linear return, by eq. (2.4) and eq. (2.25) we get

$$R_{ht} = e^{r_{ht}} - 1 . \tag{2.26}$$

We will show in section 2.5 that linear returns are additive across securities and for that reason we rather use linear returns when evaluating the portfolio.

2.4.2 Recovering of option returns

Options have two risk factors, implied volatility and the underlying stock price, that needs to be recovered from their invariants in order to calculate the return on the options. The first one is trivial since the "differences" in the implied percentage volatility is invariants to the implied percentage volatility. Hence, the last observed value of the implied volatility only needs to be added back in. The underlying stock price is not difficult to obtain either. In the last sub section the returns of the stocks were recovered at the investment-horizon. Now all that needs to be done is to turn them back into prices. By raising both sides of eq. (2.23) to the power of $\boldsymbol{e}$ and divide both sides with the last observed stock price, we get

$$S_{t+h} = S_t e^{r_{ht}} \,. \tag{2.27}$$

If the contract of a call option expires at the investment-horizon the payoff function can be used to evaluate the price. By substituting eq. (1.4) with eq. (2.2) the return on the call option at maturity is

$$R_{ht} = \frac{\max(S_T - K, 0) - C_t}{c_t} = \frac{\max(S_T - K, 0)}{c_t} - 1 \,. \tag{2.28}$$

Likewise, if the contract of a call option has yet to expire at the investment-horizon the Black-Sholes formula is used to evaluate the price. By substituting eq. (1.6) with eq. (2.2) the return on a premature call option is

$$R_{ht} = \frac{c_{t+h}}{c_t} - 1 \,. \tag{2.29}$$

2.5 Estimating portfolio risk and return

Financial risk is measured from the portfolio's profit and loss (P&L) but can under certain conditions be measured from the return distribution. To elaborate, assume that the holding period is one day, $t - 1$. At the end of the day we observe the value on the portfolio, P_{t-1}. A profit is realized if the value of the portfolio at the end of the period, P_t, i.e. tomorrow, is greater than it was at the beginning, i.e. today. On the other hand, a loss is realized if the value of the portfolio is less tomorrow than it was today. This means that since the future value is uncertain then so too is the profit and

loss (P&L). We will either realize a profit or a loss at the end of the period equal to the difference between the realized value and the invested value, that is

$$\Delta P = P_t - P_{t-1} \ . \tag{2.30}$$

Since the value on the portfolio at the end of the investment period has not yet been realized the risk needs to be estimated. For the risk to have meaning today it needs to discounted back to a present value by using the price of a discount bond, B_t, that matures at the end of the period. The discounted P&L is then given by

$$\Delta P = B_t P_t - P_{t-1} \ , \tag{2.31}$$

where

$$B_t = e^{-r_f t} \ . \tag{2.32}$$

If the portfolio only consist of long positions it is more convenient to express P&L as a percentage of the portfolio's current value. This means that the return distribution may be analyze instead of P&L. The discounted return on the portfolio is defined as

$$\mu_{t,P}^{(D)} = \frac{B_t P_t - P_{t-1}}{P_{t-1}} \ . \tag{2.33}$$

Another advantage by only holding long positions in the securities is that the portfolio can be written as a weighted sum of the returns of its instruments. The portfolio weight is the proportion of capital, n_i, invested in a certain instrument, i, which holds a price, p_{it}, at the specific time, t, and defined as

$$w_{it} = \frac{n_i p_{it}}{P_t} \ . \tag{2.34}$$

Suppose now that there are k instruments in the portfolio. Then, by definition we have

$$1 + \mu_P = \frac{P_1}{P_0} = \frac{\sum_{i=1}^{k} n_i p_{i1}}{P_0} = \sum_{i=1}^{k} \frac{n_i p_{i0}}{P_0} \frac{p_{i1}}{p_{i0}} \ , \tag{2.35}$$

and by eq. (2.34) and eq. (2.35), we get

$$1 + \mu_P = \sum_{i=1}^{k} w_i (1 + R_i) = \sum_{i=1}^{k} w_i + \sum_{i=1}^{k} w_i R_i = 1 + \sum_{i=1}^{k} w_i R_i \ . \tag{2.36}$$

Hence,

$$\mu_P = \sum_{i=1}^{k} w_i R_i \ . \tag{2.37}$$

Thus, the linear return on the portfolio is the weighted sum of the returns on the securities.

2.6 Square-root-of-time

The reason why we project invariants to expected values is because of that the risk and the expected return increases with time. But if the portfolio only consist of stocks it is possible to approximate the returns instead so that historical values can be scaled proportionally to time. This method works under the assumption that the log returns are independent and identically distributed. Thus the linear return can be approximated as log-normal. If the time frames are small, e.g. daily, then the return should also be small. By the first-order Taylor expansion we have

$$\ln[1 + R_t] \approx R_t \quad \text{if } R_t \ll 1 \,. \tag{2.38}$$

This means that the linear return should also be independent and identically distributed and thereby proportional to time which means that

$$E[R_t + \cdots + R_{t+h}] = E[R_t] + \cdots + E[R_{t+h}] = hE[R_t] \,, \tag{2.39}$$

$$Var(R_t + \cdots + R_{t+h}) = Var(R_t) + \cdots + Var(R_{t+h}) = hVar(R_t) \,.$$

Thus, it follows that volatility is proportional to the square root of time,

$$\sigma_{ht} = \sqrt{hVar(R_t)} = \sqrt{h}\sigma_t \,. \tag{2.40}$$

Considering that VaR and CVaR are scalars of volatility under normality the square-root-of-time rule can be applied to any risk measure[9].

2.7 Summary

We conclude this chapter by giving a short summary. First invariants of market risk factors were identified. We found that the log return on the stocks and the change in the implied volatility are time homogeneous invariants. By estimating the moments of the distributions their individual characteristics were identified. Even though studies suggest that market return have skewed, fat tailed distribution we still assume that their joint distribution is normal. Using the concept of Monte Carlo simulations the invariants distribution at the end of the investment-horizon were generated by drawing samples from a Cornish-Fisher modified i.i.d. distribution to include higher moments. Thereafter the invariants were converted back to returns. Since log returns are additive across time

they were summed up and then convert them back to linear returns. However, for option derivatives whose returns relies on several factors such as the stock price and the implied volatility the invariants had to be mapped back to their original risk factors in order to determine the return for the options contracts. If the portfolio only consist of long positions the risk can be determined by the discounted return on the portfolio. Where the return on the portfolio is determined by the weighted sum of the linear returns of the stocks and the option contracts. But if the portfolio only consist of stocks one can estimate the expected return and risk by scaling historical values proportionally to time instead of projecting the returns.

3 Method

The general task behind any optimization technique is to optimize certain properties of a system by choosing appropriate parameters of the system. For convenience, a system's parameters are usually represented as a vector. The approach to optimize a problem starts by designing an objective function that can model the problem's objectives while integrating any constraints. Commonly, the objective function defines the optimization problem as a minimization task but it may just as well be defined as a maximization task. We will in this section illustrate how the optimal portfolio is constructed from linear and nonlinear constraints. Classical programming techniques can solve for optimal portfolio models if the objective and the constraints are convex. But these techniques relies on the gradient to find a solution meaning that they cannot solve the problem if the constraints are non-convex. We will show that heuristic search algorithms may solve the problem whether the constraints are convex or non-convex by presenting an evolutionary technique, called *Differential Evolution*.

3.1 Portfolio optimization under linear constraints

The optimal portfolio is, as we know by now, considered to be a trade-off between risk and return and is determined by the allocation positions. If the objective is convex and has linear constraints we can solve and find the optimal allocation to the portfolio by using classical programming techniques. We will see, when variance is an appropriate risk measure, that the optimization problem can be stated in such a way so that quadratic programming can solve the optimal portfolio problem. But it is not unusual for stocks to have fat tails and therefore other risk measures are more appropriate to use to reflect the risk. When the risk is based on Conditional Value at Risk (CVaR) we can construct a model that optimize CVaR, and as consequent reduce VaR at the same time, where the optimal allocation positions can be solved by linear programming.

3.1.1 Mean-Variance

The classic Mean-Variance portfolio optimization model aims to determine the fraction $\boldsymbol{w_i}$ invested in each security $\boldsymbol{i}$ that belongs to a predetermined set of $\boldsymbol{n}$ securities so as to minimize the variance in the portfolio's return when targeting a certain value. We may formulate the model using matrix notation as

$$\begin{aligned} \text{Min} \quad & \tfrac{1}{2}\boldsymbol{w}'\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{s.t.} \quad & \boldsymbol{w}'\boldsymbol{R} = \bar{\mu}_p \\ & \boldsymbol{w}'\mathbf{1} = 1 \\ & \boldsymbol{w} \geq \mathbf{0} \end{aligned} \tag{3.1}$$

where $\mathbf{1} = (1,1,\dots,1_n)'$ and $\bar{\mu}_p$ is the target portfolio return from which the variance is to be minimized. Since the objective is a quadratic function with linear constraints the problem can be solved by quadratic programming if the objective is convex.

We can prove that the objective is in fact convex by first giving the definition of convexity. For any $x, y \in \mathbb{R}$, we have

$$f\left(\tfrac{x+y}{2}\right) \leq \tfrac{1}{2}\big(f(x) + f(y)\big) . \tag{3.2}$$

In this case we can interpret $\boldsymbol{x}$ and $\boldsymbol{y}$ as two portfolios. By substituting the objective in eq. (3.1) with eq. (3.2) we get

$$\begin{aligned} \tfrac{1}{2}(\boldsymbol{x}+\boldsymbol{y})'\boldsymbol{\Sigma}(\boldsymbol{x}+\boldsymbol{y}) &\leq \boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} + \boldsymbol{y}'\boldsymbol{\Sigma}\boldsymbol{y} \\ \boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{y} + \boldsymbol{y}'\boldsymbol{\Sigma}\boldsymbol{x} &\leq \boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} + \boldsymbol{y}'\boldsymbol{\Sigma}\boldsymbol{y} . \end{aligned} \tag{3.3}$$

Since $\boldsymbol{\Sigma}$ is positive definite, i.e. $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}$, it satisfies

$$(\boldsymbol{x}-\boldsymbol{y})'\boldsymbol{\Sigma}(\boldsymbol{x}-\boldsymbol{y}) \geq 0 . \tag{3.4}$$

Thus the objective is a convex quadratic programming problem and can thereby be solved by algorithms whose strategy of finding local maxima or minima relies on the gradient which means that the objective function must be continuously differentiable.

3.1.2 Mean-CVaR

If the first two moments of the market return is not sufficient to explain the distribution we need a better risk measure that can reflect the risk more appropriately and only considering tail losses. Conditional Value at Risk (CVaR) looks at the losses that exceeds the threshold of Value at Risk (VaR). VaR and CVaR are closely related and by minimizing CVaR will, usually, also lead to a reduction of

VaR of the portfolio. In order to determine CVaR of the portfolio we present the approach by R. T. Rockarfellar and S. Uryasev, 2000 to derive an expression of CVaR that can be minimized.

Consider the loss distribution function $\boldsymbol{f(x,y)}$ where $\boldsymbol{x}$ is a decision vector representing the portfolio weights and $\boldsymbol{y}$ is a random vector representing expected returns. Thus the loss function is a random variable that is decided by $\boldsymbol{x}$ and its distribution is adopted by $\boldsymbol{y}$. Furthermore we assume that the distribution of $\boldsymbol{y}$ has a density which we denote by $\boldsymbol{p(y)}$. The probability of $\boldsymbol{f(x,y)}$ not exceeding a threshold $\boldsymbol{\alpha}$ is then given by

$$\Omega(\boldsymbol{x},\alpha) = \int_{f(\boldsymbol{x},\boldsymbol{y})\leq\alpha} p(\boldsymbol{y})\, d\boldsymbol{y} \,, \tag{3.5}$$

where $\boldsymbol{\Omega(x,\alpha)}$, for a fixed $\boldsymbol{\alpha}$, is the cumulative distribution function for the loss associated with $\boldsymbol{x}$ and is assumed to be everywhere continuous with respect to $\boldsymbol{\alpha}$.

If we now consider a general case for a given probability level $\boldsymbol{\beta}$ then we can see $\boldsymbol{\alpha}$ as a function $\boldsymbol{\alpha(x,\beta)}$ that express the percentile of the loss distribution and where the lowest value $\boldsymbol{\beta}$ is defined as VaR. That is,

$$VaR_\beta = \alpha_\beta(\boldsymbol{x}) = min\{\alpha \in \mathbb{R}\colon \Omega(\boldsymbol{x},\alpha) \geq \beta\} \,. \tag{3.6}$$

If $\boldsymbol{f(x,y)}$ exceeds VaR then the expected loss, defined as CVaR, can be expressed

$$\Phi_\beta(\boldsymbol{x}) = (1-\beta)^{-1} \int_{f(\boldsymbol{x},\boldsymbol{y})\geq\alpha_\beta(\boldsymbol{x})} f(\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{y})\, d\boldsymbol{y} \,. \tag{3.7}$$

However, the VaR function in the CVaR formula given by eq. (3.7) is very complex and difficult to minimize. We consider therefore another approach by defining a the simpler function of CVaR as

$$F_\beta(\boldsymbol{x},\alpha) = \alpha + \tfrac{1}{1-\beta}\int_{y\in\mathbb{R}} [f(\boldsymbol{x},\boldsymbol{y}) - \alpha]^+ p(\boldsymbol{y})\, d\boldsymbol{y} \,, \tag{3.8}$$

where $[c]^+ = max(c,0)$. If we assume that F_β is convex and continuously differentiable we can determine the characteristics of $\Phi_\beta(\boldsymbol{x})$ and $\alpha_\beta(\boldsymbol{x})$ in terms of the function F_β. Thus, it can be shown that minimizing eq. (3.8) with respect to $\boldsymbol{\alpha}$ is equivalent to the original expression of CVaR, that is

$$\Phi_\beta(\boldsymbol{x}) = F_\beta\big(\boldsymbol{x},\alpha(\boldsymbol{x},\beta)\big) = min_\alpha\, F_\beta(\boldsymbol{x},\alpha) \,. \tag{3.9}$$

The proof behind eq. (3.9) is without the scope of this thesis and readers are referred to the paper of R. T. Rockarfellar and S. Uryasev, 2000, for the complete theorem and proof. Hence we may

approximate CVaR by generating $\boldsymbol{k}$ returns to determine the return distributions for all securities[10]. Thus expressing $\boldsymbol{F_\beta(x,\alpha)}$ as

$$\tilde{F}_\beta(\boldsymbol{x},\alpha) = \alpha + \frac{1}{k(1-\beta)}\sum_{i=1}^{k}[f(\boldsymbol{x},\boldsymbol{y}) - \alpha]^+ \,. \tag{3.10}$$

By adopting CVaR as the risk measure the optimal portfolio can be determined under linear constraints using linear programming by substituting eq. (3.10) as the objective in eq. (3.1)[11].

3.2 Portfolio optimization under nonlinear constraints

Even though classical optimization techniques under linear constraints are mathematically satisfying they cannot handle real world problems that are restricted by nonlinear constraints. Because the optimal portfolio is determined by preference and not entirely by the performance. Indeed, it is not uncommon for investors to put claims on the number of securities they wish to hold in their portfolio. These types of cardinality constraints are nonlinear and that is why portfolio managers must rely on heuristic search algorithms when faced with these types of constraints to obtain a solution to the optimal portfolio problem.

3.2.1 Differential Evolution

Differential Evolution (DE) is a greedy decision process that tries to mimic a natural selection to optimize a problem by iteratively improving a candidate solution from a generated population of $\boldsymbol{N}$ vectors, $\boldsymbol{v_i}$, $i = 1, \dots, N$, where each vector contains $\boldsymbol{n}$ elements and represents the objective variables, i.e. the portfolio weights. DE aims to optimize the trade-off between risk and return instead of minimizing the risk for a given specific return. The objective is thus defined as

$$\text{Max} \quad (1-\lambda)\boldsymbol{w'R} - \lambda\boldsymbol{risk} \,, \tag{3.11}$$

where $\lambda \in \mathbb{R}^+$ is the investor's level of risk aversion and $\boldsymbol{risk}$ may be any risk measure. The basic idea of DE is to produce a new solution for each current vector, $\boldsymbol{v_i}$, where the new solution is a combination of four current solutions in the population. It works in the following way: First we select a target vector, $\boldsymbol{v_0}$, from the current population. Then randomly select three different vectors and use one of them as a base vector and add the weighted difference of the two others to construct a new solution. We formulate the new vector as

$$\boldsymbol{v_m} = \boldsymbol{v_1} + F(\boldsymbol{v_2} - \boldsymbol{v_3}) \,, \tag{3.12}$$

where $F : (0,1+)$ is a so called mutation factor that controls the rate at which the population evolves. Finally we perform a crossover solution with the parent and the mutated vector. Each element in the new trail vector will be determined by a user-defined crossover ratio, $CR: [0,1]$, and a pseudo random generated number, z. The crossover controls the fraction of parameter values copied from the mutant vector, so that

$$\boldsymbol{v}_{\eta_j} = \begin{cases} \boldsymbol{v}_{0_j} \; if \; \boldsymbol{z}_j < CR \\ \boldsymbol{v}_{m_j} \; if \; \boldsymbol{z}_j \geq CR \end{cases}. \tag{3.13}$$

If the generated number is less than the crossover ratio the trail vector will inherit element j from the target vector. Likewise if the generated number is greater or equal to the crossover ratio the trail vector will inherit element j from the mutated vector. In order to deal with the discontinuities of the search space due to the constraints on the weights a repair function is introduced that ensures that the trail vector will always stay within the feasible set of solutions. For instance, by only allowing the portfolio to consist of long positions we can easily make sure that the solution stays within the feasible set so that

$$\sum_{i=1}^{n} \boldsymbol{v}_{\eta_i} = 1 \; . \tag{3.14}$$

Similarly way we can impose a penalty function to handle cardinality constraints so that the portfolio only consist of H elements, that is

$$\left\{ i \, \epsilon \, \mathbb{Z}^{+} \middle| \boldsymbol{v}_{\eta}(i) \neq 0 \right\} \leq H \; . \tag{3.15}$$

Once the trail vector is well secured it is compared against the target vector and the one with the greatest objective value will live on in the next generation. A scheme of the algorithm can be seen in figure 4.

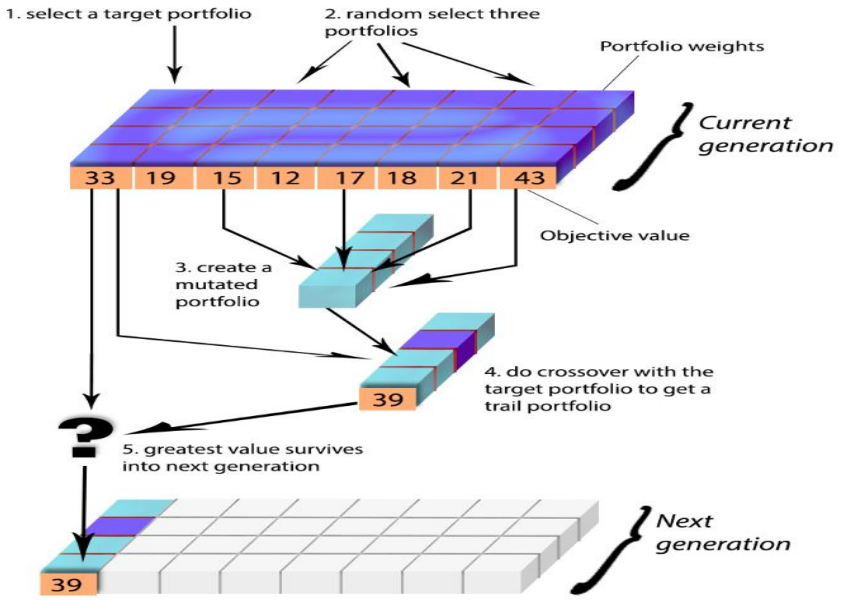

Figure 4: A scheme over the Differential Evolution algorithm.

As long as the different vectors elements do not "agree" the difference vector $\boldsymbol{v}_2 - \boldsymbol{v}_3$ will have non-zero elements and thus generate new solutions. The new solution will always move towards the solution that is considered to be the best and eventually all vectors in the population will agree by flocking around the global optimum[12]. This means that we can, after each generation, check whether the elements in the population agrees to the population mean by rounding the portfolio weights using a cut off ratio, $\boldsymbol{\varepsilon}$. Once all solutions are considered to be "close enough" to the global optimum the algorithm stops.

However there is no guarantee that DE will converge at the true optima and by wrongly specifying the parameters will most of the time lead to incorrect result. Thus, the precision of the result is dependent on the user-specified parameters and by adjusting these parameters the result can significantly be improved. Therefore, it is important to correctly specify these four variables:

- N: The number of vectors in the population
- F: Mutation factor
- CR: Crossover ratio
- $\boldsymbol{\varepsilon}$: Portfolio weights rounding cut-off ratio

Assume that we have obtained a qualified solution. The result can most likely be improved by increasing the number of vectors and the number of iterations by increasing the cut-off ratio so that

the portfolio weights are rounded at a lower decimal point. As a consequence the solution will take longer to converge. By using the qualified solution as a reference we can try to find an optimal mutation factor and crossover ratio that yields similar result for a smaller number of vectors and the cut-off ratio. We compare the solutions by the absolute error of their objective function value,

$$\boldsymbol{\xi}_i = \left|\boldsymbol{G}_{q_i} - \boldsymbol{G}_{c_i}\right| , \tag{3.16}$$

where $\boldsymbol{G}_{q_i}$ is the qualified objective function value, and $\boldsymbol{G}_{c_i}$ corresponds to the comparison objective function value at point $\boldsymbol{i}$. However a smaller population does not necessarily lead to a reduction in the time duration, $\boldsymbol{\Gamma}$. Therefore, an ideal result can be obtained from the product of the normalized mean of the absolute error and the time duration, that is

$$\bar{\bar{\boldsymbol{\xi}}}\bar{\bar{\boldsymbol{\Gamma}}} . \tag{3.17}$$

Although there is no upper limit on $\boldsymbol{F}$, effective values are seldom greater than one. Thus the ideal parameters should be within a finite set.

4 Results

We present here the choice of parameters for the DE algorithm and compare its results to other optimizing techniques. We start by considering a portfolio consisting only of stocks and assume that the daily log returns are normally distributed. Thus, we approximate the return as log-normal so that the expected return can be scaled proportionally to time. By solving the mean-variance model using quadratic programming (QP), we use the result to find good parameters for the DE algorithm so that it produces qualified results. From these we find parameter values that has the best fit for the objective and thus provides a better solution. To illustrate that DE can solve for any risk measure we use the approach by R. T. Rockarfellar and S. Uryasev to minimize CVaR and compare the results against a linear programming (LP). Finally, we simulate risk factors and show DEs full potential by including options and only allowing investors to hold a maximal number of securities in the portfolio.

4.1 Differential Evolution versus Quadratic Programming

In this case the portfolio consisted of 15 assets from S&P 500 and the returns where collected from daily stock prices. We have assumed that the log returns are normally distributed and thereby approximated the return as log-normal. Furthermore, we have assumed that there are 252 trading days a year and thus scaled the returns proportionally to time. To show that Differential Evolution can be applied to the optimal portfolio problem we had to compare the solution against MATLABs built in quadratic programming algorithm. A qualified result was obtained from adjusting the parameters by trial and error.

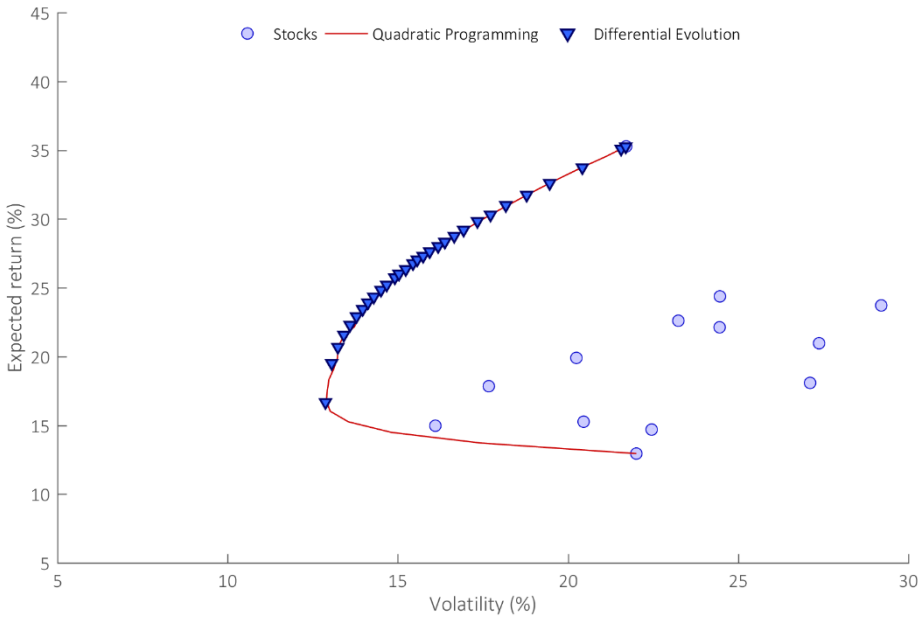

Figure 5: Optimal trade-off between the expected return and volatility determined by Quadratic Programming and Differential Evolution.

We see that the optimal expected return on the investment ranges from 16% to 35%, and the volatility from 13% to 22%, that is the movement tendency of the portfolio. Furthermore we see that diversification pose a lower risk and at the same time yield a greater return, where every optimal portfolio lies along the efficient frontier, i.e. the upper part of the parabola. Seeing that DEs solutions lies along the efficient frontier we may confirm that the solution are the same as QP by investigating the distribution of the portfolio weights.

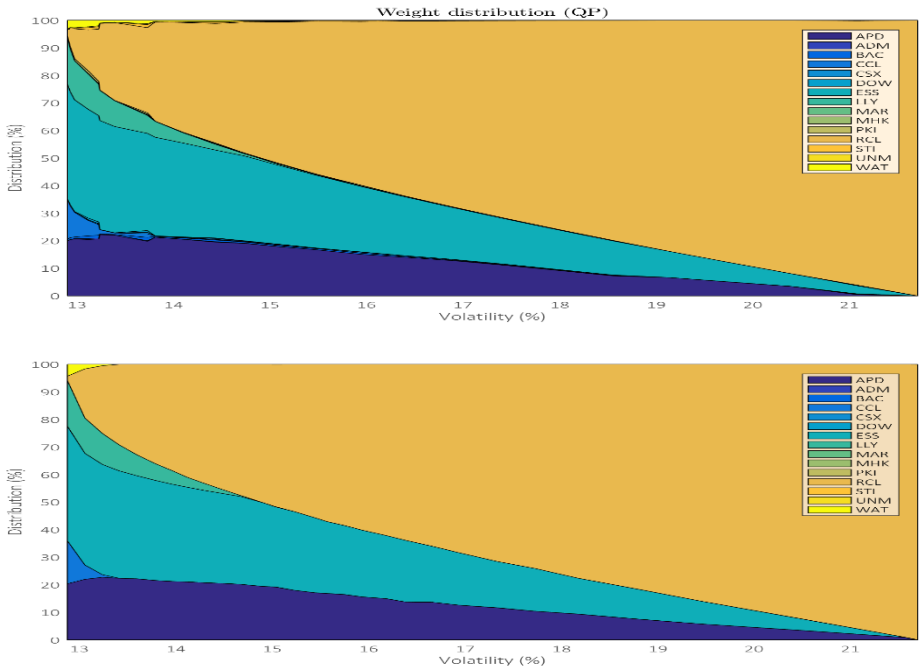

Figure 6: Area plot of the distribution among the securities for each level of risk. The upper plot is produced by Quadratic Programming and the lower by Differential Evolution.

By looking at figure 6, it is clear that the two methods have the same solution. However, DE is not as efficient as QP since its method took longer to converge. The average time duration of DE proved to be, approximately 2.74 seconds which in comparison to QP only took 0.47 seconds.

4.2 Adjusting the parameters

In the last case we found that by specifying N=15, F=0.8, CR=0.5, and ε=3 yields qualified solutions. Using variance as the risk measure we generated a reference vector of 2000 optimal portfolios along the efficient frontier for a population number of 45 vectors and the cut-off ratio of 4. Then, for a fixed population size of 15 and cut-off ratio at 3 we tried to find similar solutions by increasing F and CR by 0.1 and letting DE generate another 2000 optimal portfolios along the efficient frontier at each step.

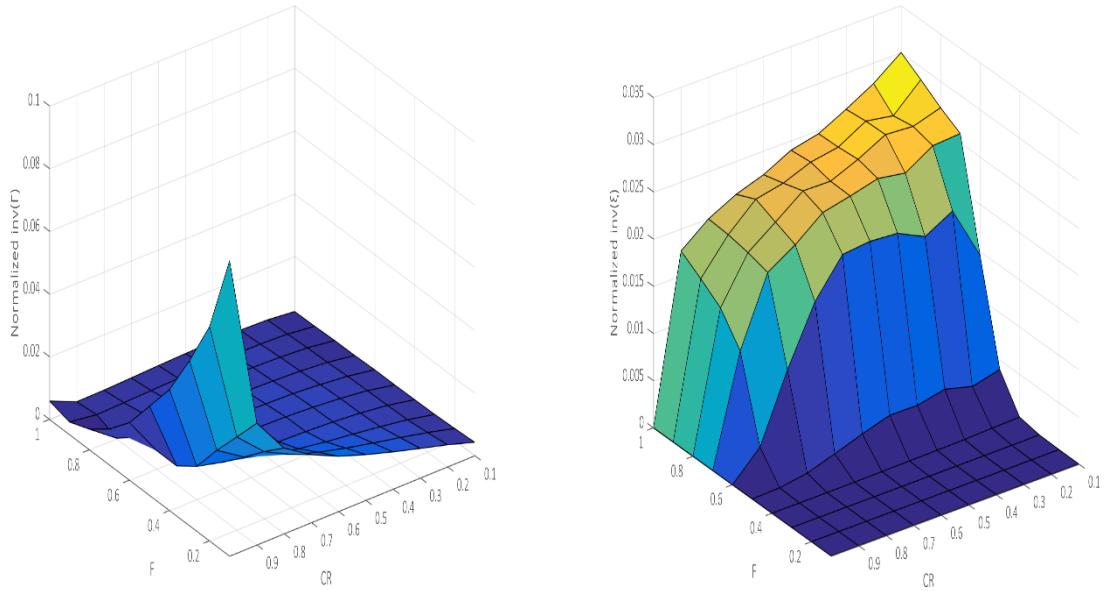

Figure 7: The plot to the left is the normalized inverse time duration and the plot to the right is the normalized inverse absolute error.

The plot to the left in figure 7 illustrate the normalized inverse of time duration and to the right is the normalized inverse of the absolute error. As we can see, we obtained a more accurate result by decreasing CR and increasing F but that also lead to an increase in the time duration. The ideal set of parameters was thereby obtained by their product given by eq. (3.16).

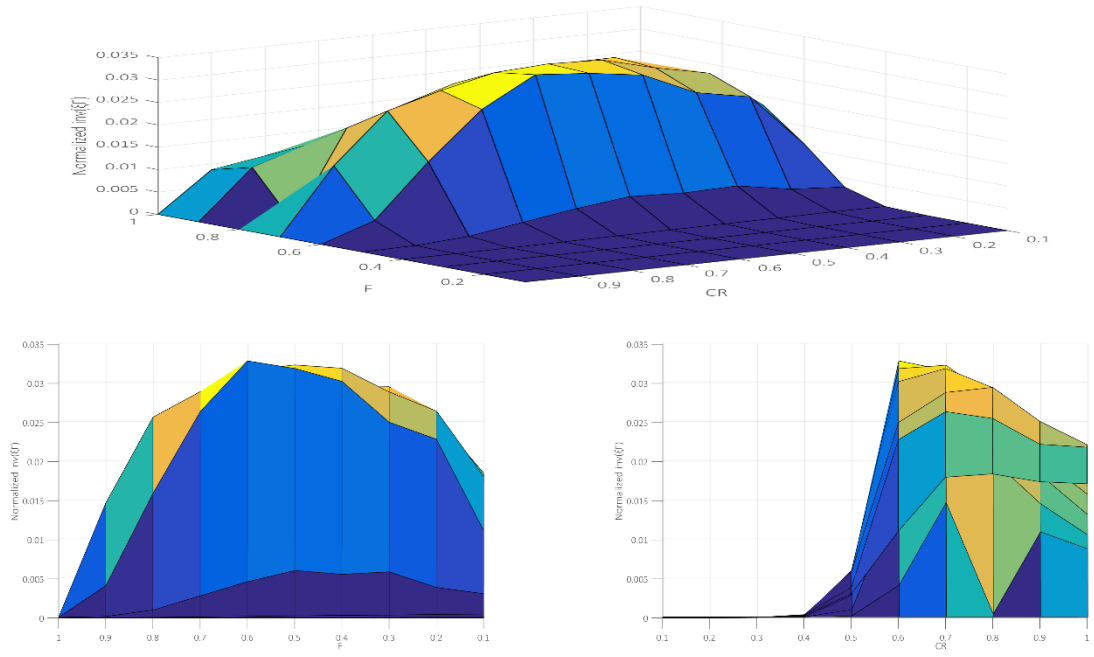

Figure 8: The normalized product of the time duration and the absolute error.

By looking at figure 8 we can see that the result indicate that the ideal parameters should be F=0.6 and CR=0.6. Similar results were obtained on another data set where we have left the result in the Appendix. Thus the results suggest the region from which ideal parameters could be found.

4.3 Differential Evolution versus Linear Programming

To show that DE can handle any risk measure we used the approach by R. T. Rockarfellar and S. Uryasev to minimize CVaR by comparing the solutions against MATLABs built in linear programming algorithm. We used here the same assumption as in the last case and approximated the returns. We then optimized CVaR at the 95% probability level using the parameters values N=15, F=0.6, CR=0.6, and ε=3.

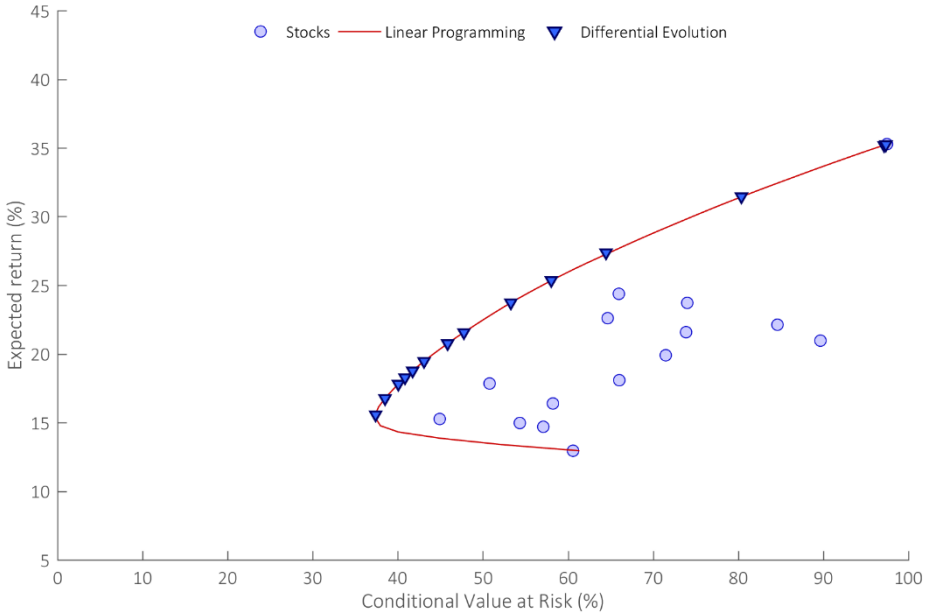

Figure 9: Optimal trade-off between the expected return and 95% Conditional Value at Risk determined by Linear Programming and Differential Evolution.

We see that $CVaR_{95\%}$ ranges from 35% to 100%, which is the average proportion of an investors capital that will be lost for the 95% probability level. This approach also led to a reduction in $VaR_{95\%}$, since it is a factor involved in the process of minimizing $CVaR_{95\%}$.

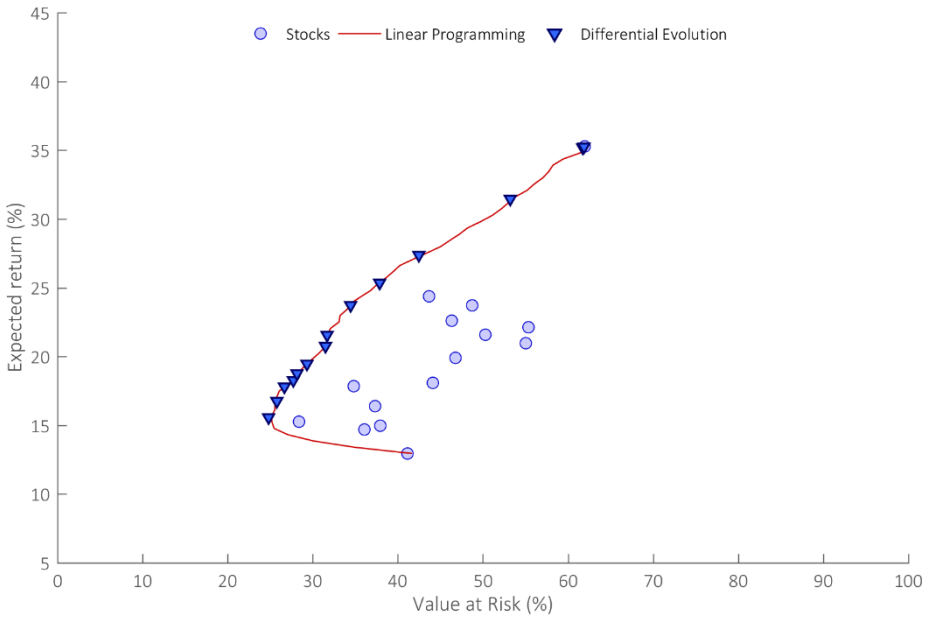

Figure 10: Optimal trade-off between the expected return and 95% Value at Risk determined by Linear Programming and Differential Evolution.

Here we see $VaR_{95\%}$ ranges from 25% to 60%, which tells us the probability that the loss will exceed the 95% probability level. In any case, we see that DE lies on the efficient frontier and we can confirm that the solutions are the same as LP by again investigating the portfolio weight distribution.

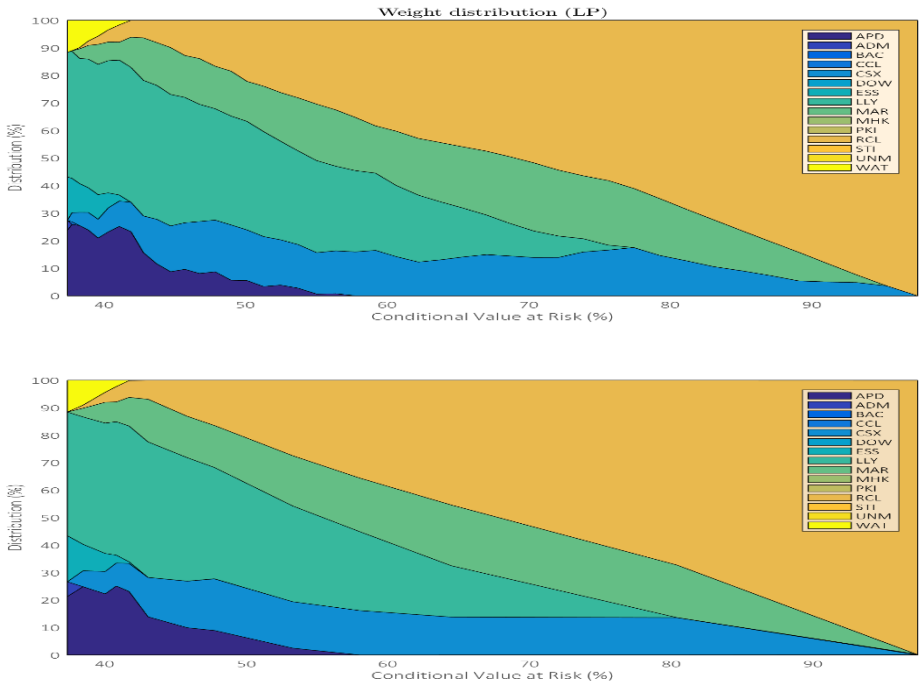

Figure 11: Area plot of the distribution among the securities for each level of risk. The upper plot is produced by Linear Programming and the lower by Differential Evolution.

Indeed, we see that the two methods have the same solution. Hence, DE can be used as an alternative to LP for solving the optimization problem when CVaR is used as the risk measure. But DE is nonetheless not as efficient as LP since it took longer to converge. The average time duration of optimization CVaR by DE took approximately 117 seconds whereas the time duration for LP was 69 seconds.

4.4 Differential Evolution under nonlinear constraints

The optimal portfolio suggest that it is possible to lower the risk by including risky securities. Therefore we included 5 stock index, and based on these, 3 put options and 2 call options in the portfolio. To accurately capture the returns we utilize the projection model of the invariants and used $CVaR_{95\%}$ as the risk measurement. The weekly data for the stock and index prices have then been projected to annual returns from a modified Monte Carlo simulator that implements Cornish-Fisher expansion to include higher moments. We have used the 30 day implied volatility data as one of the risk factors for option prices and assumed that they may be scaled to annual implied volatility using the square-root-of-time rule whereas the risk-free rate have been assumed to be constant at 3%.

The main purpose of using DE is for its ability to solve the optimal portfolio problem when the constraints are nonlinear. To show this we have imposed a cardinality constraint that only allows the portfolio to consist of no more than 6 securities where a penalty function have been included to handle these constraints.

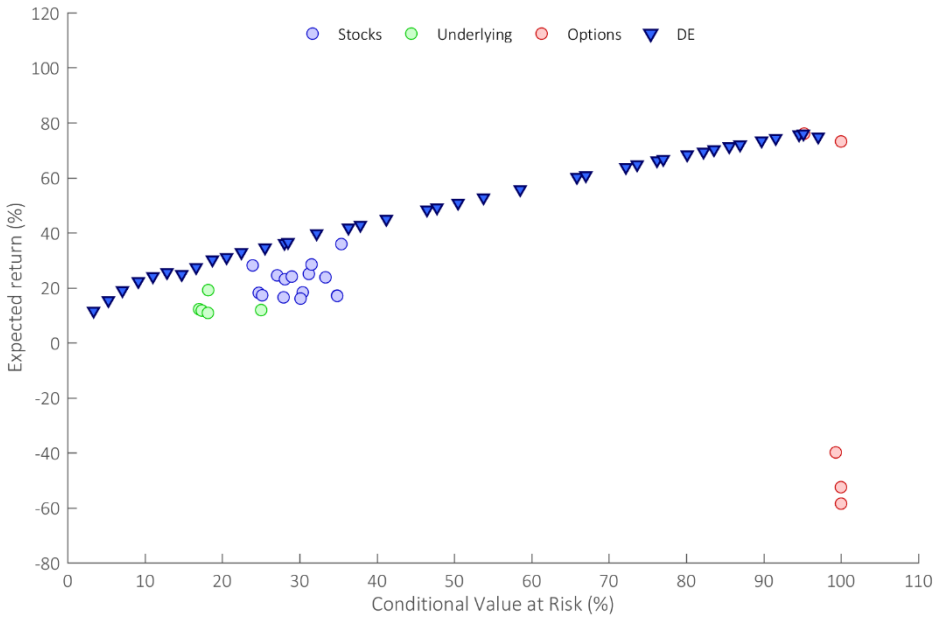

Figure 12: Optimal trade-off between the expected return and 95% Value at Risk determined by Differential Evolution.

As one can see, risk can be diversified even further by including options to the portfolio. We see that this is by analyzing the portfolio weight distribution where we may also see that DE is capable of handling nonlinear constraints.

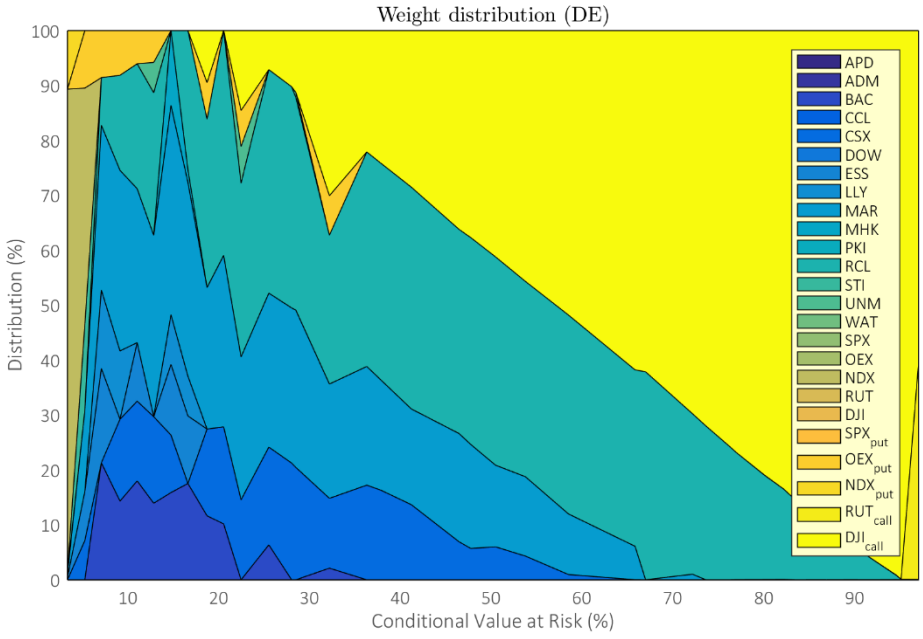

Figure 13: Area plot of the distribution among the securities for each level of risk.

Seemingly, the portfolio consist of no more than 6 securities at any level of risk and whether an investor is looking for a higher rate of return or lowering the risk it is almost always optimal to include some portion of options in the portfolio.

5 Discussion

The aim of this thesis was to use a method that may represent expected returns and to construct a model that solves the optimal portfolio problem. The purpose was to show the usability of a search algorithm based on an evolutionary technique, called Differential Evolution, and show how it can be applied to the optimal portfolio problem. When the portfolio only consisted of stocks the returns were assumed to be normally distributed. Thus, the log returns were approximated as linear returns so that the expected returns could be estimated by scaling the linear returns proportionally to time. However, the assumption that the returns are normally distributed is a bad practice since empirical studies suggest that this is not the case. This means that the assumption needs to be checked every time before use and therefore it not always appropriate to scale linear returns proportionally to time. Furthermore, since volatility can only reflect the risk appropriately if the returns are independent and identically distributed. Solving the optimal portfolio problem could leads to incorrect results if this assumption fails to hold. That is not to say that volatility is not useful because it can have practical applications when higher moments is negligible. It may nevertheless be more appropriate to project expected returns using Monte Carlo simulations even in the absent of options and reflect the risk using CVaR. There are however some difficulties in optimizing CVaR since it assumes the tail of the distribution to be continuous and without jumps which may not be the case in historical values or for projected option prices. Thus optimizing CVaR can be ill-posed since there could exist several optimal solutions.

In order simulate the returns we had to transform risk factors into invariants. We saw that the risk factor for a stock is the price itself and for options the underlying stock price and the implied volatility was considered as risk factors and the risk-free interest rate was assumed to be constant. We made assumption that the Black-Scholes formula can be applied for pricing the options. But this is only true if the risk-free interest rate and the implied volatility is constant and if the stock price is assumed to follow a Brownian motion. Still, none of these assumptions are correct and therefore another approach of reevaluating the options could lead to more accurate results. Furthermore we assumed that the invariants were joint normally distributed and that Cornish-Fisher expansion would capture the effect of higher moments. These parameters are though highly sensitive and difficult to estimate if the distributions has high skew and kurtosis which means that their estimate can lead to incorrect

results. An alternative could therefore be to represent the correlations using some type of GARCH volatility model or by including copulas. All of this factors makes much room for improvement in the task of projecting expected returns of the securities and suggestively the adequacy of the solutions to the optimal portfolio problem could be checked by back testing.

We saw that classical programming techniques can solve the optimal portfolio problem if the constraints are linear and that the Differential Evolution algorithm can be applied to obtain satisfying results. To run the Differential Evolution algorithm user are required to specify parameters. Appropriate specification on the parameters was obtained by comparing the objective function value to an already qualified solution. Then the efficiency of the Differential Evolution algorithm was explained by comparing the duration time against classical programming techniques. When the risk was based on the volatility the solution was compared with MATLABs integrated quadratic programming algorithm and when CVaR was used as the risk measurement the solution was compared with MATLABs integrated linear programming algorithm. Even though Differential Evolution may not replace classical programming techniques we still think it has a respectable time duration considering that its purpose is to solve for non-convex constraints. To illustrate that Differential Evolution may handle cardinality constraints, which are typically nonlinear, a maximal number of securities held in the portfolio was added.

5.1 Conclusion and suggestions

We showed that Differential Evolution is a robust search algorithm that can be applied to the optimal portfolio problem even for nonlinear constraints. We have seen that every optimal portfolio is a set of Pareto optimality and lies along the rand on upper part of the feasible set. However the solutions to the Differential Evolution algorithm tends to gather around the region that is most concave. This means that since we do not know what the feasible set looks like a great number of solutions may be needed if one wish to enfold the entire efficient frontier. This would of course be very time consuming and therefore it should be favorable if the duration time could be improved. The duration time depends how one specifies the parameters involved in the algorithm and to the best of our knowledge no previous studies have been done to improve the parameter selection for the optimal portfolio problem. We saw that the precision of the result were improved by correctly specifying the mutation factor and the crossover ratio and illustrated which parameters that yields the most precise

result. But striving to obtain the best precision is not ideal. In the attempt to improve the result of CVaR for the second case the duration time increased significantly, seeing that it increases exponentially, and when we specified the mutation factor as 1 and the crossover ratio as 0.1 we were, after a full day of running the simulation, unable to obtain a result. We believe the reason for this was that, by using this set up, the DE algorithm got stuck in suboptimal solutions that it had troubles to move away from. Therefore, we believe that it is ideal to specify the mutation factor within the range (0.5, 0.8] and the crossover ratio [0.2, 08].

As it comes to the other parameters little endeavor has been devoted to improving the number of vectors and the cut-off ratio. A suggestion would be to find a rule-of-thumb for the smallest number of vectors so that sufficient results can be reproduced. Alternatively other strategies in the mutation process could prove useful in the process of obtaining faster time duration. The one we used is the one given by Storn and Price but since the introduction of Differential Evolution other strategies have been develop which could possibly improve the time duration. Still, Differential Evolution has proven useful in solving for nonlinear constraints with a reasonable time duration.

6 Appendix A1

6.1 Historical stock data

Dates for the historical daily returns: `2013-05-15` to `2017-03-24`

Dates for the historical weekly returns: `2013-03-24` to `2017-03-24`

6.1.1 Stocks

Air Products and Chemicals, Inc. (APC)

Archer-Daniels-Midland Company (ADM)

Bank of America Corporation (BAC)

Carnival Corporation (CCL)

CSX Corporation (CSX)

The Dow Chemical Company (DOW)

Essex Property Trust, Inc. (ESS)

Eli Lilly and Company (LLY)

Marriott International, Inc. (MAR)

Mohawk Industries, Inc. (MHK)

PerkinElmer, Inc. (PKI)

Royal Caribbean Cruises Ltd. (RCL)

SunTrust Banks, Inc. (STI)

Unum Group (UNM)

Waters Corporation (WAT)

6.1.2 Stock indices & volatility indices

S&P 500 (^GSPC)

S&P 100 INDEX (^OEX)

NASDAQ-100 (^NDX)

RUSSELL 2000 INDEX (^RUT)

Dow Jones Industrial Average (^DJI)

VOLATILITY S&P 500 (^VIX)

CBOEO EX implied Volatility (^VXO)

CBOE NASDAQ 100 Volatility (^VXN)

CBOE RUSSELL 2000 VOLATILITY IN (^RVX)

DJIA VOLATILITY (^VXD)

6.2 Parameter simulation

Similar results were obtained on a different data set as to the one given in section 4.2

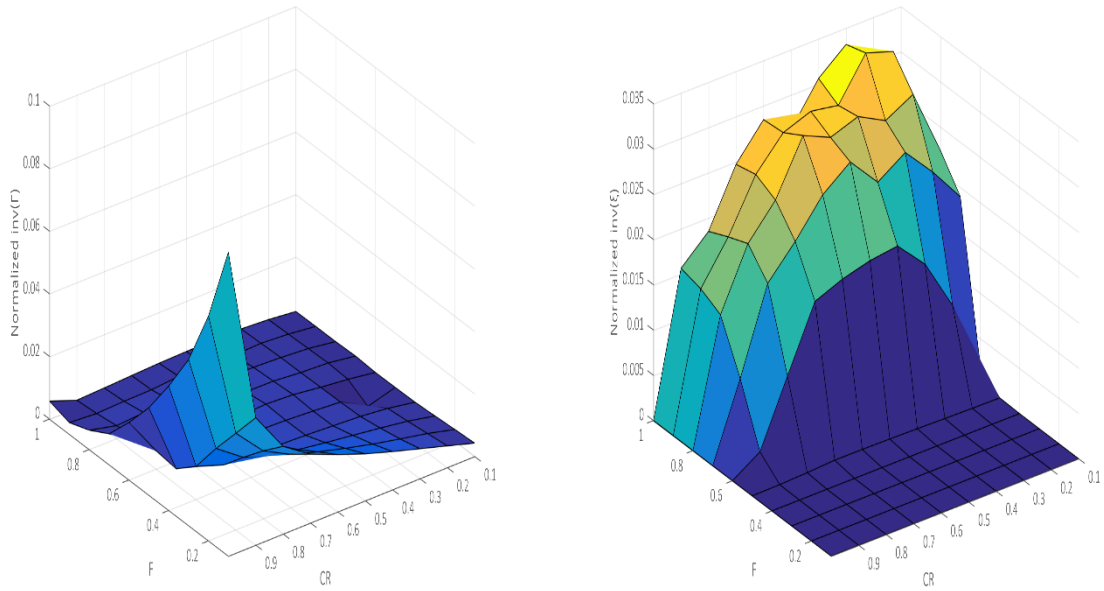

Figure 14: The plot to the left is the normalized inverse time duration and the plot to the right is the normalized inverse absolute error.

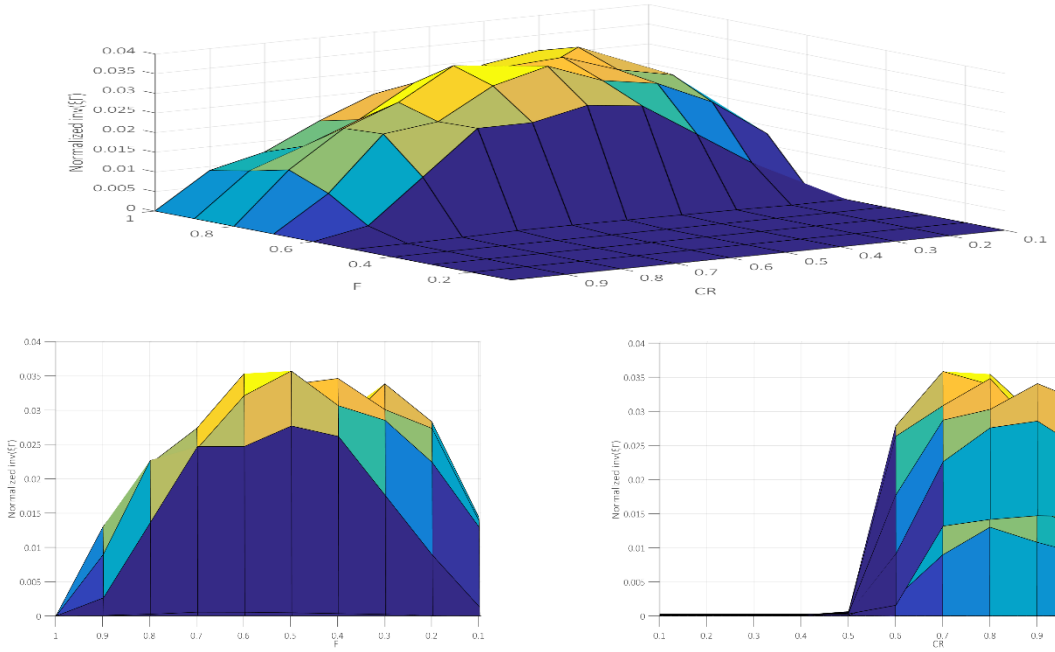

Figure 15: The normalized product of the time duration and the absolute error.

6.3 MATLAB code

The code to the Differential Evolution algorithm have been highlighted with grey background.

```
%Options based on stock indices and implied volatility data:
%SPX, OEX, NDX, RUT, DJX, VIX, VXO, VXN, RVX, VXD
%-------------------------------------------------------------------
-

%Get Data
%data1 = GetData(1) ;
load('data1.mat') ;
function data = GetData(Index)
%Returns market adjusted closing prices for assets from yahoo
%finance. Index determines the portfolio.

%Index = 1 indicates a simple portfolio consisting stocks only.
%Daily prices are retrieved from
%2009-04-14 to 2017-03-24

%Index = 2 indicates a complex portfolio consisting of stocks, stock
%indices and ATMF implied volatilities. Prices for stock indices and
%ATMF implied volatilities are used for option evaluation and
%modelling.
%Weekly prices are retrieved from
%1997-05-12 to 2017-03-24

%Assets are arranged in the following order
assets =
{'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR','MHK',...
    'PKI','RCL','STI','UNM','WAT',...

'^GSPC','^OEX','^NDX','^RUT','^DJI','^VIX','^VXO','^VXN','RVX','VXD'
} ;

c = yahoo ;
if Index == 1
    prices = zeros(2001,length(assets)-10) ;
    for i = 1:length(assets)-10
        price = fetch(c,assets(i),'Adj Close','04/15/09','03/24/17')
;
        prices(:,i) = price(:,2) ;
    end
    data = prices(end:-1:1,:)' ;
elseif Index == 2
    prices = zeros(1001,length(assets)) ;
    for i = 1:length(assets)
        if i <= length(assets)-5
            price = fetch(c,assets(i),'Adj
Close','05/12/97','03/24/17') ;
            prices(:,i) = price(1:5:end,2) ;
        else
            price = fetch(c,assets(i),'Adj
Close','04/05/13','03/24/17') ;
            prices(:,i) = price(:,2)/100 ;
```

```
        end
    end
    prices = prices(end:-1:1,:)' ;
    type = [1*ones(length(assets)-10,1); 2*ones(5,1); 3*ones(5,1)] ;
    data = table(prices,type) ;
end

end

%Expected returns and covariance
[Returns,ExpR,Sigma] = ExpRetVol(data1) ;

function [Returns,ExpR,Sigma] = ExpRetVol(data)
%Returns linear returns on stocks along with expected return and the
%co-variance matrix.
%The expected return and co-variance matrix is calculated from
%an estimation window of 1000 days. The covariance matrix have been
%estimated from an EWMA model to handle volatility clusters.
%The linear returns are assumed to be normally distributed.

Returns = zeros(size(data,2)-1,size(data,1)) ;
for i = 1:size(Returns,2)
    Returns(:,i) = data(i,2:end)./data(i,1:end-1) ;
end

wE = 1000 ; %Estimation window
a = 0.94 ; %Model parameter
N = size(Returns,2) ; %Number of assets
A = repmat(a.^(wE+1:-1:1)',1,N) ;

R = Returns(end:-1:end-wE,:) ;
ExpR = mean(R) ;
MeanR = repmat(ExpR,wE+1,1) ;
Sigma = (1-a)/(a*(1-a^wE))*(A.*(R-MeanR))'*(R-MeanR) ;

end

%Parameter test
DEparameters(ExpR,Sigma) ;
function DEparameters(ExpReturn,Riskfactor)
%This function compares the objective values for different values of
%the mutation factor F and the crossover ratio CR against an
%qualified solution fval.

load('fval') ;
N = length(ExpReturn) ; %Number of assets
n = N ; %Number of populations
eps = 3 ; %Weights boundary
b = linspace(0,1,100) ;
F = linspace(0.1,1,10) ;
CR = linspace(0.1,1,10) ;
Q = zeros(100,6) ;
wS = GenPop(N,n,1) ; %Generate initial population

objfunc = @(b,Return,Risk) (1-b)*Return - b*Risk ; %Function to
%maximize
```

```
for i = 1:length(F)
    for j = 1:length(CR)
        w = zeros(N,length(b)) ;
        fval1 = zeros(1,length(b)) ;
        for k = 1:length(b)

            w0 = wS ;
            c = 0 ; %Condition factor

            %If a member of the population is equal to the average,
            %the solution has converged, thus c = 1
            while c < 1 %Stops when DE converge
                for m = 1:n

                    v0 = w0(:,m) ;
                    vm = DErand1bin(w0,v0,m,N,n,1,F(i),CR(j)) ;

                    risk0 = sqrt(v0'*Riskfactor*v0) ; %Target
                    riskm = sqrt(vm'*Riskfactor*vm) ; %Trail
                    ret0 = ExpReturn*v0 ; %Target's expected return
                    retm = ExpReturn*vm ; %Trail's expected return
                    f1 = objfunc(b(k),ret0,risk0) ;
                    f2 = objfunc(b(k),retm,riskm) ;

                    if f1 > f2 %Compare solutions
                        w0(:,m) = v0 ;
                        fval1(k) = abs(f1-fval(k)) ;
                    else
                        w0(:,m) = vm ;
                        fval1(k) = abs(f2-fval(k)) ;
                    end
                end

                A = round(mean(w0,2),eps) ;
                B = round(w0(:,1),eps) ;
                c = isequal(A,B) ;
            end
            w(:,k) = w0(:,1) ;
        end
        Q(j,i) = mean(fval1) ;
    end
end

end

%Markowitz Mean Volatility model
tic ;
[ExpR_NS,vol_NS] = MeanVolatility(ExpR,Sigma) ;
T1a = toc ;
function [ExpR_NS,vol_NS] = MeanVolatility(ExpR,Sigma)
%Returns the expected return on the portfolio and its volatility
%using quadratic programming.

%This function solves for the Markowitz optimal
%portfolio theory for the cases:
%1. When a risk free asset is accessible.
```

```
%2. When short selling securities is allowed.
%3. When short selling securities is not allowed

%The model uses linear returns, calculated from daily generated data
%and EWMA have been used for modeling the volatilities. The linear
%returns are expected to be normal i.i.d. and have been projected to
%annual expected return and volatility. The investor's initial
%wealth (P0) is assumed to be 1, and the risk %free interest rate
%(R0) is assumed to be 1.

%Analytical solutions
%1. Access to risk free asset
[ExpR_RF,vol_RF] = RiskFree(ExpR,Sigma) ;
function [ExpR_RF,vol_RF] = RiskFree(ExpR,Sigma)
%Determines the expected return and volatility when a risk free
%asset is available. The targets is an extension to what an investor
%can expect.

P0 = 1 ;
R0 = 1 ;
N = length(ExpR) ;
volatility = (ExpR(1)-R0*P0)/sqrt((ExpR'-R0*ones(N,1))'/...
    Sigma*(ExpR'-R0*ones(N,1))) ;

Targetvol = 0.1 ;
TargetR = (ExpR(1)-R0)/volatility*Targetvol+R0 ;
ExpR_RF = linspace(R0,TargetR,200) ;
vol_RF = linspace(0,Targetvol,200) ;

end

%2. With short position
[ExpR_S,vol_S] = ShortPos(ExpR,Sigma) ;
function [ExpR_S,vol_S] = ShortPos(ExpR,Sigma)
%Determines the expected return and volatility when short selling
%securities is allowed.

V0 = 1 ;
N = length(ExpR) ;
A = ones(1,N)/Sigma*ones(N,1) ;
B = ExpR/Sigma*ones(N,1) ;
C = ExpR/Sigma*ExpR' ;
D = A*C-B^2 ;

ExpR_S = linspace(min(ExpR),max(ExpR)+0.01,200) ;
vol_S = sqrt(A/D*ExpR_S.^2-2*B*V0/D*ExpR_S+C/D*V0^2) ;

end

%Numerical solutions
%3. Without short position
[ExpR_NS,vol_NS,w,wMin] = NoShortPos(ExpR,Sigma) ;
function [ExpR_NS,vol_NS,w,wMin] = NoShortPos(ExpR,Sigma)
%Determines the expected return and volatility when short selling
%securities is not allowed.
%Quadprog is used to determine allocations with no short positions
```

```
N = length(ExpR) ;
ExpR_NS = linspace(min(ExpR),max(ExpR),30) ;
vol_NS = zeros(1,30) ;
w = zeros(N,30) ;

for i = 1:length(ExpR_NS)
    Aeq = [ExpR;ones(1,N)] ;
    beq = [ExpR_NS(i);1] ;
    w(:,i) = quadprog(Sigma,[],[],[],Aeq,beq,zeros(1,N)) ;
    vol_NS(i) = sqrt(w(:,i)'*Sigma*w(:,i)) ;
end

wMin = find(vol_NS == min(vol_NS)) ;
w = w(:,wMin:end) ;

end

h = 252 ; %Assumed number of trading days per year
ExpR = 100*(ExpR-1)*h ;
ExpR_RF = 100*(ExpR_RF-1)*h ;
ExpR_S = 100*(ExpR_S-1)*h ;
ExpR_NS = 100*(ExpR_NS-1)*h ;

vol = 100*sqrt(diag(Sigma))*sqrt(h) ;
vol_RF = 100*vol_RF*sqrt(h) ;
vol_S = 100*vol_S*sqrt(h) ;
vol_NS = 100*vol_NS*sqrt(h) ;

%Plots
figure()
stocks = {'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR',...
    'MHK','PKI','RCL','STI','UNM','WAT'} ;
area(vol_NS(wMin:end),100*w')
axis([min(vol_NS(wMin:end)),max(vol_NS(wMin:end)),0,100])
title('Weight distribution')
xlabel('Volatility (%)')
ylabel('Distribution (%)')
legend(stocks)

figure()
hold on
scatter(vol,ExpR)
plot(vol_RF,ExpR_RF,'b',vol_S,ExpR_S,'g',vol_NS,ExpR_NS,'r')
axis([5 30 5 45])
legend('Stocks','Risk-free asset','w short position',...
    'w/o short position','Location','NorthEast')
title('Markowitz Model: Mean Volatility')
xlabel('Volatility (%)')
ylabel('Expected return (%)')
hold off

end

%Differential Evolution
tic ;
```

```
Alt = 1 ; %Mean/Volatility model
DEoptimize(ExpR,Sigma,ExpR_NS,vol_NS,Alt) ;
```

```
function DEoptimize(ExpReturn,Riskfactor,CompReturn,CompRisk,Alt)
%The ExpReturn and Riskfactor parameters are used to maximize
%the objective function thus maximizing the expected return on
%a portfolio and simultaneously minimize the risk associated with
%it by using Differential Evolution algorithm.

%The solution from the Differential Evolution algorithm is then
%compared to CompReturn and CompRisk.

%Alt = 1 : Mean/Volatility model
%The model minimize the Volatility while maximizing the Expected
%return.
%The result is compared against the solution from quadratic
%programming.

%Alt = 2 : Normal linear VaR & CVaR model
%The model minimize VaR and CVaR while maximizing the Expected
%return. The result is compared against the solution from linear
programming.

%Alt = 3 : Stochastic Approximation CVaR model
%The model minimize the CVaR while maximizing the Expected return
%for a
%complex portfolio.
%The result is compared against the solution from stochastic
approximation.

N = length(ExpReturn) ; %Number of assets
n = N ; %Number of populations
eps = 3 ; %Weights boundary
h = 252 ; %Number of trading days per year
nS = length(Riskfactor) ; %Number of scenarios
a = 0.95 ; %VaR confidence level
b = linspace(0,1,50) ; %Distribution factor
F = 0.6 ; %Mutation factor
CR = 0.7 ; %Crossover ratio

ExpR_DE = zeros(1,length(b)) ;
VaR_DE = zeros(1,length(b)) ;
risk_DE = zeros(1,length(b)) ;
w = zeros(N,length(b)) ;

objfunc = @(b,Return,Risk) (1-b)*Return - b*Risk ; %Function to
maximize

for i = 1:length(b)

    w0 = GenPop(N,n,Alt) ; %Generate initial population
```

```
function w = GenPop(N,n,Alt)
%Returns a initial generated population.

%Alt == 3 impose cardinality constraints on complex portfolio.
%No more then 6 assets should be included in the optimal
%portfolio.
```

```
w = rand(N,n) ;
w = w./repmat(sum(w),N,1) ; %Initially generated population
if  Alt == 3 %Constraints
    sorted = -sort(-w) ;
    wMax = sorted(6,:) ;
    for i = 1:n
        idx = find(w(:,i) >= wMax(i)) ;
        jdx = find(w(:,i) < wMax(i)) ;
        w(idx,i) = w(idx,i) + ...
            repmat(sum(w(jdx,i)),length(idx),1)/length(idx) ;
        w(jdx,i) = 0 ;
    end
end

end
```

```
    c = 0 ; %Condition factor
    VaR0 = 0 ;
    VaRm = 0 ;

    %If a member of the population is equal to the average,
    %the solution has converged, thus c = 1
    while c < 1 %Stops when DE converge
        for j = 1:n
            v0 = w0(:,j) ;
            vm = DErand1bin(w0,v0,j,N,n,Alt,F,CR) ;
```

```
function vm = DErand1bin(w,v,j,N,n,Alt,F,CR)
%Returns a mutated portfolio weight using strategy DE/rand/1/bin
%v0 != v1 != v2 != v3

%Alt == 3 impose cardinality constraints on complex portfolio.
%No more then 8 assets should be included in the optimal
%portfolio.

% F = 0.8 ; %Mutation factor
% CR = 0.5 ; %Crossover ratio

selection = 1:n ;
selection(selection == j) = [] ;
m = selection(randi(length(selection))) ;
v1 = w(:,m) ;
selection(selection == m) = [] ;
m = selection(randi(length(selection))) ;
v2 = w(:,m) ;
selection(selection == m) = [] ;
m = selection(randi(length(selection))) ;
v3 = w(:,m) ;
vm = v1 + F*(v2 - v3) ;

rnd = rand(N,1) ;
vm = (rnd < CR).*vm + (rnd > CR).*v ;
vm = abs(vm)/sum(abs(vm)) ; %Impose constraints

if Alt == 3
    sorted = -sort(-vm) ;
```

```
    vMax = sorted(7) ;
    idx = find(vm > vMax) ;
    jdx = find(vm <= vMax) ;
    vm(idx) = vm(idx) +
repmat(sum(vm(jdx)),length(idx),1)/length(idx) ;
    vm(jdx) = 0 ;
end

end
```

```
            if Alt == 1
                risk0 = sqrt(v0'*Riskfactor*v0) ; %Target's risk
                riskm = sqrt(vm'*Riskfactor*vm) ; %Trail's risk
            else
                L0 = sum(repmat(v0',nS,1).*(1-Riskfactor),2) ;
                Lm = sum(repmat(vm',nS,1).*(1-Riskfactor),2) ;

                VaR0 = quantile(L0,a) ;
                VaRm = quantile(Lm,a) ;

                risk0 = mean(max(L0-VaR0,0))/(1-a) + VaR0 ;
                riskm = mean(max(Lm-VaRm,0))/(1-a) + VaRm ;
            end

            ret0 = ExpReturn*v0 ; %Target's expected return
            retm = ExpReturn*vm ; %Trail's expected return
            f1 = objfunc(b(i),ret0,risk0) ;
            f2 = objfunc(b(i),retm,riskm) ;

            if f1 > f2 %Compare solutions
                w0(:,j) = v0 ;
                VaR_DE(i) = VaR0 ;
                risk_DE(i) = risk0 ;
            else
                w0(:,j) = vm ;
                VaR_DE(i) = VaRm ;
                risk_DE(i) = riskm ;
            end
        end

        A = round(mean(w0,2),eps) ;
        B = round(w0(:,1),eps) ;
        c = isequal(A,B) ;
    end

    %Store solutions
    ExpR_DE(i) = ExpReturn*w0(:,1) ;
    w(:,i) = w0(:,1) ;
end

ExpR_DE = 100*(ExpR_DE-1) ;
VaR_DE = 100*VaR_DE ;
risk_DE = 100*risk_DE ;

%Remove redundant solutions from the efficient frontier
range = linspace(min(risk_DE),max(risk_DE),50) ;
```

```
idx = zeros(1,length(range)) ;
for i = 1:length(range)
    con1 = range(i) ;
    con2 = 100 ;
    for j = 1:length(risk_DE)
        if abs(con1-risk_DE(j)) < abs(con1-con2)
            k = find(risk_DE == risk_DE(j)) ;
            idx(i) = k(1) ;
            con2 = risk_DE(j) ;
        end
    end
end

ExpR_DE = ExpR_DE(idx) ;
VaR_DE = VaR_DE(idx) ;
risk_DE = risk_DE(idx) ;
w = w(:,idx) ;

%Plots
if Alt == 1
    pvol = sqrt(diag(Riskfactor)) ;

    figure()
    stocks =
{'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR',...
        'MHK','PKI','RCL','STI','UNM','WAT'} ;
    area(risk_DE*sqrt(h),100*w')
    axis([min(risk_DE*sqrt(h)),max(risk_DE*sqrt(h)),0,100])
    title('Weight distribution (DE)')
    xlabel('Conditional Value at Risk (%)')
    ylabel('Distribution (%)')
    legend(stocks)

    figure()
    hold on
    scatter(100*pvol*sqrt(h),100*(ExpReturn-1)*h)
    plot(CompRisk,CompReturn,'r')
    scatter(risk_DE*sqrt(h),ExpR_DE*h,'MarkerEdgeColor',[0 .1
.7],...
        'MarkerFaceColor',[0 .6 1],'LineWidth',1)
    axis([5 30 5 45])
    legend('Stocks','Quadratic Programming','Differential
Evolution',...
        'Location','NorthWest')
    title('Markowitz Model vs Differential Evolution')
    xlabel('Volatility (%)')
    ylabel('Expected return (%)')
    hold off

elseif Alt == 2
    Loss = 1 - Riskfactor ;
    Vi = quantile(Loss,a) ;
    CVi = mean(max((Loss-repmat(Vi,nS,1)),0))/(1-a) + Vi ;
    CompVaR = CompRisk(1,:) ;
    CompCVaR = CompRisk(2,:) ;
```

```
    figure()
    stocks =
{'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR',...
        'MHK','PKI','RCL','STI','UNM','WAT'} ;
    area(risk_DE*sqrt(h),100*w')
    axis([min(risk_DE*sqrt(h)),max(risk_DE*sqrt(h)),0,100])
    title('Weight distribution (DE)')
    xlabel('Conditional Value at Risk (%)')
    ylabel('Distribution (%)')
    legend(stocks)

    figure()
    hold on
    scatter(100*Vi*sqrt(h),100*(ExpReturn-1)*h)
    plot(CompVaR,CompReturn,'r')
    scatter(VaR_DE*sqrt(h),ExpR_DE*h,'MarkerEdgeColor',[0 .1
.7],...
        'MarkerFaceColor',[0 .6 1],'LineWidth',1)
    axis([0 100 5 40])
    legend('Stocks','Linear Programming','Differential
Evolution',...
        'Location','NorthWest')
    title('Linear Programming vs Differential Evolution')
    xlabel('Value at Risk (%)')
    ylabel('Expected return (%)')
    hold off

    figure()
    hold on
    scatter(100*CVi*sqrt(h),100*(ExpReturn-1)*h)
    plot(CompCVaR,CompReturn,'r')
    scatter(risk_DE*sqrt(h),ExpR_DE*h,'MarkerEdgeColor',[0 .1
.7],...
        'MarkerFaceColor',[0 .6 1],'LineWidth',1)
    axis([0 100 5 40])
    legend('Stocks','Linear Programming','Differential
Evolution',...
        'Location','NorthWest')
    title('Linear Programming vs Differential Evolution')
    xlabel('Conditional Value at Risk (%)')
    ylabel('Expected return (%)')
    hold off

elseif Alt == 3
    Loss = 1 - Riskfactor ;
    Vi = quantile(Loss,a) ;
    CVi = mean(max((Loss-repmat(Vi,nS,1)),0))/(1-a) + Vi ;

    figure()
    stocks =
{'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR',...

'MHK','PKI','RCL','STI','UNM','WAT','SPX','OEX','NDX','RUT',...
        'DJI','SPX_p_u_t','OEX_p_u_t','NDX_p_u_t','RUT_c_a_l_l',...
        'DJI_c_a_l_l'} ;
```

```
    area(risk_DE,100*w')
    axis([min(risk_DE),max(risk_DE),0,100])
    title('Weight distribution (DE)')
    xlabel('Conditional Value at Risk (%)')
    ylabel('Distribution (%)')
    legend(stocks)

    figure()
    hold on
    scatter(100*CVi(1:end-10),100*(ExpReturn(1:end-10)-1))
    scatter(100*CVi(end-9:end-5),100*(ExpReturn(end-9:end-5)-1))
    scatter(100*CVi(end-4:end),100*(ExpReturn(end-4:end)-1))
    scatter(risk_DE,ExpR_DE,'MarkerEdgeColor',[0 .1 .7],...
        'MarkerFaceColor',[0 .6 1],'LineWidth',1)
    axis([0 40 -10 40])

legend('Stocks','Underlying','Options','DE','Location','NorthWest')
    title('Differential Evolution')
    xlabel('Conditional Value at Risk (%)')
    ylabel('Expected return (%)')
    hold off
end

end
T1b = toc ;

%Linear programming for VaR & CVaR
tic ;
[meanR,VaR,CVaR] = LinRisks(ExpR,Returns) ;
function [meanR,VaR,CVaR] = LinRisks(ExpR,returns)
%Returns the expected return on the portfolio, VaR and CVaR
%using linear programming.

%This function is based on Rockafeller and Uryasev work on
%optimization
%of Conditional Value at Risk and Value at Risk.

N = length(ExpR) ; %Number of assets
nS = length(returns) ; %Number of scenarios
V0 = 1 ; %Initial wealth
a = 0.95 ; %Confidence level
h = 252 ; %Number of trading days per year
meanR = linspace(min(ExpR),max(ExpR),50) ;

%Allocation for linprog output
linMap = zeros(nS+N+1,length(meanR)) ;
f = [zeros(N,1);1;1/((1-a)*nS)*ones(nS,1)] ;

%Coeffcients of inequality constraints
w = ones(nS,N) - returns ;
v = -ones(nS,1) ;
y = -eye(nS) ;
A = [w,v,y] ;
b = zeros(nS,1) ;
```

```
%Coefficients of equality constraints
Aeq = [ExpR,0,zeros(1,nS);ones(1,N),0,zeros(1,nS)] ;

%Boundary constraints
lb = zeros(1,nS+N+1) ;
ub = Inf(1,nS+N+1) ;

for i = 1:length(meanR)
    beq = [meanR(i)*V0;V0] ;
    linMap(:,i) = linprog(f,A,b,Aeq,beq,lb,ub) ;
end

VaR = linMap(N+1,:) ;
CVaR = sum(linMap(N+2:end,:))/((1-a)*nS) + linMap(N+1,:) ;

w = linMap(1:length(ExpR),find(CVaR == min(CVaR)):end) ;

meanR = 100*(meanR-1)*h ;
VaR = 100*VaR*sqrt(h) ;
CVaR = 100*CVaR*sqrt(h) ;

%Plots
figure()
stocks =
{'APD','ADM','BAC','CCL','CSX','DOW','ESS','LLY','MAR','MHK',...
    'PKI','RCL','STI','UNM','WAT'} ;
area(CVaR(find(CVaR == min(CVaR)):end),100*w')
axis([min(CVaR),max(CVaR),0,100])
title('Weight distribution (LP)')
xlabel('Conditional Value at Risk (%)')
ylabel('Distribution (%)')
legend(stocks)

figure()
plot(VaR,meanR,'r')
axis([0 100 5 40])
title('Value at Risk')
xlabel('Value at Risk (%)')
ylabel('Expected return (%)')

figure()
plot(CVaR,meanR,'r')
axis([0 100 5 40])
title('Conditional Value at Risk')
xlabel('Conditional Value at Risk (%)')
ylabel('Expected return (%)')

end
T2a = toc ;

%Differential Evolution
tic ;
Alt = 2 ; %Normal linear VaR & CVaR model
DEoptimize(ExpR,Returns,meanR,[VaR;CVaR],Alt) ;
T2b = toc ;
```

```
%Get Data
%data2 = GetData(2) ;
load('data2.mat') ;

%Monte Carlo simulation
[ExpR,Returns] = MCscenarios(data2) ;
function [ExpR,Returns] = MCscenarios(data)
%This function returns simulate Stock returns and Option returns
%using Monte Carlo simulations.
%The options have different time to maturity.

%Get historical values
S = data.assets(data.type == 1,:)' ;
Si = data.assets(data.type == 2,:)' ;
vol = data.assets(data.type == 3,:)' ;

%Convert data into invariants
Inv = [diff(log(S)),diff(log(Si)),diff(vol)] ;

%Moments
wE = 208 ; %Estimation window
Mean = mean(Inv(end:-1:end-wE,:)) ;
Cov = cov(Inv(end:-1:end-wE,:)) ;
Skew = skewness(Inv(end:-1:end-wE,:)) ;
Kurt = kurtosis(Inv(end:-1:end-wE,:)) - 3 ;

N = size(Inv,2) ; %Number of invariants
Scen = 2000 ; %Number of scenarios
Steps = 52 ; %Number of time steps

Inv = MCsimulation(Mean,Cov,Skew,Kurt,N,Scen,Steps) ;
function Inv = MCsimulation(Mean,Cov,Skew,Kurt,N,Scen,Steps)
%Returns the distribution of the invariants using
%Monte Carlo simulations.

lds = sobolset(Steps,'Skip',1e3,'Leap',1e2) ; %Low-discrepancy
%sequence
qMC = net(lds,N*Scen) ;

Z = zeros(Scen,N*Steps) ;
for i = 1:N
    for j = 1:Steps
        Z(:,(i-1)*Steps+j) = norminv(qMC((i-1)*Scen+1:i*Scen,j)) ;
    end
end

C = chol(Cov,'upper') ;

%Cornish-Fisher Approximation
CF = @(z) z + Skew.*(z.^2-1)/6 + Kurt.*z.*(z.^2-3)/24 - ...
    Skew.^2.*z.*(2*z.^2-5)/36 ;

%Monte Carlo simulation
mu = repmat(Mean,Scen,1) ;
MC = zeros(Scen,N*Steps) ;
for i = 1:Steps
```

```
    for j = 1:Scen
        var = CF(Z(j,N*(i-1)+1:N*i))*C ;
        MC(j,N*(i-1)+1:N*i) = mu(j,:) + var ;
    end
end

%Sort invariants from the Monte Carlo simulation
Inv = zeros(N*Scen,Steps) ;
for i = 1:N
    Inv(Scen*(i-1)+1:Scen*i,:) = MC(:,i:N:Steps*N) ;
end

end

%Separate stock invariants and implied volatility invariants
InvS = Inv(1:Scen*(sum(data.type==1)+sum(data.type==2)),:) ;
InvI = zeros(Scen,2*sum(data.type==3)) ;

for i = 1:size(vol,2)
    InvI(:,i) = Inv(Scen*(size(S,2)+size(Si,2))+Scen*(i-1)+1:...
        Scen*(size(S,2)+size(Si,2))+Scen*i,1) ;
    InvI(:,i+size(Si,2)) = Inv(Scen*(size(S,2)+size(Si,2))+Scen*(i-
1)+1:...
        Scen*(size(S,2)+size(Si,2))+Scen*i,2) ;
end
InvI(InvI < 0) = 0 ;

%Map invariants to returns
Sret = zeros(Scen,size(S,2)) ; %Stock returns
Uret = zeros(Scen,size(Si,2)) ; %Underlying returns
SiT = zeros(Scen,size(Si,2)) ; %Stock indices returns
IV = zeros(Scen,size(vol,2)) ; %Estimated implied volatility

for i = 1:length(data.type)
    if data.type(i) == 1
        Ret = exp(cumsum(InvS((i-1)*Scen+1:i*Scen,1:end),2)) ;
        Sret(:,i) = Ret(:,end) ;
    elseif data.type(i) == 2
        k = sum(data.type==1) ;
        Ret = exp(cumsum(InvS((i-1)*Scen+1:i*Scen,1:end),2)) ;
        Uret(:,i-k) = Ret(:,end) ;
        SiT(:,i-k) = Si(end,i-k)*Uret(:,i-k) ;
    elseif data.type(i) == 3
        k = sum(data.type==1) + sum(data.type==2) ;
        IV(:,i-k) = vol(end,i-k) + (InvI(:,i-k)) ;
    end
end

%Option pricing and payoff
r = 0.03 ; %Risk-free rate
T = [1,2,2,1,2] ; %Time to maturity
PC = [0,0,0,1,1] ; %Put = 0 Call = 1
K = Si(end,:).*exp(r*T) ; %Strikes
D0 = zeros(Scen,5) ;
D1 = zeros(Scen,5) ;
for i = 1:length(T)
```

```
    D0(:,i) = BSpricing(Si(end,i),r,vol(end,i),PC(i),K(i),T(i)) ;
function price = BSpricing(S,r,vol,putcall,K,t)
%Returns the option price using Black-Scholes Pricing Model

price = zeros(size(S,1),1) ;

for i = 1:size(S,1)
    d1 = (log(S(i)/K) + (r+0.5*vol(i)^2)*t)/(vol(i)*sqrt(t)) ;
    d2 = d1 - vol(i)*sqrt(t) ;
    if putcall == 1 %Call option
        price(i) = S(i)*normcdf(d1) - K*exp(-r*t)*normcdf(d2) ;
    else %Put option
        price(i) = K*exp(-r*t)*normcdf(-d2) - S(i)*normcdf(-d1) ;
    end
end

end

    if T(i) == 1
        if PC(i) == 0
            D1(:,i) = max(K(i)-SiT(:,i),0) ;
        else
            D1(:,i) = max(SiT(:,i)-K(i),0) ;
        end
    else
        D1(:,i) = BSpricing(SiT(:,i),r,IV(:,i),PC(i),K(i),T(i)/2) ;
    end
end

Dret = D1./D0 ; %option returns

Returns = [Sret,Uret,Dret] ;
ExpR = mean(Returns) ;
Returns = Returns*exp(-r*T(1)) ; %Discount returns for risk
estimation

end

%Differential Evolution
tic ;
Alt = 3 ; %CVaR model
DEoptimize(ExpR,Returns,[],[],Alt) ;
T3 = toc ;

T = [T1a,T1b;T2a,T2b;0,T3]' ;
```

7 References

1. Sandra Paterlini. 2009. Multiobjective optimization using differential evolution for real-world portfolio optimization. ResearchGate. April. 2-4
2. Harry Markowitz. 1952. Portfolio Selection. The Journal of Finance, Vol. 7, No. 1. Mar. 77-83
3. John C. Hull. 2008. Options, futures, and other derivatives. The Black-Scholes-Merton Model. Pearson Prentice Hall. Vol. 7. 6-8, 277-292
4. Andreas Krause. 2003. Winter. Exploring the limitations of Value at Risk: How Good Is It in Practice?. The Journal of Risk Finance. 19-23
5. Attilio Meucci. 2005. Risk and Asset Allocation. Modeling the market. Springer. 101-122
6. Kevin Dowd. 2002. An Introduction to Market Risk Measurement. Measures of Financial Risk. John Wiley and Sons. 13-32
7. Paul Glasserman. 2003. Monte Carlo Methods in Financial Engineering. Foundations. Springer. 1-19
8. Carol Alexander. 2008. Market Risk Analysis Volume IV: Value-at-Risk Models. Historical simulation. John Wiley and Sons. 170-172
9. Carol Alexander. 2008. Market Risk Analysis Volume I: Quantitative Methods in Finance. Basic Calculus for Finance. John Wiley and Sons. 16-26
10. R. Tyrrell Rockafellar, Stanislav Uryasev. 1999. Optimization of Conditional Value-at-Risk. September 5. 1-6
11. Gerard Cornuejols, Reha Tutuncu. 2006. Optimization Methods in Finance. Carnegie Mellon University. January. 47-57, 141-148
12. Kenneth V. Price, Rainer M. Storn, Jouni A. Lampinen. Differential Evolution: A practical Approach to Global Optimization. The Differential Evolution Algorithm. Springer. 37-47