An algorithm for moment-matching scenario generation with application to financial portfolio optimization

K. Ponomareva [a], D. Roman [a], P. Date [a]

[a] *School of Information Systems, Computing and Mathematics, Brunel University, UK.*

Abstract

We present an algorithm for moment-matching scenario generation. This method produces scenarios and corresponding probability weights that match exactly the given mean, the covariance matrix, the average of the marginal skewness and the average of the marginal kurtosis of each individual component of a random vector. Optimisation is not employed in the scenario generation process and thus the method is computationally more advantageous than previous approaches. The algorithm is used for generating scenarios in a mean-CVaR portfolio optimisation model. For the chosen optimization example, it is shown that desirable properties for a scenario generator are satisfied, including in-sample and out-of-sample stability. It is also shown that optimal solutions vary only marginally with increasing number of scenarios in this example; thus, good solutions can apparently be obtained with a relatively small number of scenarios. The proposed method can be used either on its own as a computationally inexpensive scenario generator or as a starting point for non-convex optimisation based scenario generators which aim to match all the third and the fourth order marginal moments (rather than average marginal moments).

1 Introduction and Motivation

One of the traditional approaches for decision-making under uncertainty and risk is stochastic programming. It has wide ranging applications such as financial planning, energy systems management, supply chain logistics, agricultural planning; an extensive discussion of these may be found in [22].

Stochastic programming involves optimisation problems in which (some of) the parameters are not certain, but are described by statistical distributions. In order for the stochastic programs to be numerically solved, the distributions involved are approximated by discrete distributions with a finite number of outcomes (scenarios). The approximation process, called “scenario generation”, is important for the quality of the solution obtained: using “poor” scenarios could only result in obtaining a “poor” approximation of the true optimal solution.

One way to obtain scenarios is by sampling from an assumed distribution (or, simply, from historical data). As an example, future prices of financial assets may be assumed to follow a Geometric Brownian Motion, or a GARCH process ([1]). Sampling methods have clear advantages; however, it may be argued that such distributional assumptions are too strong, or are only applicable to one domain, viz. finance.

Other approaches to scenario generation with specific emphasis on operations research applications include principal component analysis-based simulation [21], stochastic approximation based on transportation metrics ([16], [7]) and hidden Markov models as in ([4], [14] and [19]). A detailed survey of different scenario generation methods appears in [11].

Another class of scenario generation methods are based on matching a small set of statistical properties, e.g. moments (e.g., see [9]). These methods can be broadly divided into two groups.
Under the first approach, the statistical properties of the joint distribution are specified in terms of moments, usually including the covariance matrix. In [8], cubic transformation of univariate, standard normal random variables and Cholesky factorization of covariance matrix are used to produce a multivariate distribution which approximately matches a given set of marginal central moments and the covariance matrix. Similar moment matching approach is employed to generate probability weights and support points using non-convex optimization in [6]. In [20], entropy maximization method is used to generate a discrete approximation to a given continuous distribution.
In the second group, specified (parametric) marginal distributions are sampled independently and the samples are then used along with Cholesky factorization of the covariance matrix to generate the necessary multivariate distribution. An iterative procedure of this type in described in [12], where specified marginal distributions and correlation matrix are used to produce the correlated vectors of random numbers.

The moment-matching scenario generation methods have been successful in the practical applications of stochastic optimization. They have several advantages: they are not specific to a particular field, they do not assume a specific parameterised family of distributions and allow for very different shapes of the marginals. It is only the moments that have to be specified - using historical data, predictions, specialist knowledge or a combination of these.
However, the moment matching approaches described in the above papers

have certain limitations.

Firstly, they use non-convex optimization to generate scenarios which match a specified set of statistical properties, in addition to a needed factorization of the covariance matrix. Given a univariate random variable with known first 12 central moments, the approach used in [8] and [9] finds a cubic polynomial function of this random variable which has the required four central moments. This requires a non-convex optimization in terms of the coefficients of the polynomial. The procedure has to be repeated iteratively for each marginal distribution. Similarly, the algorithm in [12] requires a non-convex optimization over the space of lower triangular matrices.

Secondly, the achieved moments of the generated samples match the target moments only approximately. There are two sources of error in these moment matching methods: one is due to the fact that only local optima are found for the non-convex optimization problem and the other is the inexact starting moments of samples of univariate random variables. Since these procedures employ samples from a known, "simple" univariate distribution, the achieved moments usually depend on the sample moments of univariate random variables used.

In [2], the authors developed a method which uses *convex* optimization to match the given mean vector, covariance matrix exactly, and to minimise the mismatch between the marginal kurtosis across all the variables.

In this paper, we propose a scenario generation method that modifies the algorithm in [2] to match the given mean vector, the covariance matrix, the average marginal skewness as well as average marginal kurtosis (thus catering for assymetric marginals) of each individual component of the random vector, without needing an optimization procedure.

A modified version of the same method in [2] was used in [17] in the context of nonlinear time series filtering. However, to authors' knowledge, a moment-matching sigma point generation algorithm, which generates probability weights as well, has not been employed in the context of financial optimization before. This algorithm may be used as a scenario generator on its own or its scalar version may be adopted to produce an initial guess for the optimization routines proposed by other authors. Being able to match a small set of statistical properties exactly, possibly with a very small set of scenarios, may be preferable to generating a very large number of scenarios to model the entire distribution. This is especially true when the scenarios are to be used in stochastic optimization procedures.
Specifically, the proposed moment-matching scenario generation algorithm has two major advantages over the existing methods:

(1) It is computationally inexpensive - generating scenarios does not involve optimisation. This is in contrast with all the other existing methods for moment matching which cater for moments higher than order two.
(2) It generates scenarios (*i.e.* support points of the multivariate distribution) together with corresponding probability weights, which are unequal in general. This represents a big advantage not only because it eliminates the need to attach user-defined probabilities, but also because the computational difficulty and time for solving the stochastic program can be much decreased - a relatively modest number of *distinct* scenarios is needed to capture the spacial properties of multivariate density surface, such as marginal tail weights and asymmetry.

The significant reduction in computational complexity comes at the cost of matching only the *average* of third marginal moments and the average of fourth marginal moments exactly (instead of matching *all* third and fourth marginal moments approximately, as in [8]). However, the computational simplicity as well as stability of results demonstrated in this paper arguably outweigh this shortcoming. If better moment-matching is needed for higher order marginals, the proposed method can provide an inexpensive initial guess for another moment matching algorithm such as the one proposed in [8].

We test the quality of the proposed scenario generator in a financial portfolio optimisation problem. We generate scenarios for the future asset returns and use them as parameters in an optimisation problem in which the portfolio's CVaR (Conditional Value-at-Risk) is minimised ([18]). We test the in-sample stability as well as out-of-sample stability of our scenario generator, using a variety of options as tuning parameters. The performance of scenario generator is also compared to using the historic returns as scenarios.

The rest of the paper is organised as follows. Section 2 gives details of the algorithm for the scenario generation that matches the mean vector, the covariance matrix and the average marginal third and fourth moments. Section 3 describes the mean-CVaR model for portfolio optimisation. Computational set up, stability of the results and analysis of the optimal solutions are covered in Section 4. Conclusions are drawn in Section 5.

2 Algorithm for scenario generation

We assume that the information about the distribution of an N-variate random vector $r = \left[r^1 \; r^2 \cdots r^N\right]^\top$ is available in terms of the following quantities being known:

μ	target mean vector,
$\mathbf{\Sigma}$	target (positive definite) covariance matrix,
κ_j	marginal third central moment of r^j, $j = 1, 2, \ldots, N$
ζ_j	marginal fourth central moment of r^j, $j = 1, 2, \ldots, N$

Let r_k^j denote the outcome of r^j under scenario k. For notational brevity, denote the average marginal moments as

$$\frac{1}{N}\sum_{j=1}^{N}\kappa_j = \overline{\kappa}, \ \frac{1}{N}\sum_{j=1}^{N}\zeta_j = \overline{\zeta}.$$

For example, r could represent the future returns of N assets, r_k^j being the return of asset j under scenario k. The algorithm is generic and not specific to financial portfolio applications; it is valid for generating scenarios of any random vector r with moment properties specified as above. In the light of the application considered in this paper, we will stick to notation prevalent in the financial optimization literature.

The outline of how the algorithm which follows works is as follows. An even number of scenarios, $2Ns$, proportional to the vector's dimension, are generated such that they match the first and second moments. These scenarios are symmetrically distributed around the expected value such that the variance-covariance matrix is matched. Three additional scenarios are generated in order to match the average marginal skewness and the average marginal kurtosis of each individual component of the random vector r.

The specific steps of the scenario generation algorithm are described next. The motivation behind these steps will be clear from the proof of proposition 1 and the subsequent remarks later in this section.

Scenario generation algorithm

Step 1: Parameters for scenario generation

Scenario generation procedure has the following inputs: mean $\boldsymbol{\mu}$ of the random vector $\boldsymbol{r}$, vector's covariance matrix $\mathbf{\Sigma}$, dimension N and average marginal moments $\overline{\kappa}$ and $\overline{\zeta}$ defined above. User needs to choose an arbitrary positive integer s, an aribtrary non-zero deterministic vector Z, such that $\mathbf{\Sigma} - ZZ^\top > 0$, and a scalar $\rho \in (0, 1)$. Once these are determined, the remaining parameters can be calculated as follows:

$$\begin{aligned}
\alpha &= \frac{1}{2}\phi_1 + \frac{1}{2}\sqrt{4\phi_2 - 3\phi_1^2},\\
\beta &= -\frac{1}{2}\phi_1 + \frac{1}{2}\sqrt{4\phi_2 - 3\phi_1^2},\\
w_1 &= \frac{1}{\alpha(\alpha+\beta)},\\
w_2 &= \frac{1}{\beta(\alpha+\beta)},\\
w_0 &= 1 - \frac{1}{\alpha\beta},\\
\gamma &= 2s^2 \frac{N\bar{\zeta} - \frac{3}{4}\sum_{j=1}^N Z_j^4 \left(\frac{N\bar{\kappa}}{\sum_{j=1}^N Z_j^3}\right)^2}{\sum_{l,k} L_{lk}^4},
\end{aligned}$$

where

$$\phi_1 = \frac{N\bar{\kappa}\sqrt{p_{s+1}}}{\sum_{j=1}^N Z_j^3},\ \phi_2 = p_{s+1}\frac{N\bar{\zeta} - \frac{1}{2s^2}\sum_{l,k} L_{lk}^4 \left(\sum_{i=1}^s \frac{1}{p_i}\right)}{\sum_{j=1}^N Z_j^4}$$

and where L is a positive definite matrix such that $\mathbf{\Sigma} = LL^\top + ZZ^\top$ holds. Constraint γ is required in step 2 to ensure the expression under square root for α and β does not become negative. Given all the inputs and the calculated parameters, probability weights can now be generated.

Step 2: Probability weights

(1) Generate real scalars $p_i \in (0,1)$ for i=1,2,..., s, such that $\sum_{i=1}^s p_i < \frac{1}{2N}$ and $\sum_{i=1}^s \frac{1}{p_i} < \gamma$, $p_{s+1} = 1 - 2N\sum_{i=1}^s p_i$.

Probability weights p_i correspond to $2Ns$ symmetric scenarios and p_{s+1} is associated with the three additional scenarios that match higher order moments. More details are provided in *Proposition 1.*

(2) Given $p_1, p_2 ... p_{s+1}$ and w_0, w_1, w_2 from step 1, vector P of the $2Ns + 3$ probability weights is formed as follows:

$$P = \{p_1, p_2, \cdots, p_s, p_1, p_2, \cdots, p_s, \cdots, p_{s+1}w_0, p_{s+1}w_1, p_{s+1}w_2\}.$$

(3) Note that Z can also be generated randomly, provided it satisfies the above mentioned constraint. In the numerical experiments, we used $Z_j = \rho\sqrt{\Sigma_{jj}}$, where $\rho \in (0,1)$ is a scalar. However, this is only one of many possible ways to generate Z vector.

(4) Also note that p_i can be generated using a random number generator, and s is an arbitrary positive integer. Hence the number of scenarios generated is independent of the dimension of the random vector r. Note also that the distribution of p_i is of no consequence since the moment-matching is independent of how p_i are generated.

(5) One way to generate p_i that satisfy the constraints is to choose:

$$p_i = \frac{s}{N\gamma} + \left(\frac{1}{2Ns} - \frac{s}{N\gamma}\right) U,$$

where $U \in (0,1)$ is a uniformly distributed random variable. Another way is to generate p_i from a gamma distribution, for example $p_i = -\ln(U)$ with $U \in (\frac{1}{e}, 1)$. Note it is necessary to normalise the weights generated using this method, for example

$$p'_i = \frac{p_i}{Ns(2\sum p_i + max(p_i))},$$

to ensure that $\sum_{i=1}^{s} p'_i < \frac{1}{2N}$ constraint is satisfied. There are many more different ways to generate p_i and Z while matching the moments, and a natural question to ask whether the results are sensitive to the choice of these parameters. We will investigate empirically the impact of the choice of p_i on the generated moment-matching scenarios later in section 4. In particular, in section 4.2 the (admittedly limited) evidence provided by our numerical experiments indicates that this might not be the case.

Step 3: Support points

Generate $S = 2Ns + 3$ support points r_k of the multivariate distribution of returns, with their probability weights given by P_k from step 2 and $k = 1, ..., 2Ns + 3$, as follows:

$$\mathbb{P}\left(r_{(i-1)N+c} = \mu + \frac{1}{\sqrt{2sp_i}} L_c\right) = p_i = P_{i+s(c-1)}, \tag{1}$$

$$\mathbb{P}\left(r_{(s+i-1)N+c} = \mu - \frac{1}{\sqrt{2sp_i}} L_c\right) = p_i = P_{i+s(N+c-1)}, \tag{2}$$

$$\mathbb{P}\left(r_{2Ns+1} = \mu\right) = p_{s+1} w_0 = P_{2Ns+1}, \tag{3}$$

$$\mathbb{P}\left(r_{2Ns+2} = \mu + \frac{\alpha}{\sqrt{p_{s+1}}} Z\right) = p_{s+1} w_1 = P_{2Ns+2}, \tag{4}$$

$$\mathbb{P}\left(r_{2Ns+3} = \mu - \frac{\beta}{\sqrt{p_{s+1}}} Z\right) = p_{s+1} w_2 = P_{2Ns+3}, \tag{5}$$

where $i = 1, 2, \ldots, s$, $c = 1, 2, \ldots, N$ and L_c is c^{th} column of L.

Note that there is no optimization involved in generation of r_k.
The ouput of the scenario generation algorithm are the $2Ns+3$ support points

$$\{r_1, r_2, \cdots, r_{2Ns+1}, r_{2Ns+2}, r_{2Ns+3}\}$$

and the corresponding probability weights

$$P = \{p_1, p_2, \cdots, p_s, p_1, p_2, \cdots, p_s, \cdots, p_{s+1}w_0, p_{s+1}w_1, p_{s+1}w_2\}.$$

The proposition below summarises the properties of the discrete distribution of the random variable r, based on the output support points and probability weights:
Proposition 1 For r_k, P_k defined in (1)-(5), the following moment properties hold:

$$2N\sum_{i=1}^{s} p_i + p_{s+1}\left(\sum_{i=0}^{2} w_i\right) = 1, \tag{6}$$

$$\mathbb{E}\left(r\right) = \mu, \tag{7}$$

$$\mathbb{E}\left((r-\mu)(r-\mu)^\top\right) = \mathbf{\Sigma}, \tag{8}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} \mathbb{E}\left(r_k^j - \mu_j\right)^3 = N\overline{\kappa}, \tag{9}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} \mathbb{E}\left(r_k^j - \mu_j\right)^4 = N\overline{\zeta}, \tag{10}$$

where the expected values are taken with respect to the probability measure defined by P_k.

Proof: The first two equations are straightforward. The third follows since

$$\begin{aligned}\mathbb{E}\left((r-\mu)(r-\mu)^\top\right) &= 2\sum_{i=1}^{s} p_i \frac{1}{2sp_i}\sum_{j=1}^{N} L_j L_j^\top + p_{s+1}\left(w_1\frac{\alpha^2}{p_{s+1}} + w_2\frac{\beta^2}{p_{s+1}}\right) ZZ^\top \\ &= LL^\top + ZZ^\top = \mathbf{\Sigma},\end{aligned} \tag{11}$$

by definition of L_j and w_i. For the two last equations, substituting expression for r_k and P_k from step 2 into left-hand side of equations (9)-(10) we get:

$$\sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} \mathbb{E}\left(r_k^j - \mu_j\right)^3 = \sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} P_k(r_k^j - \mu_j)^3 = \frac{(\alpha-\beta)\sum_{j=1}^{N} Z_j^3}{\sqrt{p_{s+1}}},$$

$$\begin{aligned}\sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} \mathbb{E}\left(r_k^j - \mu_j\right)^4 = \sum_{j=1}^{N}\sum_{k=1}^{2Ns+3} P_k(r_k^j - \mu_j)^4 &= \frac{(\alpha^2-\alpha\beta+\beta^2)\sum_{j=1}^{N} Z_j^4}{p_{s+1}} \\ &+ \frac{1}{2s^2}\sum_{l,k} L_{lk}^4\left(\sum_{i=1}^{s}\frac{1}{p_i}\right),\end{aligned}$$

where P_k is the k^{th} element of the probability weights vector P. Using definitions of α and β from step 2 provides the required result:

$$\frac{(\alpha-\beta)\sum_{j=1}^{N} Z_j^3}{\sqrt{p_{s+1}}} = \phi_1 \frac{\sum_{j=1}^{N} Z_j^3}{\sqrt{p_{s+1}}} = N\overline{\kappa},$$

$$\frac{(\alpha^2-\alpha\beta+\beta^2)\sum_{j=1}^{N} Z_j^4}{p_{s+1}} + \frac{1}{2s^2}\sum_{l,k} L_{lk}^4 \left(\sum_{i=1}^{s}\frac{1}{p_i}\right)$$

$$= \phi_2 \frac{\sum_{j=1}^{N} Z_j^4}{p_{s+1}(N\overline{\zeta} - \frac{1}{2s^2}\sum_{l,k} L_{lk}^4 \left(\sum_{i=1}^{s}\frac{1}{p_i}\right))} = N\overline{\zeta},$$

which completes the proof.

To summarise, the above algorithm generates moment matching scenarios which match a given mean vector, covariance matrix, average marginal third central moment and average marginal fourth central moment. Average computational time for this algorithm ranges from 0.034 seconds for 123 scenarios to 0.088 seconds for 5043 scenarios. All the numerical experiments were performed using Matlab 7.2 on a desktop with a dual core Pentium IV processor, 2.40GHz and 3.24Gb RAM.

Some remarks on the above proposition are in order.

- Equation (11) makes the motivation behind the chosen structure of support points clear; due to $\frac{1}{2sp_i}$ term in the denominator of the support points defined by (1)-(5), p_i cancels out while constructing a covariance matrix and we end up with known mean vector and covariance matrix *irrespective of* choice of p_i (provided they form a valid probability measure). This freedom of choosing p_i can then be partially exploited to match more moment properties. Matching two more scalar moment properties (in addition to the mean and the covariance) is feasible in closed-form, as we end up with only having to solve a quadratic equation. Technically, it is possible to match higher order moments, although it might require optimization and it might hence defeat the whole purpose of this exercise, which is to provide a inexpensive moment-matching scenario generation method.
- The number of scenarios generated is $S = 2Ns + 3$, where s is an arbitrary integer and N is the dimension of r. The distribution of these scenarios is defined by the set of support points and their corresponding probability weights, as given by equations (1)-(5). We don't have a function closed-form expression for this distribution. However, the purpose is to match the given moment properties, which is achieved (as shown by proposition 1).
- The use of a similar algorithm in unscented Kalman filters reported elsewhere (e.g. in [17]) is unrelated to the application proposed here and is unrelated in general to applications within operations research.

- Technically, the choices made in generating p_i and Z affects the generated scenarios. In the admittedly limited number of numerical experiments reported here, it was found that the effect was marginal. In case a severe variation is observed in the generated scenarios or in decisions taken based on those scenarios, one can always use one of the choices of p_i and Z as an initial feasible solution to a more sophisticated scenario generation algorithm which uses optimization.

In section 4, we evaluate the performance of this scenario generator for a real financial portfolio selection problem.

3 The mean-CVaR model for portfolio optimisation

In our numerical experiments, we generate scenarios for a one-period portfolio optimisation problem: given a set of N assets in which one may invest, how to divide *now* an amount of money between these assets, such that, *after a specified period of time* T, we obtain a return on investment as high as possible?

Let us denote by $r = \left[r^1\ r^2 \cdots r^N\right]^\top$ the random vector representing the future returns of the assets.

Let x_j be the proportion of capital invested in asset j and let $x = (x_1, \dots, x_N)$ represent the portfolio resulting from this choice. This portfolio's return is the random variable: $r_x = x_1 r^1 + \cdots + x_N r^N$; its distribution depends on the choice $x = (x_1, \dots, x_N)$ and on the distribution of $r^1, \dots r^N$.

With the popular "mean - risk" theory, a portfolio's return distribution is described by two statistics: the expected value and a "risk" value (desired to be kept low). The portfolio chosen for implementation should be "efficient", meaning, it should have the lowest risk value for a given expected return. An efficient portfolio is found by solving an optimisation problem in which, for example, the risk of the portfolio is minimised, while a constraint on the expected return is imposed.

In most cases, there are no closed form solutions for these optimisation problems. They have to be solved numerically, by approximating the distributions of the future asset returns with discrete ones with finite number of realisations; that is, by generating scenarios for the future asset returns.

Traditionally, risk is measured by variance ([13]). In the mean-variance optimisation problem, we do not need scenarios for the future asset returns, but only their expected values and the covariance matrix. However, it has been pointed out that risk may be better quantified and several alternatives risk measures have been proposed. More recently, portfolio optimisation problems include more sophisticated risk measures, most notably those concerned with left tails of distributions. Risk measures in this category include Conditional Value-at-Risk (CVaR), which has good theoretical and computational properties and has gained wide acceptance among academics and practitioners ([18], [15]).

In our numerical experiments, we generate scenarios for future asset returns in a mean-CVaR optimisation model, which, unlike the mean-variance model, requires the full set of scenario returns.

The definition of CVaR and the model formulation, are presented below.

Let r_x be a random variable representing the return of a portfolio x over a given holding period and $A \in (0,1)$ a percentage which represents a sample of "worst cases" for the outcomes of r_x (usually, $A = 1\%$, $A = 5\%$ or $A = 10\%$).

The definition of CVaR at the specified level A is the mathematical transcription of the concept "average of losses in the worst A of cases". More formally, the CVaR at level A of r_x is defined as minus the mean of the A-tail distribution of r_x, where the A-tail distribution is obtained by taking the lower A part of the distribution of r_x (corresponding to extreme unfavourable outcomes) and rescaling it to span [0,1]. The A-tail distribution of r_x considers only losses above Value-at-Risk; for a detailed definition of CVaR, see [18].

An important result is that CVaR can be computed and optimised by solving convex optimisation problems. In ([18]), an auxiliary function is used, $F : X \times \mathbb{R} \to \mathbb{R}$, where X is the set of feasible portfolios.

$$F_A(x, v) = \frac{1}{A}\mathbb{E}[-r_x + v]^+ - v,$$

where $[u]^+ = u$ if $u \geq 0$ and 0 otherwise. It was proved that minimising CVaR over X can be done by minimising F_A over $X \times \mathbb{R}$.

When the random asset returns are represented as discrete random variables (via scenarios), the CVaR optimisation problem can be formulated as a linear program (LP) as below:

$$\min \quad v + \frac{1}{A}\sum_{k=1}^{S} P_k \cdot y_k \tag{M-CVaR}$$

$$\text{Subject to:}$$

$$v - \sum_{j=1}^{N} r_k^j x_j \le y_k, \quad \forall k \in \{1 \dots S\}$$

$$y_k \ge 0, \quad \forall k \in \{1 \dots S\}$$

$$\sum_{j=1}^{N} \mu_j x_j \ge d; \quad x \in X,$$

where the parameters of the model are:

- S = the number of scenarios, (for example, with our proposed algorithm, $S = 2Ns + 3$),
- N = the number of assets,
- P_k = the probability of scenario k occuring, $k = 1 \dots S$,
- r_k^j = return of asset j under scenario k, $k = 1 \dots S$, $j = 1 \dots N$,
- μ_j = the expected return of asset j, $j = 1 \dots N$,
- d = the desired expected return of the portfolio (investor-specified).

Here, P_k and r_k^j can be obtained from scenario generation or sampling of historical data, μ_j are estimated prior to optimisation (possibly from historical data). The parameter d is decided by the investor. By not imposing the last constraint on portfolio's expected return, we obtain the (absolute) minimum CVaR portfolio.

The decision variables of the model are:

- x_j = the fraction of portfolio wealth invested in asset j, $j = 1 \dots N$,
- v = an A-quantile of the portfolio return distribution,
- y_k = the magnitude of the negative deviations of the portfolio return from the A-quantile, for every scenario $k \in \{1 \dots S\}$ (they are 0 if the portfolio return $\sum_{j=1}^{N} r_k^j x_j$ is higher than the A-quantile).

4 Numerical experiments: A case study for an investment problem

4.1 Scope and computational set-up

We investigate the behaviour of the proposed scenario generator, as used in conjunction with a mean-CVaR optimisation problem (M-CVaR). We are interested in:

(a) the stability of the scenario generator, both in-sample and out-of-sample;
(b) how the number of scenarios considered affects the optimal solution and the optimum;
(c) quality of the solution as compared to the solution obtained using historical data as scenarios;
(d) testing the effect of different parameter selection on the scenario generation algorithm.

We use a dataset drawn from FTSE 100, the share index of the stocks of the 100 companies listed on the London Stock Exchange with the highest market capitalisation. The dataset has N=20 stocks and prices monitored weekly over the period of over 14 years from July 1997 until November 2011 (747 time periods of historical data). These 20 stocks were chosen in such a way that their combined market capitalisation was roughly over 60% of FTSE 100. Please see ([10]) for more information about the FTSE 100 index.

We compute the corresponding historical returns and calculate the moments for these series; they are displayed in Tables A1 - A3 in Appendix.

We generate scenarios for the future weekly returns of these stocks using the proposed moment-matching method, with the first four moments as above. We consider various number of scenarios: $S = 43$, $S = 123$, $S = 243$, $S = 363$, $S = 603$, $S = 723$, $S = 1083$ and $S = 5043$. Since $S = 2Ns + 3 = 40s + 3$, this corresponds to generating $s = 1, 3, 6, 9, 15, 18, 27$ and 126 distinct probability weights respectively. A wide set of values is chosen in order to test the impact of choosing small or large number of scenarios on the achieved optimum.

We solve (M-CVaR) using the generated P_k and r_k^j, $k = 1 \dots S$, d is fixed to 0.3%. We consider CVaR at confidence level $A = 10\%$.

It is common practice to use for comparison the “benchmark” scenario generator with the same optimisation model. In this study, we are using the historical returns as “benchmark” scenarios. Thus, we also solve (M-CVaR) using P_k and r_k^j as given by historical data, i.e. $S = 747$, $P_k = 1/747$, r_k^j =historical return of stock j at time period k, $k = 1 \dots S$, $j = 1 \dots N$.

The optimisation problems are formulated in AMPL ([5]) and solved with the FortMP solver ([3]).

4.2 Stability of the scenario generator

Stability is a basic and very important requirement. Since any scenario generator has an element of randomness, the final outcome depends on the values drawn from the corresponding distributions. Stability guarantees that the optimum and the optimal solution of the optimisation problem of interest does not vary (but, possibly, only to a small extent) with the specific scenario set chosen.

A scenario generator is said to manifest *in-sample stability* if, when generating several scenario sets of the same size and solving the optimisation problem on each of these scenario sets, the optimums are similar (here, the optimum is evaluated on the same scenarios used for obtaining the solution, see for example [11]).

A scenario generator is said to manifest *out-of-sample stability*, if, when generating several scenario sets of the same size and solving the optimisation model on each of these scenario sets, the optimal solutions obtained yield similar "true" objective function values (i.e., the solutions obtained with in-sample scenarios and then they are evaluated using another scenario set, standing for the "true" distribution of the random variables involved).

In [11] it is emphasised that, while it is straightforward to test in-sample stability (we only solve the scenario-based optimisation problems), it is difficult to test the out-of-sample stability - that would mean knowing the "true" distribution of the random vector involved. What is usually done in practice is to use a "benchmark" scenario tree: a large scenario set obtained exogenously, that is known to be stable; this scenario set will stand for the "true" distribution.

In-sample stability does not imply the out-of sample one or vice versa. It is possible to have in-sample instability (of the objectives) but not stability of the solutions - in this case, it is likely to have out-of-sample stability, since, in this case, all the solutions are tested on the same scenario set, representing the "true" distribution.

We test the in-sample stability of our scenario generator in the following way. For each number of scenarios S considered in our example, we generate 20 sets of scenarios and use them as input in the mean-CVaR optimisation problem (M-CVaR).

As seen in section 2, the choice of p_i and the choice of Z are free parameters in our algorithm. In our numerical experiments, we consider three different ways of generating moment matching scenarios:

(1) case 1: p_i are generated by sampling from a uniform distribution method and $Z_j = 0.7\sqrt{\Sigma_{jj}}$,
(2) case 2: p_i are generated by sampling from a uniform distribution method and $Z_j = 0.45\sqrt{\Sigma_{jj}}$,
(3) case 3: p_i are generated by sampling from a gamma distribution method and $Z_j = 0.45\sqrt{\Sigma_{jj}}$.

For each scenario size, we obtain a set of 20 optimal objective values; their statistics are displayed in Tables 4.1 -4.3 below.

Table 4.1 In-sample stability: Statistics of the sets of optimums for various scenario sizes for case 1 (uniform distribution for p_i, $Z_j = 0.7\sqrt{\Sigma_{jj}}$)

S	Mean	StDev	Min	Max	Range
43	0.032613	0.000431	0.032142	0.033444	0.001302
123	0.032736	0.000336	0.032209	0.033004	0.000795
363	0.033326	0.000300	0.033053	0.033699	0.000646
723	0.036456	0.000276	0.036120	0.036751	0.000630
1083	0.036835	0.000234	0.036581	0.037163	0.000581
5043	0.037465	0.000128	0.037277	0.037702	0.000425

Table 4.2 In-sample stability: Statistics of the sets of optimums for various scenario sizes for case 2 (uniform distribution for p_i, $Z_j = 0.45\sqrt{\Sigma_{jj}}$)

S	Mean	StDev	Min	Max	Range
43	0.030776	0.000417	0.030115	0.031210	0.001095
123	0.031461	0.000377	0.031076	0.031790	0.000714
363	0.032186	0.000361	0.031553	0.032205	0.000652
723	0.033283	0.000297	0.032620	0.033140	0.000519
1083	0.035876	0.000239	0.035581	0.036033	0.000452
5043	0.037767	0.000184	0.037477	0.037844	0.000367

Table 4.3 In-sample stability: Statistics of the sets of optimums for various scenario sizes for case 3 (gamma distribution for p_i, $Z_j = 0.45\sqrt{\Sigma_{jj}}$)

S	Mean	StDev	Min	Max	Range
43	0.037160	0.000411	0.036549	0.037361	0.000812
123	0.037259	0.000391	0.036875	0.037604	0.000729
363	0.037863	0.000362	0.037531	0.038204	0.000673
723	0.038202	0.000262	0.038755	0.039263	0.000508
1083	0.038918	0.000237	0.038857	0.039293	0.000436
5043	0.039146	0.000166	0.038915	0.039244	0.000329

Tables 4.1 - 4.3 show that, when p_i's are generated from the same distribution and the vector Z is fixed, the in-sample stability is very good, even with scenario sets of relatively small sizes. As expected, stability is increased with increasing number of scenarios. For example, in case 1 (p_i's generated using uniform distribution, $Z_j = 0.7\sqrt{\Sigma_{jj}}$), when considering 123 scenarios, the optimal CVaRs range between 3.22% and 3.3%, while for 5043 scenarios, the optimum values range between 3.73% and 3.77% (table 4.1). Also expected is the increase in optimal CVaR values as the number of scenarios increases. This is due to the fact that CVaR only takes into account a specified number

of worst ourcomes (left tail) and ignores the rest of the distribution; in this case, CVaR is the negative of the expected loss under the worst 10% scenarios, expressed as a percentage of investment value.

Generating scenarios with different values of Z makes the procedure somewhat less stable; however, for large scenario sets, the differences between optimal CVaRs are marginal. For example, for 5,043 scenarios and p_i's generated from the uniform distribution, the optimal CVaRs range between 3.73% and 3.77% when $Z_j = 0.7\sqrt{\Sigma_{jj}}$ and between 3.75% and 3.78% when $Z_j = 0.45\sqrt{\Sigma_{jj}}$. Even when p_i's are generated from the gamma distribution, the optimal CVaRs are between 3.89% and 3.92% which, for the practical purposes of the problem considered, represents only a minor difference: it tells the decision maker that, irrespective of the choice of parameters p_i' and Z, the expected loss under the worst 10% cases is between 3.7% and 3.9% of capital invested.

Based on our numerical experiments we have found the following range $\rho \in [0.4, 0.8]$ to provide stable results for our specific data set. However, differences upon using different choices for generating the free parameters should be tested with what seems more reasonable for the specific problem.

For testing the out-of-sample stability, we use the historical data as the benchmark scenario set. Each of the optimal solutions obtained before are evaluated on the historical data, i.e. we use the portfolio weights previously obtained (using our scenario generator) and compute the corresponding CVaRs using historical scenarios. Thus, we obtain 20 "true" optimums ("historical CVaRs") for each scenario size. Their statistics are displayed in Tables 4.4-4.6.

Table 4.4. Out-of-sample stability: Statistics of the sets of "true" optimum CVaRs for case 1

S	Mean	StDev	Min	Max	Range
43	0.041506	0.000702	0.040648	0.042075	0.001427
123	0.040995	0.000613	0.040315	0.041675	0.001360
363	0.040461	0.000554	0.039921	0.041088	0.001167
723	0.040277	0.000506	0.039700	0.040855	0.001155
1083	0.040221	0.000342	0.039397	0.040514	0.001117
5043	0.039621	0.000283	0.039211	0.040264	0.001053

Table 4.5. Out-of-sample stability: Statistics of the sets of "true" optimum CVaRs for case 2

S	Mean	StDev	Min	Max	Range
43	0.042010	0.000608	0.041813	0.043311	0.001498
123	0.041894	0.000599	0.041315	0.042639	0.001324
363	0.041680	0.000556	0.041192	0.042447	0.001255
723	0.041357	0.000346	0.040700	0.041872	0.001172
1083	0.040495	0.000275	0.040397	0.041525	0.001128
5043	0.040395	0.000226	0.040211	0.041275	0.001064

Table 4.6. Out-of-sample stability: Statistics of the sets of "true" optimum CVaRs for case 3

S	Mean	StDev	Min	Max	Range
43	0.043075	0.000733	0.042312	0.043890	0.001578
123	0.042337	0.000611	0.041315	0.042686	0.001371
363	0.041135	0.000545	0.040921	0.042185	0.001264
723	0.040311	0.000458	0.039860	0.041040	0.001180
1083	0.040297	0.000322	0.039820	0.040991	0.001171
5043	0.040188	0.000192	0.039526	0.040618	0.001092

The "true" optimums (i.e. the CVaRs of the portfolios obtained with our scenario generator, evaluated on historical data) are very close in value, irrespective of the way p_i and Z are generated for running the optimisation: they range between 3.9% and 4.1% (tables 4.4 - 4.6). We notice that the best out-of-sample stability and also the closest match between in-sample and out-of sample CVaRs are obtained when p_i are generated from the gamma distribution: with 5,043 scenarios, the in-sample CVaRs range around 3.9% and 3.9% (table 4.3) while the out-of-sample CVaRs range around 4%.

The out-of-sample stability is also obvious, even for small in-sample scenario sets, particularly when using a consistent method for generating p_i and Z. For example, in case 1 (p_i's generated using uniform distribution, $Z_j = 0.7\sqrt{\Sigma_{jj}}$), with 123 in-sample scenarios, the "true" CVaRs range between 4.03% and 4.16%. As expected, we get better solutions with increasing number of (in-sample) scenarios: not only even more stable, but also resulting in a better (i.e. smaller) out-of-sample CVaRs. Again we notice that, when increasing the number of (in-sample) scenarios, the difference between the out-of-sample CVaRs is small: an average "true" CVaR of 4.01% is obtained for 123 in-sample scenarios; the average "true" CVaR for 5043 in-sample scenarios is 3.96% (table 4.4).
Similar results are obtained for the other two cases (tables 4.5 and 4.6). We notice that, in all three cases, for 5043 in-sample scenarios, the average "true" CVaR is in the range of 4%. The specific method used for generating the free parameters seems again to result in (only) marginal differences.

Moreover, with increasing number of in-sample scenarios, the optimum (in-sample) CVaRs get progressively close in value to the out-of-sample CVaRs (evaluated on historical data). In the case of 5043 in-sample scenarios, the in-sample CVaRs are in the range of 3.8% when p_i are generated using the uniform distribution (cases 1 and 2; tables 4.1 and 4.2) and in the range of 3.9% when p_i are generated using the gamma distribution (case 3; table 4.3) ; the out-of-sample CVaRs are in the range of 4%, in all three cases (tables 4.4, 4.5, 4.6).

4.3 Optimal solutions

We notice that the proposed scenario generator is not only stable in-sample and out-of-sample (with respect to the optimum CVaR), but also manifests "stability of optimal solutions" for the chosen problem. With a specified number of in-sample scenarios, the optimal portfolio weights obtained are very similar under different runs. This is valid even in the case of smallest scenario sets (123 scenarios). Table A4 in the Appendix displays the optimal portfolio weights obtained in each of the 20 runs, when the scenario size is 5043 and p_i's are generated using the uniform distribution, $Z_j = 0.7\sqrt{\Sigma_{jj}}$ (case 1). Similar results are obtained for the other cases. The full set of results would considerably lengthen the paper without adding much value; they can be obtained upon request from the authors.

Moreover, with increasing number of scenarios, the optimal portfolio weights change only *marginally*, i.e. the portfolio weights obtained by using 123 scenarios are similar to the portfolio weights obtained by using 5043 scenarios (please see table 4.7).

This supports the idea that stable solutions may be obtained with relatively small number of scenarios and increasing the number of scenarios leads only to a marginal change in the optimal solution.

A natural question to ask is: how do these solutions compare with the portfolio solution obtained by using the historical data as *in-sample* scenarios?

Table 4.7 displays a summary of optimal solutions obtained by using in-sample: (a)the proposed scenario generator with p_i generated using the uniform distribution, $Z_j = 0.7\sqrt{\Sigma_{jj}}$ (case 1), various scenario set sizes and (b) historical data. For a given scenario set size, the weights displayed are obtained by averaging the (very similar) optimal weights obtained for the 20 different runs. "HD" signifies the portfolio weights obtained using historical data as in-sample scenarios (747 scenarios equally probable scenarios). Note that "HD" solutions are included for comparison purposes only and not as part of stability analysis of our scenario generation methodology.
The "HD" optimal portfolio is different from the solutions obtained via the

proposed scenario generator - we notice however that assets with strictly positive weight are roughly the same in both cases; also, there are roughly the same assets with highest weights (e.g. assets 6, 8, 12, 15, 16).

Table 4.7. Optimal portfolio weights obtained using (a) the proposed method with various number of scenarios and (b) historical data(HD) as scenarios

S	CVaR(in)	CVaR(out)	x_2	x_6	x_7	x_8	x_9	x_{10}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
HD	0.038712		13.83%	24.50%	0.00%	16.37%	3.55%	8.33%	5.69%	6.50%	0.60%	9.35%	9.64%	1.64%	0.00%	0.00%	0.00%
43	0.032554	0.041192	4.15%	38.89%	0.88%	7.94%	5.63%	4.52%	5.27%	5.38%	2.27%	7.11%	5.08%	2.73%	0.74%	5.08%	4.34%
123	0.032736	0.040995	4.20%	37.92%	0.92%	7.99%	5.66%	4.62%	5.37%	5.40%	2.26%	7.09%	5.38%	2.74%	0.84%	5.11%	4.50%
243	0.032828	0.040559	4.28%	36.24%	0.51%	8.58%	5.48%	4.76%	5.18%	5.71%	2.11%	8.95%	5.63%	2.61%	0.79%	4.85%	4.30%
363	0.033326	0.040461	3.70%	34.92%	0.36%	9.21%	5.82%	3.97%	5.23%	6.33%	2.40%	9.75%	4.97%	2.98%	0.86%	4.93%	4.57%
603	0.033704	0.040449	4.38%	35.27%	1.18%	8.90%	2.88%	6.00%	5.90%	5.24%	3.39%	8.12%	6.72%	1.64%	0.31%	5.34%	4.72%
723	0.036456	0.040277	4.41%	35.62%	0.61%	9.66%	5.09%	4.72%	5.32%	6.05%	2.62%	7.24%	6.18%	2.92%	0.00%	5.07%	4.49%
1083	0.036835	0.040221	4.62%	35.87%	0.65%	9.45%	5.05%	5.27%	5.11%	5.67%	2.39%	7.77%	6.16%	2.72%	0.00%	5.02%	4.24%
5043	0.037465	0.039621	5.00%	32.85%	0.87%	9.46%	4.69%	6.02%	5.20%	5.87%	2.86%	7.84%	7.01%	3.17%	0.00%	4.88%	4.28%

Another encouraging result is that the "true" CVaR of the portfolio obtained using historical data in-sample (that is, the optimisation *and* evaluation of CVaR is done on historical data) is 3.87%.

Thus, as measured on the benchmark scenario generator, a solution obtained using our scenario generator in-sample gives a "true" CVaR of 3.96%, while the solution obtained using historical data in-sample gives a value only 0.09% lower.

5 Conclusions

We proposed a moment-matching scenario generation method. Given a distribution of a random vector that is partially specified in terms of first four marginal moments and covariance matrix, this method generates scenarios and corresponding probabilities that match exactly the first two moments, the covariance matrix, the average marginal skewness and the average marginal kurtosis.

The method presents several advantages over the existing approaches. First, it is computationally cheap; there is no optimisation involved in generating scenarios. Secondly, due to the (unequal) generated probabilities of the scenarios, this method may perform well even with a relatively small number of scenarios. In contrast, methods that would assume by default equal probabilities would need a larger number of scenarios.

These assertions are supported by the numerical results. We tested the quality of the proposed scenario generator in a mean-CVaR portfolio optimisation model. Several observations were made. First, the method appears to be remarkably stable, both in-sample and out-of-sample. For the problem solved, it also demonstrates stability of optimal solutions, i.e. not only the optimums are similar, but also the optimal portfolio weights, representing the solution to implement. Secondly, the optimal solutions vary only marginally with increasing number of scenarios. That means, by using only a small number of scenarios, we may obtain a reasonably good solution. Since the method allows for some freedom for generating the probability weights and scenarios which match the given moments, we tested the algorithm for three different ways of generating the probability weights and scenarios. The results obtained by varying the underlying distribution of p_i and/or the value of the free parameter matrix Z appear to be qualitatively similar.

In the numerical case presented here, direct comparison with solutions obtained via using historical data could be made. We comment that the proposed scenario generation method is applicable to cases where there is no (or not enough) historical data available, but only expert opinion on the statistical properties involved.
This method may also present a big advantage when used with a computationally difficult optimisation model, requiring only a limited number of scenarios; stable and good quality solutions may be obtained with a relatively small number of scenarios. Intuitively, this may be attributed to the fact that unequal and random probability weights are generated along with the support points.

The numerical results are very encouraging. Note that the single example chosen focuses on the tail behaviour of the distribution and is hence challenging from a scenario generation point of view. This scenario generation method may work well for multi-stage stochastic optimisation, where generally only a limited number of scenarios can be considered and there are scale-up issues; this is something we plan to investigate in the future.

6 Appendix

Table A1. Means for weekly returns of 20 assets, 10^{-4}

	1	2	3	4	5	6	7	8	9	10
Mean	6.27	25.51	8.9	13.79	7.06	38.6	37.67	36.97	13.08	19.25
	11	12	13	14	15	16	17	18	19	20
Mean	4.43	16.6	46.45	17.22	29.19	18.98	16.33	21.79	20.65	17.72

Table A2. Marginal 3rd and 4th moments for weekly returns of 20 assets, 10^{-6}

	1	2	3	4	5	6	7	8	9	10
Marginal 3rd moment	-8.36	25.34	1.74	5.29	9.27	74.39	86.66	26.93	-3.76	-3.81
Marginal 4th moment	26.5	24.22	11.6	14.93	10.32	44.83	133.18	23.55	15.04	7.93
	11	12	13	14	15	16	17	18	19	20
Marginal 3rd moment	-29.87	-63.57	191.96	-13.48	7.29	-8.86	3.33	285.29	157.21	42.79
Marginal 4th moment	21.5	29.4	121.67	12.31	31.28	9.97	6.96	459.01	87.35	23.84

Table A3. Covariance matrix for weekly returns of 20 assets, 10^{-4}

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	18.58	6.18	7.14	7.03	4.48	3.47	10.37	5.21	4.22	4.42	7.11	4.58	10.55	4.93	5.76	2.95	4.52	16.51	6.73	6.16
2	6.18	22.69	3.03	3.07	5.01	2.89	5.13	3.44	5.09	1.28	3.4	2.37	4.94	2.26	4.72	5.43	2.33	7.71	8.89	8.01
3	7.14	3.03	15.15	11.67	4.17	3.59	10.65	7.79	3.79	3.04	4.96	3.16	10.99	3.95	3.8	2.89	3.37	9.31	4.08	4.45
4	7.03	3.07	11.67	16.58	4.77	4.04	11.87	7.94	4.33	3.64	4.93	3.42	12.93	5.64	3.92	3.13	4.28	8.47	4.42	4.49
5	4.48	5.01	4.17	4.77	13.25	3.36	2.94	3.18	8.54	3.52	2.51	3.05	2.14	4.44	3.2	4.03	4.19	7.36	3.02	3.27
6	3.47	2.89	3.59	4.04	3.36	17.6	4.54	3.37	3.07	4.43	2.99	3.09	4.06	5.42	1.54	3.75	5.24	4.24	1.13	2.49
7	10.37	5.13	10.65	11.87	2.94	4.54	37.81	12.57	2.56	3.66	6.96	4.44	25.27	4.36	5.75	2.65	4.3	13.66	4.87	6.55
8	5.21	3.44	7.79	7.94	3.18	3.37	12.57	18.32	3.62	3.64	6.31	3.5	11.52	3.26	4.06	3.71	3.73	8.99	4.16	3.66
9	4.22	5.09	3.79	4.33	8.54	3.07	2.56	3.62	16.04	3.92	2.68	3.5	2.17	4.18	2.92	4.47	3.61	6.91	3.74	4.17
10	4.42	1.28	3.04	3.64	3.52	4.43	3.66	3.64	3.92	11.43	3.89	3.24	3.64	4.03	4.46	2.85	3.44	5.48	1.89	2.7
11	7.11	3.4	4.96	4.93	2.51	2.99	6.96	6.31	2.68	3.89	14.97	4.24	7.4	3.05	6.09	3.41	2.28	11.38	4.54	4
12	4.58	2.37	3.16	3.42	3.05	3.09	4.44	3.5	3.5	3.24	4.24	19.85	4.71	4.98	4.02	2.5	4.36	7.32	0.96	2.4
13	10.55	4.94	10.99	12.93	2.14	4.06	25.27	11.52	2.17	3.64	7.4	4.71	34.81	4.25	5.79	2.19	4.51	13.44	6.9	5.63
14	4.92	2.26	3.95	5.64	4.44	5.42	4.36	3.26	4.18	4.03	3.05	4.98	4.25	14.12	4.34	2.46	6.24	6.3	2.29	2.49
15	5.76	4.72	3.8	3.92	3.2	1.54	5.75	4.06	2.92	4.46	6.09	4.02	5.79	4.34	22.62	2.84	3.28	9.1	5.02	4.01
16	2.95	5.43	2.89	3.13	4.03	3.75	2.65	3.71	4.47	2.85	3.41	2.5	2.19	2.46	2.84	10.65	3	5.11	4.23	3.79
17	4.52	2.33	3.37	4.28	4.19	5.24	4.3	3.73	3.61	3.44	2.28	4.36	4.51	6.24	3.28	3	11.76	6.9	1.36	1.77
18	16.51	7.71	9.31	8.47	7.36	4.24	13.66	8.99	6.91	5.48	11.38	7.32	13.44	6.3	9.1	5.11	6.9	46.21	10.27	8.75
19	6.73	8.89	4.07	4.42	3.02	1.13	4.87	4.16	3.74	1.89	4.54	0.96	6.9	2.29	5.02	4.23	1.36	10.27	28.29	12.14
20	6.16	8.01	4.45	4.49	3.27	2.49	6.55	3.66	4.17	2.7	4	2.4	5.63	2.49	4.01	3.79	1.77	8.75	12.14	19.22

Table A4. Optimal portfolio weights for S= 5043 scenarios for case 1

CVaR	x_2	x_6	x_7	x_8	x_9	x_{10}	x_{12}
0.040264	4.63%	36.00%	0.82%	9.76%	5.16%	5.06%	5.18%
0.039535	5.07%	32.67%	0.81%	9.37%	4.62%	6.16%	5.25%
0.039442	5.11%	32.10%	0.90%	9.58%	4.70%	6.24%	5.27%
0.039774	5.09%	33.65%	0.70%	9.54%	4.46%	5.99%	5.06%
0.039512	5.02%	32.36%	0.95%	9.09%	4.88%	6.14%	5.31%
0.039266	5.10%	30.83%	0.89%	9.68%	4.57%	6.34%	5.26%
0.039261	5.14%	30.87%	0.99%	9.60%	4.67%	6.31%	5.23%
0.040007	4.81%	34.80%	0.59%	9.40%	4.89%	5.59%	5.17%
0.039491	5.08%	32.08%	0.95%	9.75%	4.39%	6.20%	5.25%
0.039575	4.95%	32.53%	0.74%	9.98%	4.53%	5.94%	5.28%
0.039644	5.18%	33.00%	0.76%	9.40%	4.79%	6.20%	5.01%
0.039828	4.88%	33.83%	0.96%	9.24%	4.62%	5.82%	5.29%
0.039679	5.07%	33.16%	0.98%	9.29%	4.53%	6.04%	5.24%
0.039211	5.28%	30.75%	0.92%	9.63%	4.79%	6.58%	5.06%
0.039463	5.10%	32.34%	0.97%	9.20%	4.80%	6.23%	5.17%
0.039384	5.10%	31.40%	0.83%	9.82%	4.63%	6.28%	5.13%
0.039395	5.20%	31.85%	0.84%	9.68%	4.74%	6.39%	4.99%
0.040039	4.75%	35.02%	0.97%	9.04%	4.72%	5.52%	5.22%
0.039810	4.77%	33.74%	1.10%	8.87%	4.74%	5.70%	5.35%
0.039845	4.73%	33.94%	0.73%	9.39%	4.68%	5.61%	5.19%
CVaR	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{19}	x_{20}
0.040264	5.47%	2.21%	7.80%	6.09%	2.59%	4.95%	4.28%
0.039535	5.76%	2.87%	7.89%	7.13%	3.22%	4.90%	4.27%
0.039442	5.79%	2.98%	7.68%	7.08%	3.35%	4.85%	4.38%
0.039774	6.00%	2.63%	7.84%	7.16%	2.99%	4.86%	4.02%
0.039512	5.90%	2.95%	7.78%	7.08%	3.27%	4.87%	4.39%
0.039266	6.24%	3.13%	7.84%	7.44%	3.45%	4.84%	4.40%
0.039261	6.07%	3.14%	7.96%	7.24%	3.54%	4.91%	4.33%
0.040007	5.99%	2.64%	7.65%	6.52%	2.83%	4.87%	4.27%
0.039491	5.81%	2.96%	7.85%	7.01%	3.41%	4.91%	4.34%
0.039575	5.94%	2.89%	7.73%	6.77%	3.35%	4.95%	4.41%
0.039644	6.15%	2.95%	7.89%	7.06%	2.84%	4.72%	4.06%
0.039828	5.64%	2.68%	7.88%	6.67%	3.15%	4.95%	4.39%
0.039679	5.67%	2.89%	7.84%	6.93%	3.15%	4.91%	4.29%
0.039211	6.16%	3.13%	7.92%	7.41%	3.40%	4.83%	4.14%
0.039463	5.64%	2.95%	8.02%	7.10%	3.30%	4.90%	4.28%
0.039384	6.33%	2.93%	8.03%	7.37%	3.18%	4.72%	4.26%
0.039395	6.05%	2.99%	7.90%	7.54%	3.02%	4.70%	4.11%
0.040039	5.41%	2.71%	7.73%	6.76%	2.91%	4.98%	4.27%
0.039810	5.52%	2.85%	7.78%	6.74%	3.31%	5.07%	4.46%
0.039845	5.96%	2.75%	7.72%	7.04%	3.03%	4.91%	4.33%

References

[1] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31, 1986.

[2] P. Date, L. Jalen, and R. Mamon. A new algorithm for latent state estimation in nonlinear time series models. *Applied Mathematics and Computation*, 203:224–232, 2008.

[3] E.F.D. Ellison, M. Hajian, H. Jones, R. Levkovitz, I. Maros, G. Mitra, and D. Sayers. *FortMP Manual*, 2008.

[4] C. Erlwein, G. Mitra, and D. Roman. HMM based scenario generation for an investment optimisation problem. *Annals of Operations Research*, 193-1:173–192, 2012.

[5] R. Fourer, D. M. Gay, and B. Kernighan. *AMPL: A Mathematical Programming Language*, 1989.

[6] N. Gulpinar, B. Rustem, and R. Settergren. Optimisation and simulation approaches to scenario tree generation. *Journal of economic dynamics and control*, 28:1291–1315, 2004.

[7] R. Hochreiter and G.Ch. Pflug. Financial scenario generation for stochastic multi-stage decision processes as facility location problems. *Annals of Operations Research*, 152:257–272, 2007.

[8] K. Høyland, M. Kaut, and S.W. Wallace. A heuristic for moment matching scenario generation. *Computational Optimization and Applications*, 24:169–185, 2003.

[9] K. Høyland and S.W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47:295–307, 2001.

[10] http://www.londonstockexchange.com.

[11] M. Kaut and S.W. Wallace. Evaluation of scenario generation methods for stochastic programming. *Pacific Journal of Optimization*, 3:257–271, 2007.

[12] P.M. Lurie and M.S. Goldberg. An approximate method for sampling correlated random variables from partially specified distributions. *Management Science*, 44:203–218, 1998.

[13] H. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[14] E. Messina and D. Toscani. Hidden markov models for scenario generation. *IMA Journal of Management Mathematics*, 4:379–401, 2008.

[15] G. C. Pflug. *Some remarks on the Value-at-Risk and the Conditional Value-at-Risk, in"Probabilistic Constrained Optimization: Methodology and Applications"*. Ed. S. Uryasev, Kluwer Acadedemic Publishers, 2000.

[16] G.C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89:251–271, 2001.

[17] K. Ponomareva and P. Date. Higher order sigma point filter: A new heuristic for nonlinear time series filtering. *Applied Mathematics and Computation*, 22:662–671, 2013.

[18] R.T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26:1443–1471, 2002.

[19] D. Roman, G. Mitra, and N. Spagnolo. Hidden markov models for financial optimization problems. *IMA Journal of Management Mathematics*, 21(2):111–129, 2010.

[20] J.E. Smith. Moment methods for decision analysis. *Management Science*, 39:340–358, 1993.

[21] N. Topaloglou, H. Vladimirou, and S.A. Zenios. CVaR models with selective hedging for international asset allocation. *Journal of banking and finance*, 26:1535–1561, 2002.

[22] S.W. Wallace and W.T. Ziemba. *Applications of Stochastic Programming.* SIAM, 2005.