Michal Kaut

Scenario tree generation for stochastic programming: Cases from finance

Michal Kaut
Department of Mathematical Sciences
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
N-7491 Trondheim
michal.kaut@iot.ntnu.no
http://www.iot.nt.nu.no/~mkaut/

Dr. ing. thesis
July 2003

Thesis supervisor:
Stein W. Wallace, Molde University College
Co-supervisors:
Harald Krogstad, Norwegian University of Science and Technology
Kjetil Høyland, Gjensidige NOR

Evaluation committee:
Roger J-B Wets, University of California, Davis
Kurt Jörnsten, Norwegian School of Economics and Business Administration
Stein-Erik Fleten, Norwegian University of Science and Technology

Keywords:
stochastic programming, scenario tree, scenario generation

Typeset in LaTeX
NTNU Ingeniøravhandling 2003:55
ISBN 82-471-5606-7
ISSN 0809-103X

Preface

This thesis is a result of my Dr. ing. study at Department of Mathematical Sciences at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The described work was carried out in the period from August 1999 to May 2003, with Stein W. Wallace as the main supervisor, and Harald Krogstad and Kjetil Høyland as co-supervisors. The whole doctoral program was financed by Gjensidige NOR Asset Management, part of the Gjensidige NOR group. Out of the four years of the program, three years were dedicated to the completion of the doctoral degree, and one year to duties for the sponsor.

During the whole period, I was an employee at Department of Industrial Economics and Technology Management at NTNU. In addition to NTNU, the work was partially carried out at these locations: Gjensidige NOR Asset Management, Oslo, Norway; University of Edinburgh, Edinburgh, Scotland; University of Cyprus, Nicosia, Cyprus; and Molde University College, Molde, Norway.

The thesis' main subject is practical aspects of scenario generation in a context of stochastic programming. Since the project was financed by an insurance company, all the described applications are financial ones—mostly on portfolio management. However, most of the results are general and in no way restricted to finance.

The thesis consists of four papers, plus an introduction that presents the papers and describes the background of the project, as well as the practical achievements. One of the papers has been published.

Acknowledgements

I am deeply grateful to my supervisor, Stein W. Wallace, for his excellent supervision during the duration of the project. He has always been available for questions or discussion, and it was these that led to many of the results

presented in this thesis. In addition, his extensive network of contacts has allowed me to meet, and work with, some of the top researchers in the field. Further thanks go to my co-supervisors, Kjetil Høyland from Gjensidige NOR Asset Management, and Harald Krogstad from NTNU, as well as to my other co-authors, Hercules Vladimirou and Stavros Zenios from the University of Cyprus.

During the project I visited several institutions, and at all of them received great help and support from the local hosts. Hence, I would like to thank to Ken McKinnon from the University of Edinburgh; Kjetil Høyland, Erik Ranberg, and the whole team at Gjensidige NOR Asset Management; Hercules Vladimirou and Stavros Zenios from the University of Cyprus; and Ser-Huang Poon from the University of Strathclyde, Glasgow.

Since studying in a foreign country brings a lot of practical problems, I would like to thank those who helped me to solve them: Guri Andresen, department secretary at NTNU; Ragnhild Lundgren, secretary at Gjensidige NOR; and, most importantly, my supervisor Stein W. Wallace, without whose help it would have been much more difficult to survive the first year in Norway.

Last, but not least, I am very grateful to Gjensidige NOR Asset Management for opening and financing the project, and to Miloslav S. Vošvrda, supervisor for my Master thesis in Prague, and Vlasta Kaňková, both from the Czech Academy of Science, who pointed out the project for me, and supported me during the application process.

Contents

Introduction

Stochastic programming and scenario generation

In recent years, stochastic programming has gained an increasing popularity within the mathematical programming community, mainly because the present computing power allows users to add stochasticity to models that were difficult to solve in deterministic versions only a few years ago. For general information about stochastic programming, see for example Dantzig (1955); Birge and Louveaux (1997), or Kall and Wallace (1994).

As a result, a lot of research has been done on various aspects of stochastic programming. However, scenario generation has remained out of the main field of interest. In this thesis, we try to explain the importance of scenario generation for stochastic programming, as well as provide some methods for both generating the scenarios and testing their quality.

If we simplify the matters slightly, a stochastic programming model can be viewed as a mathematical programming model with uncertainty about the values of some of the parameters. This uncertainty is then described in statistical terms, so these parameters are described by their distributions (in a single-period case), or by stochastic processes (in the multi-period case).

Except for some trivial cases, stochastic programming models can not be solved directly with continuous distributions—in order to solve a typical stochastic programming model, we need to have a discrete distribution of limited cardinality. Hence, the "true" distribution has to be discretized, i.e. approximated by a discrete distribution. While some solution methods do the discretization (sampling) internally, most methods need a discrete distribution as an input, so the discretization has to be done prior to the solution of the stochastic programming model. The outcomes of the discretization are then called *scenarios*, and the whole distribution a *scenario tree*. By *scenario generation* we understand the process of discretizing the true distribution, and creating the scenario tree.

Scientific contribution

It should be rather obvious that scenario generation is an important part of the modelling process, since a "bad" tree can lead to a "bad" solution. We believe, however, that this importance is not understood and appreciated—which is why we wrote **Paper 1**. In this paper, we discuss the influence of scenario generation on the solution of the optimization model, and propose tests of quality/suitability of a given scenario-generation method for a given stochastic programming model. The paper also includes a short overview of different scenario-generation methods.

The approach from Paper 1 was later applied for testing the stability of an optimization model—in our case a portfolio-optimization problem based on a `CVaR` risk measure. In the tests, we investigate the model's sensitivity to instabilities and errors (mis-specifications) in the scenario tree. The tests are described in **Paper 2**.

At the time when I started the project, there was not any efficient scenario-generation method that would allow the user to control the statistical properties (moments and correlations) of the marginals, without any distributional assumptions. At Gjensidige NOR Asset Management (GNAM), who initiated the project, they used the method from Høyland and Wallace (2001). This method fulfills the mentioned requirements, but is very slow for large trees—it could take several hours to generate one scenario tree. Hence, development of a new, faster, method was pointed out as the reason why GNAM initiated—and sponsored—the position, and therefore the main practical goal of the project.

This goal was achieved by the algorithm described in **Paper 3**. Already in the first implementation, this algorithm was more than hundred times faster than the original one. After the paper has been published, we have come up with a new implementation, which is at least another ten times faster—for more information, see the note "Updates to the published version" after the Paper 3.

The most obvious shortcoming of Paper 3 is that it does not address the issue of multi-period trees. For this reason, we came back to this problem in **Paper 4**, which shows how to control the final-stage moments and correlations in a multi-period tree. Even though the aim of this paper is option pricing, instead of stochastic programming, most of the results related to scenario generation are general and therefore valid also in the context of stochastic programming.

Practical contribution

Apart from the scientific contribution, I have contributed to an improvement of the portfolio-optimization setup at Gjensidige NOR Asset Management. This section describes the achievements.

About Gjensidige NOR Asset Management

Gjensidige NOR Asset Management (GNAM) is the asset-management company within the Gjensidige NOR group. The group consists of Union Bank Norway, Gjensidige NOR Life Insurance, and Gjensidige NOR Non-Life, and is one of the three largest financial groups in Norway. GNAM manages a large part of the assets of the Life and Non-life companies, as well as funds for external clients. At the moment, the value of funds under management is approximately NOK 100 billion (14 billion USD).

Both the life insurance company and GNAM use stochastic programming models for their asset-allocation decisions. The original stochastic programming model was developed for the strategic level, and is described in Høyland (1998). Later, the model was modified also for the tactical level. See Wallace and Høyland (2003) for a comparison of the two models, as well as more information about GNAM.

As a part of my project, I spent one year working at GNAM. There I was involved with the stochastic programming model and the corresponding scenario-generation procedure, which form the backbone of the tactical asset-allocation system. Working with an optimization system that is being used in a company has showed me that there are many important "practical aspects" of an optimization system, something that would be difficult to learn by working with constructed examples.

Starting point

The main setup was all created by my predecessor Kjetil Høyland. Hence, when I started the project in August 1999, there was already a system in place at GNAM. The system consisted of the following parts:

asset-allocation model The portfolio optimization was formulated as a stochastic programming problem and implemented in `AMPL`[1]. It was a

[1] `AMPL`® – *A Modeling Language for Mathematical Programming*, developed at AT&T Bell Laboratories. See `http://www.ampl.com/` for more information.

single-period model, formulated and solved as a deterministic equivalent. MINOS[2] was being used for solving the problem.

scenario generation For generating scenarios, the procedure from Høyland and Wallace (2001) was used. The problem was formulated as a non-linear least-squares problem, implemented in AMPL and again solved using MINOS.

user interface User interface to both the scenario generation and the asset allocation was implemented as a system of VBA macros in Excel. The Excel sheet also accessed the required online data using REUTERS links.

The bottleneck of the portfolio-construction procedure was the scenario generation: Generating a tree for 12 assets and 1000 scenarios could take several hours, and had to be run overnight. This excluded any chance of an interactive approach to the optimization. In addition, it was obvious that 12 assets and 1000 scenarios is close to the maximal tree that could be generated using this method.

In addition to the speed of the scenario generation, there were other minor problems: The solution time of the optimization model was well over one hour, which was less than the scenario generation part, yet still quite long. The other problem was that the Excel user interface was rigid.

Improvements

Scenario generation

During the first year of the project, we developed and implemented the scenario-generation algorithm described in Paper 3. In the first step, the new algorithm was implemented in AMPL, and MINOS was used to solve subproblems. In every iteration, we had to solve one non-linear subproblem for every asset in the scenario tree. For a scenario tree with 12 assets and 1000 scenarios, the new algorithm was approximately 100 times faster than the original one, bringing the solution time down from several hours to several minutes. In addition, the speed-up was increasing with the size of the problem, so it became possible to create significantly larger trees.

To speed up the algorithm even further, I implemented it in the C programming language, using LOQO[3] callable libraries to find the coefficients of

[2] MINOS™– solver for sparse linear, quadratic and nonlinear problems, developed at Stanford University. See http://www.sbsi-sol-optimize.com/ for more information.

[3] LOQO – an interior-point solver for smooth optimization problem, developed by Robert J. Vanderbei from Princeton University. See http://www.orfe.princeton.edu/~loqo.

the cubic transformation. The new code was more than ten times faster than our `AMPL` implementation, but still had the disadvantage of depending on a commercial solver.

Finally, in the summer of 2002, Diego Mathieu from INSA Toulouse, France, who had a summer project at Molde University College, implemented the cubic transformation in the `C` programming language. Hence, we could replace the `LOQO` libraries and make the code completely self-contained.

With the latest implementation, and with slightly faster machines compared to those we used in 1999, a scenario tree with 20 assets and 2000 scenarios is typically created in few seconds. Hence, the problem with the duration of the scenario generation was solved.

Optimization model

The most significant increase of the speed of the optimization problem came from a change of the solver: The optimization problem is quadratic, while `MINOS` does not have any algorithms for quadratic programming (QP), and solves it as a general non-linear problem. At the end we have decided to use `LOQO`, which uses a barrier algorithm to solve QPs. `LOQO` turned out to be more than ten times faster than `MINOS`, bringing the solution time of a typical problem down to less than two minutes.

In addition to the objective and the constraints in the optimization model, the decision maker wishes to have a control over the tracking error of the portfolio. This could not be written as a constraint in the model, since none of the available solvers could not handle quadratic constraints. Instead, the tracking error is handled ex-post, using the risk-aversion parameter of the model: We solve the model, and if the tracking error is too high/low, we increase/decrease the value of the risk-aversion parameter, and run the optimization again.

Originally, this was being done manually. I have written a script that automatizes the procedure. Typically, we need 2–5 runs, so the total solution time is 5–10 minutes.

The optimization model was partially rewritten, even if the main structure was not changed. The most significant change was an introduction of options to the model. In addition, some minor errors and mis-specifications were corrected.

User interface

The `Excel` interface was completely restructured and the macros rewritten. The main objectives were to increase the flexibility of the `Excel` sheets, and

the user comfort. The added/improved features were:

- Possibility to add new regions and asset classes.
- Possibility to control most of the model parameters (including bounds) from the sheet.
- Visualisation of both scenario distributions and the portfolio positions.
- Addition of options (accompanied by corresponding extension of the decision model).
- Transition from `Reuters` to `Bloomberg` as suppliers of data.

The papers

This section presents the papers. Since all the papers are written in collaboration with other authors, I should specify my contribution. In all the papers, I have done most of the writing, all the programming, and all the testing. More details are given where needed.

Paper 1 – Evaluation of scenario-generation methods for stochastic programming

This paper was written together with my supervisor Stein W. Wallace. It summarizes some of our experience with scenario generation gained during the whole duration of the project, and is thus probably the most important paper in the collection. Some ideas from the paper were presented at the Nordic MPS'02 meeting in Bergen, Norway, September 2002. The paper is posted at SPEPS (Stochastic Programming E-Print Series), `http://www.speps.info`.

Paper 2 – Stability analysis of a portfolio management model based on the conditional value-at-risk measure

This paper was written together with my supervisor Stein W. Wallace, and Hercules Vladimirou and Stavros Zenios from the University of Cyprus. It contains results of work done during my one-semester visit at the University of Cyprus, in autumn 2001. Some of the results were presented at the Nordic MPS'02 meeting in Bergen, Norway, September 2002, and at the 32^{th} meeting of the EURO Working Group on Financial Modelling, London, UK, in April 2003.

Unlike the other papers, most of the text in the final version is not mine: Even though I wrote the first version of the paper, it was later revised by Hercules Vladimirou.

Paper 3 – A heuristic for moment-matching scenario generation

This paper was written together with my supervisor Stein W. Wallace, and Kjetil Høyland from Gjensidige NOR. The first version of the algorithm was finished at the beginning of 2000, and the algorithm was first presented at the 26^{th} meeting of the EURO Working Group on Financial Modelling in Trondheim, Norway, in May 2000. The first version of the paper was finished in June 2000, and was later presented at 7^{th} ELAVIO (Latin-American OR Summer School), Viña del Mar, Chile, in January 2001; at IFIP/IIASA/GAMM Workshop on Dynamic Stochastic Optimization, IIASA, Laxenburg, Austria, in

March 2002; and at APMOD 2002, Varenna, Italy, in June 2002. The paper was submitted to Computational Optimization and Applications in May 2001, and accepted after two revisions in August 2002. It was published in *Computational Optimization and Applications*, vol. 24(2–3), pages 169–185, 2003.

The first idea of the algorithm was Kjetil Høyland's, while I made the idea implementable by introducing the cubic transformation. Also the later refinements of the algorithm are mine.

In addition to the published version of the paper, we present a detailed description of data used in the numerical tests in the paper, and a note describing new development of the algorithm.

Paper 4 – Multi-period scenario tree generation using moment-matching: Example from option pricing

This paper was written together with my supervisor Stein W. Wallace. The original impulse for the paper came from Ser-Huang Poon from the University of Strathclyde, Glasgow, Scotland. Some of the ideas in the paper I learned during SIRIF/ESRC Postgraduate Training Activities: Derivatives and Computational Methods, University of Strathclyde, Glasgow, Scotland, September 2002. An early version of the paper was presented at the Half-Day Meeting on Stochastics and Computation in Mathematical Finance, University of Strathclyde, Glasgow, Scotland, September 2002.

Bibliography

J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer-Verlag, New York, 1997. ISBN 0-387-98217-5.

G. Dantzig. Linear programming under uncertainty. *Management Science*, 1: 197–206, 1955.

K. Høyland. *Asset liability management for a life insurance company. A stochastic programming approach*. PhD thesis, Norwegian University of Science and Technology, Trondheim, 1998.

K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.

P. Kall and S. Wallace. *Stochastic Programming*. Wiley, Chichester etc., 1994.

S. Wallace and K. Høyland. Using stochastic programming models for ALM and tactical asset allocation – a Norwegian study. In S. A. Zenios and W. T. Ziemba, editors, *Handbook of Asset and Liability Management*. Elsevier, 2003. ISBN 0-444-50875-9. Included in the series Handbooks in Finance.

Paper 1

Evaluation of scenario-generation methods for stochastic programming

Posted at SPEPS
(Stochastic Programming E-Print Series)
http://www.speps.info, ref. no. 14-2003

Evaluation of scenario-generation methods for stochastic programming

Michal Kaut
michal.kaut@iot.ntnu.no*

Stein W. Wallace
stein.w.wallace@himolde.no†

May 2003

Abstract

In this paper, we discuss the evaluation of quality/suitability of scenario-generation methods for a given stochastic programming model. We formulate minimal requirements that should be imposed on a scenario-generation method before it can be used for solving the stochastic programming model. We also show how the requirements can be tested.

The procedure of testing a scenario-generation method is illustrated on a case from portfolio management. In addition, we provide a short overview of the most common scenario-generation methods.

Keywords: stochastic programming, scenario tree, scenario generation

1 Introduction

In recent years, stochastic programming has gained an increasing popularity within the mathematical programming community. Present computing power allows users to add stochasticity to models that had been difficult to solve as deterministic models only a few years ago. In this context, a stochastic programming model can be viewed as a mathematical programming model with uncertainty about the values of some of the parameters. Instead of single values, these parameters are then described by distributions (in a single-period case), or by stochastic processes (in a multi-period case). A single-period

*Norwegian University of Science and Technology, N-7491 Trondheim, Norway
†Molde University College, Postboks 2110, N-6402 Molde, Norway

stochastic programming model can thus be formulated ([20]) as:

$$\begin{aligned} &\text{“min”}\ g_0(\boldsymbol{x}, \tilde{\boldsymbol{\xi}}) \\ &\quad \text{s.t. } g_i(\boldsymbol{x}, \tilde{\boldsymbol{\xi}}) \leq 0,\ i = 1, \ldots, m \\ &\qquad x \in \mathbf{X} \subset \mathbb{R}^n\ , \end{aligned} \tag{1}$$

where $\tilde{\boldsymbol{\xi}}$ is a random vector, whose distribution must be independent of the decision vector $\boldsymbol{x}$. Note that the formulation is far from complete—we still need to specify the meanings of “min” and the constraints.

Except for some trivial cases, (1) can not be solved with continuous distributions—most solution methods need discrete distributions. In addition, the cardinality of the support of the discrete distributions is limited by the available computing power, together with a complexity of the decision model. Hence, in most practical applications, the distributions of the stochastic parameters have to be approximated by discrete distributions with a limited number of outcomes. The discretization is usually called a *scenario tree* or an event tree – see Figure 1 for an example.

Hence, we solve only an approximation of (1), with the quality of the approximation directly linked to a quality of the scenario tree: *garbage in, garbage out* holds here as anywhere else. Surprisingly, there has been little focus on measuring the quality of scenario trees. In this paper, we thus ask the question of what is a good scenario-generation method for a given stochastic programming model. The link to the decision model is very important, we do not believe there is a scenario-generation method that would be best for all possible models, even if these models were subject to the same random phenomena.

When comparing scenario-generation methods, we focus on practical performance, not on the theoretical properties: it may be comforting to know that a certain method approximates the distribution perfectly when the number of outcomes goes to infinity, yet it does not mean that the method is good for generating a tree with just a few scenarios. Indeed, some of the methods mentioned in Section 2 do not guarantee convergence to the true distribution, but perform very well in real-life problems. For more information on the theoretical properties, see for example [8].

Because of the variety of both scenario-generation methods and decision models, we do not provide a guideline of the type “for this model use that method”. Instead, we formulate two important properties that a scenario-generation method should satisfied in order to be usable for a given model. We also show how to test the properties. The user can thus test several scenario-generation methods, and choose the one that is best suitable for the

given decision model.

The rest of the paper is organised as follows: Section 2 presents a short overview of the most important scenario-generation methods. Section 3 then describes the terminology and notation for the paper. Section 4 provides two criteria for the quality of a scenario tree, and Section 5 shows how to test them. Section 6 then demonstrates the tests on a case from portfolio management. Finally, Section 7 discusses some more aspects of scenario generation, before we conclude the paper.

2 Short overview of scenario-generation methods

2.1 "Pure" Scenario-generation methods

Conditional sampling.

These are the most common methods for generating scenarios. At every node of a scenario tree, we sample several values from the stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}$. This is done either by sampling directly from the distribution of $\{\tilde{\boldsymbol{\xi}}_t\}$, or by evolving the process according to an explicit formula $\tilde{\boldsymbol{\xi}}_{t+1} = z(\boldsymbol{\xi}_t, \tilde{\boldsymbol{\varepsilon}})$, or even $\tilde{\boldsymbol{\xi}}_{t+1} = z(\{\boldsymbol{\xi}_\tau, \tau < t\}, \tilde{\boldsymbol{\varepsilon}})$, sampling from $\tilde{\boldsymbol{\varepsilon}}$.

Traditional sampling methods can sample only from a univariate random variable. When we want to sample a random vector, we need to sample every marginal (the univariate component) separately, and combine them afterwards. Usually, the samples are combined all-against-all, resulting in a vector of independent random variables. The obvious problem is that the size of the tree grows exponentially with the dimension of the random vector: if we sample s scenarios for k marginals, we end-up with s^k scenarios.

Another problem is how to get correlated random vectors – a common approach ([23, 13, 31]) is to find the principal components (which are independent by definition) and sample those, instead of the original random variables. This approach has the additional advantage of reducing the dimension, and therefore reducing the number of scenarios.

There are several ways to improve a sampling algorithm. Instead of a "pure" sampling, we may, for example, use integration quadratures or low discrepancy sequences, if appropriate – see [27]. For symmetric distributions, [22] uses an antithetic sampling. Another way to improve a sampling method is to re-scale the obtained tree, to guarantee the correct mean and variance – see [1].

Sampling from specified marginals and correlations.

As mentioned in the previous section, the traditional sampling methods have problems generating multivariate vectors, especially if they are correlated. However, there are sampling-based methods that solve this problem, using various transformations.

In those methods, the user specifies the marginal distributions and the correlation matrix. In general, there is no restriction on the marginal distributions, they may even be from different families. Examples of such methods can be found in [2, 24, 6].

Moment matching.

The methods from the previous section may be used only if we know the distribution functions of the marginals. If we do not know them, we may describe the marginals by their moments (mean, variance, skewness, kurtosis etc.) instead. In addition, we specify the correlation matrix and possibly—if the method allows us—other statistical properties (percentiles, higher comoments, etc). Then we *construct* a discrete distribution satisfying those properties. Examples of this approach include [32, 30, 24, 15, 22, 25, 12, 16].

Path-based methods.

These methods start by generating complete paths, i.e. the scenarios, by evolving the stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}$. The result of this step is not a scenario tree, but a set of paths, also called a "fan". To transform a fan to a scenario tree, the scenarios have to be clustered (bound) together, in all-but-the-last period. This process is called clustering or bucketing. Examples of these methods can be found in [8, 17].

"Optimal discretization".

[28] describes a method that tries to find an approximation of a stochastic process (i.e. scenario tree) that minimizes an error in the objective function of the optimization model. Unlike the methods from the previous sections, the whole multi-period scenario tree is constructed at once. On the other hand, it works only for univariate processes. We use some of the methodology from [28] in Section 4.

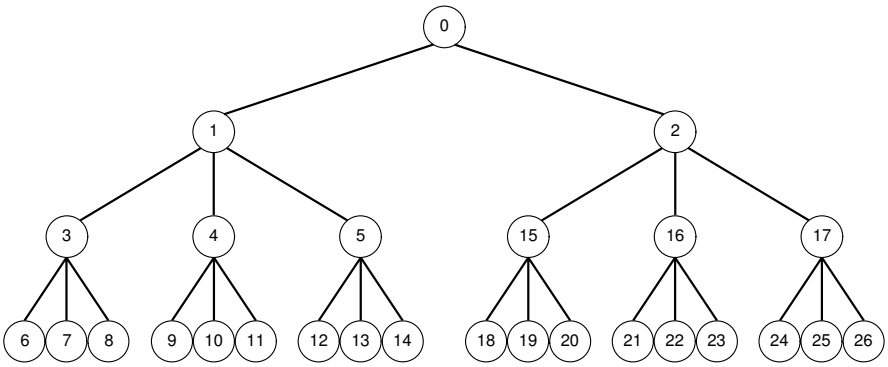

Figure 1: Example of a three-period tree

2.2 Related methods

Scenario reduction.

This is a method for decreasing the size of a given tree. This method tries to find a scenario subset of prescribed cardinality, and a probability measure based on this set, that is closest to the initial distribution in terms of some probability metrics. The method is described in [9, 29].

Internal sampling methods.

Instead of using a pre-generated scenario tree, some methods for solving stochastic programming problems sample the scenarios during the solution procedure. The most important methods of this type are: *stochastic decomposition* [14], importance sampling within Benders' (L-shaped) decomposition [5, 19, 18], and *stochastic quasigradient* methods [10, 11].

In addition, there are methods that proceed iteratively: they solve the problem with the current scenario tree, add or remove some scenarios and solve the problem again. Hence, at least in principle, the scenarios are added exactly where needed. The methods differ in the way they decide where to add/remove the scenarios: [3] uses dual variables from the current solution, while [7] measures the "importance of scenarios" by EVPI (expected value of perfect information).

3 Notation and terminology

Throughout the paper, we use the following conventions: stochastic variables are denoted by tilde (as in $\tilde{\xi}$), and discrete stochastic variables by breve ($\breve{\xi}$).

Stochastic processes are described as $\{\tilde{\xi}_t\}_{t\in\mathbf{T}}$, or only $\{\tilde{\xi}_t\}$. The notation can combine, so $\{\breve{\boldsymbol{\xi}}_t\}$ denotes a discrete multivariate process.

Let us have a stochastic programming model with uncertainty described by a stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}_{t\in\mathbf{T}}$. To be able to approximate the process by a scenario tree, the process has to be discrete in time, i.e. $\mathbf{T} = \{0,\ldots,T\}$. We call the points in time $t \in \mathbf{T}$ *stages*.[1] Since choosing the stages is often a natural part of the modelling process, we assume that the time discretization has already been done, so that we have the set $\mathbf{T}$.

In a scenario tree, the "true" stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}$ is approximated by a discrete process $\{\breve{\boldsymbol{\xi}}_t\}$. Since there is a unique relation between the scenario tree and the process $\{\breve{\boldsymbol{\xi}}_t\}$, we often refer to a "$T$-period scenario tree $\{\breve{\boldsymbol{\xi}}_t\}$". For example, the three-period tree in Figure 1 represents a stochastic process with two outcomes in the first period, and three outcomes per node in the last two periods.

In the rest of the paper, we focus on the objective function of the stochastic programming model (1). To simplify the formulas, we denote the whole model by

$$\min_{x\in\mathbf{X}} \; F\big(\boldsymbol{x};\tilde{\boldsymbol{\xi}}_t\big) \;, \tag{2}$$

where $\tilde{\boldsymbol{\xi}}_t$ is to be understood as $\{\tilde{\boldsymbol{\xi}}_t\}$. When we approximate the process $\{\tilde{\boldsymbol{\xi}}_t\}$ by a scenario tree $\{\breve{\boldsymbol{\xi}}_t\}$, the objective function becomes $F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_t\big)$.

4 Measure of quality of a scenario tree

We should always remember that our goal is to solve a stochastic program. The only reason why we need a scenario tree is that we do not know how to solve the problem directly with the process $\{\tilde{\boldsymbol{\xi}}_t\}$. Hence, we should judge a scenario tree (and, consequently, a scenario-generation method) by the quality of the decision it gives us. *We are not concerned about how well the distribution is approximated, as long as the scenario tree leads to a "good" decision.* In other words, we are not necessarily searching for a discretization of a distribution that is optimal (or even good) in the statistical sense. See [30] for discussion and examples of this topic.

The error of approximating a stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}$ by a discretization $\{\breve{\boldsymbol{\xi}}_t\}$, for a given stochastic programming problem (2), is thus defined as the difference between the value of the true objective function at the optimal solutions of the true and the approximated problems. The following definition

[1] There is no general agreement on what should be called stages: in some contexts, stages are only those points in time where a decision is made.

of the error is from [28]:

$$\begin{aligned} e_f(\tilde{\boldsymbol{\xi}}_t, \breve{\boldsymbol{\xi}}_t) &= F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}_t\big); \tilde{\boldsymbol{\xi}}_t\Big) - F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x}; \tilde{\boldsymbol{\xi}}_t\big); \tilde{\boldsymbol{\xi}}_t\Big) \\ &= F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}_t\big); \tilde{\boldsymbol{\xi}}_t\Big) - \min_{\boldsymbol{x}} F\big(\boldsymbol{x}; \tilde{\boldsymbol{\xi}}_t\big) \end{aligned} \tag{3}$$

Note that $e_f(\tilde{\boldsymbol{\xi}}_t, \breve{\boldsymbol{\xi}}_t) \geq 0$, since the second element is the true minimum, while the first one is a value of the (true) objective function at an approximate solution. Note also that we do not compare the optimal solutions $\boldsymbol{x}$, but their corresponding values of the objective function. The reason is that the objective function of a stochastic programming problem is typically flat, so there can be different solutions giving very similar objective values.[2]

Definition (3) has one rather obvious problem: the error is, in most practical problems, impossible to calculate. [28] solves this by proving that, under certain uniform Lipschitz conditions,

$$e_f(\tilde{\boldsymbol{\xi}}_t, \breve{\boldsymbol{\xi}}_t) \leq 2 \sup_{\boldsymbol{x}} \Big| F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}_t\big) - F\big(\boldsymbol{x}; \tilde{\boldsymbol{\xi}}_t\big) \Big| \leq 2L\, d(\breve{\boldsymbol{\xi}}_t, \tilde{\boldsymbol{\xi}}_t) \ ,$$

where L is a Lipschitz constant of $F()$,[3] and $d(\breve{\boldsymbol{\xi}}_t, \tilde{\boldsymbol{\xi}}_t)$ is a Wasserstein (transportation) distance of the distribution functions of the processes $\{\breve{\boldsymbol{\xi}}_t\}$ and $\{\tilde{\boldsymbol{\xi}}_t\}$. An algorithm is then developed to construct a scenario tree that minimizes the upper bound, i.e. the Wasserstein distance $d(\breve{\boldsymbol{\xi}}_t, \tilde{\boldsymbol{\xi}}_t)$.

This approach has several shortcomings: The bounds can, in general, be quite loose, so even if we find a scenario tree that minimizes the upper bound, there is no guarantee that we will be close to the minimum of $e_f()$. In addition, minimization of the upper bound does not depend on the optimization problem, so we have missed the link between the scenario generation and the problem. (Only the constant L, i.e. the tightness of the bound, depends on the problem.)

In this paper, we have therefore taken a different approach: instead of trying to find the optimal scenario-generation method, we focus on evaluation of a given method. In this context, a scenario-generation method may be seen as a heuristic for minimizing the error $e_f()$, as opposed to [28], which comes with an exact method for minimizing an upper bound of $e_f()$.

There are two problematic operations in definition (3) of the error $e_f(\tilde{\boldsymbol{\xi}}_t, \breve{\boldsymbol{\xi}}_t)$:

[2] In addition, we would need to define a meaningful metric on the space of $\boldsymbol{x}$, which could itself be a problem.

[3] Actually, L is a Lipschitz constant of $f()$, where $F(\boldsymbol{x}, \tilde{\boldsymbol{\xi}}) = \mathbb{E}^{\tilde{\boldsymbol{\xi}}}[f(\boldsymbol{x}, \tilde{\boldsymbol{\xi}})]$. See [28] for details.

1. finding the "true" objective value $F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_t\big)$ for a given solution $\boldsymbol{x}$.

2. finding the "true" optimal solution to (2): $\operatorname{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x};\tilde{\boldsymbol{\xi}}_t\big)$

While the second is almost always prohibitive, since it needs solving the optimization problem with the continuous process, the first one may be possible, for example via simulation. In the next section, we discuss different approaches for testing the discretization error, together with other tests of the quality of the discretization.

5 Testing a scenario-generation method

There are (at least) two minimal requirements a scenario-generation method must satisfy. Since most of the methods involve some randomness, the first requirement is stability: if we generate several trees (with the same input) and solve the optimization problem with these trees, we should get the same optimal value of the objective function. The other requirement is that the scenario tree should not introduce any bias, compared to the true solution.

There is a conceptual difference between the two requirements: while the first one can, at least to some degree, be tested, a direct testing of the second is in most cases impossible.

5.1 Stability requirement

This requirement can be stated as follows: If we generate several scenario trees (discretizations $\{\breve{\boldsymbol{\xi}}_t\}$) for a given process $\{\tilde{\boldsymbol{\xi}}_t\}$, and solve the stochastic programming problem with each tree, we should get (approximately) the same optimal value of the objective function.

Let us say that we generate K scenario trees $\breve{\boldsymbol{\xi}}_{tk}$, solve the optimization problem with each one of then, and obtain optimal solutions $\boldsymbol{x}^*_k,\ k = 1 \dots K$. By an *in-sample stability* we then understand

$$F\big(\boldsymbol{x}^*_k;\breve{\boldsymbol{\xi}}_{tk}\big) \approx F\big(\boldsymbol{x}^*_l;\breve{\boldsymbol{\xi}}_{tl}\big) \qquad k,l \in 1 \dots K\ ,$$

while an *out-of-sample stability* is defined as

$$F\big(\boldsymbol{x}^*_k;\tilde{\boldsymbol{\xi}}_t\big) \approx F\big(\boldsymbol{x}^*_l;\tilde{\boldsymbol{\xi}}_t\big) \qquad k,l \in 1 \dots K\ .$$

Or, equivalently:

$$\begin{aligned}
&\text{in-sample:} && \min_{\boldsymbol{x}} F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_{tk}\big) \approx \min_{\boldsymbol{x}} F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_{tl}\big)\\
&\text{out-of-sample:} && F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_{tk}\big);\tilde{\boldsymbol{\xi}}_t\Big) \approx F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x};\breve{\boldsymbol{\xi}}_{tl}\big);\tilde{\boldsymbol{\xi}}_t\Big)\\
&\text{out-of-sample, using (3):} && e_f(\tilde{\boldsymbol{\xi}}_t,\breve{\boldsymbol{\xi}}_{tk}) \approx e_f(\tilde{\boldsymbol{\xi}}_t,\breve{\boldsymbol{\xi}}_{tl})
\end{aligned}$$

There is an important difference between the two definitions: while for the in-sample stability we need only solve the scenario-based optimization problem, for the out-of-sample stability we have to be able to evaluate the "true" objective function $F\big(\boldsymbol{x};\tilde{\boldsymbol{\xi}}_t\big)$. To be able to do this, we need to have a full knowledge of the distribution of $\{\tilde{\boldsymbol{\xi}}_t\}$, and even then it may not be straightforward to evaluate $F\big(\boldsymbol{x};\tilde{\boldsymbol{\xi}}_t\big)$.

It is important to realize that the two stabilities are different and that there is no simple relationship between them. This can be demonstrated on the following one-period, one-dimensional example:

$$\min_{x\in\mathbb{R}} \; F\big(x;\tilde{\xi}\big) = \mathbb{E}^{\tilde{\xi}}\Big[\big(x-\tilde{\xi}\big)^2\Big]$$

This problem can be solved explicitly, for any distribution of $\tilde{\xi}$ (we drop the distribution index):

$$\begin{aligned}
F\big(x;\tilde{\xi}\big) &= \mathbb{E}\left[\big(\tilde{\xi}-x\big)^2\right]\\
&= \mathbb{E}\left[\Big(\big(\tilde{\xi}-\mathbb{E}[\tilde{\xi}]\big)+\big(\mathbb{E}[\tilde{\xi}]-x\big)\Big)^2\right]\\
&= \mathbb{E}\left[\big(\tilde{\xi}-\mathbb{E}[\tilde{\xi}]\big)^2\right]+\mathbb{E}\left[2\big(\tilde{\xi}-\mathbb{E}[\tilde{\xi}]\big)\big(\mathbb{E}[\tilde{\xi}]-x\big)\right]+\mathbb{E}\left[\big(\mathbb{E}[\tilde{\xi}]-x\big)^2\right]\\
&= \operatorname{Var}[\tilde{\xi}]+0+\big(x-\mathbb{E}[\tilde{\xi}]\big)^2,
\end{aligned}$$

so the optimal solution is

$$\begin{aligned}
x^* &= \operatorname*{argmin}_{x\in\mathbb{R}} \; F\big(x;\tilde{\xi}\big) = \mathbb{E}[\tilde{\xi}]\\
F\big(x^*;\tilde{\xi}\big) &= \min_{x\in\mathbb{R}} \; F\big(x;\tilde{\xi}\big) = \operatorname{Var}[\tilde{\xi}]
\end{aligned}$$

Now, assume we generate sample trees $\breve{\xi}_k,\ k=1\dots K$, and get the solutions $x_k^* = \mathbb{E}\big[\breve{\xi}_k\big]$. Let us first assume that the scenario-generation method is such that all the samples $\breve{\xi}_k$ have the correct means (i.e. $\mathbb{E}\big[\breve{\xi}_k\big] = \mathbb{E}\big[\tilde{\xi}\big]$), but the variances are different in all the samples. Hence $F\big(x_k^*;\breve{\xi}_k\big) = \operatorname{Var}\big[\breve{\xi}_k\big]$ is different for all the samples, so we do not have in-sample stability.

At the same time, $x_k^* = x^*$, so $F\big(x_k^*; \tilde{\xi}\big) = F\big(x^*; \tilde{\xi}\big)$, and the out-of-sample stability holds.

If we instead assume that we have a scenario-generation method that produces samples with correct variances (i.e. $\mathrm{Var}\big[\breve{\xi}_k\big] = \mathrm{Var}\big[\tilde{\xi}\big]$), but the means are different in all the samples,[4] we would have $F\big(x_k^*; \breve{\xi}_k\big) = \mathrm{Var}\big[\breve{\xi}_k\big] = \mathrm{Var}\big[\tilde{\xi}\big]$, so we would have the in-sample stability. On the other hand, $F\big(x_k^*; \tilde{\xi}\big) = \mathrm{Var}\big[\tilde{\xi}\big] + \big(\mathbb{E}\big[\breve{\xi}_k\big] - \mathbb{E}\big[\tilde{\xi}\big]\big)^2$ would be different for all the samples, so the problem would be out-of-sample unstable.

We may ask what is the practical difference between the in-sample and out-of-sample stability, and which of them is more important to have. Having out-of-sample stability means that the real performance of the solution $\boldsymbol{x}_k^*$ is stable, i.e. it does not depend on which scenario tree $\{\breve{\boldsymbol{\xi}}_t\}$ we choose. However, if we do not have the in-sample stability as well, we may be getting good solutions, but without knowing how good they really are (unless we solve several instances and take an average, or do the out-of-sample evaluation). The opposite (in-sample without out-of-sample stability) is even more dangerous, since the real performance of the solutions depends on which scenario tree we pick—without the possibility of detecting it by solving the problem on several trees.

In the example above, we could see that it is possible to have an in-sample instability in the objective function, but still have an in-sample stability of the solutions—in our case, the solutions were the same in all the sample trees. This obviously guarantees an out-of-sample stability. Therefore, if we detect an in-sample instability of the objective, we should look at the solutions as well. However, it does not work the other way around, i.e. we can have the out-of-sample stability even if the in-sample solutions vary, because the objective functions of stochastic programming problems are typically flat.

It can be expected that in most practical applications we will have either both the stabilities or none, so the in-sample tests should be sufficient in detecting a possible instability. However, if there is a way to perform the out-of-sample test, we would recommend to do that as well.

There are several possible ways to do the out-of-sample testing, i.e. the *evaluation* of the objective function $F\big(\boldsymbol{x}_k; \tilde{\boldsymbol{\xi}}_t\big)$ for a given decision $\boldsymbol{x}_k$. If we know the true stochastic process $\{\tilde{\boldsymbol{\xi}}_t\}$, the obvious choice is some Monte-Carlo-like simulation method. If we, on the other hand, use historical data in the scenario generation, back-testing may be an appropriate option. Or, if we have another scenario-generation method we believe to be stable, we may use it to create a reference scenario tree and evaluate the solutions $\boldsymbol{x}_k$ on that tree—notice that the tree can be quite big, since we are not solving a stochastic

[4] This may not be a very realistic example, but that is not the point here.

programming problem on it, we are only evaluating the objective function for a given decision.

To conclude the section, we would like to repeat that stability is the minimal requirement we should put on a scenario-generation method. Hence, before we start to work with a new optimization model, or a new scenario-generation method (remember that we test the two together), we should always run the stability tests: the in-sample test and, if feasible, the out-of-sample test.

5.2 Testing for a possible bias

In addition to being stable (both in-sample and out-of-sample), the scenario-generation method should not introduce any bias into the solution. In other words, the solution of the scenario-based problem,

$$\breve{\boldsymbol{x}}^* = \operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}_t\big) \ ,$$

should be an (almost) optimal solution of the original problem (2). Hence, the value of the "true" objective function at the scenario solution, $F\big(\breve{\boldsymbol{x}}^*; \tilde{\boldsymbol{\xi}}_t\big)$, should be (approximately) equal to the "true" optimal value $\min_{\boldsymbol{x}} F\big(\boldsymbol{x}; \tilde{\boldsymbol{\xi}}_t\big)$:

$$F\big(\breve{\boldsymbol{x}}^*; \tilde{\boldsymbol{\xi}}_t\big) = F\Big(\operatorname*{argmin}_{\boldsymbol{x}} F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}_t\big); \tilde{\boldsymbol{\xi}}_t\Big) \approx \min_{\boldsymbol{x}} F\big(\boldsymbol{x}; \tilde{\boldsymbol{\xi}}_t\big) \ .$$

Or, using the definition (3),

$$e_f(\tilde{\boldsymbol{\xi}}_t, \breve{\boldsymbol{\xi}}_t) \approx 0 \ .$$

The problem is that testing of this property is in most practical problems impossible, since it needs solving the optimization problem with the (true) continuous process—and if we could solve that, we would not need scenario trees in the first place.

In some cases, however, it can be possible to do some approximate tests. One possibility is to built a *reference tree*, and use it as a representation (approximation) of the true stochastic process. Typically, such a tree should be as big as possible, i.e. the biggest tree for which we can still solve the optimization problem. To create such a tree, we would need a method that is guaranteed to be unbiased—we can not use the method we want to test! For example, if we use a data series as an input for the scenario generation, we may try using all the history as scenarios.

5.3 Improving the performance

When the testing shows that our scenario-generation method is instable or biased (for the given stochastic programming model), the next question is what are the possible causes of the problem. The answer depends to a large degree on the type of the scenario-generation method used:

Sampling methods.

When we use a sampling method, the strongest candidate for the source of the instability or bias is a lack of scenarios—we know that, with an increasing number of scenarios, the discrete distribution converges to the true distribution. Hence, by increasing the number of scenarios, the trees will be closer to the true distribution, and consequently also closer to each other. As a result, both the instability and the bias should decrease.

In addition to increasing the number of scenarios (which is usually limited by the solution time for the optimization model), we can also try to improve the sampling method. Some of the options are included in the overview in Section 2.

Moment-matching methods.

With moment-matching methods, the situation is more complicated. Since these methods generally do not guarantee convergence, increasing the number of scenarios is not guaranteed to help. We thus need to look at different issues. In the following discussion we assume that in all the tested scenario trees $\breve{\boldsymbol{\xi}}_{tk}$, we have managed to match all the required properties perfectly, i.e. the instability/bias has to come from some properties we do not control (and that can, thus, vary between the tested trees).

Even without the convergence guarantee, the first thing to test is still the number of scenarios: There is an obvious difference between a discrete distribution with three points, and a discrete distribution with thousand points, even if their first four (or even five) moments can be equal. The difference is in the *smoothness* of the distribution, and our experience shows that this is often an important factor. In this context, it is important to understand that not all the moment-matching methods show increasing smoothness with increasing number of scenarios: while this is typically the case for transformation-based methods (for example [16]), it is not true for optimization-based methods like [15].

The important issue of the moment-matching methods is whether we match the right properties—an issue that is obviously dependent on the optimization

model. While for a mean-variance model it is enough to match the means, variances, and the correlation matrix, most optimization models will require more. Our experience shows that the first four moments are often a good enough description of the marginals, at least for financial models. On the other hand, a correlation matrix may not be enough to describe the multi-variate structure. In such a case, we may try to match also higher co-moments, or use a copula ([26, 4]), if we have the necessary data and a scenario-generation method that can work with these properties.

What shall we then do, when we discover that a moment-matching method is either instable or biased? The first thing to try is to increase the number of scenarios as much as possible (we still have to be able to solve the optimization model in a reasonable time). If this helps, the problems were probably caused by the lack of smoothness in the original trees. Otherwise, it means that there is some property the decision model reacts to, but we do not control it in the scenario-generation process. We have no general advices on identifying the missing property—it depends on the decision model, and is typically done by a trial-and-error approach, based on problem understanding.

6 Test case: a portfolio optimization

As a test case, we use a simple one-period portfolio optimization problem: we consider one-month investments in three indices (stocks, short-term bonds, and long-term bonds), in four markets (USA, UK, Germany, Japan), giving us twelve assets in total. We model the situation of a US investor, so we have to include the exchange rates of the three foreign currencies to USD. Hence, we have fifteen random variables in the scenario trees. In the model, we do not allow short positions. In addition, it is possible to hedge the currency risk with forward contracts.

As an objective function, we use the expected return and quadratic penalties for shortfalls (returns under a given threshold):

$$\begin{aligned} \mathrm{sf}(\boldsymbol{\xi}) &= \max\left(\mathrm{Tg} - \mathrm{ret}(\boldsymbol{x}, \boldsymbol{\xi}), 0\right) \\ F\big(\boldsymbol{x}; \breve{\boldsymbol{\xi}}\big) &= \mathbb{E}\left[\mathrm{ret}(\boldsymbol{x}, \breve{\boldsymbol{\xi}}) - \alpha\left(\mathrm{sf}(\breve{\boldsymbol{\xi}}) + \beta\,\mathrm{sf}(\breve{\boldsymbol{\xi}})^2\right)\right] , \end{aligned}$$

where α is a risk-aversion parameter, and β is a weight of the quadratic term. In the test, we used the following values: $\mathrm{Tg} = 0$, $\alpha = 1$, and $\beta = 10$.

We use the moment-matching scenario-generation method from [16] to generate the scenarios. This method generates scenario trees with specified first four moments of the marginal distributions (mean, standard deviation, skewness and kurtosis), and correlation matrix.

For the out-of-sample test, and for the test of a bias, we need a representation of the "real world". In our case, we take a large scenario set— that we refer to as the "benchmark scenario set". *It is important that the benchmark set is provided exogenously, that is, it is not generated by the same method which we want to test.* In our case, the benchmark scenario set (tree) was generated by a method based on principal component analysis described in [31]. The benchmark tree has $20,000$ scenarios, and is based on data in the period from January 1990 to April 2001. See [21] for a detailed description of properties of the benchmark scenario set. We note that the scenarios of the benchmark tree are not equiprobable.

Based on the benchmark tree, we compute the moments and correlations of the differentials of the random variables. The values of these statistical properties constitute the targets to match with our scenario generation procedure.

Since we have the benchmark scenario tree as an representation of the true distribution, we can perform all the tests from Section 5: For a given size of the tree, we generate 25 scenario trees, solve the optimization model on each of them, and then evaluate the solutions on the benchmark tree. This is repeated for several different sizes of the tree.

Results of the test are presented in Table 1. We see that the scenario-generation method used gives a reasonable stability, both in-sample and out-of-sample. We see also that the out-of-sample values have a smaller variance then the in-sample values. The reason is that in in-sample tests we evaluate (different) solutions on different trees, while in the out-of-sample tests we evaluate all the solutions on the common benchmark tree. Note also that the performance (true objective value of the solutions) improves as the number of scenarios increases.

Another important observation is the fact that, in the case of 50 scenarios, the in-sample objective values are significantly higher than the out-of-sample (true) values. In other words, the solution is notably worse than the model tells us. This is a common observation: when we do not have enough scenarios, the model overestimates the quality of its own solution. Only out-of-sample evaluations can tell us how good a solution really is.

In addition to the stability tests, we have solved the optimization model on the benchmark tree, and obtained the "true" optimal solution: 0.00930. Hence, we see that the scenario generation method does not introduce any significant bias, given there are enough scenarios. We also see that there is a noticeable bias in the case of 50 scenarios.

The conclusion of the tests is that the tested scenario-generation method is suitable for the given optimization model: it is stable and does not introduce

Table 1: Stability tests for the optimization model. For every size of the scenario tree, 25 trees were generated, and the model was solved on each of them. The solutions were then evaluated on the benchmark tree to obtain the out-of-sample values. The table presents sample means and standard deviations of the optimal values, for the different sizes.

description of the test			# of scenarios			
type of test	objective f.	value	50	250	1000	5000
in-sample	$F(\boldsymbol{x}_k^*; \breve{\boldsymbol{\xi}}_{tk})$	average	0.00948	0.00936	0.00931	0.00929
		std. dev.	0.00023	0.00011	0.00005	0.00002
out-of-sample	$F(\boldsymbol{x}_k^*; \tilde{\boldsymbol{\xi}}_t)$	average	0.00902	0.00926	0.00928	0.00930
		std. dev.	0.00015	0.00003	0.00001	0.00000

any significant bias, provided we have enough scenarios. The results also suggest that we should not use trees smaller than 1000 scenarios.

7 How far can we get?

So far, we have implicitly assumed that all distributions are known. In reality this is very rarely the case. What does this lack of knowledge mean, particularly for the issue we raise here, that of generating good scenario trees? First, let us distinguish between three cases:

- The distributions are fully known.
- We have theoretical knowledge about the distribution family, plus data.
- We only have data.

Although it is common to assume in theoretical papers that a distribution is fully known—very often this is done indirectly by basing the paper (or the tests within the paper) on certain distributions—this is in our view not the case in many applications. An exception may be planning under well-known stochasticity, such as the roll of a (fair) die or the flip of a (fair) coin. But most interesting applications concern real-life phenomena such as price, demand or quality. So, if the distribution is not known, what can we then say about scenario trees? The most important, and also obvious, observation is that certain theoretical properties of scenario generation methods become less useful. For example, sample-based methods will, if the sample is large enough, produce scenario trees arbitrarily close to the distribution from which we sample. But how useful is it to know that we have convergence towards something that is not the real thing?

This becomes even more important if we have do not know a distribution family. A common approach in this case is to assume some family. If we do so, we *know* that sampling will converge to a distribution that contains information we have added without foundation in data or theory. An alternative is using the empirical distribution directly. In this way we avoid adding any subjective information to the data. On the other hand, this approach will prevent the use of any methodology that requires knowledge of the distribution itself.

In the (scarce) occasions that we have distributional information, it is normally advisable to estimate the distribution, since otherwise that information is lost. But we still face the problem of having convergence to a distribution that may not be the right one.

But there is more to the problem than this. To use data we have to assume that the past is a reasonable description of the future. Whether this is the case cannot be tested, it is a question of belief. We can of course test, at least in many cases, whether or not the past would have been a good description of the future, in the past. But it is still a question of belief if this is the case now.

We have mentioned simulation as a way to evaluate the true value of a certain decision. The good thing about simulation models is that they can be allowed to contain details that we are unable to put into the optimization model. But all the problems discussed above remain. Within the simulation model we need to sample from distributions, and they are normally not fully known.

This discussion may seem to be very negative, we seem to be saying that nothing works. That is not the point. The point is to be aware of the shortcomings of modeling in general, and stochastic programming in particular. We can get to a certain point, but thereafter empirical testing becomes impossible, and we have to start believing in what we do. And in our view, this also means that we should be very sceptical to high accuracy statements from models. We may know that a given method retains two digits of accuracy, but we cannot know how many correct digits were in the input.

Hence, the convergence properties are not so important in real-life applications. Instead, it is more important to have scenario generation tools that give us reasonable control over the tree with a limited number of scenarios. What we want is a method that is stable, unbiased and produces small trees. But there is a limit to what we require in terms of accuracy, given these properties.

Conclusions

In this paper, we have discussed how to evaluate the suitability of a given scenario-generation method for a given stochastic programming problem. We have identified the main properties the scenario-generation method should satisfied, and suggested a way to test them. We have also demonstrated the methodology on an example from portfolio management.

References

[1] D. R. Cariño, T. Kent, D. H. Myers, C. Stacy, M. Sylvanus, A. L. Turner, K. Watanabe, and W. T. Ziemba. The Russell-Yasuda Kasai model: an asset liability model for a Japanese insurance company using multistage stochastic programming. *INTERFACES*, 24(1):29–49, 1994.

[2] M. C. Cario and B.L. Nelson. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 1997.

[3] Michael Casey and Suvrajeet Sen. The scenario generation algorithm for multistage stochastic linear programming. Available at `http://www.math.ups.edu/~mcasey/`, 2002.

[4] Robert T. Clemen and Terence Reilly. Correlations and copulas for decision and risk analysis. *Management Science*, 45(2):208–224, February 1999.

[5] George B. Dantzig and Gerd Infanger. Large-scale stochastic linear programs—importance sampling and Benders decomposition. In *Computational and applied mathematics, I (Dublin, 1991)*, pages 111–120. North-Holland, Amsterdam, 1992.

[6] B. Deler and B. L. Nelson. Modeling and generating multivariate time series with arbitrary marginals using an autoregressive technique. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 2000.

[7] M. A. H. Dempster and R. T. Thompson. EVPI-based importance sampling solution procedures for multistage stochastic linear programmes on parallel MIMD architectures. *Annals of Operations Research*, 90:161–184,

1999. also in Proceedings of the POC96 Conference, Versailles, March, 1996.

[8] Jitka Dupačová, Giorgio Consigli, and Stein W. Wallace. Scenarios for multistage stochastic programs. *Ann. Oper Res.*, 100:25–53 (2001), 2000.

[9] Jitka Dupačová, Nicole Gröwe-Kuska, and Werner Römisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.

[10] Yu. Ermoliev. *Methods of Stochastic Programming*. Nauka, Moscow, 1976. In Russian.

[11] Yury M. Ermoliev and Alexei A. Gaivoronski. Stochastic quasigradient methods for optimization of discrete event systems. *Ann. Oper. Res.*, 39(1-4):1–39 (1993), 1992.

[12] N. Gülpınar, B. Rustem, and R. Settergren. Optimisation and simulation approaches to scenario tree generation. *Journal of Economics Dynamics and Control*, to appear, 2002.

[13] Roger Halldin. *Scenario Trees for Inflow Modelling in Stochastic Optimisation for Energy Planning*. PhD thesis, Lund Unversity, Sweden, 2002.

[14] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

[15] K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.

[16] Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2-3):169–185, 2003.

[17] IBM Corp. *IBM Optimization Library Stochastic Extensions Users Guide*, 1998.

[18] G. Infanger. *Planning under Uncertainty: Solving Large-Scale Stochastic Linear Programs*. Boyd and Fraser, Danvers, 1994.

[19] Gerd Infanger. Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Ann. Oper. Res.*, 39(1-4):69–95 (1993), 1992.

[20] P. Kall and S.W. Wallace. *Stochastic Programming*. Wiley, Chichester, 1994.

[21] Michal Kaut, Stein W. Wallace, Hercules Vladimirou, and Stavros Zenios. Stability analysis of a portfolio management model based on the conditional value-at-risk measure. Feb 2003.

[22] R. R. P. Kouwenberg. Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134(2):51–64, 2001.

[23] Mico Loretan. Generating market risk scenarios using principal components analysis: Methodological and practical considerations. In *The Measurement of Aggregate Market Risk, CGFS Publications No. 7*, pages 23–60. Bank for International Settlements, November 1997. Available at `http://www.bis.org/publ/ecsc07.htm`.

[24] P. M. Lurie and M. S. Goldberg. An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science*, 44(2):203–218, 1998.

[25] Johan Lyhagen. A method to generate multivariate data with moments arbitrary close to the desired moments. Working paper 481, Stockholm School of Economics, December 2001.

[26] Roger B. Nelsen. *An Introduction to Copulas*. Springer-Verlag, New York, 1998.

[27] T. Pennanen and M. Koivu. Integration quadratures in discretization of stochastic programs. Stochastic Programming E-Print Series, `http://www.speps.info`, May 2002.

[28] G. C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271, 2001.

[29] W. Römisch and H. Heitsch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2-3):187–206, 2003.

[30] James E. Smith. Moment methods for decision analysis. *Management Science*, 39(3):340–358, March 1993.

[31] N. Topaloglou, Vladimirou H., and S. A. Zenios. CVaR models with selective hedging for international asset allocation. *Journal of Banking and Finance*, 26(7):1535–1561, 2002.

[32] C. David Vale and Vincent A. Maurelli. Simulating multivariate nonnormal distributions. *Psychometrika*, 48(3):465–471, 1983.

Paper 2

Stability analysis of a portfolio management model based on the conditional value-at-risk measure

Stability analysis of a portfolio management model based on the conditional value-at-risk measure

Michal Kaut
michal.kaut@iot.ntnu.no*

Stein W. Wallace
stein.w.wallace@himolde.no†

Hercules Vladimirou
hercules@ucy.ac.cy‡

Stavros A. Zenios
zenioss@ucy.ac.cy‡

February 2003

Abstract

We examine the stability of a portfolio management model based on the conditional value-at-risk (CVaR) measure. The stochastic programming model controls total risk exposure of an international investment portfolio. This includes both market risk in multiple countries as well as currency exchange risk. Uncertainty in asset returns and exchange rates is modeled in terms of discrete scenarios that are generated by means of a moment matching method. The procedure generates a set of scenarios with statistical properties of the random variables matching specific target values that are estimated using historical market data.

First, we establish that the scenario generation procedure does not bias the results of the optimization program, and we determine the required number of scenarios to attain stable solutions. We then investigate the sensitivity of the CVaR model to errors (mis-specifications) in the statistics of stochastic parameters (i.e., mean, variance, skewness and kurtosis of the marginal distributions, as well as correlations). We empirically determine the effects on the model's results due to systematic changes in the target statistics at the scenario generation phase. The results are most sensitive to changes in the means of the stochastic parameters (domestic asset returns and currency exchange rates). As expected, accurate estimation of the means is a most important concern in the portfolio management problem. However, errors in the variance,

*Norwegian University of Science and Technology, N-7491 Trondheim, Norway.

†Molde University College, Postboks 2110, N-6402 Molde, Norway.

‡HERMES Center on Computational Finance and Economics, School of Economics and Management, University of Cyprus, P.O. Box 20573, CY-1678 Nicosia, Cyprus.

skewness and correlations of the random parameters also have considerable — and approximately equal — impact on the solutions. The effect from errors in the values of kurtosis is about one half of the effects of the previous statistics, but still is not negligible.

This study demonstrates that optimization programs that use risk measures concerned with the tail of the portfolio's return distribution indeed exhibit sensitivity to the higher moments of stochastic inputs. Our results, moreover, quantify the relative impact of errors in the various statistical properties of these inputs. We conclude that care is needed when applying stochastic programming models for financial risk management to (a) reliably estimate the statistical properties of the random variables, and (b) generate scenario sets that reflect as closely as possible the target statistics, since mis-specifications of these inputs can influence the results.

1 Introduction

The mean-variance approach from Markowitz' seminal work in the 1950's formed the basis for portfolio selection in recent decades. The mean-variance method assumes that the returns of assets follow normal distributions and/or that the investor has a quadratic utility function. Despite the long and widespread use of the mean-variance paradigm in portfolio management, its fundamental assumptions often do not hold in practice. The returns of many financial securities exhibit skewed and leptokurtic distributions. The inclusion of derivatives, or securities with embedded options, in portfolios further stresses the problem as derivatives, by construction, have highly skewed return distributions. Many other investments are exposed to multiple risk factors whose joint effect on returns often can not be modeled by a normal distribution.

Substantial research effort has been directed toward the development of models that properly capture asymmetries and dynamic effects in the observed behavior of asset returns. At the same time, alternative risk metrics have been sought. Such measures are concerned with other, or additional, characteristics of the return distribution (e.g., the tails) besides the variance, and are designed to accommodate a wider range of investor priorities and regulatory requirements for risk management.

The value-at-risk (`VaR`) has essentially attained the status of a de-facto standard in financial practice. `VaR` is customarily defined as the maximal loss (or minimal return) of a portfolio over a specific time horizon and a specified confidence level. Let $\boldsymbol{x}$ denote the composition of a portfolio whose assets have

random returns $\tilde{\boldsymbol{r}}$.[1] The return of the portfolio is a random variable $\tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}})$ with a distribution function $F(\boldsymbol{x}, u) = P\{\tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \leq u\}$. VaR, defined in terms of portfolio return, is the minimal return for a prespecified (critical) confidence level $\alpha \times 100\%$:

$$\mathtt{VaR}(\boldsymbol{x}, \alpha) = \min\{u : F(\boldsymbol{x}, u) \geq 1 - \alpha\} = \min\{u : P\{\tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \leq u\} \geq 1 - \alpha\}.$$

Hence, $\mathtt{VaR}(\boldsymbol{x}, \alpha)$ corresponds to the $(1 - \alpha) \times 100\%$ percentile of the portfolio's return distribution. If we assume normality of returns then VaR is easily computable. Alternatively, it can be estimated via Monte Carlo simulation procedures.

The use of VaR as a risk measurement benchmark has been promoted by a leading provider of risk management tools [14]. It had also been embraced by academic researchers (see, for example, Jorion [8]). Many financial institutions compute the VaR value of their portfolios either to comply with regulatory requirements or for internal risk measurement purposes. The Basle Accord [11, 12] dictates that banks should maintain capital reserves that are determined as multiples of the VaR characterizing their portfolios (see Lucas [10]).

Although its calculation for a certain portfolio $\boldsymbol{x}$ indicates that the portfolio return will be below $\mathtt{VaR}(\boldsymbol{x}, \alpha)$ with likelihood $(1-\alpha)\times 100\%$, it provides no information on the extent of the distribution's tail which may be quite long; in such cases, the portfolio return may take substantially lower values than VaR and result in severe losses. Despite its widespread adoption in practice, the use of VaR for risk management purposes is being increasingly challenged by the research community both on theoretical, as well as on practical grounds. Theoretical limitations of VaR are discussed, for example, in Pflug [13], Acerbi and Tasche [1], Frey and McNeil [5], Rockafellar and Uryasev [16], and Szegö [17]. As pointed out by Pflug [13], VaR is not a coherent risk measure in the sense defined by Artzner et al. [2]. Specifically, VaR is not sub-additive. Hence, the VaR of a diversified portfolio can be larger than the sum of the VaRs of its constituent asset components.

It has been empirically observed that, when determined by means of simulations, the computed values of VaR can be rather unstable. Berkowitz and O'Brien [3] provide evidence that VaR models can have poor performance in practice. Moreover, when the returns of assets are expressed in terms of discrete distributions (i.e., scenarios) VaR is a non-smooth and non-convex function of the portfolio positions $\boldsymbol{x}$ and exhibits multiple local extrema (see, e.g., Rockafellar and Uryasev [16]). Incorporating such functions in mathematical programs is very difficult, thus making impractical the use of VaR in optimal portfolio management models.

[1] In this paper we use boldface type for vectors and tilde (˜) for random variables.

In response to the limitations of `VaR`, alternative risk metrics have been sought. Artzner et al. [2] discuss several such metrics and specify the required properties for coherent risk measures. Notable among them is the conditional value-at-risk (`CVaR`) — also called "mean excess loss", "expected shortfall" or "tail `VaR`". This risk measure has been analyzed in a number of recent papers; see, for example, Rockafellar and Uryasev [16], Acerbi and Tasche [1], Frey and McNeil [5], Tasche [18], Topaloglou et al. [19].

`CVaR` is a risk measure related to `VaR` as it quantifies the conditional expectation of losses exceeding `VaR`. When expressed in terms of portfolio return, `CVaR` is essentially the expected value of the $(1-\alpha)100\%$ worst (lowest) returns of portfolio $\boldsymbol{x}$. If $F(\boldsymbol{x}, u)$ is a continuous function of u, we have

$$\texttt{CVaR}(\boldsymbol{x}, \alpha) = \mathbb{E}\left[\tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \mid \tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \leq \texttt{VaR}(\boldsymbol{x}, \alpha)\right]. \tag{1}$$

As indicated by Pflug [13], unlike `VaR`, `CVaR` possesses the required properties of coherent risk measures, including subadditivity.

Models based on the `CVaR` risk measure are being increasingly applied to various portfolio management problems. As `CVaR` is concerned with the lower tail of the portfolio's return distribution, such models should be sensitive to higher moments of the constituent assets' random returns. If that were not the case, then the models' effectiveness for managing risks of portfolios whose returns have skewed distributions, or heavy tails, would be questionable. The purpose of this paper is specifically to study the stability of a such a portfolio management model with respect to changes in input specifications.

Kallberg and Ziemba [9] and Chopra and Ziemba [4] examined the relative effects of errors in the mean, variance and covariance of asset returns on mean-variance efficient portfolios. They found that the results are most sensitive to mis-specifications in the means of the asset returns. They reported that the impact of errors in the variance of asset returns was about an order of magnitude lower than that of errors in the means, while errors in covariance values had about half the impact of errors in the variance. The sensitivity of their model's results to mis-specifications of statistical properties of asset returns depended on the risk aversion parameter. Nevertheless, the relative effects of errors in these statistics were approximately as stated above.

Here, we similarly investigate the effects of errors (mis-specifications) in statistical properties — including higher moments — of asset returns on the results of a model based on the `CVaR` measure. As a test case we use a portfolio management model for international investments. The portfolio is exposed to market risk in multiple countries and to currency exchange risk. We use discrete scenarios to model the uncertainty in asset returns and spot exchange rates. The scenarios are generated by a moment matching method, so that in

the set of generated scenarios the random variables have statistical properties that match specific target values. These target statistics are determined from historical market data.

First, we define in-sample and out-of-sample stability and we demonstrate that the scenario generation procedure does not bias the results of the optimization model. That is, for sufficiently large representative scenario sets, the portfolio model produces stable solutions that are not dependent on the specific scenario sets (i.e., the results are stable with respect to sample). We empirically determine the required number of scenarios to attain stable solutions. We then conduct extensive computational experiments to determine the effects on the model's results due to controlled changes in the target statistics (i.e., mean, variance, skewness, kurtosis and correlations) of the random variables. We demonstrate that the portfolio optimization model that is based on the CVaR risk measure is indeed sensitive — as it should — to the higher moments of the stochastic inputs. Moreover, we quantify the relative impact of errors in the various statistical properties of these inputs on the model's results.

The rest of the paper is organized as follows. Section 2 states a linear program for a generic asset-allocation model that manages risk by constraining the CVaR measure of portfolio returns. Section 3 presents the specific international portfolio management model that we use as a test case in this study. Section 4 describes the scenario generation method, the input data, and the tests to establish the stability of the optimization model with respect to the scenario generation procedure. In Section 5 we describe the computational experiments involving the introduction of controlled errors in the statistical properties of stochastic input parameters (i.e., domestic asset returns and spot exchange rates) and we present the effects of these errors on the model's results. Finally, Section 6 concludes the paper.

2 Risk Management Model Constraining CVaR

Let the random asset returns $\tilde{\boldsymbol{r}}$ be expressed in terms of a finite set of discrete scenarios $\boldsymbol{S} = \{s : s = 1, 2, \ldots, S\}$, whereby the returns under a particular scenario $s \in \boldsymbol{S}$ take the values $\boldsymbol{r}_s$ with associated probability $p_s > 0$, $\sum_{s=1}^{S} p_s = 1$. A portfolio $\boldsymbol{x}$ has a return $\rho(\boldsymbol{x}, \boldsymbol{r}_s) = \boldsymbol{x}^T \boldsymbol{r}_s$ under scenario $s \in \boldsymbol{S}$; the expected return of the portfolio is $R(\boldsymbol{x}) = \mathbb{E}\left[\tilde{\rho}(\boldsymbol{x}, \tilde{\boldsymbol{r}})\right] = \sum_{s=1}^{S} p_s \rho(\boldsymbol{x}, \boldsymbol{r}_s)$. We need to formulate a mathematical program in order to determine the portfolio $\boldsymbol{x}$ that maximizes the expected portfolio return $R(\boldsymbol{x})$ while controlling the respective risk measure $\texttt{CVaR}(\boldsymbol{x}, \alpha)$.

The definition of CVaR in equation (1) applies to the case of a continuous

distribution $F(\boldsymbol{x}, u)$. When the random asset returns $\tilde{\boldsymbol{r}}$ are modeled in terms of a discrete distribution (i.e., scenarios) the function $F(\boldsymbol{x}, u)$ is discontinuous in u and $\boldsymbol{x}$. A problem particularly arises when $F(\boldsymbol{x}, u)$ exhibits a jump at the critical percentile $u = \texttt{VaR}(\boldsymbol{x}, \alpha)$. An alternative definition of `CVaR` for general distributions (including discrete distributions) is introduced in Rockafellar and Uryasev [16]:

$$\texttt{CVaR}(\boldsymbol{x}, \alpha) = \left(1 - \frac{1}{1-\alpha} \sum_{\{s \in \boldsymbol{S} | \rho(\boldsymbol{x}, \boldsymbol{r}_s) \leq z\}} p_s\right) z \; + \; \frac{1}{1-\alpha} \sum_{\{s \in \boldsymbol{S} | \rho(\boldsymbol{x}, \boldsymbol{r}_s) \leq z\}} p_s \rho(\boldsymbol{x}, \boldsymbol{r}_s)\,, \quad (2)$$

where $z = \texttt{VaR}(\boldsymbol{x}, \alpha)$. As we use scenarios to model the random variables in this paper, we will adopt this alternative definition of `CVaR`. This definition of `CVaR` for discrete distributions may not be exactly equal to the conditional expectation of portfolio returns below $\texttt{VaR}(\boldsymbol{x}, \alpha)$ when $F(\boldsymbol{x}, u)$ has a jump at $\texttt{VaR}(\boldsymbol{x}, \alpha)$. Rockafellar and Uryasev [16] discuss the subtle differences in this case; they also show that the `CVaR` function in (2) is a coherent risk measure in the sense of Artzner et al. [2]. It is also a continuous and convex function in the portfolio positions $\boldsymbol{x}$.

Let us define for every scenario $s \in \boldsymbol{S}$ an auxiliary variable

$$y_s \;=\; \max\left[0,\, z - \rho(\boldsymbol{x}, \boldsymbol{r}_s)\right],$$

which measures the shortfall of returns with respect to $\texttt{VaR}(\boldsymbol{x}, \alpha)$ under the respective scenario. These non-smooth functions can be expressed in terms of the following set of linear inequalities

$$\left\{\, y_s \,:\, y_s \geq 0\,,\; y_s \geq z - \rho(\boldsymbol{x}, \boldsymbol{r}_s)\,,\; s \in \boldsymbol{S} \,\right\}.$$

Using the shortfall variables y_s, Rockafellar and Uryasev [16] (see also Topaloglou et al. [19]) show that, as defined in (2),

$$\texttt{CVaR}(\boldsymbol{x}, \alpha) \;=\; z \;-\; \frac{1}{1-\alpha} \sum_{s=1}^{S} p_s y_s\,. \quad (3)$$

A model that maximizes the portfolio's expected return, while maintaining the respective `CVaR` measure of returns within an allowable limit, can thus be stated as follows:

$$\texttt{maximize} \quad \sum_{s=1}^{S} p_s \, \rho(\boldsymbol{x}, \boldsymbol{r}_s) \tag{4a}$$

$$\texttt{s.t.} \quad \boldsymbol{x} \in \boldsymbol{X}, \; z \in \mathcal{R}\,, \tag{4b}$$

$$z - \frac{1}{1-\alpha} \sum_{s=1}^{S} p_s \, y_s \geq \vartheta\,, \tag{4c}$$

$$y_s \geq z - \rho(\boldsymbol{x}, \boldsymbol{r}_s)\,, \qquad s = 1, \ldots, S \tag{4d}$$

$$y_s \geq 0\,, \qquad s = 1, \ldots, S \tag{4e}$$

The set $\{x \in \boldsymbol{X}\}$ denotes generic constraints on the portfolio composition. The parameter ϑ specifies the minimal allowable `CVaR` value of portfolio returns at the $(1-\alpha)100^{\text{th}}$ percentile. If the `CVaR` constraint (4c) is active (which it usually is), $z = \texttt{VaR}(\boldsymbol{x}, \alpha)$. Hence, we get the value of `VaR` as a by-product of solving the `CVaR` problem.

Scenario-based LP formulations of models that use `CVaR` either in the constraints or in the objective function can be found in Uryasev [20]; a complete theoretical derivation is given in Rockafellar and Uryasev [16].

3 The International Portfolio Management Model

We apply a single-period (2-stage[2]) stochastic program drawn from Topaloglou et al. [19] to model an international portfolio management problem. The problem of portfolio (re)structuring is viewed from the perspective of a US investor who may hold assets denominated in multiple currencies. To reposition his investments from one market (currency) to another, the investor must first convert to USD the proceeds of foreign asset sales in the market in which he reduces his presence and then purchase the foreign currency in which he wishes to increase his investments. The current spot exchange rates of foreign currencies to USD apply in the currency exchange transactions. At the end of the holding period (one month in our case) we compute the scenario-dependent value of each investment using its projected price under the respective scenario. The USD-equivalent value is determined by applying the corresponding estimate of the appropriate spot exchange rate to USD at the end of the period under the same scenario.

[2]There is no general rule on how to count (or define) periods and stages. We define a period to be a time at which portfolio (re)structuring decisions are made; it is also a time interval between two events (stages) at which new information on random variables is revealed. Hence, we have $number\ of\ stages = number\ of\ periods + 1$.

The investor's portfolio is exposed to market risk in the various currencies, as well as to currency exchange risk. To (partly) hedge the currency risk, the investor may enter into currency exchange contracts in the futures market. In our model, the monetary amounts (in USD) of contracts in currency futures are decided in the first stage, so the decisions are implementable. We do not constrain these decisions; thus, it is possible to enter into a currency exchange in futures even if we do not invest in the respective market.

We define the following notation:
User-specified parameters:

α	confidence level for `VaR` and `CVaR`
ϑ	minimal allowable `CVaR` of portfolio returns

Deterministic input data:

$\boldsymbol{M}$	set of markets (synonymously, countries, currencies)
$\ell \in \boldsymbol{M}$	index of investor's base (reference) currency (in our case USD)
$\boldsymbol{M}_f$	set of foreign markets; $\boldsymbol{M}_f = \boldsymbol{M} \setminus \{\ell\}$
$\boldsymbol{I}_m$	set of available investments (asset classes) in market $m \in \boldsymbol{M}$
b_{im}	number of assets in class $i \in \boldsymbol{I}_m$ of market $m \in \boldsymbol{M}$ in initial portf.
P^0_{im}	current price of asset $i \in \boldsymbol{I}_m,\ m \in \boldsymbol{M}$; in units of domestic curr. m
c_m	initially available cash in currency $m \in \boldsymbol{M}$
γ_{im}	transaction cost rate for asset class $i \in \boldsymbol{I}_m,\ m \in \boldsymbol{M}$
λ_m	transaction cost rate for currency $m \in \boldsymbol{M}$
e^0_m	current spot exchange rate of currency $m \in \boldsymbol{M}$
φ_m	futures exchange rate of currency $m \in \boldsymbol{M}$ (i.e., the deterministic rate for a currency exchange to be executed at the end of the planning horizon)
W_0	total value of initial portfolio (i.e., initial wealth), in units of reference currency: $W_0 = \sum_{m \in \boldsymbol{M}} \left(c_m + \sum_{i \in \boldsymbol{I}_m} b_{im} P^0_{im} \right) e^0_m$

Scenario dependent data:

S	number of scenarios
$\boldsymbol{S}$	set of scenarios: $\boldsymbol{S} = \{1, \ldots, S\}$
p_s	probability of scenario $s \in \boldsymbol{S}$ — in our tests, scenarios are equiprobable (i.e., $p_s = 1/S$)
P^s_{im}	price of asset $i \in \boldsymbol{I}_m,\ m \in \boldsymbol{M}$ at the end of the holding period under scenario $s \in \boldsymbol{S}$; in units of domestic currency m
e^s_m	spot exchange rate of currency $m \in \boldsymbol{M}$ at the end of the holding period under scenario $s \in \boldsymbol{S}$

Decision variables:

x_{im}	number of assets $i \in \boldsymbol{I}_m,\ m \in \boldsymbol{M}$ to buy
w_{im}	number of assets $i \in \boldsymbol{I}_m,\ m \in \boldsymbol{M}$ to sell
g_m	base currency used to buy currency $m \in \boldsymbol{M}_f$ in the spot market
q_m	base currency collected from sale of currency $m \in \boldsymbol{M}_f$ in the spot market
f_m	base currency collected from sale of currency $m \in \boldsymbol{M}_f$ in the futures market (i.e., amount of futures contract, in units of the base currency). A negative value indicates a sale of the base currency in the futures market.

Auxiliary variables:

v_m^s	total value of investments in market $m \in \boldsymbol{M}$ under scenario $s \in \boldsymbol{S}$ after settlement of futures currency contracts at the end of the holding period, expressed in units of the respective curr. $m \in \boldsymbol{M}$
y_s	return shortfall below `VaR` under scenario $s \in \boldsymbol{S}$
z	variable in definition of `CVaR` — equals to `VaR` at the optimal sol.

All exchange rates are expressed as the equivalent amount of the base currency for one unit of the foreign currency. Obviously, $e_\ell^0 = e_\ell^s = 1,\ \forall\, s \in \boldsymbol{S}$. Also, we must have $g_\ell = q_\ell = f_\ell = 0$.

We formulate the international portfolio management model as follows:

$$\texttt{maximize} \quad \frac{1}{W_0} \sum_{s \in \boldsymbol{S}} p_s \sum_{m \in \boldsymbol{M}} v_m^s e_m^s - 1 \tag{5a}$$

$$\texttt{s.t.} \quad c_\ell + \sum_{i \in \boldsymbol{I}_\ell} w_{i\ell} P_{i\ell}^0 (1 - \gamma_{i\ell}) + \sum_{m \in \boldsymbol{M}_f} q_m (1 - \lambda_m) = \sum_{i \in \boldsymbol{I}_\ell} x_{i\ell} P_{i\ell}^0 (1 + \gamma_{i\ell}) + \sum_{m \in \boldsymbol{M}_f} g_m (1 + \lambda_m) \tag{5b}$$

$$c_m + \sum_{i \in \boldsymbol{I}_m} w_{im} P_{im}^0 (1 - \gamma_{im}) + \frac{g_m}{e_m^0}(1 - \lambda_m) = \sum_{i \in \boldsymbol{I}_m} x_{im} P_{im}^0 (1 + \gamma_{im}) + \frac{q_m}{e_m^0}(1 + \lambda_m)\,, \quad \forall\, m \in \boldsymbol{M}_f \tag{5c}$$

$$y_s \geq z - \frac{1}{W_0} \left(\sum_{m \in \boldsymbol{M}} v_m^s e_m^s - W_0 \right), \qquad \forall\, s \in \boldsymbol{S} \tag{5d}$$

$$z - \frac{1}{1 - \alpha} \sum_{s \in \boldsymbol{S}} p_s y_s \geq \vartheta \tag{5e}$$

$$0 \leq x_{im}\,, \quad 0 \leq w_{im} \leq b_{im}\,, \quad \forall\, m \in \boldsymbol{M}\,, \forall\, i \in \boldsymbol{I}_m \tag{5f}$$

$$g_m \geq 0\,, \quad q_m \geq 0\,, \qquad \forall\, m \in \boldsymbol{M}_f \tag{5g}$$

$$y_s \geq 0\,, \quad \forall\, s \in \boldsymbol{S} \tag{5h}$$

$$\text{where} \qquad v_m^s = \sum_{i \in \boldsymbol{I}_m} (b_{im} + x_{im} - w_{im})\, P_{im}^s + \begin{cases} \sum_{n \in \boldsymbol{M}_f} f_n & \texttt{if } m = \ell \\ -\frac{f_m}{\varphi_m} & \texttt{if } m \in \boldsymbol{M}_f \end{cases} \tag{5i}$$

This formulation maximizes the expected total return of the international portfolio. Obviously, this is equivalent to maximizing the expected terminal value of the portfolio (in units of the base currency) at the end of the holding period — i.e., by maintaining only the sum in the objective function (5a).

Equations (5b) and (5c) impose the cash balance conditions in every currency; the former for the base currency ℓ and the latter for the foreign currencies $m \in \boldsymbol{M}_f$. These constraints equate the sources and uses of funds in each currency. Note that no holdings in cash are allowed after portfolio restructuring. Hence, we do not need to model the interest rates in each market. The constraints in (5f) disallow short positions in the assets.

Constraints (5d), (5e) and (5h) are the definitional constraints for determining `CVaR`, as they were introduced in the general formulation (4). Constraints (5i) simply define the auxiliary variables v_m^s (scenario-dependent values of holdings in each market at the end of the holding period, after settlement of currency futures contracts). Obviously, these constraints can be eliminated by substituting the expressions for v_m^s directly to (5a) and (5d).

The model takes into account the initial composition of the portfolio and determines optimal rebalancing decisions. Proportional transaction costs are also taken into account. A bid-ask spread of $2\gamma_{im}$ (proportional to the current price P_{im}^0) is imposed for every asset $i \in \boldsymbol{I}_m$, $m \in \boldsymbol{M}$, in the cash balance equations (5b) and (5c). A similar spread is imposed to currency transactions with the parameters λ_m. Note that the model jointly determines in an integrated manner both the asset selection (portfolio composition), as well as the appropriate level of currency hedging for each foreign market $m \in \boldsymbol{M}_f$ — with the variables f_m for exchanges in currency futures.

4 Scenario generation

We used the method of Høyland et al. [6] to generate scenarios; see also Høyland and Wallace [7] for the general method of scenario generation by matching statistical properties to specific targets. The method generates a set of discrete scenarios so that statistical properties of the random variables match specified target values. Specifically, we match the following statistics: the first four moments of the marginal distributions (mean, standard deviation, skewness and kurtosis), as well as the correlation matrix. We estimate

the target values for these statistics on the basis of historical data. However, this is not a prerequisite for the scenario generation procedure. We could, as easily, use subjective estimates for the target statistics, as well as targets determined through alternative estimation means.

We start with data series $\{V_{jt} | \, t = 0, \ldots, T\}$ of observed market values for all random variables in the problem. The index j indicates the data series for a particular variable (e.g., the market price of an asset $i \in \boldsymbol{I}_m$, $m \in \boldsymbol{M}$ (in domestic terms), or the spot exchange rate of a foreign currency $m \in \boldsymbol{M}_f$). The time-step of the data series coincides with the length of the planning horizon in the optimization model (i.e., one month). Our scenario generation procedure then proceeds as follows:

i. Transform each of the initial data series $\{V_{jt} | \, t = 0, \ldots, T\}$ to a corresponding series of differentials: $\{{}^{V_{jt}}\!/_{V_{j,t-1}} - 1 | \, t = 1, \ldots, T\}$.

ii. Compute the required target statistics — i.e., the first four marginal moments and the correlation matrix — of the transformed series.

iii. Decide the required number of scenarios S to be generated. The scenario generation method additionally needs as inputs the scenario probabilities. We have made all scenarios equiprobable by setting $p_s = {}^1\!/_{\mathrm{S}}\,, \; \forall \, s \in \boldsymbol{S}$.

iv. Call the scenario generation routine with the target statistics and the scenario probabilities as input. The output is a discrete distribution (set of scenarios) of the differentials for all the random variables whose statistical properties match the specified target values.

v. Set the current values of the asset prices P^0_{im} and the spot exchange rates e^0_m equal to the last values in the respective initial data series. This simulates the situation of an investor who makes a decision based on data up to the present day.

vi. Compute the scenario-dependent asset prices P^s_{im} and spot currency exchange rates e^s_m at the end of the planning horizon based on their current values and the scenario outcomes for the respective differentials.

vii. Set the current values of the futures exchange rates equal to the expected values of the spot exchange rates at the same term (i.e., $\varphi_m = \sum_{s=1}^{S} p_s e^s_m\,, \; \forall m \in \boldsymbol{M}_f$). In this way we ensure that there is no arbitrage.[3]

[3] Moreover, these can be regarded as "fair values" for the futures exchange rates.

4.1 Data

Our test problem considers investments in four markets: United States, United Kingdom, Germany and Japan. The available asset classes in each of these markets include a stock index, denoted as `Stk`, and bond indices of short-term (1–3 years) and long-term (7–10 years) maturity bands, denoted `Bnd1` and `Bnd7`, respectively. Thus, a total of 12 assets are considered in each portfolio. The problem is viewed from the perspective of a US investor; hence, data for the exchange rates of the three foreign currencies to USD are also needed.

The data for the stock indices were obtained from the Morgan Stanley Capital International, Inc. database (`www.mscidata.com`). The data for the bond indices and the currency exchange rates were collected from `DataStream`. All time series have a monthly time-step and cover the period from January 1990 to April 2001 (i.e., a total of 136 monthly observations). The statistical properties of these data series — used as target values in the scenario generation procedure — are reported in Tables 3 and 4 in the Appendix.

The inputs for the portfolio optimization model were set to:

$\alpha = 0.95$	confidence level for `VaR` and `CVaR`
$\vartheta = -0.01$	lower bound on `CVaR` of portfolio return (monthly)
$\boldsymbol{M} = \{\texttt{USA}, \texttt{UK}, \texttt{Ger}, \texttt{Jap}\}$	set of markets (countries, currencies)
$\boldsymbol{I}_m = \{\texttt{Stk}, \texttt{Bnd1}, \texttt{Bnd7}\},\ \forall m \in \boldsymbol{M}$	set of asset classes in each market
$\ell = \texttt{USA}$	base country (reference currency)
$b_{im} = 0,\ \forall m \in \boldsymbol{M},\ \forall i \in \boldsymbol{I}_m$	no initial holdings in any asset
$c_\ell = 100$	initial cash amount in the base currency (USD)
$c_m = 0,\ \forall m \in \boldsymbol{M}_f$	no initial cash in foreign currencies
$\gamma_{\texttt{Stk},m} = 0.001,\ \forall m \in \boldsymbol{M}$	transaction cost rate for stocks
$\gamma_{\texttt{Bnd1},m} = \gamma_{\texttt{Bnd7},m} = 0.0005,\ \forall m \in \boldsymbol{M}$	transaction cost rate for bonds
$\lambda_m = 0.0001,\ \forall m \in \boldsymbol{M}_f$	transaction cost rate for currencies

4.2 Assessment of the scenario generation method

In Section 5, we investigate the behavior of model (5) with respect to the number of scenarios and with respect to mis-specifications in the statistical properties of stochastic inputs. To validate the results, however, we must first show that the scenario generation method used in the tests is "neutral"; that is, it does not unduly influence the results by causing instability. Such tests should, in general, be conducted for every application of a stochastic program

that relies on a specific scenario generation procedure to depict uncertainty in model parameters.

Ideally, we would like to determine whether the scenario generation can effectively represent the "real world", but that is not an attainable goal. As a representation of the "real world", we take a large scenario set — that we refer to as the "benchmark scenario set". *It is important that the benchmark set is provided exogenously, that is, it is not generated by the same method which we want to test.* In our case, the benchmark scenario set (tree) was generated by a method based on principal component analysis described in Topaloglou et al. [19]. The benchmark tree has $15,000$ scenarios to jointly depict the co-variation of the 15 random variables in our international portfolio management problem. We note that the scenarios of the benchmark tree are not equiprobable.

Based on the benchmark tree, we compute the moments and correlations of the differentials of the random variables. The values of these statistical properties constitute the targets to match with our scenario generation procedure. We then proceed from Step iii. of the scenario generation procedure described at the beginning of Section 4.

First, we verify that the scenario generation procedure produces scenario sets in which the statistical properties of the random variables match the target values. Moreover, we check that the generated scenario sets reproduce other distributional characteristics of the random variables.

Match of the marginal distribution functions

The easiest to check is a match of the marginal distributions. We generated several scenario sets ranging in size from 10 to $5,000$ scenarios. For each set we determined the marginal distributions of the random variables and compared them to the corresponding distributions from the benchmark tree. The comparison of the whole distribution function of monthly returns of the US stock index (`StkUSA`) is depicted in Figure 1, while Figure 2 shows the lower tail in detail. The reproduction of the marginal distributions of the remaining random variables is quite similar.

We observe that even with moderate-size scenario trees (> 250 scenarios) we can closely reproduce the marginal distributions from the 1st to the 99th percentile. At the extreme tails the distribution is not as accurately matched unless a sufficiently large number of scenarios is generated. This is understandable, since we should expect to get more samples in the tails as the number of scenarios increases. In fact, if we generate larger scenario sets than the benchmark tree (i.e., with $> 15,000$ scenarios) then we get more scenarios in

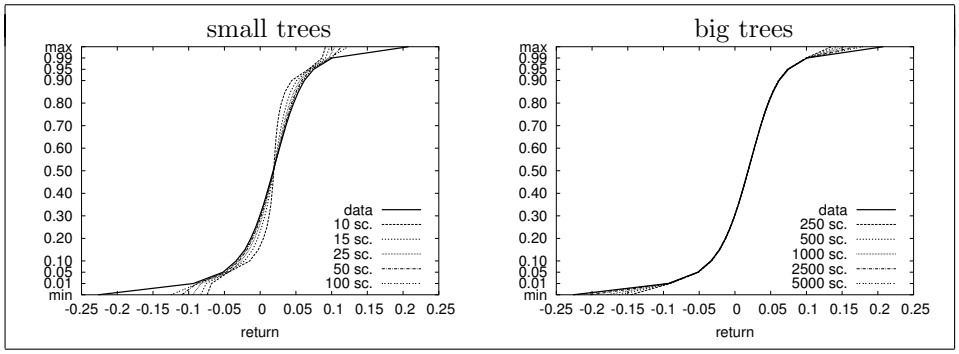

Figure 1: Match of the distribution function for the US stock index
Comparison of the distribution function for the monthly returns of the US stock index
in the benchmark tree and the generated trees with different number of scenarios.
Note that the scale of the vertical axis is not linear, the two outer intervals are prolonged.

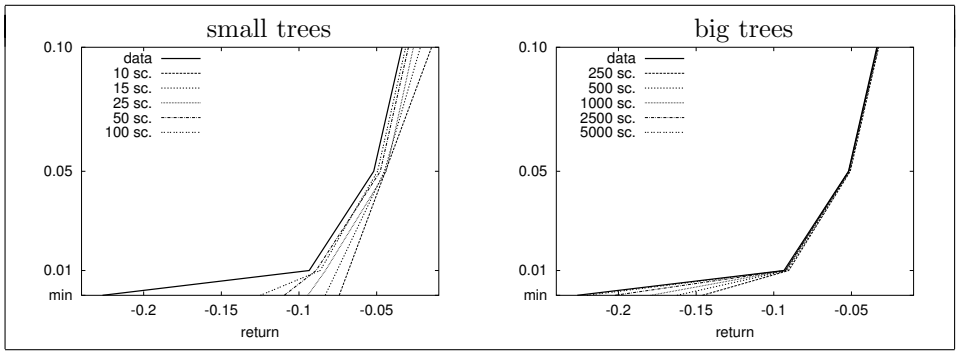

Figure 2: Detail from Figure 1 — match of the lower tail

the extreme tails than we have in the benchmark tree.

The desired degree of matching the distributions depends on the decision model in which the scenarios will be used. For example, if the optimization model applies the mean-variance paradigm then the accuracy of match at the tails will not make any difference, as long as the first two marginal moments and the correlations are matched. A rather close match of the tails becomes relevant for the `CVaR` model which is concerned with the lower tail of the portfolio's return distribution.

The test on the match of the marginal distributions of the random variables is indeed illustrative. Yet, it is not sufficient, even though we know that

the generated scenario sets also match the desired correlations as well. We aim to establish that the portfolio optimization model produces stable results regardless of the specific scenario set generated in any given run — i.e., that it is stable with respect to sample. Evidently, representative scenario sets of sufficiently large size are needed to ensure such stability. Moreover, we need to test jointly the scenario generation method and the optimization model in order to assess whether the scenario generation method is effective and "neutral" for the `CVaR` model, in the sense that it does not cause instability in its solutions.

Joint stability test of scenario generation and the `CVaR` model

Here we examine the stability of the results of the `CVaR` model. We generate 25 scenario trees with a given number of scenarios S, each matching the moments and correlations of the benchmark tree. We solve the optimization model with each tree and save both the expected portfolio return and the value of `CVaR` at the respective optimal solution.[4] As stochastic programs tend to have multiple optimal or near-optimal solutions, we study the stability in terms of the optimal objective function value; we do not compare the optimal decisions (portfolio compositions). We then simulate all the solutions on the benchmark tree and record the "true" values of both the expected return and `CVaR`. The confidence level in all tests is $\alpha = 0.95$; thus, `CVaR` is the expected return for the 5% worst scenarios.

We investigate two types of stability of the scenario sets:

In-sample stability: The solutions of the model should not depend on the specific scenario tree used, as long as it is representative. Hence, the optimal values should ideally be equal for all the 25 scenario trees of a given size.

Out-of-sample stability: The true expected portfolio returns and the true values of `CVaR` should ideally be equal for all the scenario trees. In addition, they should be equal to the in-sample values.

The two notions of stability are not equivalent. We can have in-sample stability without out-of-sample stability. Consider, for example, a case in which all the scenario trees are identical but incorrect in comparison to the benchmark. On the other hand, we can have alternative scenario trees for which the `CVaR` model yields the same optimal portfolio. Then the in-sample

[4] Since we constrain `CVaR` and maximize expected return, the `CVaR` is always at its minimal value ($\vartheta = -1\%$) at the optimal solution.

objective values would differ for different scenario trees, but the "true" out-of-sample objective values would be exactly equal.

In practice we can not test precisely the out-of-sample stability as we do not have the true distribution of the random variables. We may only employ a benchmark distribution, as we do in this study. Verifying out-of-sample stability in terms of a representative benchmark provides an indication that the model should also be effective in a practical setting. Hence, the purpose of our tests is to determine whether we can achieve both types of stability, although the benchmark tree was generated by a different method.

Results of the tests are depicted in Figure 3. We see that as we increase the number of scenarios we can indeed achieve both in-sample and out-of-sample stability. Thus, the scenario generation method is effective and "neutral", in the sense that it does not cause any instability in the solutions of the `CVaR` model.

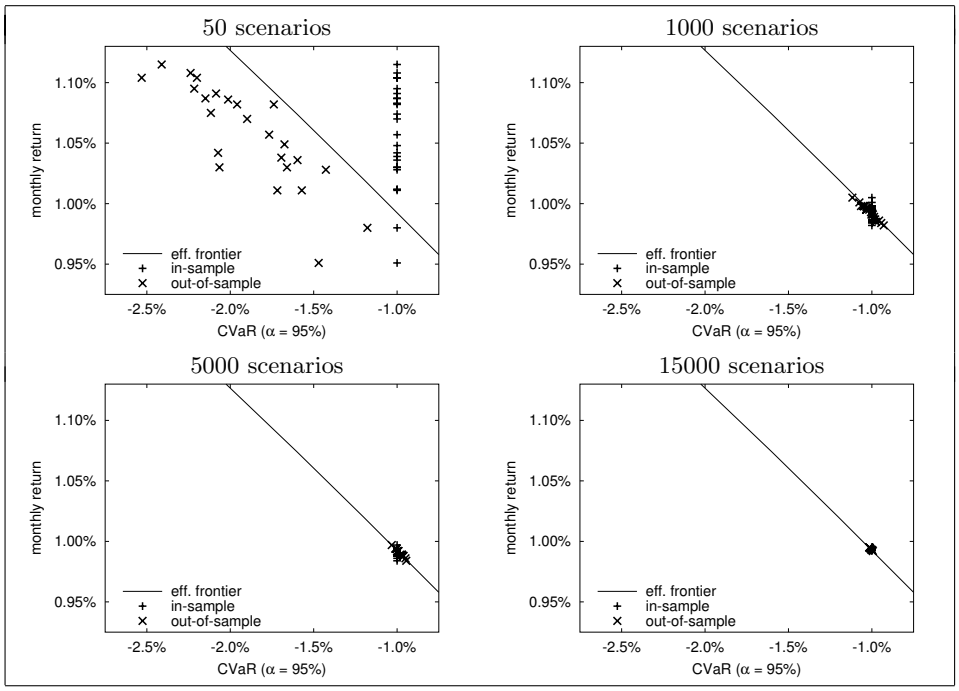

Figure 3: Stability of the `CVaR` model with respect to the exogenous benchmark tree

5 Sensitivity tests of the CVaR model

This section studies the sensitivity of the CVaR model with respect to mis-specifications in the statistical properties of stochastic parameters. Again we need a benchmark as a reference. Here we calibrate the scenario generation method using the statistical properties estimated from historical market data, as reported in Tables 3 and 4. All scenario sets mentioned in the following tests are generated so as to match these target statistics. First, we generate a benchmark tree with 20,000 scenarios.[5] This scenario set is sufficiently large to ensure both in-sample and out-of-sample stability of the solutions, while it is still practically solvable so as to trace the "reference" efficient frontier depicting the tradeoff between expected portfolio return and the CVaR risk measure. The frontier is obtained by repeatedly solving the parametric optimization model for different allowable limits ϑ on CVaR.

To interpret the results of the tests, we must understand the source of possible differences between the in-sample and out-of-sample expected portfolio return of a given portfolio. In the absence of foreign assets, the portfolio return would depend only on the portfolio composition and the means (expected values) of the asset returns. The contribution of a foreign asset, however, on the portfolio's expected return depends on the product of the mean asset return (in its domestic currency) and the expected change of the exchange rate to the reference currency. Since $\mathbb{E}\big[\tilde{X}\tilde{Y}\big] = \mathbb{E}\big[\tilde{X}\big]\mathbb{E}\big[\tilde{Y}\big] - Cov(\tilde{X}, \tilde{Y})$, the expected portfolio return depends not only on the means of the random variables, but also on their covariances. Thus, for a given portfolio, the in-sample and the out-of-sample expected portfolio returns would be equal only if the random variables have the same means, standard deviations, and correlations in the respective scenario sets (i.e., the test tree and the benchmark tree). This condition is satisfied by construction in our scenario generation method as the random variables have matching moments and correlations in the benchmark and in the test trees. Hence, a portfolio has the same in-sample and out-of-sample expected return, but its CVaR value is different when it is simulated on the benchmark scenario set in comparison to its value on a test tree.

5.1 Finding a minimal number of scenarios

Before we test the sensitivity of the CVaR model to mis-specifications in the statistical properties of its stochastic inputs, we must ensure that we employ

[5] Having established with the tests in Section 4.2 that our scenario generation method is indeed "neutral" and "unbiased" (i.e., can effectively reproduce the characteristics of a desired distribution and lead to stable solutions of the portfolio management model) we can employ it to establish a representative benchmark.

representative and sufficiently large scenario sets in our tests. That is, we must ensure that any variation in the model's results stems from induced changes in the statistical properties of the stochastic inputs and not from insufficiency of the scenario test sets.

The results of Section 4.2 indicate that at least $5,000$ scenarios are needed to attain both in-sample and out-of-sample stability. We repeat the same tests here, with the only difference that the target statistics in the scenario generation procedure, for both the benchmark as well as the test trees, are estimated from the historical time series. Again, we use 25 scenario trees for every tested number of scenarios S, and we test for both in-sample and out-of-sample stability.

The results of the tests are summarized in Figure 4. Again we observe that the required number of scenarios to ensure adequate stability of the `CVaR` model is rather high (i.e., $> 5,000$ scenarios).[6] This is not surprising. Since we have equiprobable scenarios, and work with `CVaR` at a confidence level of 5%, with 100 scenarios the `CVaR` is an average of only 5 values. This can hardly be expected to be stable. Hence, much larger scenario sets are needed to reliably depict the tail of the portfolio's return distribution.
Some comments on the figures:

- The in-sample results always lie on a vertical line, as the `CVaR` value is always equal to its minimal allowable limit $\vartheta = -1\%$ at the optimal solution. The range of this line indicates the in-sample instability of the model with respect to scenario sets of a given size.
- Since the in-sample values are not the "true" estimates of the portfolio's `CVaR`, they can cross the "reference" efficient frontier that is generated using the benchmark scenario tree.
- Because the random variables have the same moments and correlations in the test trees and in the benchmark tree, the in-sample and the out-of-sample expected portfolio returns are the same for a given portfolio — see Section 5 for explanation. Only the `CVaR` values of a portfolio change when it is simulated on the benchmark scenario set.

Table 1 presents sample ranges of the expected portfolio returns for tests using scenario sets of increasing size. The values are annualized for easier

[6] This is about two orders of magnitude higher than we need to achieve stability in the results of a comparable mean-variance model for the same problem. However, a mean-variance model would be inappropriate for this problem as the random variables are not normally distributed (see, the Appendix). Moreover, in the case of a mean-variance model we could directly input the variance-covariance matrix, without any need to model the co-variation of the random variables by means of scenarios.

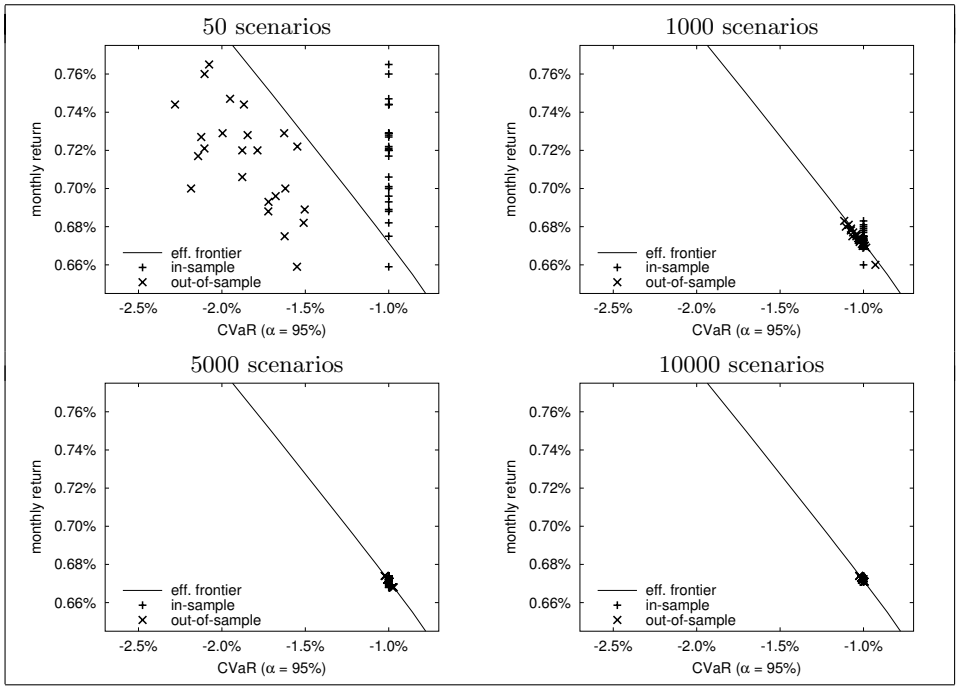

Figure 4: Stability of CVaR model with respect to the number of scenarios

interpretation. Since in-sample and out-of-sample expected portfolio returns are the same, their range is also the same. Note that this sample range — proxy of instability measure for the model — monotonicaly decreases with increasing number of scenarios. This is a helpful consistency check. As a compromise between accuracy in stability and computational speed, we decided to use trees with $5,000$ scenarios in the tests of the next section.

Table 1: Sample ranges (max − min) of the expected annualized portfolio returns

# of scenarios	50	100	250	500	1000	2500	5000	10000
variation	1.46%	0.76%	0.41%	0.30%	0.28%	0.11%	0.07%	0.04%

5.2 Effects of mis-specifications in statistical properties

In this section we test the sensitivity of the CVaR model (5) with respect to mis-specifications in the statistical properties — including higher moments —

of the stochastic inputs (domestic asset returns and currency exchange rates). Controlled errors are systematically introduced to the target statistics (first four moments and correlations) of the random variables at the scenario generation phase. Multiple scenario test trees are then generated to match the perturbed statistical properties. Similar tests for the mean-variance model are presented in Kallberg and Ziemba [9] and Chopra and Ziemba [4].

We quantify the induced errors by means of the following approach. We compute the moments and correlations of the random variables based on subsets of our data set, using a moving time window of size 50% of the available time series. Thus, we obtain a series of plausible estimates for the moments and correlations of the random variables. For each moment/correlation, we take the interval from the minimal to the maximal estimated value from the respective series and call it a *feasible interval* for the corresponding moment/correlation. These feasible intervals for moments and correlations are reported in Tables 5 and 6 in the Appendix. In addition, we define the value of the respective moment/correlation, calculated on the basis of the entire data set, the *true value*.

We define a δ-percent error in a statistical property as

$$true\ value + \varepsilon\, \tfrac{\delta}{100}\ \texttt{length}(feasible\ interval), \quad \varepsilon \in \mathcal{U}(-1,1), \tag{6}$$

where $\mathcal{U}(-1,1)$ is a random number from the uniform distribution on the interval $[-1,1]$. With this definition, the average absolute error is

$$\tfrac{1}{2}\tfrac{\delta}{100}\ \texttt{length}(feasible\ interval).$$

Note that this is different from the corresponding definition in Chopra and Ziemba [4]. There, the δ-percent error was defined as

$$true\_value\left(1 + \varepsilon\, \tfrac{\delta}{100}\right),\ \varepsilon \in \mathcal{N}(0,1).$$

This definition is more natural, but not as suitable for our situation. If we have a statistical property (e.g., skewness or correlation) with a *true value* equal to zero, then this property will never be changed if we introduce errors using the approach of Chopra and Ziemba.

A potential problem when introducing random errors to statistical properties is that we may specify a property, or a combination of properties, that is not feasible. For example, we may end up with specifications that may violate the condition $kurt_i > 1 + skew_i^2$, or we may specify a correlation matrix that is not positive definite. When this happens, we simply discard these particular specifications and create new ones.

In all tests we use scenario trees with 5,000 scenarios. As we showed in the previous section, this size of scenario sets is sufficient to achieve both in-sample and out-of-sample stability in the results of the CVaR model. On the other hand, it is still small enough to allow the execution of multiple tests with a reasonable number of alternative scenario trees within acceptable computational time. To test the effect of errors in the marginal moments of the inputs we used 100 trees for every test. To test the impact of mis-specifications in correlations of the random variables we increased the number of tests to 250.

The procedure for every test is summarized as follows:

i. Decide the size of the relative error δ.

ii. Decide the statistical property to test (i.e, means, standard deviations, skewness, kurtosis, correlations).

For every test do:

iii. For each random variable, perturb the selected property based on (6).

iv. Generate a scenario set, matching the perturbed statistical properties.

v. Solve the portfolio optimization model and store the expected portfolio return and the value of CVaR at the optimal solution.

vi. Simulate the solution on the benchmark tree — which was generated with unperturbed statistics, — again storing the expected portfolio return and the value of CVaR.

Results of the sensitivity tests

We ran the tests for 10% and 25% errors (i.e., for $\delta = 0.1$ and $\delta = 0.25$). In the case of errors in moments, we never obtained an infeasible specification of moments. In the case of 10% error in correlations, there were very few (less than 10) cases where the resulting correlation matrices were not positive definite. In the case of 25% error in correlations, however, most of the specified correlation matrices were not positive definite, and had to be discarded. Thus, the results in this case are somewhat biased; they, in fact, correspond to a smaller error.[7] In this case, we also ended up with a sample size of 100 test trees, instead of the 250 initially planned.

The results for 10% error in the statistical properties of the random variables are summarized in Figure 5, while the results for 25% error are shown in

[7] Most of the rejected cases resulted from the introduction of the larger errors, i.e. $\varepsilon \approx \pm 1$. So, in this case, the actual distribution of ε would not be $\mathcal{U}(-1, 1)$, but it would be more concentrated around zero. Thus, the average absolute error would be smaller.

Figure 6. Table 2 reports the sample ranges of expected (annualized) portfolio returns when errors were introduced to the respective statistics. In this table, we distinguish between in-sample and out-of-sample results only in the case of errors in the means of the random variables. For errors in the remaining statistical properties, the difference between the in-sample and the out-of-sample results was at most 0.01%, so we report only the latter.

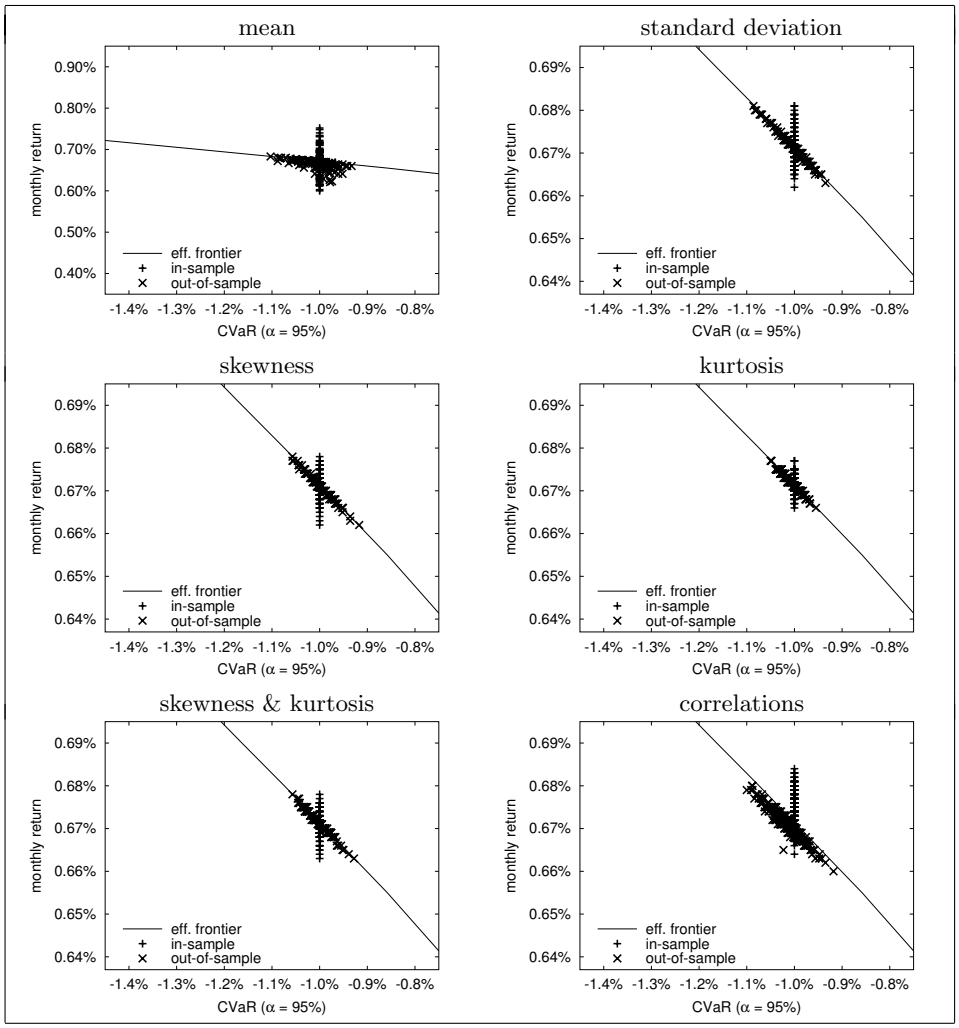

Figure 5: Stability of `CVaR` model — 10% error in inputs
Note that the graph for errors in means has a bigger scale on the y-axis.

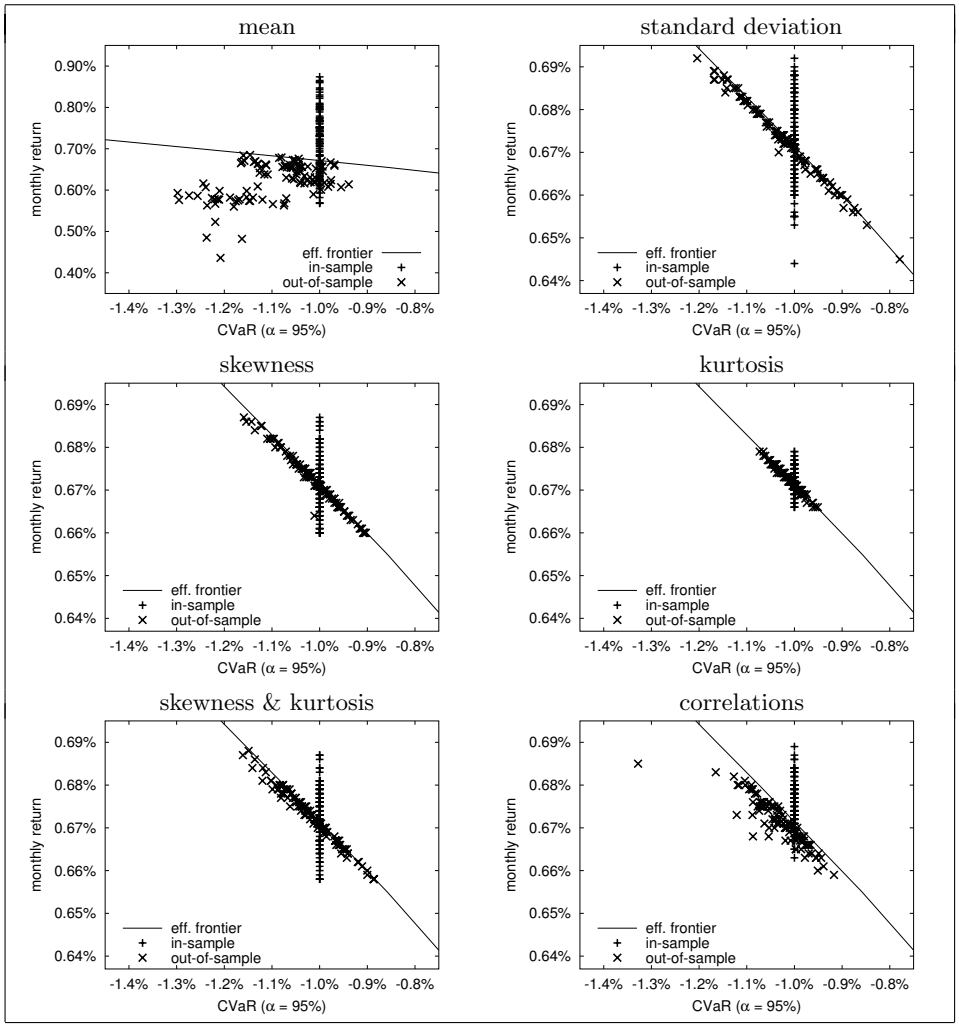

Figure 6: Stability of `CVaR` model — 25% error in inputs
Note that the graph for errors in means has a bigger scale on the y-axis.

Table 2: Sample ranges (`max` − `min`) of the expected annualized portfolio returns

error	mean		stdev	skew	kurt	corr
	in-sample	out-of-sample				
10%	1.84%	0.75%	0.22%	0.19%	0.13%	0.24%
25%	3.73%	3.03%	0.57%	0.32%	0.16%	0.31%

We summarize our observations on the results as follows:

- In all tests, the `CVaR` model is most sensitive to errors in the means of the stochastic inputs. As expected, the mean proves to be by far the most important statistical property to estimate accurately, as errors in the means of the random variables have the most significant impact on the results — almost an order of magnitude higher than the effect of errors in any other statistical property. This finding is consistent with the results of Chopra and Ziemba [4].
- After errors in the means, the `CVaR` model exhibits approximately equal sensitivity — measured by the variation of expected portfolio return — to mis-specifications in the standard deviation, skewness and correlations of the random variables. Hence, estimating correctly these statistics seems to be of approximately equal importance, in terms of their effect on the results of the `CVaR` model.
- Errors in the values of kurtosis have a detectable and non-negligible influence on the results of the `CVaR` model. The effect of errors in the values of kurtosis is about one half of the effect of errors in standard deviations.
- The mean and the correlations are the only properties whose mis-specifications result in portfolios that deviate significantly from the efficient frontier. Errors — especially when they are relatively small — in the other properties seem to result in portfolios with different `CVaR` values, which are, however, still close to the efficient frontier.

Our results demonstrate that errors in higher-order moments of stochastic inputs do indeed play a role in portfolio management models that use the `CVaR` risk measure. Moreover, they quantify the relative effects of errors in these statistics on the model's results. Our results imply that it pays to devote care and effort so as to accurately estimate the values of higher-order moments when employing risk measures concerned with the tails of the return distribution in the context of portfolio management models.

When assessing the potential effects of errors in statistical properties, we should have a sense of the expected magnitude of such errors in practice. Admittedly, errors of a specific magnitude in the estimation of the means or the standard deviations of the random variables may be less likely than comparable errors in the higher moments. This is because the importance of the first two moments is well understood. Much more care is exercised in obtaining reliable estimates of these critical statistics, and more effective tools are available for their estimation, in comparison to higher-order moments. This

is partly because the estimation and verification of higher-order moments is typically more difficult. Higher-order moments are often neglected in portfolio management models as their potential impact may not be as well understood and appreciated. We hope that this study sheds some light in this respect, by indicating the relative importance of accurate estimates of higher-order moments for random variables in risk management models that employ risk measures tailored to control the tails of the portfolio's return distribution.

6 Conclusions

We tested a risk management model for international portfolios based on the `CVaR` risk metric. We employed a scenario generation procedure based on principles of moment matching. We showed that this scenario generation method is effective and "unbiased", in the sense that it can closely reproduce the characteristics of a desired distribution and it leads to stable solutions of the portfolio optimization model.

We determined the required size of a scenario set to achieve stable results in the `CVaR` model; the required number of scenarios is rather large — about $5,000$ in our test problem that involves 15 random variables (3 assets in each of 4 countries plus 3 exchange rates). Evidently, this result depends on the scenario generation method used. It may be possible to devise a method that would decrease this requirement, for example, by selectively sampling scenarios in the extreme tails. However, the identification of relevant extreme scenarios can be quite difficult in a multi-dimensional space. Moreover, the tail of a portfolio's return distribution clearly depends on the portfolio composition — i.e., on the solution of the portfolio optimization model.

We investigated the sensitivity of the `CVaR` model with respect to errors introduced to the statistical properties of stochastic inputs, as represented in discrete scenario sets. The statistical properties investigated included the first four marginal moments and the correlations of the random variables (domestic assets returns and spot currency exchange rates). The results of our tests confirm that the mean value of the random variables is the most important statistic to accurately estimate; the `CVaR` model exhibits high sensitivity to mis-specifications of the means. But, unlike the mean-variance model, the `CVaR` model shows sensitivity to errors in the estimates of higher-order moments as well. Errors in the standard deviation, skewness, and correlations of the random variables have considerable — and approximately equal — impact on the model's results. Errors in the values of kurtosis have lesser, yet non-negligible, effect.

Our overall conclusion is that accurate estimates of higher-order moments

of random variables are important in risk management models for investment portfolios. We also observe that sufficiently large scenario sets are needed to properly capture the risks associated with the tails of the distribution. We hope that the results of this study will motivate further development of effective tools for modeling extreme events in financial risk management models.

References

[1] C. Acerbi and D. Tasche. On the coherence of expected shortfall. *Journal of Banking and Finance*, 27(6):1487–1503, 2002.

[2] P. Artzner, F. Delbaen, J.M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[3] J. Berkowitz and J. O'Brien. How accurate are value-at-risk models at commercial banks? *Journal of Finance*, 57(3):1093–1111, 2002.

[4] V. Chopra and W.T. Ziemba. The effects of errors in means, variances, and covariances on optimal portfolio choice. *The Journal of Portfolio Management*, Winter:6–11, 1993.

[5] R. Frey and A. McNeil. VaR and expected shortfall in portfolios of dependent credit risk: Conceptual and practical insights. *Journal of Banking and Finance*, 27(6):1317–1334, 2002.

[6] K. Høyland, M. Kaut, and S.W. Wallace. A heuristic for moment matching scenario generation. *Computational Optimization and Applications*, to appear.

[7] K. Høyland and S.W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.

[8] P. Jorion. *Value at Risk: The New Benchmark for Managing Financial Risk.* McGraw-Hill, New York, 2001.

[9] J.G. Kallberg and W.T. Ziemba. Mis-specifications in portfolio selection problems. In G. Bamberg and K. Spremann, editors, *Risk and Capital*, pages 74–87. Springer Verlag, New York, 1984.

[10] A. Lucas. Evaluating the Basle guidelines for backtesting of banks' internal risk management models. *Journal of Money, Credit and Banking*, 33(3):826–846, 2001.

[11] Basle Committee on Banking Supervision. International convergence of capital measurements and capital standards. BIS, July 1988.

[12] Basle Committee on Banking Supervision. Supervisory framework for the use of "Backtesting" in conjunction with the internal models approach to market risk capital requirements. BIS, Report No. 22, 1996.

[13] G. Ch. Pflug. Some remarks on the value-at-risk and the conditional value-at-risk. In S. Uryasev, editor, *Probabilistic Constrained Optimization: Methodology and Applications*, pages 272–281. Kluwer Academic Publishers, 2001.

[14] RiskMetrics. Technical document, 4th edition. J.P. Morgan Inc., December 1996.

[15] R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000.

[16] R.T. Rockafellar and S. Uryasev. Conditional value-at-risk for general distributions. *Journal of Banking and Finance*, 26(7):1443–1471, 2002.

[17] G. Szegö. Measures of risk. *Journal of Banking and Finance*, 26(7):1253–1272, 2002.

[18] D. Tasche. Expected shortfall and beyond. *Journal of Banking and Finance*, 27(6):1253–1272, 2002.

[19] N. Topaloglou, H. Vladimirou, and S.A. Zenios. CVaR models with selective hedging for international asset allocation. *Journal of Banking and Finance*, 26(7):1535–1561, 2002.

[20] S. Uryasev. Conditional value-at-risk: Optimization algorithms and applications. *Financial Engineering News*, 14:1–5, February 2000.

Appendix: Properties of the data

Tables 3 and 4 present the first four moments and the correlation matrix of the differentials in the historical market data of the random variables (domestic returns of the stock and bond indices, as well as of the spot currency exchange rates). These statistics constitute the targets matched in the scenario sets used in the empirical tests of Section 5. Note that the random variables have skewness ranging from -1.00 to 1.36 and kurtosis ranging from 2.78 to 7.39. Clearly, the historical observations indicate that the random variables in the international portfolio management problem are not normally distributed — Jarge-Berra tests reject the normality hypothesis for these data. This was a primary motivation for our entire study. That is,

1. To apply the `CVaR` risk measure that is concerned with skewness and the tails of the portfolio's return distribution.

2. To investigate the effects of mis-specifications in the higher moments of the random variables on the model's results.

3. To employ a scenario generation method based on principles of moment matching.

Table 3: Moments of the differentials of the historical market data

	StkUSA	StkUK	StkGer	StkJap	Bnd1USA	Bnd7USA	Bnd1UK
Mean	0.01296	0.01047	0.01057	-0.00189	0.00553	0.00702	0.00718
StDev	0.04101	0.04150	0.05796	0.06184	0.00467	0.01620	0.00688
Skew	-0.47903	-0.19051	-0.47281	0.04768	-0.18341	-0.07482	1.36036
Kurt	3.76519	3.11399	4.11970	3.62119	2.77801	3.23974	7.38764

Bnd7UK	Bnd1Ger	Bnd7Ger	Bnd1Jap	Bnd7Jap	ExRUK	ExRGer	ExRJap
0.00894	0.00535	0.00671	0.00318	0.00622	-0.00077	-0.00152	0.00179
0.01884	0.00455	0.01368	0.00506	0.01681	0.02801	0.03021	0.03607
0.12127	0.55214	-0.87820	0.54803	-0.53562	-0.99772	-0.25505	1.09286
3.52858	5.13927	4.42483	4.28775	5.23964	6.51592	3.80887	6.75996

Tables 5 and 6 present the lengths of the *feasible intervals* for the moments and correlations. The *feasible intervals* are defined and used in the model sensitivity tests in Section 5.2.

Table 4: Correlations of the differentials of the historical market data

	StkUSA	StkUK	StkGer	StkJap	Bnd1USA	Bnd7USA	Bnd1UK
StkUK	0.6651						
StkGer	0.5573	0.5911					
StkJap	0.3568	0.3601	0.3429				
Bnd1USA	0.1965	0.0844	-0.0578	-0.0105			
Bnd7USA	0.2656	0.1150	0.0027	0.0205	0.8768		
Bnd1UK	0.0853	0.4014	0.0276	0.0018	0.3600	0.3176	
Bnd7UK	0.2258	0.5075	0.1714	0.0392	0.4314	0.4815	0.8175
Bnd1Ger	0.0556	0.2642	0.0536	0.0081	0.3466	0.3574	0.6121
Bnd7Ger	0.1687	0.3066	0.2326	0.0408	0.4385	0.5453	0.4639
Bnd1Jap	0.0557	0.0814	-0.0005	0.0226	0.2513	0.2186	0.3274
Bnd7Jap	0.0463	0.0493	0.0140	-0.0029	0.2831	0.3235	0.1815
ExRUK	0.0247	-0.2177	-0.1062	0.1162	0.2422	0.1911	-0.2811
ExRGer	-0.0643	-0.2263	-0.2651	-0.0828	0.2716	0.2129	-0.1429
ExRJap	0.1126	0.0945	-0.1414	0.0475	0.1319	0.0975	0.0927

	Bnd7UK	Bnd1Ger	Bnd7Ger	Bnd1Jap	Bnd7Jap	ExRUK	ExRGer
Bnd1Ger	0.5688						
Bnd7Ger	0.6627	0.7779					
Bnd1Jap	0.2645	0.4008	0.2853				
Bnd7Jap	0.2100	0.3025	0.3093	0.7827			
ExRUK	-0.1588	-0.2227	-0.0948	0.0145	0.0262		
ExRGer	-0.1332	-0.0638	-0.0598	0.1021	0.1375	0.6949	
ExRJap	0.0680	0.0825	-0.0072	0.0334	-0.0122	0.2680	0.4236

Table 5: Lengths of the *feasible intervals* of moments

	StkUSA	StkUK	StkGer	StkJap	Bnd1USA	Bnd7USA	Bnd1UK
Mean	0.00946	0.00764	0.01836	0.01055	0.00166	0.00435	0.00351
StDev	0.02035	0.01217	0.02153	0.02321	0.00171	0.00293	0.00485
Skew	1.62256	0.8916	1.0513	0.54337	0.79795	0.62041	2.32511
Kurt	3.87492	1.82066	2.56178	0.99143	1.2739	0.891	7.2035

Bnd7UK	Bnd1Ger	Bnd7Ger	Bnd1Jap	Bnd7Jap	ExRUK	ExRGer	ExRJap
0.00591	0.00366	0.00284	0.00433	0.00487	0.0057	0.0107	0.01108
0.00952	0.00174	0.00262	0.00249	0.00351	0.01533	0.00954	0.0132
0.78578	1.18387	0.55199	1.21049	1.48868	1.62235	1.22593	1.55678
1.04044	3.12805	2.33646	2.18066	5.13353	6.35287	2.086	4.0713

Table 6: Lengths of the *feasible intervals* of correlations

	StkUSA	StkUK	StkGer	StkJap	Bnd1USA	Bnd7USA	Bnd1UK
StkUK	0.28245						
StkGer	0.41613	0.1814					
StkJap	0.33692	0.27907	0.38272				
Bnd1USA	0.48044	0.43639	0.3613	0.21907			
Bnd7USA	0.43832	0.44669	0.37637	0.28547	0.07029		
Bnd1UK	0.25762	0.63827	0.47922	0.29948	0.32842	0.3272	
Bnd7UK	0.18449	0.53845	0.3226	0.18788	0.30047	0.29125	0.10638
Bnd1Ger	0.32837	0.53526	0.44861	0.20951	0.23945	0.21754	0.23587
Bnd7Ger	0.28289	0.4632	0.52714	0.18048	0.25918	0.25712	0.1254
Bnd1Jap	0.39658	0.30524	0.38022	0.47467	0.35529	0.34925	0.42718
Bnd7Jap	0.53725	0.42657	0.43082	0.53627	0.1201	0.19085	0.34086
ExRUK	0.32612	0.2294	0.23331	0.34925	0.37893	0.34466	0.48376
ExRGer	0.34274	0.23943	0.30395	0.41129	0.4128	0.42665	0.49304
ExRJap	0.35474	0.3389	0.22861	0.29166	0.23326	0.21832	0.19624

	Bnd7UK	Bnd1Ger	Bnd7Ger	Bnd1Jap	Bnd7Jap	ExRUK	ExRGer
Bnd1Ger	0.18615						
Bnd7Ger	0.19871	0.08346					
Bnd1Jap	0.39379	0.41627	0.31011				
Bnd7Jap	0.39649	0.35563	0.26651	0.11535			
ExRUK	0.41516	0.57385	0.39195	0.26009	0.36195		
ExRGer	0.43402	0.33271	0.3127	0.22496	0.23136	0.34187	
ExRJap	0.18262	0.24788	0.25119	0.44653	0.5184	0.32304	0.27371

Paper 3

A Heuristic for Moment-Matching Scenario Generation

Computational Optimization and Applications,
vol. 24, pp. 169–185, 2003.

A Heuristic for Moment-matching Scenario Generation

Kjetil Høyland
kjetil.hoyland@gjensidige.no*

Michal Kaut
michal.kaut@iot.ntnu.no†

Stein W. Wallace
stein.w.wallace@himolde.no‡

June 20, 2000; revised May 11, 2001, August 12, 2002

Abstract

In stochastic programming models we always face the problem of how to represent the random variables. This is particularly difficult with multidimensional distributions. We present an algorithm that produces a discrete joint distribution consistent with specified values of the first four marginal moments and correlations. The joint distribution is constructed by decomposing the multivariate problem into univariate ones, and using an iterative procedure that combines simulation, Cholesky decomposition and various transformations to achieve the correct correlations without changing the marginal moments.

With the algorithm, we can generate 1000 one-period scenarios for 12 random variables in 16 seconds, and for 20 random variables in 48 seconds, on a Pentium III machine.

Keywords: stochastic programming, scenario tree generation, Cholesky decomposition, heuristics

1 Introduction

Gjensidige Nor Asset Management (GNAM) has NOK 65 billion (7 billion US$) under management. During the last few years they have used stochastic-programming-based asset allocation models in their asset allocation processes.

*Gjensidige Nor Asset Management, POBox 276, N-1326 Lysaker, Norway
†Norwegian University of Science and Technology, N-7491 Trondheim, Norway
‡Molde University College, Servicebox 8, N-6405 Molde, Norway

The most important step in that process is to establish the market expectations, i.e. to establish what are their beliefs for the major asset categories (bonds, stocks, cash, commodities and currencies) in different major regions of the world. The decision-makers prefer to express their expectations in terms of marginal distributions of the return / interest rates for the different asset classes in addition to correlations. The challenge in converting these expectations into an asset allocation mix is twofold: First we need to convert the expectations into a format which a stochastic programming model can handle, second we need an optimization model which gives us the optimal asset mix, given this input.

Practical experience has told us that the first part, the scenario generation, can in fact be the most challenging and (computer-) time consuming one. For the purpose of generating the input data, GNAM has been using the method described in *Høyland and Wallace, 2001*, a method developed in 1996. For larger problems, with many asset classes, the scenario generation became the bottleneck in the portfolio optimization process. This paper presents an algorithm that reduces the computing time for the scenario generation substantially.

The most well-known applications of asset allocation models are the Russell-Yasuda-Kasai model in *Cariño and Ziemba, 1998* and the models implemented by Towers Perrin in *Mulvey, 1996*. Other applications can be found in *Consigli and Dempster, 1998* and *Dert, 1995*.

These models are all focused on long term strategic asset liability planning. Stochastic processes are widely used for scenario generation in such models. The big challenge with such processes is to calibrate the parameters so that the generated scenarios are consistent with the decision-maker's beliefs of the future development of the asset classes. In many applications, the parameters are calibrated so that the future scenarios are consistent with the past. This might be acceptable for long term strategic planning. For tactical short term planning, i.e. for the question of how to construct an asset allocation mix relative to an asset allocation benchmark, such approaches are, however, inappropriate. The user wishes to express views on the future which deviate from the past. It is then important that the decision-maker can express the market expectations in a way that he or she finds convenient and that these expectations are converted to model input in a consistent manner.

This is the basic philosophy of the scenario generation method proposed in *Høyland and Wallace, 2001*. The user specifies his or her market expectations in terms of marginal distributions for each asset class, in addition to correlations among the different asset classes, and possibly other distributional properties. The stochastic, possibly multi-period, asset allocation model requires

discrete outcomes for the uncertain variables. To generate these outcomes a least squares model is applied. The idea is to minimize the distance between some specified properties of the generated outcomes and their target values (either specified directly or derived from the marginal distributions, which may themselves be calculated from data or specified explicitly).

In the general form of the algorithm presented by Høyland and Wallace, outcomes of all the random variables (assets) are generated simultaneously. Such an approach becomes slow when the number of random variables increases. In this paper we generate one marginal distribution at a time and create the joint distribution by putting the marginal distributions together in the following way: All marginal distributions are generated with the same number of realizations, and the probability of the i'th realization is the same for each marginal distribution. The i'th scenario, that is, the i'th realization of the joint distribution, is then created by using the i'th realization from each marginal distribution, and given the corresponding probability. We then apply various transformations in an iterative loop to reach the target moments and correlations.

The presented algorithm is inspired by the work of *Fleishman, 1978*, *Vale, 1983* and *Lurie and Goldberg, 1998*. Fleishman presents a cubic transformation that transforms a sample from a univariate normal distribution to a distribution satisfying some specified values for the first four moments. Vale and Maurelli address the multivariate case and analyse how the correlations are changed when the cubic transformation is applied. The algorithm assumes that we start out with multivariate normals. The initial correlations are adjusted so that the correlations after the cubic transformation are the desired ones. The algorithm is only approximate with no guarantee about the level of the error.

Lurie and Goldberg outlined an algorithm that is of similar type as ours. They also generate marginal distributions independently and transform them in an iterative procedure. There are, however, two major differences between the two algorithms. One is in the way they handle the (inevitable) change of distribution during the transition to the multivariate distribution — while they modify the correlation matrix in order to end up with the right distribution, we modify the starting moments. The other major difference is that they start out with parametric marginal distributions whereas we start out with the marginal moments. We believe that specifying marginal moments is a more flexible approach and we certainly could also derive the marginal moments (up to the desired number) from the parametric distribution and apply our approach.

The rest of the paper is organized as follows: In Section 2 we present the algorithm. Numerical results are presented in Section 3, while possible future research areas are discussed in Section 4.

2 The algorithm

In Høyland and Wallace's method, a scenario tree can in principle be constructed to fit all distributional properties that can be formulated as functions of probabilities and outcomes. In this section, we will assume that the properties are the first four marginal moments and the correlations. This assumption is consistent with many other studies, as well as our own empirical analysis in *Høyland and Wallace, 2001* of what were the important properties in the given case study. The presented methodology can, in fact, treat more than four marginal moments and correlations. We could specify even higher moments, but the method is more restrictive than our original approach, which also allowed for such as extreme values.

The general idea of the algorithm is as follows: Generate n discrete univariate random variables, each satisfying a specification for the first four moments. Transform them so that the resulting random vector is consistent with a given correlation matrix. The transformation will distort the marginal moments of higher than second order. Hence, we need to start out with a different set of higher moments, so that we end up with the right ones.

The procedure would lead to the exact desired values for the correlations and the marginal moments if the generated univariate random variables were independent. This is, however, true only when the number of outcomes goes to infinity and all the scenarios are equally probable.[1] With a limited number of outcomes, and possibly distinct probabilities, the marginal moments and the correlations will therefore not fully match the specifications. To be able to secure that the error is within a pre-specified range, we have developed an iterative algorithm, which is an extension of the core algorithm.

Section 2.1 discusses the assumptions we have on the correlation matrix, Section 2.2 introduces necessary notation, Section 2.3 explains the key transformations used in the algorithm, Section 2.4 describes the core module of the algorithm, while Section 2.5 explains the iterative procedure.

[1] With unequal probabilities, we can expect the extreme cases to accumulate in the scenarios with the smallest probabilities, while the most probable scenarios would end up with outcomes close to their respective means. That would result in dependencies.

2.1 Assumption on the correlation matrix

There are two assumptions on the specified correlation matrix R. The first is a general assumption that R is a possible correlation matrix, i.e. that it is a symmetric positive semi-definite matrix with 1's on the main diagonal. While implementing the algorithm there is no need to check positive semi-definiteness directly, as we do a Cholesky decomposition of the matrix R at the very start. If R is not positive semi-definite, the Cholesky decomposition will fail.

Note that having an R that is not positive semi-definite means having some internal inconsistency in the data, so we should re-run our analysis. As an alternative, there exist several algorithms that find a possible correlation matrix that is, in some sense, closest to the specified matrix R. One such algorithm can be found in *Higham, 2000*. Another approach is used in *Lurie and Goldberg, 1998*, where a QP model is formulated to solve the problem. The latter approach has an advantage of being very flexible and allowing, for example, for specifying weights that express how much we believe in every single correlation. We can also use bounds to restrict the possible values.

The other assumption is that the random variables are not collinear, so that R is a non-singular — hence positive definite — matrix. For checking this property we can again use the Cholesky decomposition, because the resulting lower-triangular matrix L will have zero(s) on its main diagonal in a case of collinearity.

This is not a serious restriction, since having collinearity means that at least one of the variables can be computed from the others after the generation. We can thus decrease the dimension of the problem.

2.2 Notation

To formulate the model we introduce the following notation. Note that vectors are columns by default.

n	number of random variables
s	number of scenarios
$\tilde{X}$	general n-dimensional random variable $\rightarrow\ \tilde{X} = (\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_n)$ $\rightarrow$ every moment of $\tilde{X}$ is a vector of size n $\rightarrow$ the correlation matrix of $\tilde{X}$ is a matrix of size $n \times n$
$\mathbb{X}$	matrix of s scenario outcomes — $\mathbb{X}$ has dimension $n \times s$
$\mathbb{X}_i$	row vector of outcomes of the i'th random variable — dim. s
$\mathbb{P}$	row vector of scenario probabilities — given by the user
$\tilde{\mathcal{X}}$	discrete n-dimensional random variable given by $\mathbb{X}$ and $\mathbb{P}$

$\mathbb{E}[\tilde{X}]$ or $\mathbb{E}[\tilde{\mathcal{X}}]$	vector of means of a random var. (general or discrete)
$\mathcal{RV}(mom; corr)$	the set of all random variables with moments $mom = mom_1 \dots mom_4$ and a correlation matrix $corr$, where every mom_i is a vector of size n and $corr$ is a matrix of size $n \times n$
$TARMOM$	matrix of target moments $(4 \times n)$
R	target correlation matrix $(n \times n)$

Note the use of calligraphic letters — $\tilde{\mathcal{X}}$ — to represent a discrete random variable given by $\mathbb{X}$ and $\mathbb{P}$. Hence, we use $\tilde{X}$ when we refer to a general distribution (continuous or discrete), while $\tilde{\mathcal{X}}$ is always discrete, and always part of a construction eventually leading to the scenario tree we are about to make. Since the scenario probabilities $\mathbb{P}$ are given, generating the discrete random variable $\tilde{\mathcal{X}}$ is equivalent to generating a matrix of its outcomes $\mathbb{X}$. The two terms are thus very closely connected, even if $\tilde{\mathcal{X}}$ is a random variable, while $\mathbb{X}$ is a matrix.

Our goal is to generate scenarios with outcomes $\mathbb{Z}$, such that the discrete random variable $\tilde{\mathcal{Z}}$ defined by those outcomes and the scenario probabilities $\mathbb{P}$ has moments equal to $TARMOM$ and a correlation matrix equal to R. In our notation we want to generate a discrete random variable $\tilde{\mathcal{Z}}$ such that $\tilde{\mathcal{Z}} \in \mathcal{RV}(TARMOM; R)$.

2.3 Key transformations

The core module, which will be presented in the next section, has two key transformations. One is a cubic transformation used for generating univariate distributions with specified moments. The other is a matrix transformation used to transform a multivariate distribution to obtain a given correlation matrix.

2.3.1 Cubic transformation

This transformation comes from *Fleishman, 1978*, where a method to generate a univariate non-normal random variable $\tilde{Y}_i$ with given first four moments is introduced.[2] It takes a $\mathcal{N}(0,1)$ variable $\tilde{X}_i$ and uses a cubic transformation

$$\tilde{Y}_i = a + b\tilde{X}_i + c\tilde{X}_i^2 + d\tilde{X}_i^3$$

[2]The index i is obsolete in this section, but we use it for consistence with the rest of the paper.

to obtain $\tilde{Y}_i$ with the target moments. Parameters a, b, c and d are found by solving a system of non-linear equations. Those equations utilize normality of the variable $\tilde{X}_i$.

The problem of this approach is that $\tilde{X}_i$ must have the first 12 moments equal to those of $\mathcal{N}(0,1)$ in order to get exactly the target moments of $\tilde{Y}_i$. Since this is hard to achieve, either by sampling or discretization, the results with those formulas are only approximate. We have thus dropped the assumption of normality and derived formulas that work with an arbitrary random variable $\tilde{X}_i$ — see Appendix A. Parameters a, b, c and d are now given by a system of four implicit equations.

We have used a non-linear mathematical-programming (least-squares) model to solve the system. In this model we have a, b, c and d as decision variables, and we express the moments of $\tilde{\mathcal{Y}}_i$ as functions of these variables and the first 12 moments of $\tilde{\mathcal{X}}$.[3] We then minimize the distance between those moments and their target values. We do not need to assume that the system has a solution — if the solution does not exist, the model gives us the $\tilde{\mathcal{Y}}_i$ with the closest possible moments.

Our method for generating a discrete approximation $\tilde{\mathcal{Y}}_i$ of $\tilde{Y}_i$ is thus as follows:

- take some discrete r.v. $\tilde{\mathcal{X}}_i$ with the same number of outcomes as $\tilde{\mathcal{Y}}_i$
- calculate the first 12 moments from $\tilde{\mathcal{X}}_i$
- compute the parameters a, b, c, d
- compute the outcomes $\mathbb{Y}_i$ of $\tilde{\mathcal{Y}}_i$ as $\mathbb{Y}_i = a + b\mathbb{X}_i + c\mathbb{X}_i^2 + d\mathbb{X}_i^3$

2.3.2 Matrix transformation

Our other main tool in the algorithm is a matrix transformation of a random variable $\tilde{X}$

$$\tilde{Y} = L\,\tilde{X}$$

where L is a lower-triangular matrix. The matrix L always comes from a Cholesky decomposition of the correlation matrix R, so we have $L\,L^T = R$.[4]

From theory we know that if $\tilde{X}$ is an n-dimensional $\mathcal{N}(0,1)$ random variable with correlation matrix I (and therefore with $\tilde{X}_i$ mutually independent), then the $\tilde{Y} = L\,\tilde{X}$ is an n-dimensional $\mathcal{N}(0,1)$ random variable with correlation matrix $R = L\,L^T$.

[3] Since we have switched from general random variables to a discrete scenario model, we have to switch the notation from $\tilde{Y}$ to $\tilde{\mathcal{Y}}$.

[4] Note that L always exists, since we assume R to be positive semi-definite.

Since we do not have normal variables, we need a more general result. To make the formulas as simple as possible, we restrict ourselves to the case of zero means and variances equal to 1. In the beginning of Section 2.4 we show how to deal with this restriction. Note that $E[\tilde{X}] = 0$ leads to $mom_i = E[\tilde{X}^i]$. We will thus use the two interchangeably for the rest of the section.

In Appendix B, we show that, in that case, $\tilde{Y}(= L\,\tilde{X})$ is an n-dimensional random variable with zero means, variances equal to 1, and correlation matrix $R = L\,L^T$. In addition, the higher moments of $\tilde{Y}$ are as follows:

$$\begin{aligned}
\mathbb{E}\left[\tilde{Y}_i^3\right] &= \sum_{j=1}^{i} L_{ij}^3\,\mathbb{E}\left[\tilde{X}_j^3\right] \\
\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 &= \sum_{j=1}^{i} L_{ij}^4\left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)
\end{aligned}$$

We will need also the opposite direction of the transformation:

$$\tilde{X} = L^{-1}\,\tilde{Y}.$$

Since L^{-1} is a triangular matrix, it is easy to invert the formulas:

$$\begin{aligned}
\mathbb{E}\left[\tilde{X}_i^3\right] &= \frac{1}{L_{ii}^3}\left(\mathbb{E}\left[\tilde{Y}_i^3\right] - \sum_{j=1}^{i-1} L_{ij}^3\,\mathbb{E}\left[\tilde{X}_j^3\right]\right) \\
\mathbb{E}\left[\tilde{X}_i^4\right] - 3 &= \frac{1}{L_{ii}^4}\left[\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 - \sum_{j=1}^{i-1} L_{ij}^4\left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)\right]
\end{aligned}$$

We divide only by the diagonal elements L_{ii} in these formulas. We can do it since L_{ii} are positive due to regularity (positive definiteness) of R.

2.4 The core algorithm

This section presents the core algorithm. It runs as follows: Find the target marginal moments from stochastic processes, from statistics or by specifying them directly. Generate n discrete random variables with these moments. Create the multivariate random variable by combining the univariate variables, as explained in the Introduction. Transform this variable so that it has the desired correlations and marginal moments. If the random variables $\tilde{\mathcal{X}}_i$ were independent, we would end up with $\tilde{\mathcal{Y}}$ having exactly the desired properties.

To facilitate the reading, we have divided the algorithm in two parts. In the *input phase* we read the target properties specified by the user and transform them to a form needed by the algorithm. In the *output phase* we generate the distributions and transform them to the original properties.

2.4.1 The input phase

In this phase we work only with the target moments and correlations, we do not yet have any outcomes. This means that all operations are fast and independent of the number of scenarios s.

Our goal is to generate a discrete approximation $\tilde{\mathcal{Z}}$ of an n-dimensional random variable $\tilde{Z}$ with moments $TARMOM$ and correlation matrix R. Since the matrix transformation needs zero means and variances equal to 1, we have to change the targets to match this requirement. Instead of $\tilde{\mathcal{Z}}$ we will thus generate random variables $\tilde{\mathcal{Y}}$ with moments MOM (and correlation matrix R), such that $MOM_1 = 0$, and $MOM_2 = 1$. $\tilde{\mathcal{Z}}$ is then computed at the very end of the algorithm as

$$\tilde{\mathcal{Z}} = \alpha\tilde{\mathcal{Y}} + \beta.$$

It is easily shown that the values leading to the correct $\tilde{\mathcal{Z}}$ are:

$$\alpha = TARMOM_2^{1/2} \qquad MOM_3 = \frac{TARMOM_3}{\alpha^3}$$
$$\beta = TARMOM_1 \qquad MOM_4 = \frac{TARMOM_4}{\alpha^4}$$

The final step in the input phase is to derive moments of independent univariate random variables $\tilde{X}_i$ such that $\tilde{Y} = L\,\tilde{X}$ will have the target moments and correlations. To do this we need to find the Cholesky-decomposition matrix L, i.e. a lower-triangular matrix L so that $R = L\,L^T$.

The input phase then contains the following steps:

1. Specify the target moments $TARMOM$ and target correlation matrix R of $\tilde{Z}$ (and $\tilde{\mathcal{Z}}$)
2. Find the normalized moments MOM for $\tilde{\mathcal{Y}}$
3. Compute L and find the transformed moments $TRSFMOM$ for $\tilde{\mathcal{X}}$ — see Section 2.3.2

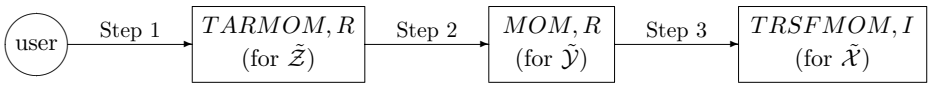

Figure 1: Input Phase

2.4.2 The output phase

In this phase we start by generating the outcomes for the independent random variables. Next, we transform them to get the intermediate-target moments and target correlations, and finally obtain the moments specified by the user. Since the last transformation is a linear one, it will not change the correlations. All the transformations in this phase are with the outcomes, so the computing time needed for this phase is longer and increases with the number of scenarios.

We start by generating n discrete univariate random variables $\tilde{\mathcal{X}}_i$. This is a well-known problem and there are several possible ways to do it. We have used a method from *Fleishman, 1978*, in a way described in Section 2.3.1. The method starts by sampling from $\mathcal{N}(0,1)$ and afterwards uses the cubic transformation to get the desired moments. For the starting $\mathcal{N}(0,1)$ sample we use a random-number generator. An alternative would be to use a discretization of the distribution, or some other method for discretizing $\mathcal{N}(0,1)$, for example the method described in *Høyland and Wallace, 2001.*

Once we have generated the outcomes $\mathbb{X}_i$ for the random variables $\tilde{\mathcal{X}}_i$, we can proceed with the transformations. First $\mathbb{Y} = L\,\mathbb{X}$ to get the target correlations and then $\mathbb{Z} = \alpha\mathbb{Y} + \beta$ to get the user-specified moments.[5]

The output phase of the core algorithm consists of the following steps:

4. Generate outcomes $\mathbb{X}_i$ of 1-dimensional variables $\tilde{\mathcal{X}}_i$
   (independently for $i = 1 \ldots n$)
5. Transform $\tilde{\mathcal{X}}$ to the target correlations: $\mathbb{Y} = L\,\mathbb{X} \;\rightarrow \tilde{\mathcal{Y}} \in \mathcal{RV}(MOM; R)$
6. Transform $\tilde{\mathcal{Z}}$ to the original moments:
   $\mathbb{Z} = \alpha\mathbb{Y} + \beta \;\rightarrow \tilde{\mathcal{Z}} \in \mathcal{RV}(TARMOM; R)$

2.5 The modified algorithm

We know that the core algorithm gives us the exact results only if the random variables $\tilde{\mathcal{X}}_i$ are independent. Since we generate each set of outcomes $\mathbb{X}_i$

[5]Note that as we have stopped to speak about distributions and started to speak about outcomes, we have to change the notation from $\tilde{\mathcal{X}}$ to $\mathbb{X}$.

Figure 2: Output Phase

separately, and have a limited number of outcomes, the resulting co-moments will most certainly differ from zero and the results of the core algorithm will be only approximate.

The modified algorithm is iterative, so all results are still approximate, but with a pre-specified maximal error. Hence, we can control the quality of the results. We will again use the matrix transformation $\mathbb{Y} = L\,\mathbb{X}$, this time both forward and backward, as mentioned in 2.3.2. Recall that this transform allows us to obtain a desired correlation matrix, but it changes the marginal moments while doing so.

In Section 2.3.1 we showed that the cubic transformation allows us to transform a general univariate random variable $\tilde{X}_i$ to a variable with some target moments. This transformation changes the correlations, but if the starting $\tilde{\mathcal{X}}_i$'s have moments close to their targets, the changes in the correlations are expected to be small.

The idea of the new algorithm is thus to introduce iterative loops in the core algorithm, namely in Steps 4 and 5, in the following way:

The purpose of Step 4 is to generate the independent random variables $\tilde{\mathcal{X}}_i$. Since independence is very hard to achieve, we change our target to generating uncorrelated r.v. $\tilde{\mathcal{X}}_i$, i.e. we seek to get $\tilde{\mathcal{X}} \in \mathcal{RV}(TRSFMOM; I)$. We use an iterative approach to achieve those properties.

Since we do not control the higher co-moments, they will most likely not be zero, and the $\tilde{\mathcal{Y}}$ obtained in Step 5 will not have the target properties. We thus need another loop there to end up with the desired $\tilde{\mathcal{Y}}$.

The iterative versions of Steps 4 and 5 are:[6]

Step 4

4.*i.* Generate n univariate random variables with moments $TRSFMOM$ (independently) $\rightarrow$ we get $\tilde{\mathcal{X}}$ with correlation matrix R_1 close to I due to the independent generation

4.*ii.* `let` $p = 1$ and $\tilde{\mathcal{X}}_1 = \tilde{\mathcal{X}}$

4.*iii.* `while` $dist(R_p; I) > \varepsilon_x$ `do`

[6] As we have not found any better notation, in the rest of this section lower index p denotes an iteration counter, not a matrix column.

4.*iv.* `do` Cholesky decomposition: $R_p = L_p\, L_p^T$

4.*v.* `do` backward transform $\mathbb{X}_p^* = L^{-1}\, \mathbb{X}_p \;\rightarrow \tilde{\mathcal{X}}^*$ has zero correlations, wrong moments

4.*vi.* `do` cubic transform of $\tilde{\mathcal{X}}_p^*$ with $TRSFMOM$ as the target moments; store results as $\tilde{\mathcal{X}}_{p+1} \;\rightarrow$ right moments, wrong correlations

4.*vii.* compute correlation matrix R_{p+1}

4.*viii.* `let` $p = p + 1$

4.*ix.* `let` $\tilde{\mathcal{X}} = \tilde{\mathcal{X}}_p \;\rightarrow \tilde{\mathcal{X}} \in \mathcal{RV}(TRSFMOM; R_p)$ with correlation error $dist(R_p; I) \leq \varepsilon_x$

A root-mean-squared-error is used as the measure $dist()$ in 4.*iii*, see Section 3 for an exact definition. The same distance is used also in Step 5 below (in steps 5.*iii* and 5.*ix*). Since $\tilde{\mathcal{X}}$ is not a final output, the maximum error ε_x in Step 4 is typically higher than the corresponding ε_y in Step 5.

There are two possible outcomes from Step 4: $\tilde{\mathcal{X}}_p$ corresponding to random variables with right moments and slightly off correlations, and $\tilde{\mathcal{X}}_{p-1}^*$ corresponding to random variables with slightly off moments and zero correlations. We start with the latter in Step 5, and denote it $\tilde{\mathcal{X}}^*$.

Step 5

5.*i.* $\tilde{\mathcal{Y}}_1 = L\, \tilde{\mathcal{X}}^* \;\rightarrow$ both moments and correlations are incorrect (due to higher co-moments different from zero)

5.*ii.* `let` $p = 1$ and let R_1 be the correlation matrix of $\tilde{\mathcal{Y}}_1$

5.*iii.* `while` $dist(R_p; R) > \varepsilon_y$ `do`

5.*iv.* `do` Cholesky decomposition: $R_p = L_p\, L_p^T$

5.*v.* `do` backward transform $\mathbb{Y}_p^* = L_p^{-1}\, \mathbb{Y}_p$ $\rightarrow \tilde{\mathcal{Y}}_p$ has zero correlations, incorrect moments

5.*vi.* `do` forward transform $\mathbb{Y}_p^{**} = L\, \mathbb{Y}_p^*$ $\rightarrow \tilde{\mathcal{Y}}_p$ has correct correlations (R), incorrect moments

5.*vii.* `do` cubic transform of $\tilde{\mathcal{Y}}_p^{**}$ with MOM as the target moments; store results as $\tilde{\mathcal{Y}}_{p+1} \rightarrow \tilde{\mathcal{Y}}_{p+1} \in \mathcal{RV}(MOM; R_{p+1})$

5.*viii.* `let` $p = p + 1$

5.*ix.* `let` $\tilde{\mathcal{Y}} = \tilde{\mathcal{Y}}_p \;\rightarrow\; \tilde{\mathcal{Y}} \in (MOM; R_p)$ with correlation error $dist(R_p; R) \leq \varepsilon_y$

Note that we can again choose two different outcomes from Step 5 (and thus from the whole algorithm). After Step 5.*ix*, $\tilde{\mathcal{Y}}$ has the right moments and (slightly) off correlation. If we prefer exact correlations, we can either use the last $\tilde{\mathcal{Y}}_p^{**}$, or repeat Steps 5.*iv* – 5.*vi* just before we go to Step 5.*ix*.

Note also that Steps 5.*v* and 5.*vi* are written as individual steps for the sake of clarity. Since we have always $s > n$ (usually $s \gg n$), it is much more efficient to join the two steps and calculate $\mathbb{Y}_p^{**} = \left(L \times L_p^{-1}\right) \mathbb{Y}_p$ directly.

2.5.1 Convergence

The issue of convergence to the target moments and correlations is difficult. First, we know that the method cannot converge in general, since it is possible to specify combinations of moments that cannot exist. In addition, we need to have enough scenarios, where the minimal number of scenarios depends not only on the number of random variables n, but also on the structure of the problem. For example, random variables with distribution close to the normal, and with small correlations, typically need fewer scenarios than random variables with fat tails and high correlations.

We have not succeeded in producing a convergence proof, but more than two years of active use in Gjensidige NOR has so far never left us with an unsolvable case. The algorithm is stopped whenever a prespecified number of iterations has been performed, or the convergence criteria are satisfied. Whatever the reason, it is straightforward to test if the resulting scenario tree has the required properties (and also if it is arbitrage-free). Hence, when a tree is used in an optimization model, it always has the required properties.

Therefore, the concern is not that we risk using a tree with bad properties, but rather that we are left with no tree at all because the algorithm either converges to the wrong solution or diverges.

If the algorithm does not converge to the right solution, our first approach is always to try to rerun the algorithm a few times, in reality, trying to start the whole process with a different set of independent random variables in Step 4.*i*. If that does not work, we try to increase the number of scenarios. Unless we know for sure that the specified set of properties exist (for example because they are calculated from a data set or from given distributions) we next try to find out if we may have defined random variables that do not exist. By these three means, we have always found a solution.

It is worth noting that when the method is used on a continuous basis, the user learns how many scenarios are needed for his problem. Also, he will know whether or not he has specified the properties in such a way that the implied distribution exists. This will limit the possible actions. Our experience is that

it is enough to first check for actual errors (bugs) in the specifications, and then rerun the algorithm a few times from different starting points.

A numerical test of convergence is in Section 3.1.

3 Numerical results

This section presents the times needed to generate scenario trees with different numbers of random variables and scenarios.

To see how the generation time depends on the number of random variables and the number of scenarios, we have generated trees with 4, 8, 12 and 20 random variables and 40, 100, 200 and 1000 scenarios. Except the case of four random variables, we have used actual data from Gjensidige NOR as input. In addition, we have used two different data sets in the cases of 12 and 20 random variables to improve the estimates. Since we did not have any distinct case with only four random variables, we have used the first four variables from the first 12-variable case. A more detailed description of the input data can be found in `http://home.himolde.no/~wallace/reports.htm`.[7]

The algorithm is implemented in `AMPL` with `MINOS 5.5` as a solver. The test was done on a Pentium III 500 MHz machine with 256 MB memory, running Windows NT 4.0. As a distance used for evaluating the quality of the distributions in Steps 4 and 5 of the algorithm, we have used a root-mean-squared-error defined as

$$RMSE = \sqrt{\frac{1}{N_{el}} \sum_k (value_k - TARGET_k)^2}$$

where N_{el} is the number of elements in the sum. The distance was evaluated separately for moments ($N_{el} = 4n$) and for correlations ($N_{el} = \frac{n(n-1)}{2}$).

The stopping-values were $\varepsilon_x = 0.1$ ($\varepsilon_x = 0.2$ in the case of 20 r.v.) and $\varepsilon_y = 0.01$. Note that the distances are evaluated inside the algorithm, where all the data are scaled to variance equal to 1, so they are scale-independent. Using the formulas in Section 2.4.1, one can show that the maximum error of the i'th moment in the final output is $TARMOM_i^i \cdot \varepsilon_y$.

Results of the tests are in Table 1. As we see, we can find trees with as much as 1000 scenarios in less than a minute. It is worth observing that as the number of scenarios increases, the computing time decreases. This strange behavior is caused by a better convergence for larger trees, i.e. the fact that the number of iterations decreases with the size of a tree: with 40 scenarios

[7] In this thesis, the description of the input data follows after the paper.

Table 1: Running times for different numbers of random variables and scenarios. Times are given in format `mm:ss`.

	Number of scenarios			
r.v.	40	100	200	1000
4	00:01	00:01	00:01	00:05
8	00:04	00:04	00:03	00:10
12	00:09	00:07	00:06	00:16
20	01:05	00:44	00:31	00:48

we need typically 1–2 iterations in Step 4 of the algorithm (generation of $\tilde{\mathcal{X}}$) and 2–3 iterations in Step 5 (generation of $\tilde{\mathcal{Y}}$), whereas with 1000 scenarios we usually need only one iteration in each part. Since we cannot have less than one iteration, there will be no more improvement in the convergence for trees with more than 1000 scenarios. We can thus expect approximately linear time-dependency for very large trees.

It may be of interest to compare these times with what we observe by using the method in *Høyland and Wallace, 2001*. Table 2 shows the results for 1000 scenarios. As we see, the savings are substantial.

Table 2: Running times for creating 1000 scenarios. In the first column are times for the algorithm from *Høyland and Wallace, 2001*, in the second column times for the new algorithm. The last column presents the speed-up. Times are given in format `[h:]mm:ss`.

r.v.	old alg.	new alg.	speed-up
4	00:35	00:05	7.5×
8	08:39	00:10	53×
12	17:54	00:16	66×
20	1:34:46	00:48	119×

3.1 Convergence test

As mentioned in Section 2.5.1, we do not have a convergence proof for the iterative algorithm presented in Section 2.5. We have thus tested the convergence numerically. The iterative procedure is used in steps 4 and 5 of the algorithm. Step 4 is, however, used only to obtain a reasonable starting point for

Step 5, so the convergence is not as important there. Hence, we only present convergence tests for Step 5.

For the test, we have chosen the smallest and the biggest cases from the data used in the main test, namely 40 scenarios for 4 random variables and 1000 scenarios for 20 random variables. For both sets, we have run the algorithm 25 times, each time with a different initial set of scenarios.

We have measured an error in moments after the matrix transformation and an error in correlations after the cubic transformation. The other two combinations are not interesting, since both transformations set "their" errors to zero. Hence, the two errors are equal to the total errors after the given transformation.[8]

Results of the test are in Figure 3. Every iteration consists of the matrix transformation (Steps 5.*iv*...5.*vi*) and the cubic transformation (Step 5.*vii*). We see that both the errors are monotonously decreasing. Hence, also the total error at the end of an iteration is monotonously decreasing. This is actually true not only for the average values depicted in the graphs, but for all the 25 runs, in both test cases.

4 Future work

The most important subject for future research is the convergence of the algorithm. Since we do not have a convergence proof, we should try to obtain a better understanding of the convergence. In particular, we should try to identify statistical properties that would guarantee/prevent convergence. Another interesting problem is the minimum number of scenarios needed to achieve given precision for a given set of properties.

In many cases we are interested in multi-period scenario trees. It is a future challenge to see how the presented method can be used as a building block for a larger tree, particularly how to preserve time series properties.

Even though we have achieved a substantial computational speed, the tested implementation is still far from optimal. In the current `AMPL` implementation, the algorithm spends most of its time with the Cholesky transformation and the communication with the solver before and after the cubic transform.

Both these critical times can be eliminated or decreased by implementing the algorithm in a compiled form. A `C++` implementation is currently being

[8] We define the total error as a sum of the errors in moments and correlations. After every transformation one of the errors is zero, hence the total is equal to the other one.

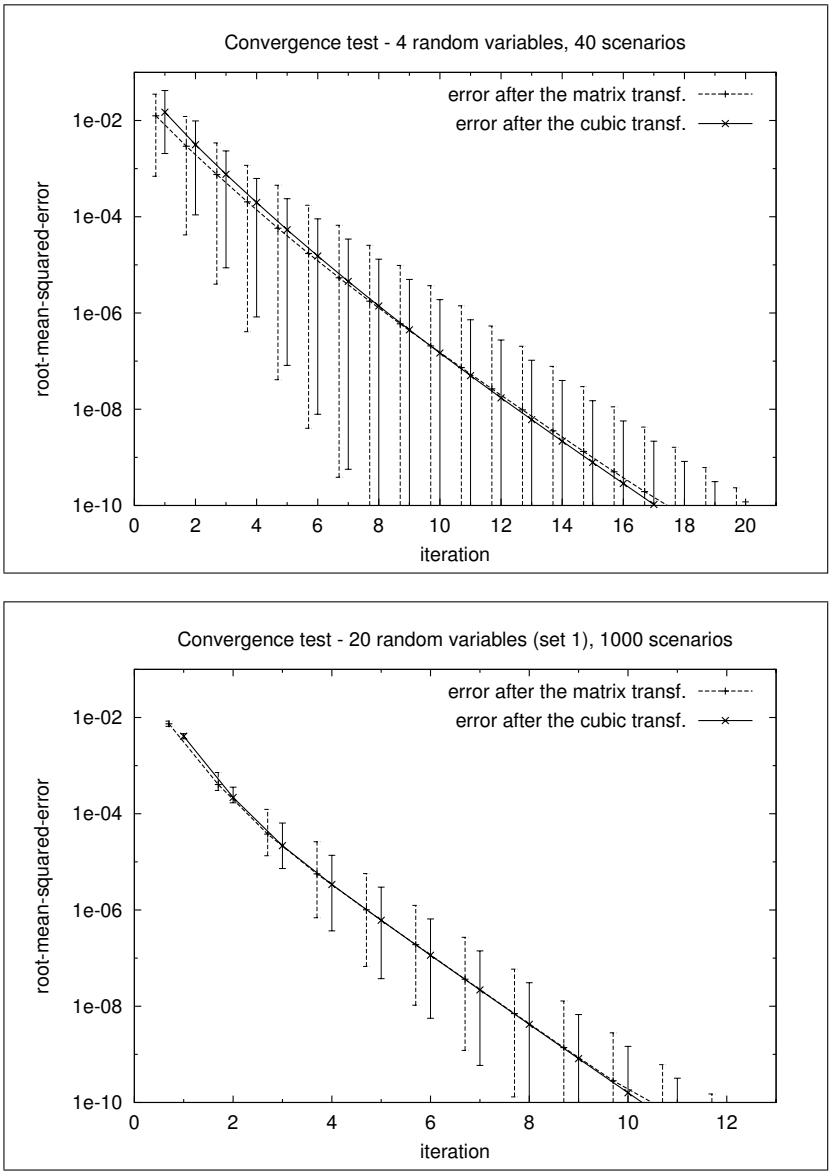

Figure 3: Convergence of the iterative algorithm. In both cases, the algorithm was run 25 times. Lines represent average errors after every iteration, bars represents the best and the worst cases. The dashed lines represents errors in moments after the matrix transformation, the solid line errors in correlations after the cubic transformation. The root-mean-squared-error is defined in Section 3.

developed at SINTEF[9] research institute, and according to the first tests is more than 10 times faster than our AMPL code. This implementation is part of OMEGA-IST-1999-12088, an EU financed project on electricity production i deregulated markets.

The presented algorithm is well suited for parallel implementation, because most of its parts can be processed independently for each random variable. The only step of the algorithm that uses considerable amount of time and cannot be parallelized in this simple way is the Cholesky decomposition of the correlation matrix, but parallel codes for the Cholesky decomposition can also be found.

5 Conclusions

We have presented an algorithm to generate scenarios for multivariate random variables. The purpose of the algorithm is to speed up an existing scenario generation algorithm, which constructs multi-dimensional scenario trees with specified moments and correlations.

The original algorithm constructs the multidimensional scenario tree by solving a single, potentially very large, least squares problem. The main idea of the new algorithm is to decompose the least squares problem so that each marginal distribution is constructed separately. To combine the different marginal distributions so that the joint distribution satisfies the specified correlations, we apply a Cholesky decomposition and a cubic transformation in an iterative procedure.

Even if we cannot guarantee convergence of this procedure, our experience shows that it does converge if the specifications are possible and there are enough scenarios. In addition, a potential divergence or convergence to the wrong solution is easy to detect. Hence, we never end up using an incorrect tree in the optimization procedure.

Testing shows that our algorithm is reasonably fast. We can find trees with 1000 scenarios representing 20 random variables in less than one minute.

Acknowledgments

We would like to thank Erik Kole from the Maastricht University for pointing out an error in an earlier version of the paper, and Matthias Nowak from SINTEF, Trondheim, for suggesting some important changes in notation and

[9]The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology

structure of the paper. Furthermore, we would like to thank our colleague Halvard Arntzen for helping us with cleaning up the terminology. We are also in debt to anonymous referees. Much of this work was done while Stein W. Wallace visited the Centre for Advanced Study at the Academy of Science and Letters in Oslo.

References

Cariño, D. R. and Ziemba, W. T. (1998). Formulation of the Russell-Yasuda Kasai financial planning model. *Operations Research*, 46(4):443–449.

Consigli, G. and Dempster, M. A. H. (1998). Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81:131–162.

Dert, C. (1995). *Asset Liability Management for Pension Funds, A Multistage Chance Constrained Programming Approach*. PhD thesis, Erasmus University, Rotterdam, The Netherlands.

Fleishman, A. I. (1978). A method for simulating nonnormal distributions. *Psychometrika*, 43:521–532.

Higham, N. J. (2000). Computing the nearest correlation matrix—A problem from finance. Numerical Analysis Report No. 369, Manchester Centre for Computational Mathematics, Manchester, England.

Høyland, K. and Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307.

Lurie, P. M. and Goldberg, M. S. (1998). An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science*, 44(2):203–218.

Mulvey, J. M. (1996). Generating scenarios for the Towers Perrin investment system. *Interfaces*, 26:1–13.

Vale, C. David & Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. *Psychometrika*, 48(3):465–471.

Appendix

A Cubic transformation $\tilde{Y} = a + b\tilde{X} + c\tilde{X}^2 + d\tilde{X}^3$

The purpose of this transformation is to produce a univariate random variable $\tilde{Y}_i$ with specified first four moments $\mathbb{E}\big[\tilde{Y}_i^k\big], k = 1, \ldots, 4$, given the random variable $\tilde{X}_i$ with known first 12 moments $\mathbb{E}\big[\tilde{X}_i^k\big], k = 1, \ldots, 12$.

The problem is to find the transform parameters a_i, b_i, c_i and d_i. For this we have to express $\mathbb{E}\big[\tilde{Y}_i^k\big]$ as functions of $\mathbb{E}\big[\tilde{X}_i^k\big]$. The formulas can be stated either for univariate random variables $\tilde{Y}_i$, or as a vector equations for the random vector $\tilde{Y}$. The only difference is the presence/absence of index i in all elements. To make the formulas easier to read, we use the vector form.

$$
\begin{aligned}
\mathbb{E}[\tilde{Y}] &= a + b\mathbb{E}[\tilde{X}] + c\mathbb{E}[\tilde{X}^2] + d\mathbb{E}[\tilde{X}^3] \\
\mathbb{E}[\tilde{Y}^2] &= d^2\mathbb{E}[\tilde{X}^6] + 2cd\mathbb{E}[\tilde{X}^5] + (2bd + c^2)\mathbb{E}[\tilde{X}^4] + (2ad + 2bc)\mathbb{E}[\tilde{X}^3] \\
&+ (2ac + b^2)\mathbb{E}[\tilde{X}^2] + 2ab\mathbb{E}[\tilde{X}] + a^2 \\
\mathbb{E}[\tilde{Y}^3] &= d^3\mathbb{E}[\tilde{X}^9] + 3cd^2\mathbb{E}[\tilde{X}^8] + (3bd^2 + 3c^2d)\mathbb{E}[\tilde{X}^7] + (3ad^2 + 6bcd + c^3)\mathbb{E}[\tilde{X}^6] \\
&+ (6acd + 3b^2d + 3bc^2)\mathbb{E}[\tilde{X}^5] + (a(6bd + 3c^2) + 3b^2c)\mathbb{E}[\tilde{X}^4] \\
&+ (3a^2d + 6abc + b^3)\mathbb{E}[\tilde{X}^3] + (3a^2c + 3ab^2)\mathbb{E}[\tilde{X}^2] + 3a^2b\mathbb{E}[\tilde{X}] + a^3 \\
\mathbb{E}[\tilde{Y}^4] &= d^4\mathbb{E}[\tilde{X}^{12}] + 4cd^3\mathbb{E}[\tilde{X}^{11}] + (4bd^3 + 6c^2d^2)\mathbb{E}[\tilde{X}^{10}] \\
&+ (4ad^3 + 12bcd^2 + 4c^3d)\mathbb{E}[\tilde{X}^9] + (12acd^2 + 6b^2d^2 + 12bc^2d + c^4)\mathbb{E}[\tilde{X}^8] \\
&+ (a(12bd^2 + 12c^2d) + 12b^2cd + 4bc^3)\mathbb{E}[\tilde{X}^7] + (6a^2d^2 + a(24bcd + 4c^3) \\
&+ 4b^3d + 6b^2c^2)\mathbb{E}[\tilde{X}^6] + (12a^2cd + a(12b^2d + 12bc^2) + 4b^3c)\mathbb{E}[\tilde{X}^5] \\
&+ (a^2(12bd + 6c^2) + 12ab^2c + b^4)\mathbb{E}[\tilde{X}^4] + (4a^3d + 12a^2bc + 4ab^3)\mathbb{E}[\tilde{X}^3] \\
&+ (4a^3c + 6a^2b^2)\mathbb{E}[\tilde{X}^2] + 4a^3b\mathbb{E}[\tilde{X}] + a^4
\end{aligned}
$$

Note that since the matrix transformation $\tilde{Y}_i = L\,\tilde{X}_i$ does not change the first two moments, we have $E[\tilde{X}_i] = 0$ and $E[\tilde{X}_i^2] = 1$. We could thus simplify the above formulas slightly by assuming that the first two moments are exactly 0 and 1. In our implementation we have, however, used the formulas in the presented form, computing the actual moments.

B Matrix transformation $\tilde{Y} = L\,\tilde{X}$

We seek an n-dimensional random variable $\tilde{Y}$ with zero means, variances equal to 1, skewness MOM_3, kurtosis MOM_4, and a correlation matrix $R = L\,L^T$, where L is a lower-triangular matrix. To obtain $\tilde{Y}$, we start with an n-dimensional random vector $\tilde{X}$ with independent components $\tilde{X}_i$. Thereafter,

$\tilde{Y}$ is computed as $\tilde{Y} = L\,\tilde{X}$. Note that the k'th moment of $\tilde{Y}$ is equal to $\mathbb{E}\big[\tilde{Y}^k\big]$ because of the zero means.

During the algorithm we use the matrix transformation in two different cases. First it is used on distributions, which are abstract objects, so we work only with their distributional properties. In the second part of the algorithm, we transform the outcomes, which are matrices of numbers. The formulas are the same in both cases, the only difference is the use of $\tilde{X}$ in the first case and $\mathbb{X}$ in the latter.

We use the distribution-notation in the rest of the appendix. The matrix transformation in a column-wise form is then:

$$\tilde{Y}_i = \Big(L\,\tilde{X}\Big)_i = \sum_{j=1}^{n} L_{ij}\,\tilde{X}_j = \sum_{j=1}^{i} L_{ij}\,\tilde{X}_j$$

where the last equality comes from the fact that L is a lower-triangular matrix and therefore $L_{ij} = 0$ for $j > i$.

Theorem – properties of $\tilde{Y} = L\,\tilde{X}$

Assume we have an n-dimensional random variable $\tilde{X}$ with the following properties:

i. $\mathbb{E}\big[\tilde{X}^k\big]$ exists for $k = 1 \ldots 4$

ii. $\mathbb{E}\big[\tilde{X}\big] = 0 \quad$ and $\quad \mathbb{E}\big[\tilde{X}^2\big] = 1$

iii. the univariate random variables $\tilde{X}_i$, $\tilde{X}_j$ are independent for $i \neq j$

Assume further that L is a lower-triangular matrix of size n such that $R = L\,L^T$, where R is a correlation matrix, i.e. R is a symmetric positive semi-definite matrix with 1's on the main diagonal.

If we then define a random variable $\tilde{Y}$ as $\tilde{Y} = L\,\tilde{X}$, it has the following properties:

iv. $\mathbb{E}\big[\tilde{Y}^k\big]$ exists for $k = 1 \ldots 4$

v. $\mathbb{E}\big[\tilde{Y}\big] = 0 \quad$ and $\quad \mathbb{E}\big[\tilde{Y}^2\big] = 1$

vi. $\tilde{Y}$ has a correlation matrix $R = L\,L^T$

vii. $\mathbb{E}\left[\tilde{Y}_i^3\right] = \sum_{j=1}^{i} L_{ij}^3\,\mathbb{E}\left[\tilde{X}_j^3\right]$

viii. $\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 = \sum_{j=1}^{i} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)$

The proof is straightforward, and is left to the reader.

Consequence

Under the assumptions of the theorem, with an additional assumption that R is regular (and therefore positive-definite), we can express the moments of $\tilde{X}$ as:

ix. $\mathbb{E}\left[\tilde{X}_i^3\right] = \frac{1}{L_{ii}^3}\left(\mathbb{E}\left[\tilde{Y}_i^3\right] - \sum_{j=1}^{i-1} L_{ij}^3\, \mathbb{E}\left[\tilde{X}_j^3\right]\right)$

x. $\mathbb{E}\left[\tilde{X}_i^4\right] - 3 = \frac{1}{L_{ii}^4}\left[\mathbb{E}\left[\tilde{Y}_i^4\right] - 3 - \sum_{j=1}^{i-1} L_{ij}^4 \left(\mathbb{E}\left[\tilde{X}_j^4\right] - 3\right)\right]$

Note that the set of moments $(0, 1, 0, 3)$ is an invariant of this transformation. This confirms the known theoretical result that the linear transformations preserve normality. In the context of our algorithm it means that if we generate normal variables, we can skip Step 3 of the algorithm.

Description of data used in the numerical tests

Michal Kaut
michal.kaut@iot.ntnu.no*

Abstract

This note provides a complete description of input data used in the numerical tests in the paper "A Heuristic for Moment-matching Scenario Generation", published in *Computational Optimization and Applications*, vol. 24, pp. 169–185, 2003.

In the paper, we test the presented heuristics on trees with 4, 8, 12 and 20 random variables and 40, 100, 200 and 1000 scenarios. All the cases—except the case of four random variables—are actual data files used in an asset allocation model at Gjensidige NOR. All the random variables are thus one-month returns of some assets. The scale of the returns may vary, since different types of assets are handled differently in the model.

All the assets in the tables have four-letter codes. The first two letters designate a type of the asset: `Cs`, `C1`, and `C2` stands for cash; `Bs`, `Bl`, and `Bn` for short-, long- and unspecified bonds, respectively; and `st` for stocks. The second two letters designate country of the asset: in addition to the standard abbreviations, `Jp` stands for Japan, `Ge` for Germany, `Eu` for the Eurozone,[1] and `No` for Norway.

For the tests with twelve and twenty random variables, we have used two different data sets for each case to improve the estimates. The input data are in Tables 2, 3, 4, and 5, respectively. For the tests with four and eight random variables, only one data set was used. The data set for eight random variables is in Table 1. Since we did not have any distinct data set with only four random variables, we have used the first four variables from the data in Table 2.

*Norwegian University of Science and Technology, N-7491 Trondheim, Norway

[1] Except for the currency, Eurozone means Germany.

Table 1: Statistical properties — 8 random variables

	BnUS	BnJp	BnUK	BnGe	StUS	StJp	StUK	StGe
mean	-0.032	0.14	0.174	0.214	0.0	3.0	2.0	3.0
stdev	0.1	0.25	0.6	0.5	10.8	10.8	10.8	14.8
skew	0.0	0.5	0.4	0.4	-0.4	0.5	0.2	0.2
kurt	3.0	2.5	3.0	2.5	4.0	3.0	4.0	4.0

	BnUS	BnJp	BnUK	BnGe	StUS	StJp	StUK
BnJp	0.2						
BnUK	0.5	0.2					
BnGe	0.5	0.2	0.5				
StUS	-0.4	-0.1	-0.2	-0.2			
StJp	-0.1	-0.4	-0.1	-0.1	0.2		
StUK	-0.2	-0.1	-0.4	-0.2	0.5	0.2	
StGe	-0.2	-0.1	-0.2	-0.4	0.5	0.2	0.5

Table 2: Statistical properties — 12 random variables, set 1

	CsUS	CsJp	CsUK	CsGe	BnUS	BnJp	BnUK	BnGe	StUS	StJp	StUK	StGe
mean	0.03	-0.09	0.15	0.04	0.12	0.14	0.187	0.138	-3.0	2.0	3.0	-2.0
stdev	0.6	0.15	0.6	0.35	0.6	0.3	0.6	0.4	10	10.5	10.5	10.7
skew	0.25	0.6	0.0	0.5	0.3	0.5	0.3	0.25	-0.6	0.5	0.0	0.0
kurt	3.0	2.0	3.0	2.5	3.0	2.5	3.0	2.5	4.0	3.0	4.0	4.0

	CsUS	CsJp	CsUK	CsGe	BnUS	BnJp	BnUK	BnGe	StUS	StJp	StUK
CsJp	0.1										
CsUK	0.2	0.1									
CsGe	0.2	0.1	0.2								
BnUS	0.4	0.0	0.1	0.1							
BnJp	0.0	0.4	0.0	0.0	0.2						
BnUK	0.1	0.0	0.4	0.1	0.5	0.2					
BnGe	0.1	0.0	0.1	0.4	0.5	0.2	0.5				
StUS	-0.3	0.0	-0.1	-0.1	-0.4	-0.1	-0.2	-0.2			
StJp	0.0	-0.3	0.0	0.0	-0.1	-0.4	-0.1	-0.1	0.2		
StUK	-0.1	0.0	-0.3	-0.1	-0.2	-0.1	-0.4	-0.2	0.5	0.2	
StGe	-0.1	0.0	-0.1	-0.3	-0.2	-0.1	-0.2	-0.4	0.5	0.2	0.5

Table 3: Statistical properties — 12 random variables, set 2

	CsUS	CsUK	CsEu	CsNo	BsEu	BlUS	BlJp	BlUK	BlEu	BlNo	StUS	StJp
mean	0.641	-0.114	0.303	0.529	-0.207	-0.145	0.36	-0.707	-0.075	0.765	4.0	2.5
stdev	0.35	0.3	0.3	0.3	1.05	2.38	1.62	2.1	2.25	2.475	12.59	16.14
skew	0.2	0.2	0.2	0.3	0	-0.1	-0.2	-0.2	-0.1	-0.1	0.2	0.2
kurt	3.2	3.2	3.2	3.3	2.8	2.8	2.8	2.8	2.8	2.8	3.2	3.2

	CsUS	CsUK	CsEu	CsNo	BsEu	BlUS	BlJp	BlUK	BlEu	BlNo	StUS
CsUK	0.2										
CsEu	0.2	0.3									
CsNo	0.2	0.2	0.2								
BsEu	0.2	0.1	0.4	0.0							
BlUS	0.2	0.0	0.0	0.0	0.0						
BlJp	0.0	0.0	0.0	0.0	0.0	0.1					
BlUK	0.1	0.3	0.1	0.0	0.3	0.4	0.1				
BlEu	0.1	0.1	0.3	0.0	0.6	0.4	0.1	0.4			
BlNo	0.0	0.2	0.0	0.2	0.4	0.4	0.1	0.4	0.6		
StUS	0.1	0.0	0.0	0.0	0.0	0.25	0.1	0.2	0.2	0.1	
StJp	0.0	0.0	0.0	0.0	0.0	0.2	0.3	0.1	0.2	0.1	0.3

Table 4: Statistical properties — 20 random variables, set 1

	C1US	C1Jp	C1UK	C1Eu	C1No	C2US	C2Jp	C2UK	C2No	BsUS
mean	0.217	0.073	-0.13	-0.085	0.0	0.378	0.0226	-0.0237	0.108	-0.235
stdev	0.3	0.08	0.25	0.2	0.5	0.5	0.16	0.4	1	0.525
skew	-0.2	-0.4	0.0	0.0	0.0	0.2	-0.4	0.0	0.0	-0.2
kurt	2.5	2.2	2.5	3	3.3	2.5	2.2	2.5	3.3	2.65
	BlUS	BlJp	BlUK	BlEu	StUS	StJp	StUK	StEu	StNo	BsEu
mean	-0.793	-0.765	-0.721	-0.45	2.0	0.0	2.0	1.0	-1.0	0.042
stdev	1.75	1.35	1.05	1.875	9.496	9.927	8.633	10.359	9.064	0.525
skew	-0.2	-0.3	-0.6	-0.2	0.0	0.3	0.2	0.2	0.0	-0.1
kurt	2.8	2.5	2.8	2.8	3.2	3.2	3.2	3.3	3.5	2.9

	C1US	C1Jp	C1UK	C1Eu	C1No	C2US	C2Jp	C2UK	C2No	BsUS
C1Jp	0.0									
C1UK	0.1	0.0								
C1Eu	0.1	0.0	0.3							
C1No	0.0	0.0	0.1	0.1						
C2US	0.7	0.0	0.0	0.0	0.0					
C2Jp	0.0	0.7	0.0	0.0	0.0	0.0				
C2UK	0.0	0.0	0.7	0.0	0.0	0.1	0.0			
C2No	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.1		
BsUS	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	
BsEu	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.2
BlUS	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.5
BlJp	0.0	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
BlUK	0.1	0.0	0.3	0.0	0.0	0.1	0.0	0.3	0.0	0.1
BlEu	0.1	0.0	0.1	0.2	0.0	0.1	0.0	0.1	0.0	0.1
StUS	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.2
StJp	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
StUK	0.0	0.0	-0.3	-0.1	0.0	0.0	0.0	-0.3	0.0	0.0
StEu	0.0	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.0
StNo	0.0	0.0	-0.1	-0.1	-0.3	0.0	0.0	-0.1	-0.3	0.0

	BsEu	BlUS	BlJp	BlUK	BlEu	StUS	StJp	StUK	StEu
BlUS	0.1								
BlJp	0.0	0.0							
BlUK	0.1	0.4	0.1						
BlEu	0.5	0.4	0.1	0.5					
StUS	0.0	0.3	0.1	-0.1	-0.1				
StJp	0.0	0.1	0.25	0.0	0.1	0.4			
StUK	0.0	0.1	0.1	-0.2	-0.1	0.5	0.3		
StEu	0.0	0.1	0.1	-0.1	0.0	0.4	0.2	0.5	
StNo	0.0	0.1	0.0	0.0	0.0	0.3	0.3	0.4	0.5

Table 5: Statistical properties — 20 random variables, set 2

	C1US	C1Jp	C1UK	C1Eu	C1No	C2US	C2Jp	C2UK	BsEu	BnEu
mean	0.602	0.055	0.175	0.531	0.432	0.655	-0.005	0.472	0.291	0.249
stdev	0.35	0.1	0.3	0.3	0.3	0.35	0.15	0.28	0.495	1.05
skew	0.3	0.0	0.2	0.2	0.3	0.2	-0.2	0.3	0.2	0.0
kurt	3.2	3.2	3.2	3.2	3.3	3	3	3	2.8	2.8

	BlUS	BlJp	BlUK	BlEu	BlNo	StUS	StJp	StUK	StEu	StNo
mean	-0.354	-0.405	0.742	0.165	1.238	3.5	3.5	4	3.5	2.5
stdev	2.38	1.62	2.1	2.25	2.475	10.527	10.075	8.6	10.636	10.948
skew	-0.1	-0.2	-0.2	-0.1	-0.1	0.2	0.2	0.2	0.2	-0.3
kurt	2.8	2.8	2.8	2.8	2.8	3.2	3.2	3.2	3.3	3.5

	C1US	C1Jp	C1UK	C1Eu	C1No	C2US	C2Jp	C2UK	BsEu	BnEu
C1Jp	0.0									
C1UK	0.2	0.0								
C1Eu	0.2	0.0	0.3							
C1No	0.2	0.0	0.2	0.2						
C2US	0.6	0.2	0.2	0.2	0.2					
C2Jp	0.0	0.6	0.0	0.0	0.0	0.0				
C2UK	0.12	0.0	0.6	0.2	0.2	0.2	0.0			
BsEu	0.2	0.0	0.2	0.6	0.2	0.2	0.0	0.2		
BnEu	0.2	0.0	0.1	0.4	0.0	0.1	0.0	0.1	0.6	
BlUS	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	0.0
BlJp	0.0	0.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
BlUK	0.1	0.0	0.3	0.1	0.0	0.1	0.0	0.3	0.2	0.3
BlEu	0.1	0.0	0.1	0.3	0.0	0.1	0.0	0.1	0.4	0.6
BlNo	0.0	0.0	0.2	0.0	0.2	0.0	0.0	0.2	0.0	0.4
StUS	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
StJp	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
StUK	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	-0.1
StEu	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	-0.1	-0.1
StNo	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	-0.1

	BlUS	BlJp	BlUK	BlEu	BlNo	StUS	StJp	StUK	StEu	
BlJp	0.1									
BlUK	0.4	0.1								
BlEu	0.4	0.1	0.4							
BlNo	0.4	0.1	0.4	0.6						
StUS	0.25	0.1	0.2	0.2	0.1					
StJp	0.2	0.3	0.1	0.2	0.1	0.3				
StUK	0.2	0.0	0.3	0.2	0.1	0.5	0.2			
StEu	0.2	0.0	0.2	0.3	0.1	0.4	0.2	0.5		
StNo	0.2	0.0	0.2	0.2	0.1	0.3	0.2	0.3	0.3	

Updates to the published version

Michal Kaut
michal.kaut@iot.ntnu.no*

May 2003

Abstract

This note describes a new development of the scenario-generation algorithm from the paper "A Heuristic for Moment-matching Scenario Generation", published in *Computational Optimization and Applications*, vol. 24, pp. 169–185, 2003. The presented results lead to a better performance of the algorithm, so the note should be of interest to anybody considering implementing the algorithm.

Throughout the note, we assume that the reader is familiar with the paper, so we can use the notation and refer to parts of the algorithm.

Redundance of Step 4 of the algorithm

In the core (idealised) algorithm, presented in Section 2.4 of the paper, we start the generation with an independent random vector $\tilde{\mathcal{X}}$ with moments $TRSFMOM$. These moments are computed in such a way that $\tilde{\mathcal{Y}} = L\,\tilde{\mathcal{X}}$ has moments MOM (which then lead to the specified moments $TARMOM$). In the modified algorithm in Section 2.5, the outcomes $\mathbb{X}$ of the random vector $\tilde{\mathcal{X}}$ serve as a starting point for the iterative loop in Step 5 of the algorithm.

Unfortunately, the moments $TRSFMOM$ required for the random vector $\tilde{\mathcal{X}}$ are often quite extreme—they may not even exist. As a result, $\tilde{\mathcal{X}}$ may be both hard to obtain and, even worse, it may lead to "strange" (non-smooth and/or truncated) distributions.

On the other hand, our testing shows that the iterative procedure in Step 5 converges even if we start with a random vector $\tilde{\mathcal{X}}$ with moments different from $TRSFMOM$. In particular, the algorithm works well if we start with marginals $\tilde{\mathcal{X}}_i$ with standard normal distributions. Our recommendation is thus to skip Step 4 of the algorithm (generation of $\tilde{\mathcal{X}}$) altogether, and sample the marginals $\tilde{\mathcal{X}}_i$ from the standard normal distribution.

*Norwegian University of Science and Technology, N-7491 Trondheim, Norway

Instead of sampling, it is also possible to use a pre-defined discretization of the distribution. This variant then leads to an "almost deterministic" algorithm, i.e. it decreases the differences between trees coming from several runs of the algorithm. (Whether the algorithm becomes truly deterministic depends on the implementation of the solution method used for finding coefficients of the cubic transformation. In our case, the solution method contains some randomness, so the algorithm may give different trees even if we start with the same discretizations of $\tilde{\mathcal{X}}_i$.)

The normal distribution is chosen mostly for convenience, since it is easy to sample from. In addition, the normal distribution is smooth, which seems to be important for stability of optimization models—see Paper 1 of this thesis for a discussion of stability. Even if other distributions have not been tested, we believe that any smooth distribution would work as well.

Problems with low kurtosis

Unfortunately, not all distributions can be obtained by a single cubic transformation of the standard normal distribution: When the kurtosis is too small, the difference between the normal distribution and the target distribution becomes too big.

In this section, we use notation from Paper 4 of this thesis, and denote[1]

$$\text{skewness: } \gamma = \frac{\mu_3}{\sigma^3}$$
$$\text{kurtosis: } \delta = \frac{\mu_4}{\sigma^4} ,$$

where μ_k is the k-th central moment, $\mu_k = \mathbb{E}\big[\big(\tilde{X} - \mathbb{E}\big[\tilde{X}\big]\big)^k\big]$, and $\sigma^2 = \mu_2$. Note that both moments are independent of the value of the mean and variance. For the rest of the section, we thus set mean to zero and variance to one—in conformity with the paper.

First, we should explain what is meant by "too small kurtosis". *Pearson, 1916* proved that, for a given value of skewness γ, there is a lower bound on the possible value of the kurtosis,

$$\delta \geq 1 + \gamma^2 .$$

In addition, *Klaassen et al., 2000* showed that, in the case of unimodal distributions, the bound is

$$\delta \geq \frac{189}{125} + \gamma^2 = 1.52 + \gamma^2 ,$$

[1]The standard notation would be γ_1 for skewness and γ_2 for normalised kurtosis, so our notation is $\gamma = \gamma_1$ and $\delta = \gamma_2 + 3$. The reason for the choice is that we need to divide by the kurtosis later, which is not possible with the standard definition, since γ_2 can be zero.

so all distributions between these two bounds are multi-modal. It is thus not surprising that they can not be obtained by a single cubic transformation of the (unimodal) normal distribution.

The easiest remedy of the problem is to repeat the cubic transformation several times. This, together with trying several starting samples, usually solves the problem, at least with our implementation of the cubic transformation. To illustrate the approach, we have tested $50,000$ combinations of skewness and kurtosis, sampled uniformly from

$$\{(\gamma, \delta),\ \gamma \in [0, 10],\ \delta \in [\delta_\gamma, 2\delta_\gamma]\}\ ,$$

where δ_γ denotes the minimal kurtosis, $\delta_\gamma = 1 + \gamma^2$. For every combination, we start with a sample of $10,000$ outcomes[2] from the standard normal distribution, and try to transform the sample to a distribution with the given skewness and kurtosis, using the cubic transformation. When we do not obtain the desired moments after ten transformations, we mark the combination as inaccessible, otherwise we store the number of transformations.

From the $50,000$ combinations, only 55 were not obtained in 10 transformations—and all were very close to the lower bound: Table 1 presents the maximal and average distances from the bound, both in absolute ($\delta - \delta_\gamma$) and relative ($\delta/\delta_\gamma - 1$) values. For comparison: from the combinations we were able to obtain, the one closest to the lower bound had the absolute distance of 0.014, and relative distance of 0.01%.

Table 1: Distance of the combinations (γ, δ) we were not able to generate, from the theoretical lower bound (γ, δ_γ).

	distance from bound	
statistics	absolute	relative
average	0.015	0.17%
max	0.051	0.93%

We have also created a "map" showing the number of transformations needed to achieve different combinations of skewness and kurtosis. To obtain the map, we had to run the test on the whole region, not only along the bound as in the previous test. The result of the test is presented in Figure 1. An interesting observation is that there is also an upper bound for kurtosis that can be achieved by a single cubic transformation of the standard normal distribution—as far as we know, this has not been reported before. With our

[2]The relatively high number of scenarios was chosen in order to ensure that the starting discretization is sufficiently close to the standard normal distribution.

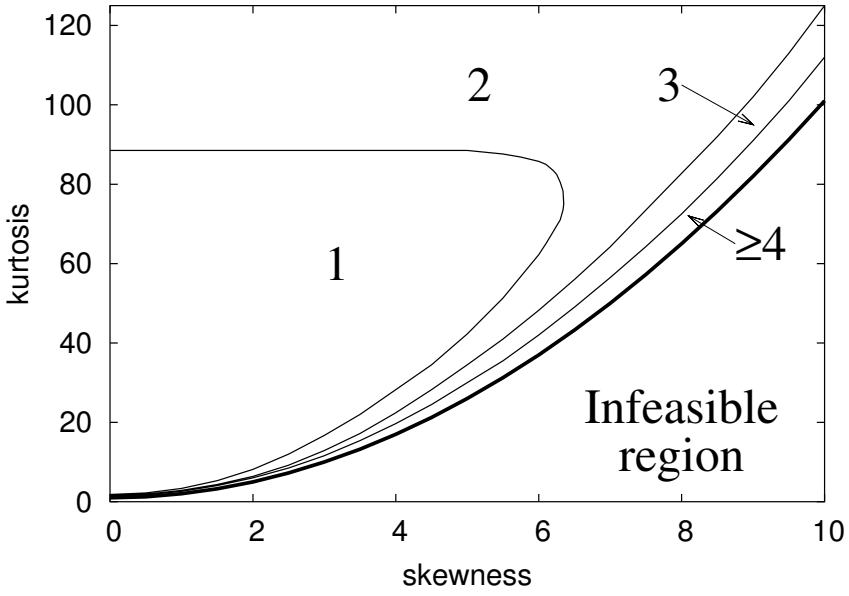

Figure 1: Combinations of skewness and kurtosis accessible by a repeated cubic transformation of the standard normal distribution. The numbers show the number of cubic transformations needed to obtain distributions from the corresponding areas. The lowest region contains infeasible combinations of skewness and kurtosis.

implementation, the upper bound is approximately 88.5. (There is a corresponding bound for two transformations, but it starts at kurtosis of about 3000 for zero skewness and increases to more than 4000 for skewness of 50. Hence, the maximal achievable kurtosis can be seen as unlimited for most practical purposes.)

It is important to realize that the repetition of the cubic transformation would be impossible with the original formulation from *Fleishman, 1978*, since it assumes that the starting distribution is *exactly* normal. Our implementation, however, allows starting with arbitrary distribution, as long as we can compute the first twelve moments.[3] This makes the computation of the coefficients more difficult, but on the other hand practically eliminates the biggest problem of Fleishman's method, the inability of generating distributions with low kurtosis.

An alternative to repeating the cubic transformation is to start with a different distribution than the standard normal. For example, if the kurtosis

[3]This is why we can use it in the iterative loop in Step 5 of the algorithm.

is below the bound for unimodal distributions, we may try a mixture of two normal distributions: Kurtosis decreases with the distance of the two components of the mixture. In some occasions, uniform distribution may help—this is, however, not recommended because of the non-smoothness of the distribution.

As a more sophisticated alternative, we may consider using a different method than the cubic transformation to obtain the starting value $\mathbb{Y}_i$ for the random variable $\tilde{\mathcal{Y}}_i$ in Step 5 of the algorithm. A good source of information is *Tadikamalla, 1980*, who presents and compares six different methods, where three are capable of generating distributions with any feasible combination of skewness and kurtosis: the Johnson system of distributions from *Johnson, 1949*, the Tadikamalla-Johnson system from *Tadikamalla and Johnson, 1979*, and the Schmeiser-Deutch system from *Schmeiser and Deutch, 1997*.

Note that the special approach—either repeating the cubic transformation, or using some of the mentioned alternatives—is needed only in order to get a starting point for the iterative procedure in Section 5. Inside the loop, we always use the cubic transformation: Even if the cubic transformation is not capable of entering the low-kurtosis region, it works inside it.

New implementation

As mentioned in the "Future work" part of the paper, we have implemented the algorithm in the `C` programming language. The cubic transformation, which is the crucial part of the algorithm, was implemented by Diego Mathieu from INSA Toulouse, France, during his visit at Molde University College in the summer of 2002. The code has been compiled for `Win32` using both `Microsoft Visual C++` and `MinGW` (`GCC` for `Win32`), so it should compile on other platforms as well.

The new implementation provides two major improvements compare to the original `AMPL` implementation: It is more than ten times faster, and it is a stand-alone code, so it is not dependent on any commercial solver.

The issue of distributions with low kurtosis has been addressed by allowing several repetitions of the cubic transformation. Diego Mathieu has also implemented the Schmeiser-Deutch system from *Schmeiser and Deutch, 1997*, but it has not yet been included in the code—mostly because we have not yet encountered a real case where we could not generate the scenarios using the current implementation.

References

Fleishman, A. I. (1978). A method for simulating nonnormal distributions. *Psychometrika*, 43:521–532.

Høyland, K., Kaut, M., and Wallace, S. W. (2003). A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2-3):169–185.

Johnson, N. L. (1949). Systems of frequency curves generated by methods of translation. *Biometrika*, 36(1):149–176.

Klaassen, C., Mokveld, P., and Es, B. V. (2000). Squared skewness minus kurtosis bounded by 186/125 for unimodal distributions. *Statistics & Probability Letters*, 50:131–135.

Pearson, K. (1916). *Mathematical Contributions to the Theory of Evolution, XIX: Second Supplement to a Memoir on Skew Variation*. Ser. A. Philos. Trans. Roy. Soc., London.

Schmeiser, B. W. and Deutch, S. J. (1997). A versatile four parameter family of probability distributions suitable for simulation. *AIIE Transactions*, 9(2):176–182.

Tadikamalla, P. and Johnson, N. L. (1979). Systems of frequency curves generated by transformations of logistic variables. Mimeo Series 1226, Department of Statistics, University of North Carolina at Chapel Hill, Chapel Hill.

Tadikamalla, P. R. (1980). On simulating non-normal distributions. *Psychometrika*, 45(2):273–279.

Paper 4

Multi-period scenario tree generation using moment-matching: Example from option pricing

Multi-period scenario tree generation using moment-matching: Example from option pricing

Michal Kaut
michal.kaut@iot.ntnu.no*

Stein W. Wallace
stein.w.wallace@himolde.no†

June 2003

Abstract

This paper presents an algorithm for generating multi-period non-recombining scenario trees, and shows how it can be used for pricing multi-variate path-dependent options.

Recently, the moment-matching approach to scenario generation has gained some popularity in the stochastic programming community. In most cases, the moment matching is done on one-period trees. Multi-period trees are then generated by single-period subtrees, so it is possible to evolve the conditional moments and correlations of the subtrees according to specified rules (for example autocorrelation or mean-reversion). Unfortunately, this approach does not allow the user to control the unconditional distribution of the final period of the whole multi-period tree.

Option pricing is an example of an application where the final-period distribution plays a crucial role, and the above approach is thus not directly applicable. In this paper, we present formulas for moments and correlations of the one-period subtrees that result in given moments and correlations of the distribution of the final period of the multi-period tree, in the case without inter-period dependencies. With inter-period dependencies, we provide an ex-post correction procedure.

Non-recombining, multi-period trees typically lead to an explosion in both storage space and computation time. We show, however, that in the case of option pricing we can avoid the explosion in the storage space by simultaneous tree generation and pricing. The presented algorithm assumes that we know the moments and correlation of the risk-neutral distribution (equivalent martingale measure).

Keywords: non-recombining trees, scenario generation, option pricing, multivariate options

*Norwegian University of Science and Technology, N-7491 Trondheim, Norway
†Molde University College, Postboks 2110, N-6402 Molde, Norway

1 Introduction

Despite some criticism—for example by Pflug and Hochreiter (2003)—the moment matching approach to scenario generation has been accepted by the stochastic programming community, mostly because of its reasonable performance in practical applications: See, for example, Kouwenberg (2001), or Kaut and Wallace (2003).

In most cases, the matching of moments (and possibly other properties) is done on one-period trees. Hence, when we need to generate a multi-period tree, we generate it by the constituting one-period subtrees. Høyland and Wallace (2001) show how to use this procedure to evolve the conditional moments according to specified rules, such as mean-reversion or volatility clumping. Consequently, there is no direct control over the *unconditional* distribution of the final period of the multi-period tree. Whether this is a problem depends on the use of the scenario tree.

In this paper, we present a method for generating multi-period trees with control over the moments and correlations of the unconditional distribution. In the case without inter-period dependencies, we present formulas for moments and correlations of the one-period subtrees, that result in specified moments and correlations of the unconditional distribution. With inter-period dependencies, we present an iterative procedure for ex-post correction of the first two moments.

Tree-based option pricing is an example of an application where the unconditional (final-period) distribution is crucial for the result (the option price). In the standard case of binomial trees, we actually specify only the final-period distribution, and the number of periods (the size of the tree).

To price an option on a tree, we need to know the risk-neutral distribution (equivalent martingale measure). In the binomial trees, the risk-neutral probabilities are computed directly from the prices and the risk-free rate. For non-recombining trees with more than two branches, however, this is no longer possible, and we need to know the risk-neutral distribution in advance. Typically, the risk-neutral distribution is estimated from option prices. In this paper, we assume that we have already made the estimation, and thus know the risk-neutral distribution—or, more precisely, its moments and correlations. For information about recovering the risk-neutral distribution from option prices, see for example Jackwerth and Rubinstein (1996), Fornari and Violi (1998), or Rosenberg (1999, 2003).

Since we use non-recombining trees, we face the usual "curse of dimensionality", i.e. the exponential growth of the tree with the number of periods. Unlike stochastic programming, where the whole tree has to be stored prior

to the solution of the optimisation problem, in option pricing we can generate the tree and price the option simultaneously. This decreases the storage-space drastically—in the presented algorithm, the storage place depends linearly on the number of stages. As a result, we can process a tree with 250 million terminal nodes in approximately half a minute on a 1GHz PC. This number of nodes corresponds to a 12-period tree with five branches per node—if there is a need for a significantly finer time-discretization, the method is not applicable.

In addition to the multi-period algorithm, we show how the moment-based approach provides an easy way to price multi-variate path-independent European options, using large one-period trees.

The rest of the paper is organised as follows: Section 2 presents the notation used throughout the paper. In Section 3, we discuss ways of controlling the unconditional distribution of a multi-period tree, by adjusting the distributions of the one-period subtrees it consists of. In Section 4, we present selected information about option pricing, needed to understand the rest of the paper. We also show how to use the moment-based approach for pricing multi-variate European options. Finally, Section 5 presents the combined generating/pricing algorithm, together with a numerical example and a discussion of numerical stability.

2 Description of a multi-period tree

Throughout the paper, we assume that the reader is familiar with the concept of scenario trees. For information about scenario trees and their role in stochastic programming, see for example Dupačová et al. (2000).

In the description of a multi-period scenario tree, we use the following conventions: We call the points in time represented by nodes *stages*, and the time intervals between them *periods*. Hence, a tree with p periods has $p + 1$ stages. The root of the tree is indexed as stage zero, so period t is an interval between stages $t-1$ and t. A distribution of random variables with outcomes at stage t can thus be referred to as a distribution *of* period t, or as a distribution *at* stage t.

In a multi-period tree, we have to distinguish between conditional and unconditional distributions. For example, in Figure 1, nodes $\{3, 4, 5\}$ represent a distribution of the second period (or a distribution at stage 2), conditional on the values at node 1. The *conditional probabilities* of node $\{3, 4, 5\}$ sum up to one. On the other hand, the *unconditional distribution* of the second period is represented by nodes $\{3, 4, 5, 15, 16, 17\}$. Specifically, the unconditional distribution of the final period of a tree is hereafter referred to as a *final-stage*

distribution.

In addition, by generating a subtree of node n we understand generating a one-period tree rooted at node n, i.e. finding the outcomes at the direct successors (children) of node n, and the probabilities thereof. For example, generating a subtree of node 1 in Figure 1 means finding the values of the random variables at nodes 3, 4 and 5, as well as the probabilities of these nodes.

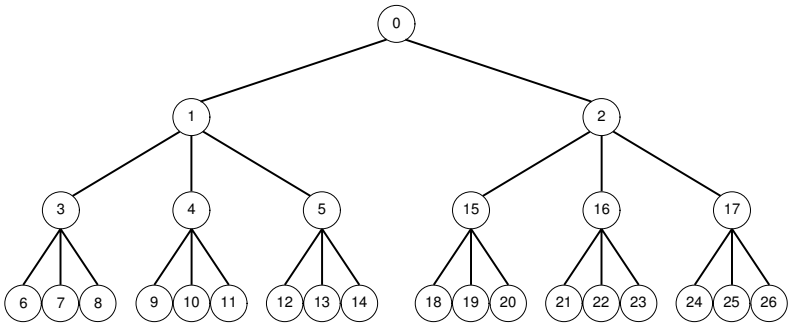

Figure 1: Example of a multi-period tree

The multi-period tree is constructed by one-period subtrees, starting in the root. The subtrees are generated using a moment-matching approach from Høyland and Wallace (2001), so every subtree is constructed to have specified first four moments (mean, variance, skewness, kurtosis) and correlations. Generating the multi-period tree by subtrees allows us to specify an inter-period dependency: We generate a subtree, move to one of the nodes, update the distribution of the consecutive subtree (based on the original distribution and the outcomes at the node and its predecessors), and generate the subtree with the updated distribution. See Appendix B for an example of the update formulas, in the case of first order autocorrelation.

When generating scenario trees for a stochastic process $\{\tilde{\boldsymbol{V}}_t\}$, we work with distributions of its returns $\tilde{X}_t$; both the arithmetic returns $X_t = {}^{V_t}/_{V_{t-1}} - 1$, and the geometric returns (also called "log-returns") $X_t = \ln(V_t) - \ln(V_{t-1})$, are considered throughout the paper. Hence, the scenarios are generated using the distribution of returns $\tilde{X}_t$, and the values of $\tilde{V}_t$ are computed afterwards as $V_t = (1 + X_t)V_{t-1}$ and $V_t = \mathrm{e}^{X_t}V_{t-1}$, respectively. This is a common approach. The main reason for using the returns is that their distributions are independent of the previous values of V_t. We may, for example, have the same distribution of returns in all the one-period subtrees of a multi-period tree, something that would be very difficult to describe in terms of $\tilde{V}_t$.

3 Controlling the final-stage distribution

There are (at least) two possible approaches to generation of multi-period scenario trees: We may be concerned with the distributions of the one-period subtrees and with the inter-period dependencies, or we may focus on the final-stage distribution. We take the latter approach—for information about the former, see for example Høyland and Wallace (2001).

As explained in Section 2, we generate the multi-period trees by their constituting one-period subtrees. Yet, we want to control the unconditional distribution of the final period of the whole tree. Hence, the problem is to find the right moments and correlations for the one-period subtrees, given the moments and correlations of the final-stage distribution, the number of periods and possibly the inter-period dependency rules and the outcomes of the predecessors of the subtree.

First, we focus on the case with no inter-period dependency, i.e. the case where the distribution of a subtree of a given node does not depend on the outcomes at the predecessors of this node. In this case, we provide formulas for moments and correlations of the subtrees that result in a final-stage distribution with given properties.

In many situations, however, the assumption of independent periods is not very realistic. In finance, for example, effects like autocorrelation and mean-reversion are commonplace. In the second part of this section we thus look at trees with inter-period dependencies. In this case, however, we did not manage to find corresponding formulas for controlling the subtree distributions. Instead, we present a procedure that generates a tree and thereafter corrects the means and variances to obtain the required unconditional distribution (distribution over all nodes in the same stage). Used in a loop, the procedure corrects the means and variances in the whole tree.

3.1 Independent periods

In this section we present formulas for the moments and correlations of the distribution on the subtrees that will result in a multi-period tree with the correct moments and correlations at the final stage. The formulas rely on an assumption that all the subtrees have the same distributions, i.e. that there is no inter-period dependency. An important property of the formulas is that they are independent of the number of branches in the scenario tree. A detailed derivation of the formulas is in Appendix A.

Throughout the section we use the following notation: In a multi-period

tree with p periods we denote by $\mu, \sigma^2, \gamma, \delta$ the first four moments[1] of the final-stage distribution, and $\mu_p, \sigma_p^2, \gamma_p, \delta_p$ the first four moments of the corresponding subtrees.[2] For two random variables $\tilde{X}$ and $\tilde{Y}$, we extend the notation by ρ for their final-stage correlation, and ρ_p for the correlation in the subtrees. In addition, we add superscripts X and Y to the moments defined above, in order to distinguish between moments of $\tilde{X}$ and $\tilde{Y}$ where needed.

Arithmetic and geometric returns, defined in Section 2, are discussed separately. Note that, in addition to the differences in formulas, there is also a computational difference between the two cases: Updating prices with the geometric returns requires a computation of the exp() function, and is therefore significantly slower than updating with the arithmetic returns.

Formulas for arithmetic returns

A process $\{\tilde{\boldsymbol{V}}_t\}$ with arithmetic returns $\tilde{X}_t$ evolves as

$$\tilde{V}_t = (1 + \tilde{X}_t)\tilde{V}_{t-1} \ ,$$

so the final-stage return $\tilde{V}_p/\tilde{V}_0 - 1$ is equal to

$$\prod_{k=1}^{p}(1 + \tilde{X}_k) - 1 \ .$$

In Appendix A we show that, in order to get a tree with properties μ, σ^2, γ, δ, and ρ, we have to use subtrees with:

$$\mu_p = (1+\mu)^{1/p} - 1 \tag{1a}$$

$$\sigma_p^2 = \left((1+\mu)^2 + \sigma^2\right)^{1/p} - (1+\mu_p)^2 \tag{1b}$$

$$\gamma_p = \frac{1}{\sigma_p^3}\left[\left((1+\mu)^3 + 3(1+\mu)\sigma^2 + \sigma^3\gamma\right)^{1/p} - (1+\mu_p)^3 - 3(1+\mu_p)\sigma_p^2\right] \tag{1c}$$

$$\begin{aligned}\delta_p = \frac{1}{\sigma_p^4}\Big[&\left((1+\mu)^4 + 6(1+\mu)^2\sigma^2 + 4(1+\mu)\sigma^3\gamma + \sigma^4\delta\right)^{1/p} \\ &-(1+\mu_p)^4 - 6(1+\mu_p)^2\sigma_p^2 - 4(1+\mu_p)\sigma_p^3\gamma_p\Big]\end{aligned} \tag{1d}$$

$$\rho_p = \frac{1}{\sigma_p^X\sigma_p^Y}\left[\left((1+\mu^X)(1+\mu^Y) + \sigma^X\sigma^Y\rho\right)^{1/p} - (1+\mu_p^X)(1+\mu_p^Y)\right] \tag{1e}$$

[1] The standard notation is γ_1 for skewness and γ_2 for normalised kurtosis. Since we need indices for the moments, we have introduced a new notation $\gamma = \gamma_1$ and $\delta = \gamma_2 + 3$.

[2] The index p thus means that these properties are implied by their respective final-stage values, and the number of periods p.

This, for example, means that in order to get the standard normal distribution at the final stage of a p-periodic tree ($\mu = 0,\ \sigma^2 = 1,\ \gamma = 0,\ \delta = 3$), the moments of the subtrees have to be

$$\begin{aligned} \gamma_p &= \frac{2^{1/p} - 2}{\sqrt{2^{1/p} - 1}} \quad \xrightarrow[p\to\infty]{} -\infty \\ \delta_p &= \frac{1}{\sigma_p^4}\left(10^{1/p} - 4\cdot 4^{1/p} + 6\cdot 2^{1/p} - 3\right) \quad \xrightarrow[p\to\infty]{} \infty. \end{aligned} \tag{2}$$

If we use subtrees with the correct mean and variance, but keep them normal (i.e. $\gamma = 0$ and $\delta = 3$), the resulting p-periodic tree would have skewness and kurtosis:

$$\begin{aligned} \gamma &= \left(3\cdot 2^{1/p} - 2\right)^p - 4 \quad \xrightarrow[p\to\infty]{} 4 \\ \delta &= \left(3\cdot 4^{1/p} - 2\right)^p - 4\left(3\cdot 2^{1/p} - 2\right)^p + 9 \quad \xrightarrow[p\to\infty]{} 41 \end{aligned} \tag{3}$$

Note also that zero correlation in the subtrees leads to a zero correlation in the final-stage distribution. Generally, the correlation is very stable, i.e. the difference between ρ and ρ_p is typically small.

Formulas for geometric returns

A process $\{\tilde{\boldsymbol{V}}_t\}$ with geometric returns $\tilde{X}_t$ evolves as

$$\tilde{V}_t = \mathrm{e}^{\tilde{X}_t}\tilde{V}_{t-1}\ ,$$

so the final-stage return $\ln\left(\tilde{V}_p/V_0\right)$ is equal to

$$\ln(\tilde{V}_p) - \ln(V_0) = \ln\left(\mathrm{e}^{\sum_{k=1}^{p}\tilde{X}_k}V_0\right) - \ln(V_0) = \sum_{k=1}^{p}\tilde{X}_k\ .$$

In Appendix A we show that in order to get a tree with properties μ, σ^2, γ, δ, and ρ, we have to use subtrees with:

$$\mu_p = p^{-1}\mu \qquad \gamma_p = \sqrt{p}\,\gamma \tag{4a}$$

$$\sigma_p^2 = p^{-1}\sigma^2 \qquad \delta_p = p(\delta - 3) + 3 \tag{4b}$$

$$\rho_p = \rho \tag{4c}$$

Unlike the arithmetic returns, for geometric returns we know that a sum of independent normal distributions is again normally distributed, so $(\gamma_p, \delta_p) = (0, 3)$ implies $(\gamma, \delta) = (0, 3)$, and vice versa. In addition, for any fixed γ_p and δ_p, we know that the final-stage distribution converges to the normal distribution as p goes to infinity, hence $\gamma \xrightarrow[p\to\infty]{} 0$ and $\delta \xrightarrow[p\to\infty]{} 3$.

Example

As an example, we use a risk-neutral distribution of returns of S&P 500 and DAX 30 indices. The distribution comes from Rosenberg (2003), and was estimated using data from December 1999. We will use the same distribution also later in the paper. Even if the data represent arithmetic returns, we use them both as arithmetic and geometric returns, to demonstrate the difference between the two.

Table 1 presents moments and correlations of the one-period subtrees, which result in the given final-stage distribution. Moments and correlations of the final period can thus be found in the first column. We see that there is a substantial difference in behaviour of the arithmetic and geometric returns: In the first case, the subtrees are much more non-normal. As a result, the minimum number of branches of the subtrees, needed to obtain the desired properties, is usually higher in the case of arithmetic returns, since more extreme distributions typically need more scenarios.

Table 1: Moments of subtrees to get a given final-stage distribution

		# periods	1	2	4	6	8	10	12
arithmetic returns	S&P 500	mean	0.0061	0.0030	0.0015	0.0010	0.0008	0.0006	0.0005
		std.dev.	0.0645	0.0454	0.0321	0.0262	0.0227	0.0203	0.0185
		skew	-0.580	-0.952	-1.438	-1.800	-2.100	-2.362	-2.598
		kurt	3.800	5.061	7.627	10.203	12.781	15.361	17.940
	DAX 30	mean	0.0036	0.0018	0.0009	0.0006	0.0004	0.0004	0.0003
		std.dev.	0.0861	0.0607	0.0429	0.0350	0.0303	0.0271	0.0247
		skew	-0.360	-0.685	-1.092	-1.388	-1.632	-1.844	-2.035
		kurt	3.380	4.183	5.867	7.568	9.273	10.980	12.688
		corr	0.5192	0.5196	0.5198	0.5198	0.5199	0.5199	0.5199
geometric returns	S&P 500	mean	0.0061	0.0030	0.0015	0.0010	0.0008	0.0006	0.0005
		std.dev.	0.0645	0.0456	0.0322	0.0263	0.0228	0.0204	0.0186
		skew	-0.580	-0.820	-1.160	-1.421	-1.640	-1.834	-2.009
		kurt	3.800	4.600	6.200	7.800	9.400	11.000	12.600
	DAX 30	mean	0.0036	0.0018	0.0009	0.0006	0.0004	0.0004	0.0003
		std.dev.	0.0861	0.0609	0.0430	0.0351	0.0304	0.0272	0.0248
		skew	-0.360	-0.509	-0.720	-0.882	-1.018	-1.138	-1.247
		kurt	3.380	3.760	4.520	5.280	6.040	6.800	7.560
		corr	0.5192	0.5192	0.5192	0.5192	0.5192	0.5192	0.5192

3.2 Dependent periods

In many cases, the assumption of independent periods is too restrictive. In finance, for example, effects like mean-reversion or autocorrelation are very common. There are many models of inter-period dependency, see for example Melamed (1991), Song et al. (1996), or Cario and Nelson (1997b).

Instead on focusing on modelling of a special type of dependency, we discuss what can be done in the case when we need a scenario tree with dependent periods, but also with some degree of control over the unconditional distributions (distributions over all nodes in the same stage). As an example, Appendix B presents a way to implement first order autocorrelation, in a way that allows, at the same time, to control the unconditional means and variances.

In the case of a general inter-period dependency, we did not manage to repeat the results from the previous section, and produce formulas for controlling the final-stage distribution. The best we can do is to control the unconditional means and variances by a procedure that first generates the tree, and then corrects the unconditional distribution. This procedure is independent of the type of the inter-period dependency used.

The unconditional distributions are corrected one period at a time. Hence, we need to know the desired means and variances for distributions of all the periods—if we know only the moments of the final-stage distribution, we have to approximate the rest, in order to do the correction.

Correcting means and variances for one period

Here we show how to correct the unconditional means and variances for a given period t, i.e. the period between stages $t-1$ and t. We want to change the means and variances of all the one-period subtrees in period t, so that the unconditional distribution of period t will get the desired means $\boldsymbol{\mu}_t$ and variances $\boldsymbol{\sigma}_t^2$.

We generate a tree with (at least) t periods, and compute the current values of the unconditional means and variances at period t, $\bar{\boldsymbol{\mu}}_t$ and $\bar{\boldsymbol{\sigma}}_t^2$. If these differ from the desired values $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t^2$, we start the update: For every node n in period $t-1$, we compute the means and variances of the one-period subtree rooted at n, $\mathbb{E}\big[\tilde{\boldsymbol{X}}_t^n\big]$ and $\mathrm{Var}\big[\tilde{\boldsymbol{X}}_t^n\big]$. The values are then updated using a linear transformation:

$$\begin{aligned} \mathbb{E}\big[\tilde{\boldsymbol{X}}_t^n\big] &\leftarrow \frac{\boldsymbol{\sigma}_t}{\bar{\boldsymbol{\sigma}}_t}\mathbb{E}\big[\tilde{\boldsymbol{X}}_t^n\big] + \boldsymbol{\mu_t} - \frac{\boldsymbol{\sigma}_t}{\bar{\boldsymbol{\sigma}}_t}\,\bar{\boldsymbol{\mu}}_t \\ \mathrm{Var}\big[\tilde{\boldsymbol{X}}_t^n\big] &\leftarrow \frac{\boldsymbol{\sigma}_t}{\bar{\boldsymbol{\sigma}}_t}\mathrm{Var}\big[\tilde{\boldsymbol{X}}_t^n\big]\ . \end{aligned} \tag{5}$$

Controlling unconditional means and variances at every period

To control the means and variances of the whole tree we apply the formulas from the previous section iteratively:

```
for t in 2 to p {
  generate tree with t periods, using corrections already found for t' < t
  compute the unconditional means and variances in period t
  compute the corrections of means and variances in period t
}
generate tree with p periods, using corrections for all t' ∈ {2...p}.
```

Note that we have to generate $p-1$ trees (growing in size from 2 to p periods), before we can generate the whole p-period tree with the right properties. Since the size of the trees grows exponentially with the number of periods, the whole procedure can be expected to take 2 to 3 times longer than the generation of one p-period tree.

4 Option pricing using a risk-neutral measure

Currently, the two most common approaches for option pricing are the Black-Scholes continuous-time framework, and Cox, Ross, Rubinstein discrete-time approach using binomial (recombining) trees. While the former is an exact formula for plain European options, the latter is an approximate approach used mainly for path-dependent and/or Bermudan options. Both of the approaches have, at least in their "classical" versions, the disadvantage of assuming that the asset returns are normally distributed. Recently, there have appeared attempts to drop the normality assumption, both in the theoretical B-S-like context (see for example Jurczenko et al. (2002), Vitiello et al. (2002)), and in the context of binomial or trinomial recombining trees (Rubinstein (1994), Derman and Kani (1994), Derman et al. (1996), Rubinstein (1998)).

Yet, most of the methods can price only options depending on one asset. In recent years, however, several types of multi-variate options (also called basket options) have appeared. An example is the option to buy the better of two indices. Methods for pricing such options were also presented, yet most of them are based on the normality assumption. Beyond normality, the pricing tools are still scarce: Cherubini and Luciano (2002) provide a theoretical derivation for pricing of bivariate options, while Rosenberg (1998, 1999, 2003) price bivariate options using binomial trees. All of these methods are based on copulas as a description of the multi-variate distribution. Since

we believe that not all the readers are familiar with copulas, we present some basic information about copulas in Figure 2. Note, however, that the figure is not a prerequisite for understanding the rest of the paper.

The presented moment-based approach is simpler than the copula-based approaches mentioned above. In addition, we do not need any distributional assumptions (assumptions on the type/family of the distribution). Hence, the method may be useful in contexts where a description of the risk-neutral distribution by moments and correlations is more natural than using copulas—for example because we do not have enough information/data to estimate the copula, or we want to avoid the various assumptions (on type of copula, or on the density function of the marginals), needed for the copula estimation.

There is, however, a price for the simplicity: For some of the basket options, the quality of the multi-variate structure of the return distribution becomes very important. In such cases, correlation coefficients may not be enough to describe the dependencies, and the resulting prices become only approximations. The method is thus best suited for cases where either the copula approach is not applicable, or only an approximation of the option prices is needed. As an example of the latter, consider the case of real options, where the risk-neutral distribution often is not known perfectly, so only an approximation of the price is obtained regardless of the method used.

4.1 Path-independent European options

Path-independent European options are a special case, since their prices are determined only by the distribution of asset prices at the exercise time, together with the initial parameters. Hence, when we price a path-independent European option using a tree, only the final-stage distribution affects the price. It is thus enough to use a single-period tree. The problem thus becomes a multivariate integration problem, with the distribution specified by its moments and correlations, instead of the distribution function.

In Section 3, we saw that it is not trivial to generate a multi-period tree with a given distribution of the last period. On the other hand, there are several methods for generating single-period trees with a specified distribution, see for example Song et al. (1996), Cario and Nelson (1997a), Kouwenberg (2001), Lurie and Goldberg (1998), Høyland et al. (2003), Lyhagen (2001), or Gülpınar et al. (2001). We use a moment-matching method from Høyland et al. (2003), which gives us control over the first four moments and the correlations of the random variables. Once we have the tree for the risk-neutral distribution, the price of the option is simply the expected value of the distribution, i.e. weighted sum of the prices in the tree.

The name *copula* was first used in Sklar (1996) to describe "a function that links a multidimensional distribution to its one-dimensional margins". The mathematical formulation comes from Sklar (1959).

For an n-dimensional stochastic vector $\tilde{\boldsymbol{X}}$, an associated *copula* is such a function $\boldsymbol{C}() : [0,1]^n \to [0,1]$ that

$$F(x_1, \ldots, x_n) = \boldsymbol{C}\big(F_1(x_1), \ldots, F_n(x_n)\big) \ ,$$

where $F()$ is the joint (n-variate) distribution function of $\tilde{\boldsymbol{X}}$, and $F_i()$ are the marginal distribution functions, $i \in 1, \ldots, n$. *Sklar's theorem* states that copula exists for every distribution function $F()$. An immediate consequence of the theorem is that, for every $\boldsymbol{u} = (u_1, \ldots, u_n) \in [0,1]^n$,

$$\boldsymbol{C}(u_1, \ldots, u_n) = F\big(F_1^{-1}(u_1), \ldots, F_n^{-1}(u_n)\big) \ ,$$

where $F_i^{-1}()$ is the generalized inverse of $F_i()$.

As an example consider two independent random variables $\tilde{X}$ and $\tilde{Y}$ with $F(x,y) = F_X(x)F_Y(y)$. The associated copula is thus $\boldsymbol{C}(u,v) = uv$.

Estimating a copula from data is analogous to estimating a distribution function in the sense that we can either use an empirical copula, or assume a parametric family for the copula and estimate the parameters. For the latter approach we may also need to assume distributional families of the marginals.

In addition to the original papers Sklar (1959, 1996), Nelsen (1998), and Clemen and Reilly (1999) present an introduction to copulas. For copula-based option-pricing techniques, see Rosenberg (1998, 1999, 2003), or Cherubini and Luciano (2002).

Figure 2: Basic information about copulas

Example

The following example comes from Rosenberg (2003), and is based on the S&P 500 and DAX 30 indices presented in Section 3.1. We consider two different one-month call options: an underperformance option, which gives the buyer a right to buy the worse of the two indices, and an outperformance option that gives a right to buy the better of the two indices. In both cases, the strike price is equal to the spot price, and we buy indices for \$100. Hence, the outcomes of the two options in scenario s are:

$$\begin{aligned} \text{underperformance:} \quad & \$100 \ \max\big\{\min\{r^s_{\text{S\&P}}, r^s_{\text{DAX}}\}, 0\big\} \\ \text{outperformance:} \quad & \$100 \ \max\big\{\max\{r^s_{\text{S\&P}}, r^s_{\text{DAX}}\}, 0\big\} \ , \end{aligned}$$

where $r^s_{\text{S\&P}}$ and $r^s_{\text{S\&P}}$ denote the arithmetic returns of the indices in scenario s.

To achieve reasonable stability, we have used trees with 10.000 scenarios—it still takes less than a second to both generate the tree and price the options. Using 10 runs, we have estimated the price of the underperformance option as $\$1.72 \pm \$.025$, and the outperformance option as $\$4.74 \pm \$.025$.

In Rosenberg (2003), the prices are estimated as \$1.52 and \$4.86, respectively, making our prices 13% and 2% different. The difference[3] comes probably from differences in the bi-variate distributions: We control only the correlation coefficient, which is not enough to get the same structure of the bi-variate distribution as in Rosenberg (2003). This is also confirmed by the fact that the difference is much bigger for the underperformance option, which is more sensitive to the bi-variate dependency structure—see Rosenberg (2003) for a detailed discussion.

The difference between our prices and the prices from Rosenberg (2003) shows that the correlation alone may not be enough for an accurate description of the bi-variate dependency structure. This may result in a bias in the estimated price. Hence, when we need a more accurate estimation of the price of an multi-variate option, we have to either add more co-moments (such as co-skewness) to our approach, or use another method. Obvious candidates are copula-based methods, provided we have enough data/information to estimate the copula.

5 Generating/pricing algorithm

In this section, we describe the algorithm for simultaneous generation of a multi-period tree, and pricing an option based on this tree. In order to use the algorithm, we need to know the distributions of all the one-period subtrees of the multi-period tree—or, more precisely, their moments and correlations. How to get these was discussed in Section 3, so we take all the distributions for granted throughout this section.

The important property of option pricing is that the price at every node depends only on the outcomes at the node and the prices of its direct successors (children). Hence, once we have computed the price at a given node, we can discard all its successors. When we use a post-order traversal of the tree, we need to store at most one path from the root to the leafs.[4] The number of trees

[3] Note that the prices in Rosenberg (2003) are approximate as well, so a difference does not necessarily mean an error.

[4] This is an important difference from generating a scenario tree for an optimization model, where we need to store the whole tree prior to the optimization. The option-pricing trees can thus be much bigger than the scenario trees used in optimization.

stored at any moment is thus at most equal to the number of periods p—an enormous decrease of the required storage, compared to storing of the whole tree.

As an example, consider generating/pricing the tree from Figure 1. Table 2 describes the procedure step-by-step, showing for every step nodes that are generated, nodes in which we calculate the price, and nodes that can be discarded. Figure 3 represents the same procedure graphically.

Table 2: Step-by-step generation and pricing of the tree from Figure 1. For example, in step 4 of the algorithm, we first generate outcomes at nodes 9–11. Since these nodes are leafs of the tree, we can compute prices at the nodes. Then we can compute price at node 4, and drop the nodes 9–11. Hence, after the step, we have outcomes in nodes 0–5, and prices at nodes 3–4.

	new values at nodes		dropped nodes	nodes stored after the step	
step	outcomes	prices	outcomes & prices	outcomes	prices
1	1–2			0,1–2	
2	3–5			0,1–2,3–5	
3	6–8	6–8,3	6–8	0,1–2,3–5	3
4	9–11	9–11,4	9–11	0,1–2,3–5	3–4
5	12–14	12–14,5,1	12–14,3–5	0,1–2	1
6	15–17			0,1–2,15–17	1
7	18–20	18–20,15	18–20	0,1–2,15–17	1,15
8	21–23	21–23,16	21–23	0,1–2,15–17	1,15–16
9	24–26	24–26,17,2,0	24–26,15–17,1–2	0	0

A complete generating/pricing algorithm is presented in Figure 4. Note that the algorithm is written in a recursive form only for the sake of simplicity: Since recursive algorithms are usually slower than their non-recursive versions, the actual implementation is not recursive (and therefore significantly more complicated than the presented version). The formulation also assumes that all the structures (outcomes and prices) are local to the function *value*(), and are disposed of automatically by the system at the moment of the departure from the function.

5.1 Pre-generation of subtrees

Since the size of a multi-period tree grows exponentially with the number of periods p, the efficiency of the whole procedure becomes crucial. The most time-consuming task of the algorithm presented in Figure 4 is the generation of subtrees, because the rest consists of simple algebraic operations. In some cases we can, however, avoid generating all the subtrees, speeding up the procedure substantially.

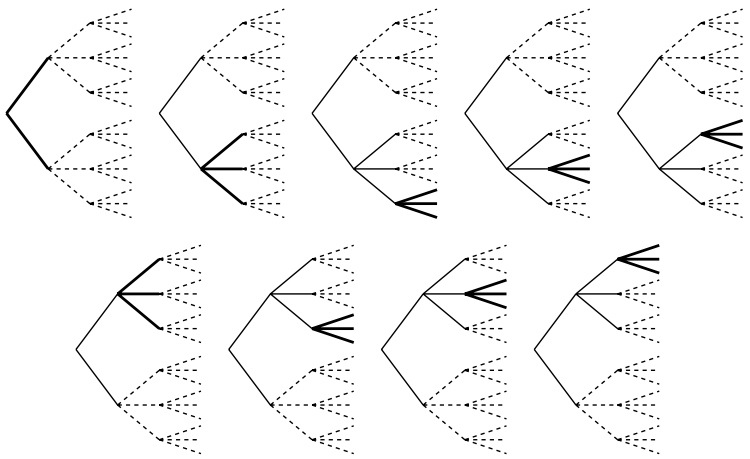

Figure 3: Example of step-by-step generation of a multi-period tree. The bold arcs represent the currently generated subtree, the other subtrees kept in the memory have solid arcs. Dashed arcs represent parts of the multi-period tree that are not being processed at the moment.

The easiest case is when all the subtrees have the same number of branches and the same distributions, i.e. the case with no inter-period dependency. In this case a single subtree can be used throughout the whole multi-period tree. This subtree can be generated prior to (or at the start of) the generating/pricing algorithm, so there would be no scenario generation during the rest of the algorithm.

If we want the number of branches of the subtrees to differ for every period (typically starting with more branches and decrease the number for later periods), we still only need one subtree for every desired size, as long as the distributions are kept constant, i.e. as long as we do not have inter-period dependencies.

When we introduce inter-period dependencies (such as mean-reversion or autocorrelation), the situation becomes more complicated. However, we may still avoid "on the fly" generation of subtrees in the case when the dependency rules change only the means and variances of the distributions, for example mean-reversion or volatility clumping. In this case we can use a linear transformation to update the distribution: If we have a pre-generated subtree $\tilde{\boldsymbol{X}}$ with zero means and variances of one, then $\boldsymbol{\sigma}\tilde{\boldsymbol{X}} + \boldsymbol{\mu}$ gives a subtree with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$, while the higher moments and the correlations remain unchanged. The linear transformation is much cheaper than generating a new

```
function value(n) {
  if n is not a leaf {
    generate subtree of n = {x(m), p(m) | m ∈ C(n)};
    for all children m ∈ C(n) do
      v(m) = value(m);
    value = max{ f_e(x(n)), Σ_{m∈C} v(m)p(m) };
  }
  else
    value = f_e(x(n));
}
```

Figure 4: The generating/pricing algorithm, using a recursive post-order traversal. For any node n of the tree, $\mathbf{C}(n)$ is a set of all children of n, $\boldsymbol{x}(n)$ are prices of the assets at n, $p(n)$ is a probability of n, $\vec{\boldsymbol{x}}(n)$ is a vector of prices of the assets on the path from the root to the node n, and $f_e(\vec{\boldsymbol{x}}(n))$ is the profit of immediate exercise of the option at node n. The price of the option is then found as $price = value(root)$.

tree, and the total running time will be of the same magnitude as in the case without inter-period dependencies.

The most difficult case is when we introduce inter-period rules that update also the higher moments and/or the correlations of the subtree distribution. If we want to avoid generating all the subtrees even in this case, we have to use an approximate approach: We pre-generate a set of subtrees with different combinations of the properties that are updated, and during the generating/pricing procedure choose the one closest to the desired properties. For example, discretizing the interval of possible values of the correlation, the skewness, and the kurtosis to 10 points each, gives $10 \times 10 \times 10 = 1000$ subtrees. These have to be generated in advance. Since a typical multi-period tree will have millions of subtrees, the speed-up of the whole generating/pricing process is still substantial.

It is important to remember that the introduction of inter-period dependency means that we lose control over the unconditional (overall) distributions. We may, however, control the means and variances of these distributions, using the procedure described in Section 3.2.[5]

[5]It may not be obvious how to compute the current values of the unconditional moments, since we do not store the whole tree. This is, however, no problem: Instead of the central moments $\mathbb{E}\big[\big(\tilde{X} - \mathbb{E}\big[\tilde{X}\big]\big)^k\big]$, we compute the non-central moments $\mathbb{E}\big[\tilde{X}^k\big]$—these are easily accumulated during the execution of the algorithm. After the run, the non-central moments are transformed to the central moments using the formulas from Appendix A.1.

5.2 Speed test

The tests were run on an `Intel Pentium III` machine with 996.76 MHz processor and 256 MB RAM, running `Windows 2000`™. The algorithm was implemented using the `C` programming language and compiled with `GNU C++` compiler.[6]

In the tests, we generated 12-period trees and priced the outperformance option from Section 4.1 on them. The option was priced as an American call. In these tests, we assume that all the subtrees have the same distributions and sizes, so we need to generate only one subtree and use it throughout the whole tree. The time for generation is not included in the reported time, but it typically is only a fraction of a second. The results are presented in Table 3.

Table 3: Time to generate a 12-period tree, and price two options on it

# branches per subtree	# subtrees	# terminal nodes	time
5	61.035.156	244.110.625	32 sec
6	435.356.467	2.176.782.336	5 min

In these tests, we have used the arithmetic returns. If we use the geometric returns ("log-returns") instead, the code runs about three times slower since evaluation of the exp() function is significantly slower than the addition and multiplication needed for the arithmetic returns.

The option in the example was dependent only on the current price and the expected future option values. The algorithm from Figure 4, however, allows pricing of "strongly path-dependent" options, i.e. options that depend also on past prices.

As an example, consider a "hedging" option that gives the buyer the right to receive, at any time t, the biggest difference between the two indices, S&P 500 and DAX 30, measured so far. Again, we buy indices for \$100. In other words, if the option is exercised at time t, the buyer receives

$$\$100 \max_{\tau \le t} \big(\mathrm{S\&P}_\tau - \mathrm{DAX}_\tau \big) \ .$$

Since we need to check the whole paths, the pricing of such an option takes more time: With a 12-period tree with 5 branches per node, it took 86 seconds.

[6] The code was also compiled with `Visual C++`, in which case it runs about 5% slower.

5.3 Stability issues

In Section 5.1 we explained that, in the case of independent periods with equal distributions, it is possible to generate only one subtree and reuse it throughout the whole multi-period tree.

While this resembles the "classical" binomial trees, there is an important difference: In a binomial tree the subtree is unique, while in our trees there are infinitely many trees with the given properties (assuming that the subtrees have enough branches). Since we randomly pick only one, we have to test how much it influences the resulting option prices.

We tested prices of the two options defined in Section 4.1, both priced as American-style[7] options. To be able to test also the influence of the size of the subtrees, the test was done for 6-period trees. The size of the subtrees varied from 5 to 40. For every size, we generated 25 trees, priced the options on them, and computed the means and variances of the prices. Results are reported in Table 4.

Table 4: Stability of option prices with respect to the size of the subtrees

	# branches	5	10	15	18	20	40
underperf.	mean	1.76	1.84	1.89	1.93	1.83	1.77
	std. dev.	0.41	0.23	0.20	0.18	0.16	0.16
outperf.	mean	4.63	4.60	4.56	4.54	4.65	4.70
	std. dev.	0.45	0.22	0.21	0.19	0.16	0.17

We see that the prices are not stable, i.e. they depend on the choice of the constituting subtree. The question is, where does this instability come from? Since all the subtrees are constructed in such a way that the unconditional final-stage distributions have the right first four moments and correlations, the variance must come from some other properties.

One possible source is the variance of the higher moments of the marginals, which we do not control. However, our previous experience shows that the first four moments give so much stability that the variance from higher moments can not explain the observed variance of prices. To test it, we have used the same trees to price common one-asset options, and in this case the prices were stable.

[7]Both options are call options. For the common one-asset call options, the price of an American call is equal to the price of an European call. In our case it turns out that the outperformance call is also never exercised before the final stage. For the underperformance call, the American call has higher price than the European, yet the difference in prices is only about 3%.

Since we price American options, another possible source of the observed variance is an instability of the paths. However, when we price the options as European, the variance of the prices does not decrease – one of the reasons is the fact that the difference between the prices of American and European options is very small.[7]

Hence, the variance of the prices must come from differences in the structure of the multi-variate dependency. As we did in Section 4.1, we again see that the correlation matrix is not enough to describe the dependency. We may also document this graphically: Figure 5 shows density maps of two distributions with the same four moments and correlations – they correspond to trees with 40 branches per subtree, taking the cases with the highest and the lowest price for the underperformance option.

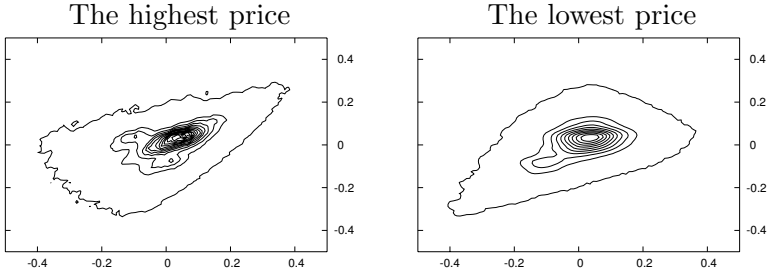

Figure 5: Two distributions with equal first four moments and correlation

This means that if we know only the moments and correlations of the return distribution, the prices of multi-variate options are not uniquely determined. Hence, the interval of prices from multiple runs can be interpreted as an interval of possible option prices, given this limited information. If we, on the other hand, know more about the multi-variate distribution, we may try to add more properties to our approach (a good candidate seems to be co-skewness). If this does not help, we would probably have to leave the moment-matching approach and try some alternatives. This is left for future research.

There is, however, an easy way to reduce the variance of the option prices: Instead of re-using the same one-period subtree throughout the whole multi-period tree, we may pre-generate many subtrees with the same properties, and sample from them during the generating/pricing algorithm (Fig. 4). The problem with this procedure is that we do not know anything about the distribution of the subtrees, so there is no guarantee that the sampling will be unbiased.

Conclusions

In this paper, we have presented a method for multi-period moment-matching scenario generation. When the distributions do not have any inter-period dependencies, the method gives the user an exact control over the first four moments and correlations of the final-stage (unconditional) distribution. With inter-period dependencies (such as autocorrelation), we provide an iterative procedure for ex-post correction of the last two moments.

In addition, we have described a method for pricing multi-variate path-dependent options, based on the (non-recombining) multi-period trees. This is done by a simultaneous tree-generation and pricing, an approach that leads to significant savings in storage place: Even with the exponential growth of the size of the trees, it is possible to price options on 12-period trees in approximately half a minute. We have also shown how to price path-independent European options using one-period trees.

Our numerical tests suggest that the moment-based approach, with correlations as the only description of the multi-variate dependency, has its limits in the option pricing context: For several distribution of prices with the same moments and correlations, the price of some multi-variate options may vary significantly—multi-variate options can be very sensitive to the dependence structure of the distribution of prices. Hence, better methods for description of the multi-variate structure should be used, if we have enough data/information. This is left for a future research.

Acknowledgments

We would like to thank Ser-Huang Poon from the University of Strathclyde, Glasgow, Scotland, for suggesting the topic of option pricing and an initial help with the finance part of the problem. We are also in debt to Stein-Erik Fleten from the Norwegian University of Science and Technology, Trondheim, Norway, as well as several anonymous referees, for pointing out problems in the earlier version of the manuscript.

References

M. C. Cario and B.L. Nelson. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 1997a.

Marne C. Cario and Barry L. Nelson. Numerical methods for fitting and simulating autoregressive-to-anything processes. *INFORMS Journal on Computing*, 10:72–81, 1997b.

Umberto Cherubini and Elisa Luciano. Multivariate option pricing with copulas. Working paper 05, International Centre For Economic Research, 2002. URL `http://www.icer.it`.

Robert T. Clemen and Terence Reilly. Correlations and copulas for decision and risk analysis. *Management Science*, 45(2):208–224, February 1999.

Emanuel Derman and Iraj Kani. Riding with a smile. *Risk*, 7(2):32–39, 1994. Previously appeared as "The Volatility Smile and Its Implied Tree".

Emanuel Derman, Iraj Kani, and Neil Chriss. Implied trinomial trees of the volatility smile. *Journal of Derivatives*, 3(4):7–22, 1996.

Jitka Dupačová, Giorgio Consigli, and Stein W. Wallace. Scenarios for multistage stochastic programs. *Ann. Oper Res.*, 100:25–53 (2001), 2000. ISSN 0254-5330.

F. Fornari and R. Violi. The probability density function of interest rates implied in the price of options. Banca d'Italia, Temi di Discussione del Servizio Studi 339, Banca d'Italia, October 1998.

N. Gülpınar, B. Rustem, and R. Settergren. Multistage stochastic programming in computational finance. Available at `http://www.doc.ic.ac.uk/~reuben/jobs/`, 2001.

K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.

Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2-3):169–185, 2003. ISSN 0926-6003.

J. C. Jackwerth and M. Rubinstein. Recovering probability distributions from option prices. *Journal of Finance*, 51(5):1611–1631, 1996.

Emmanuel Jurczenko, Bertrand Maillet, and Bogdan Negrea. Multi-moment approximate option pricing models: A general comparison (part 1). In *JMA 2002: 19èmes Journées de Micro-économie Appliquée*, Rennes and Saint-Malo, June 2002.

Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. Stochastic Programming E-Print Series, `http://www.speps.info`, May 2003.

R. R. P. Kouwenberg. Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134(2):51–64, 2001.

P. M. Lurie and M. S. Goldberg. An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science*, 44(2):203–218, 1998.

Johan Lyhagen. A method to generate multivariate data with moments arbitrary close to the desired moments. Working paper 481, Stockholm School of Economics, December 2001.

Benjamin Melamed. TES: A class of methods for generating autocorrelated uniform variables. *ORSA Journal on Computing*, 3(4):317–329, 1991.

Roger B. Nelsen. *An Introduction to Copulas*. Springer-Verlag, New York, 1998.

Georg Pflug and Ronald Hochreiter. Scenario generation for stochastic multistage decision processes as facility location problems. Technical Report 2003-01, Department of Statistics and Decision Support Systems, University of Vienna, 2003.

Joshua V. Rosenberg. Pricing multivariate contingent claims using estimated risk-neutral density functions. *Journal of International Money and Finance*, 17(2):229–247, April 1998.

Joshua V. Rosenberg. Semiparametric pricing of multivariate contingent claims. Available at `http://pages.stern.nyu.edu/~jrosenb0/`, August 1999.

Joshua V. Rosenberg. Non-parametric pricing of multivariate contingent claims. *The Journal of Derivatives*, 10(3):9–26, 2003.

Mark Rubinstein. Implied binomial trees. *Journal of Finance*, 49(3):771–818, July 1994.

Mark Rubinstein. Edgeworth binomial trees. *Journal of Derivatives*, 5(3): 20–27, 1998.

A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

A. Sklar. Random variables, distribution functions, and copulas – a personal look backward and forward. In L. Rüschendorff, B. Schweizer, and M. Taylor, editors, *Distributions with Fixed Marginals and Related Topics*, pages 1–14. Institute of Mathematical Statistics, Hayward, CA, 1996.

Wheyming Tina Song, Li-Ching Hsiao, and Yun-Ju Chen. Generating pseudo-random time series with specified marginal distributions. *European Journal of Operational Research*, 94:194–202, 1996.

Luiz Vitiello, Antonio Camara, and Ser-Huang Poon. Pricing options with high moment distributions. Working paper at Strathclyde University, Glasgow, Scotland, November 2002.

A Moments of trees with independent stages

In this section we develop the formulas for the moments of the final-stage distribution as functions of the moments of the one-period subtrees, as well as the inverse formulas. In the formulas we assume that all the one-period subtrees have the same distributions and that the periods are independent. (A typical example is repeating the same subtree over the whole multi-period tree.)

In a multi-period tree with p periods we denote $\mu, \sigma^2, \gamma, \delta$ the first four moments of the final-stage distribution, and $\mu_p, \sigma_p^2, \gamma_p, \delta_p$ the first four moments of the corresponding subtrees.

A.1 General equalities

$$
\begin{aligned}
\mu &= \mathbb{E}\left[\tilde{X}\right] \\
\sigma^2 &= \mathrm{Var}\left[\tilde{X}\right] = \mathbb{E}\left[(\tilde{X} - \mathbb{E}\left[\tilde{X}\right])^2\right] = \mathbb{E}\left[\tilde{X}^2\right] - \mathbb{E}\left[\tilde{X}\right]^2 \\
\sigma^3\gamma &= \sigma^3\,\mathrm{skew}\left[\tilde{X}\right] = \mathbb{E}\left[(\tilde{X} - \mathbb{E}\left[\tilde{X}\right])^3\right] = \mathbb{E}\left[\tilde{X}^3\right] - 3\,\mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{X}^2\right] + 2\,\mathbb{E}\left[\tilde{X}\right]^3 \\
\sigma^4\delta &= \sigma^4\,\mathrm{kurt}\left[\tilde{X}\right] = \mathbb{E}\left[(\tilde{X} - \mathbb{E}\left[\tilde{X}\right])^4\right] \\
&= \mathbb{E}\left[\tilde{X}^4\right] - 4\,\mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{X}^3\right] + 6\,\mathbb{E}\left[\tilde{X}\right]^2\mathbb{E}\left[\tilde{X}^2\right] - 3\,\mathbb{E}\left[\tilde{X}\right]^4
\end{aligned}
$$

and the inverse relations:

$$
\begin{aligned}
\mathbb{E}\left[\tilde{X}^2\right] &= \mathrm{Var}\left[\tilde{X}\right] + \mathbb{E}\left[\tilde{X}\right]^2 = \sigma^2 + \mu^2 \\
\mathbb{E}\left[\tilde{X}^3\right] &= \sigma^3\mathrm{skew}\left[\tilde{X}\right] + 3\,\mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{X}^2\right] - 2\,\mathbb{E}\left[\tilde{X}\right]^3 = \sigma^3\gamma + 3\mu(\sigma^2 + \mu^2) - 2\mu^3 \\
&= \sigma^3\gamma + 3\mu\sigma^2 + \mu^3 \\
\mathbb{E}\left[\tilde{X}^4\right] &= \sigma^4\mathrm{kurt}\left[\tilde{X}\right] + 4\,\mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{X}^3\right] - 6\,\mathbb{E}\left[\tilde{X}\right]^2\mathbb{E}\left[\tilde{X}^2\right] + 3\,\mathbb{E}\left[\tilde{X}\right]^4 \\
&= \sigma^4\delta + 4\mu\left(\sigma^3\gamma + 3\mu\sigma^2 + \mu^3\right) - 6\mu^2\left(\sigma^2 + \mu^2\right) + 3\mu^4 \\
&= \sigma^4\delta + 4\mu\sigma^3\gamma + 6\mu^2\sigma^2 + \mu^4
\end{aligned}
$$

A.2 Formulas for arithmetic returns

In this section we present formulas for processes $\{\tilde{\boldsymbol{V}}_t\}$ with arithmetic returns $\tilde{X}_t$, i.e. processes that evolve as

$$\tilde{V}_t = (1 + \tilde{X}_t)\tilde{V}_{t-1}\ .$$

The final-stage return is then given as

$$\prod_{k=1}^{p}(1+\tilde{X}_k)-1\ .$$

For computing the moments of the distribution of final-stage returns, the following equalities are useful. Note that we use the independence assumption here.

$$\begin{aligned}
\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right] &= \prod_{k=1}^{p}\mathbb{E}\left[1+\tilde{X}_k\right] = \prod_{k=1}^{p}\left(1+\mathbb{E}\left[\tilde{X}_k\right]\right) = \prod_{k=1}^{p}(1+\mu_p)\\
&= (1+\mu_p)^p\\
\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)^2\right] &= \prod_{k=1}^{p}\mathbb{E}\left[(1+\tilde{X}_k)^2\right] = \prod_{k=1}^{p}\mathbb{E}\left[1+2\tilde{X}_k+\tilde{X}_k^2\right]\\
&= \prod_{k=1}^{p}\left(1+2\mu_p+\sigma_p^2+\mu_p^2\right) = \left(1+2\mu_p+\sigma_p^2+\mu_p^2\right)^p\\
&= \left((1+\mu_p)^2+\sigma_p^2\right)^p\\
\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)^3\right] &= \prod_{k=1}^{p}\mathbb{E}\left[(1+\tilde{X}_k)^3\right] = \prod_{k=1}^{p}\mathbb{E}\left[1+3\tilde{X}_k+3\tilde{X}_k^2+\tilde{X}_k^3\right]\\
&= \prod_{k=1}^{p}\left(1+3\mu_p+3(\sigma_p^2+\mu_p^2)+(\sigma_p^3\gamma_p+3\mu_p\sigma_p^2+\mu_p^3)\right)\\
&= \left(1+3\mu_p+3\mu_p^2+\mu_p^3+3\sigma_p^2+3\mu_p\sigma_p^2+\sigma_p^3\gamma_p\right)^p\\
&= \left((1+\mu_p)^3+3(1+\mu_p)\sigma_p^2+\sigma_p^3\gamma_p\right)^p\\
\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)^4\right] &= \prod_{k=1}^{p}\mathbb{E}\left[(1+\tilde{X}_k)^4\right] = \prod_{k=1}^{p}\mathbb{E}\left[1+4\tilde{X}_k+6\tilde{X}_k^2+4\tilde{X}_k^3+\tilde{X}_k^4\right]\\
&= \prod_{k=1}^{p}\left(1+4\mu_p+6(\sigma_p^2+\mu_p^2)+4(\sigma_p^3\gamma_p+3\mu_p\sigma_p^2+\mu_p^3)\right.\\
&\quad \left.+(\sigma_p^4\delta_p+4\mu_p\sigma_p^3\gamma_p+6\mu_p^2\sigma_p^2+\mu_p^4)\right)\\
&= \left(1+4\mu_p+6\mu_p^2+4\mu_p^3+\mu_p^4+6\sigma_p^2(1+2\mu_p+\mu_p^2)\right.\\
&\quad \left.+4\sigma_p^3\gamma_p(1+\mu_p)+\sigma_p^4\delta_p\right)^p\\
&= \left((1+\mu_p)^4+6(1+\mu_p)^2\sigma_p^2+4(1+\mu_p)\sigma_p^3\gamma_p+\sigma_p^4\delta_p\right)^p
\end{aligned}$$

The formulas for moments

$$\mu = \mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)-1\right] = \prod_{k=1}^{p}\mathbb{E}\left[(1+\tilde{X}_k)\right]-1 = (1+\mu_p)^p-1$$

$$\mu_p = (1+\mu)^{1/p} - 1 \tag{6}$$

$$\begin{aligned}
\sigma^2 &= \operatorname{Var}\left[\prod_{k=1}^{p}(1+\tilde{X}_k) - 1\right] = \operatorname{Var}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right] \\
&= \mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^2\right] - \left(\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\right)^2 \\
&= \left((1+\mu_p)^2 + \sigma_p^2\right)^p - \left((1+\mu_p)^p\right)^2 = \left((1+\mu_p)^2 + \sigma_p^2\right)^p - (1+\mu)^2 \\
\sigma_p^2 &= \left((1+\mu)^2 + \sigma^2\right)^{1/p} - (1+\mu_p)^2
\end{aligned} \tag{7}$$

$$\begin{aligned}
\sigma^3\gamma &= \operatorname{skew}\left[\prod_{k=1}^{p}(1+\tilde{X}_k) - 1\right] = \operatorname{skew}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right] \\
&= \mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^3\right] - 3\,\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^2\right] \\
&\quad + 2\left(\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\right)^3 \\
&= \left((1+\mu_p)^3 + 3(1+\mu_p)\sigma_p^2 + \sigma_p^3\gamma_p\right)^p - 3(1+\mu)\left((1+\mu_p)^2 + \sigma_p^2\right)^p + 2(1+\mu)^3 \\
&= \left((1+\mu_p)^3 + 3(1+\mu_p)\sigma_p^2 + \sigma_p^3\gamma_p\right)^p - (1+\mu)\left[3\left((1+\mu_p)^2 + \sigma_p^2\right)^p - 2(1+\mu)^2\right] \\
&= \left((1+\mu_p)^3 + 3(1+\mu_p)\sigma_p^2 + \sigma_p^2\gamma_p\right)^p - (1+\mu)\left(3\sigma^2 + (1+\mu)^2\right) \\
&= \left((1+\mu_p)^3 + 3(1+\mu_p)\sigma_p^2 + \sigma_p^3\gamma_p\right)^p - (1+\mu)^3 - 3(1+\mu)\sigma^2 \\
\gamma_p &= \frac{1}{\sigma_p^3}\left[\left((1+\mu)^3 + 3(1+\mu)\sigma^2 + \sigma^3\gamma\right)^{1/p} - (1+\mu_p)^3 - 3(1+\mu_p)\sigma_p^2\right]
\end{aligned} \tag{8}$$

$$\begin{aligned}
\sigma^4\delta &= \operatorname{kurt}\left[\prod_{k=1}^{p}(1+\tilde{X}_k) - 1\right] = \operatorname{kurt}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right] \\
&= \mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^4\right] - 4\,\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^3\right] \\
&\quad + 6\left(\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\right)^2\mathbb{E}\left[\left(\prod_{k=1}^{p}(1+\tilde{X}_k)\right)^2\right] - 3\left(\mathbb{E}\left[\prod_{k=1}^{p}(1+\tilde{X}_k)\right]\right)^4
\end{aligned}$$

$$
\begin{aligned}
\sigma^4\delta &= \big((1+\mu_p)^4 + 6(1+\mu_p)^2\sigma_p^2 + 4(1+\mu_p)\sigma_p^3\gamma_p + \sigma_p^4\delta_p\big)^p \\
&\quad - 4(1+\mu)\big((1+\mu_p)^3 + 3(1+\mu_p)\sigma_p^2 + \sigma_p^3\gamma_p\big)^p \\
&\quad + 6(1+\mu)^2\left((1+\mu_p)^2 + \sigma_p^2\right)^p - 3(1+\mu)^4 \\
&= \big((1+\mu_p)^4 + 6(1+\mu_p)^2\sigma_p^2 + 4(1+\mu_p)\sigma_p^3\gamma_p + \sigma_p^4\delta_p\big)^p \\
&\quad + (1+\mu)\Big[-4\big(\sigma^3\gamma + 3(1+\mu)\sigma^2 + (1+\mu)^3\big) \\
&\quad + 6(1+\mu)\left(\sigma^2 + (1+\mu)^2\right) - 3(1+\mu)^3\Big] \\
&= \big((1+\mu_p)^4 + 6(1+\mu_p)^2\sigma_p^2 + 4(1+\mu_p)\sigma_p^3\gamma_p + \sigma_p^4\delta_p\big)^p \\
&\quad - (1+\mu)^4 - 6(1+\mu)^2\sigma^2 - 4(1+\mu)\sigma^3\gamma \\
\delta_p &= \frac{1}{\sigma_p^4}\Big[\big((1+\mu)^4 + 6(1+\mu)^2\sigma^2 + 4(1+\mu)\sigma^3\gamma + \sigma^4\delta\big)^{1/p} \\
&\quad -(1+\mu_p)^4 - 6(1+\mu_p)^2\sigma_p^2 - 4(1+\mu_p)\sigma_p^3\gamma_p\Big]
\end{aligned}
\tag{9}
$$

The formulas for correlations

Finally, for two random variables $\tilde{X}$ and $\tilde{Y}$, we can also measure the correlation ρ. For this formulas, we add upper indices X and Y to the notation for mean and variance.

$$
\begin{aligned}
\sigma^X\sigma^Y\rho &= \mathrm{Cov}\left[\prod_{k=1}^{p}\left(1+\tilde{X}_k\right) - 1, \prod_{k=1}^{p}\left(1+\tilde{Y}_k\right) - 1\right] \\
&= \mathrm{Cov}\left[\prod_{k=1}^{p}\left(1+\tilde{X}_k\right), \prod_{k=1}^{p}\left(1+\tilde{Y}_k\right)\right] \\
&= \mathbb{E}\left[\prod_{k=1}^{p}\left(1+\tilde{X}_k\right)\prod_{k=1}^{p}\left(1+\tilde{Y}_k\right)\right] - \mathbb{E}\left[\prod_{k=1}^{p}\left(1+\tilde{X}_k\right)\right]\mathbb{E}\left[\prod_{k=1}^{p}\left(1+\tilde{Y}_k\right)\right] \\
&= \mathbb{E}\left[\prod_{k=1}^{p}\left(1+\tilde{X}_k\right)\left(1+\tilde{Y}_k\right)\right] - (1+\mu^X)(1+\mu^Y) \\
&= \prod_{k=1}^{p}\mathbb{E}\big[\left(1+\tilde{X}_k\right)\left(1+\tilde{Y}_k\right)\big] - (1+\mu^X)(1+\mu^Y) \\
&= \prod_{k=1}^{p}\left(\mathbb{E}\big[\left(1+\tilde{X}_k\right)\big]\mathbb{E}\big[\left(1+\tilde{Y}_k\right)\big] + \mathtt{Cov}(\tilde{X},\tilde{Y})\right) - (1+\mu^X)(1+\mu^Y) \\
&= \big((1+\mu_p^X)(1+\mu_p^Y) + \sigma_p^X\sigma_p^Y\rho_p\big)^p - (1+\mu^X)(1+\mu^Y) \\
\rho_p &= \frac{1}{\sigma_p^X\sigma_p^Y}\left[\big(\sigma^X\sigma^Y\rho + (1+\mu^X)(1+\mu^Y)\big)^{1/p} - (1+\mu_p^X)(1+\mu_p^Y)\right]
\end{aligned}
\tag{10}
$$

A.3 Formulas for geometric returns

In this section we present formulas for processes $\{\tilde{\boldsymbol{V}}_t\}$ with geometric returns (also called "log-returns") $\tilde{X}_t$, i.e. processes that evolve as

$$V_t = \mathrm{e}^{\tilde{X}_t} V_{t-1} \ .$$

The final-stage return is then defined as

$$\tilde{X} = \ln\left(\frac{V_p}{V_0}\right) = \ln(V_p) - \ln(V_0) = \ln\left(\mathrm{e}^{\sum_{k=1}^{p} \tilde{X}_k} V_0\right) - \ln(V_0) = \sum_{k=1}^{p} \tilde{X}_k \ .$$

Using the formulas from Section A.1, we get the following relations between the subtree properties and the final-stage properties:

$$\begin{aligned} \mu = \mathbb{E}\left[\sum_{k=1}^{p} (\tilde{X}_k)\right] &= \sum_{k=1}^{p} \mathbb{E}\left[\tilde{X}_k)\right] = p\,\mu_p \\ \mu_p &= p^{-1}\mu \end{aligned} \tag{11}$$

$$\begin{aligned} \mathbb{E}\left[\tilde{X}^2\right] = \mathbb{E}\left[\left(\sum_{k=1}^{p} \tilde{X}_k\right)^2\right] &= \mathbb{E}\left[\sum_{k=1}^{p}\sum_{l=1}^{p} \tilde{X}_k \tilde{X}_l\right] = \sum_{k=1}^{p}\sum_{l=1}^{p} \mathbb{E}\left[\tilde{X}_k \tilde{X}_l\right] \\ &= \sum_{k=1}^{p} \left(\mathbb{E}\left[\tilde{X}_k\right] \sum_{l\neq k} \mathbb{E}\left[\tilde{X}_l\right] + \mathbb{E}\left[\tilde{X}_k^2\right]\right) \\ &= \sum_{k=1}^{p} \left(\mathbb{E}\left[\tilde{X}_k\right] \left(\sum_{l=1}^{p} \mathbb{E}\left[\tilde{X}_l\right] - \mathbb{E}\left[\tilde{X}_k\right]\right) + \mathbb{E}\left[\tilde{X}_k^2\right]\right) \\ &= \sum_{k=1}^{p} \left(\mu\mathbb{E}\left[\tilde{X}_k\right] + \mathbb{E}\left[\tilde{X}_k^2\right] - \left(\mathbb{E}\left[\tilde{X}_k\right]\right)^2\right) = \mu^2 + \sum_{k=1}^{p} \sigma_k^2 = \mu^2 + p\,\sigma_p^2 \\ \sigma^2 = \mathrm{Var}\left[\sum_{k=1}^{p} (\tilde{X}_k)\right] = \mathbb{E}\left[\tilde{X}^2\right] - \mathbb{E}\left[\tilde{X}\right]^2 &= \mu^2 + p\,\sigma_p^2 - \mu^2 = p\,\sigma_p^2 \\ \sigma_p^2 &= p^{-1}\sigma_p^2 \end{aligned} \tag{12}$$

$$\mathbb{E}\left[\tilde{X}^3\right] = \mathbb{E}\left[\left(\sum_{k=1}^{p}\tilde{X}_k\right)^3\right] = \mathbb{E}\left[\sum_{k,l,m=1}^{p}\tilde{X}_k\tilde{X}_l\tilde{X}_m\right] = \sum_{k,l,m=1}^{p}\mathbb{E}\left[\tilde{X}_k\tilde{X}_l\tilde{X}_m\right]$$

$$= \sum_{k=1}^{p}\left(\sum_{l,m\neq k}\mathbb{E}\left[\tilde{X}_k\right]\mathbb{E}\left[\tilde{X}_l\tilde{X}_m\right] + 2\sum_{l\neq k}\mathbb{E}\left[\tilde{X}_k^2\tilde{X}_l\right] + \mathbb{E}\left[\tilde{X}_k^3\right]\right)$$

$$= \sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k\right]\sum_{l,m\neq k}\mathbb{E}\left[\tilde{X}_l\tilde{X}_m\right] + 2\sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k^2\right]\sum_{l\neq k}\mathbb{E}\left[\tilde{X}_l\right] + \sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k^3\right]$$

$$\sum_{l,m\neq k}\mathbb{E}\left[\tilde{X}_l\tilde{X}_m\right] = \sum_{l\neq k}\left(\sum_{m=1}^{p}\mathbb{E}\left[\tilde{X}_l\tilde{X}_m\right] - \mathbb{E}\left[\tilde{X}_l\tilde{X}_k\right]\right)$$

$$= \sum_{l=1}^{p}\left(\sum_{m=1}^{p}\mathbb{E}\left[\tilde{X}_l\tilde{X}_m\right] - \mathbb{E}\left[\tilde{X}_l\tilde{X}_k\right]\right) - \sum_{m=1}^{p}\mathbb{E}\left[\tilde{X}_k\tilde{X}_m\right] + \mathbb{E}\left[\tilde{X}_k^2\right]$$

$$= \mu + \sigma^2 - 2\sum_{l=1}^{p}\mathbb{E}\left[\tilde{X}_l\tilde{X}_k\right] + \mathbb{E}\left[\tilde{X}_k^2\right]$$

$$= \mu + \sigma^2 - 2\left(\mu\mathbb{E}\left[\tilde{X}_k\right] + \mathbb{E}\left[\tilde{X}_k^2\right] - \left(\mathbb{E}\left[\tilde{X}_k\right]\right)^2\right) + \mathbb{E}\left[\tilde{X}_k^2\right]$$

$$= \mu + \sigma^2 - 2\mu\mathbb{E}\left[\tilde{X}_k\right] - \mathbb{E}\left[\tilde{X}_k^2\right] + 2\left(\mathbb{E}\left[\tilde{X}_k\right]\right)^2$$

$$\mathbb{E}\left[\tilde{X}^3\right] = \sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k\right]\left(\mu + \sigma^2 - 2\mu\mathbb{E}\left[\tilde{X}_k\right] - \mathbb{E}\left[\tilde{X}_k^2\right] + 2\left(\mathbb{E}\left[\tilde{X}_k\right]\right)^2\right)$$

$$+ 2\sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k^2\right]\left(\mu - \mathbb{E}\left[\tilde{X}_k\right]\right) + \sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k^3\right]$$

$$= \mu\left(\mu + \sigma^2\right) + 2\mu\sigma^2$$

$$+ \sum_{k=1}^{p}\left(-\mathbb{E}\left[\tilde{X}_k\right]\mathbb{E}\left[\tilde{X}_k^2\right] + 2\left(\mathbb{E}\left[\tilde{X}_k\right]\right)^3 - 2\mathbb{E}\left[\tilde{X}_k^2\right]\mathbb{E}\left[\tilde{X}_k\right] + \mathbb{E}\left[\tilde{X}_k^3\right]\right)$$

$$= \mu^3 + 3\mu\sigma^2 + \sum_{k=1}^{p}\left(\mathbb{E}\left[\tilde{X}_k^3\right] - 3\mathbb{E}\left[\tilde{X}_k\right]\mathbb{E}\left[\tilde{X}_k^2\right] + 2\left(\mathbb{E}\left[\tilde{X}_k\right]\right)^3\right)$$

$$\sigma^3\gamma = \mathbb{E}\left[\tilde{X}^3\right] - \mu^3 - 3\mu\sigma^2 = \sum_{k=1}^{p}\sigma_k^3\gamma_k = p\,\sigma_p^3\gamma_p$$

$$\sigma_p = \frac{\sigma^3}{p\,\sigma_p^3}\gamma = \frac{p^{3/2}\sigma_p^3}{p\,\sigma_p^3}\gamma = \sqrt{p}\,\gamma \tag{13}$$

Similar, but more complicated, calculations show that for kurtosis we have

$$\begin{aligned} \delta - 3 &= p^{-1}\left(\delta_p - 3\right) \\ \delta_p &= p\left(\delta - 3\right) + 3 \end{aligned} \tag{14}$$

Finally, for correlations we get:

$$\begin{aligned} \mathbb{E}\left[\tilde{X}\tilde{Y}\right] &= \mathbb{E}\left[\left(\sum_{k=1}^{p}\tilde{X}_k\right)\left(\sum_{l=1}^{p}\tilde{Y}_l\right)\right] = \sum_{k,l=1}^{p}\mathbb{E}\left[\tilde{X}_k\tilde{Y}_l\right] \\ &= \sum_{k=1}^{p}\left(\sum_{l\neq k}\mathbb{E}\left[\tilde{X}_k\right]\mathbb{E}\left[\tilde{Y}_l\right] + \mathbb{E}\left[\tilde{X}_k\tilde{Y}_k\right]\right) \\ &= \sum_{k=1}^{p}\left(\mathbb{E}\left[\tilde{X}_k\right]\left(\sum_{l=1}^{p}\mathbb{E}\left[\tilde{Y}_l\right] - \mathbb{E}\left[\tilde{Y}_k\right]\right) + \mathbb{E}\left[\tilde{X}_k\tilde{Y}_k\right]\right) \\ &= \sum_{k=1}^{p}\mathbb{E}\left[\tilde{X}_k\right]\sum_{l=1}^{p}\mathbb{E}\left[\tilde{Y}_l\right] + \sum_{k=1}^{p}\Big(\mathbb{E}\left[\tilde{X}_k\tilde{Y}_k\right] - \mathbb{E}\left[\tilde{X}_k\right]\mathbb{E}\left[\tilde{Y}_k\right]\Big) \\ &= \mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{Y}\right] + \sum_{k=1}^{p}\mathrm{Cov}\left[\tilde{X}_k,\tilde{Y}_k\right] \\ \sigma^X\sigma^Y\rho = \mathrm{Cov}\left[\tilde{X},\tilde{Y}\right] &= \mathbb{E}\left[\tilde{X}\tilde{Y}\right] - \mathbb{E}\left[\tilde{X}\right]\mathbb{E}\left[\tilde{Y}\right] = \sum_{k=1}^{p}\mathrm{Cov}\left[\tilde{X}_k,\tilde{Y}_k\right] \\ &= \sum_{k=1}^{p}\sigma_k^X\sigma_k^Y\rho_k = p\,\sigma_p^X\sigma_p^Y\rho_p \\ \rho_p &= \frac{\sigma^X\sigma^Y}{p\,\sigma_p^X\sigma_p^Y}\rho = \frac{\sqrt{p}\,\sigma_p^X\sqrt{p}\,\sigma_p^Y}{p\,\sigma_p^X\sigma_p^Y}\rho = \rho \end{aligned} \tag{15}$$

B Example of dependency: first order autocorrelation

In this section we show how to add a first order autocorrelation to the scenario-tree generating procedure, in such a way that we do not change the means and variances of the unconditional distributions. We describe only the simplest case when the distribution of a marginal $\tilde{X}_{t,i}$ of the random vector $\tilde{\boldsymbol{X}}_t$ depends explicitly[8] only on the previous value $\boldsymbol{x}_{t-1,i}$. For more general information about modelling the autocorrelation, see for example Melamed (1991), Song

[8] We do not say that $\tilde{\boldsymbol{X}}_{t,i}$ does not depend on $\{\boldsymbol{x}_{\tau,i},\, \tau < t-1\}$, we just do not control these dependencies.

et al. (1996), or Cario and Nelson (1997b). For the rest of the section, we drop the index i to simplify the formulas.

In the context of scenario trees, the first order correlation can be added during the generation of the tree, using the algorithm described in Section 5. Hence, we need formulas for adding the autocorrelation to a subtree of a given node n at stage t of the tree. Since the tree generation proceeds from the root to the leaves, we already know the outcome $\boldsymbol{x}_t$ of the random vector $\tilde{\boldsymbol{X}}_t$ at node n. We also know the unconditional distribution of the successive subtree, $\tilde{\boldsymbol{X}}_{t+1}$. We want to update the distribution in such a way that

$$\operatorname{corr}\left(\tilde{\boldsymbol{X}}_{t+1}, \tilde{\boldsymbol{X}}_t\right) = \rho_1.$$

To distinguish between the (unconditional) moments of $\tilde{\boldsymbol{X}}_t$ and $\tilde{\boldsymbol{X}}_{t+1}$, we add a time index to the symbols for moments μ and σ^2.

To introduce the correlation, we use the Cholesky decomposition of the correlation matrix

$$R = \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \rho_1 & \sqrt{1-\rho_1^2} \end{bmatrix} \cdot \begin{bmatrix} 1 & \rho_1 \\ 0 & \sqrt{1-\rho_1^2} \end{bmatrix} = LL^T.$$

Since $\tilde{X}_{t+1}$ is independent of $\tilde{X}_t$, we know that, with the random vector $\tilde{\boldsymbol{Y}} = L\left[\tilde{X}_t, \tilde{X}_{t+1}\right]^T$, we have

$$\begin{aligned} \tilde{Y}_1 &= \tilde{X}_t \\ \tilde{Y}_2 &= \rho_1 \tilde{X}_t + \sqrt{1-\rho_1^2}\tilde{X}_{t+1} \\ \operatorname{corr}\left(\tilde{\boldsymbol{Y}}_{t+1}, \tilde{\boldsymbol{Y}}_t\right) &= \rho_1 \end{aligned} \tag{16}$$

We want $\tilde{Y}_2$ to become the updated (autocorrelated) version of the random variable $\tilde{X}_{t+1}$, so their unconditional distributions should be equal. We do not know how to guarantee equal distributions, yet we can at least control the means and variances: The transformation $\tilde{\boldsymbol{Y}} = L\tilde{\boldsymbol{X}}$ does not change the means and variances, if $\tilde{\boldsymbol{X}}$ has zero means and variances of one. We can therefore first normalise $\tilde{\boldsymbol{X}}$, then do the transformation $\tilde{\boldsymbol{Y}} = L\tilde{\boldsymbol{X}}$, and finally transform $\tilde{\boldsymbol{Y}}$ to the moments μ_{t+1}, σ^2_{t+1}. Since all of these are linear transformations, they will not change the correlation ρ_1. The formula then becomes

$$\begin{aligned} \tilde{Y}_2 &= \left(\rho_1 \frac{x_t - \mu_t}{\sigma_t} + \sqrt{1-\rho_1^2}\frac{\tilde{X}_{t+1} - \mu_{t+1}}{\sigma_{t+1}}\right)\sigma_{t+1} + \mu_{t+1} \\ &= \mu_{t+1} + \sigma_{t+1}\rho_1 \frac{x_t - \mu_t}{\sigma_t} + \sqrt{1-\rho_1^2}\left(\tilde{X}_{t+1} - \mu_{t+1}\right). \end{aligned} \tag{17}$$

From this formula, we get the following properties:

$$\text{conditional:} \quad \begin{cases} \mathbb{E}\left[\tilde{Y}_2 | \tilde{X}_t = x_t\right] = \mu_{t+1} + \sigma_{t+1}\rho_1 \dfrac{x_t - \mu_t}{\sigma_t} \\ \text{Var}\left[\tilde{Y}_2 | \tilde{X}_t = x_t\right] = \left(1 - \rho_1^2\right)\sigma_{t+1}^2 \end{cases} \tag{18}$$

$$\text{unconditional:} \quad \begin{cases} \mathbb{E}\left[\tilde{Y}_2\right] = \mu_{t+1} \\ \text{Var}\left[\tilde{Y}_2\right] = \sigma_{t+1}^2 \end{cases} \tag{19}$$

We see that the unconditional distribution of $\tilde{Y}_2$ has the same unconditional mean and variance as the distribution of $\tilde{X}_{t+1}$. Unfortunately, this is not case with the higher moments. For example, even in the simple case of constant skewness, $\text{skew}\left[\tilde{X}_{t+1}\right] = \text{skew}\left[\tilde{X}_t\right]$, we get

$$\text{skew}\left[\tilde{Y}_2\right] = \left(\rho_1^3 + \left(1 - \rho_1^2\right)^{3/2}\right) \text{skew}\left[X_t\right].$$

Note also that there are two equivalent ways of implementing the transformation: We may either transform the vector of outcomes $\left\{x_{t+1}^m \mid m \in \mathbf{C}(n)\right\}$, or we may transform its mean and variance before we generate the outcomes.