Randomized quasi-Monte Carlo methods with applications to quantitative risk management

by

Erik Hintz

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Statistics

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Shane G. Henderson Professor, School of Operations Research & Information Eng. Cornell University
Supervisors:	Christiane Lemieux Professor, Dept. of Stats. and Act. Sci. University of Waterloo
	Marius Hofert Associate Professor, Dept. of Stats. and Act. Sci. University of Waterloo
Internal Members:	Adam Kolkiewicz Associate Professor, Dept. of Stats. and Act. Sci. University of Waterloo
	Martin Lysy Associate Professor, Dept. of Stats. and Act. Sci. University of Waterloo
Internal-External Member:	Yuying Li Professor, School of Computer Science University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We use randomized quasi-Monte Carlo (RQMC) techniques to construct computational tools for working with normal mixture models, which include automatic integration routines for density and distribution function evaluation, as well as fitting algorithms. We also provide open source software with all our methods implemented.

In many practical problems, combining RQMC with importance sampling (IS) gives further variance reduction. However, the optimal IS density is typically not known, nor can it be sampled from. We solve this problem in the setting of single index models by finding a near optimal location-scale transform of the original density that approximates the optimal IS density for the univariate index.

Sampling from complicated multivariate models, such as generalized inverse Gaussian mixtures, often involves sampling from a multivariate normal by inversion and from another univariate distribution, say W, whose quantile function is not known nor easily approximated. We explore how we can still use RQMC in this setting and propose several methods when sampling of W is only possible via a black box random variate generator. We also study different ways to feed acceptance rejection (AR) algorithms for W with quasi-random numbers.

RQMC methods on triangles have recently been developed by K. Basu and A. Owen. We show that one of the proposed sequences has suboptimal projection properties and address this issue by proposing to use their sequence to construct a stratified sampling scheme. Furthermore, we provide an extensible lattice construction for triangles and perform a simulation study.

Acknowledgements

First and foremost, I am deeply grateful to my doctoral supervisors, Dr. Christiane Lemieux and Dr. Marius Hofert, who have supported me with their knowledge, insightful comments, encouragement and endless patience in difficult times. I cannot imagine a better supervision than the one Christiane and Marius have given me.

I would also like to thank Dr. Adam Kolkiewicz, Dr. Martin Lysy, Dr. Yuying Li and Dr. Shane Henderson for agreeing to serve on the thesis committee and reading my thesis.

My career path would not have been possible without the support and encouragement of my high school mathematics teacher, Winfried Schindele, who handed me my A-level exam papers with the words: "This is your first step to success!". I would also like to thank Prof. Dr. Hajo Zwiesler and Prof. Dr. An Chen at the University of Ulm: It was Hajo who convinced me to go to Canada for my graduate studies, and it was An who supervised my Bachelor's and Master's theses and by doing so sparked my interest in research.

I would like to extend my gratitude to all my friends in Germany, Canada and elsewhere. Without your warmth and support I would not have been able to make it through the many difficult times in my PhD studies.

I am indebted to my partner Ashton, who constantly encouraged me to not give up and supported me in any way he could - be it cooking food when I was working all night or putting a smile on my face in my saddest moments.

Last but foremost, I am thanking my parents, Sirje and Fred, for their endless emotional and financial support, their unconditional love and for helping me become the person I am now, even though it meant moving across the ocean.

Dedication

Für meine Eltern,
Sirje und Fred.

For my parents,
Sirje and Fred.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AR acceptance-rejection.

CI confidence interval.

dof degrees of freedom.

ECME Expectation/Conditional Maximization Either.

EM expectation maximization.

GIG Generalized inverse Gaussian.

IG Inverse gamma.

iid independent and identically distributed.

IS Importance sampling.

LDS low-discrepancy sequence.

MC Monte Carlo.

ML Maximum likelihood.

MLE Maximum likelihood estimator.

MSE Mean squared error.

MVN Multivariate normal.

MVT Multivariate t.

NRVG non-uniform random variate generator.

QMC Quasi-Monte Carlo.

RE Relative error.

RQMC Randomized quasi-Monte Carlo.

SIS Single-index importance sampling.

SSIS Stratified single-index importance sampling.

vdC van der Corput.

Chapter 1

Introduction

A nearly omni-present problem in many disciplines is the approximation of an unknown but existing expectation $\mu = \mathbb{E}(f(\boldsymbol{X}))$ where $\boldsymbol{X}$ is a random vector and f is a measurable, integrable function.

Various integration methods exist to estimate μ. Monte Carlo (MC) methods estimate the unknown quantity by the sample average of n independent and identically distributed (iid) copies of $f(\boldsymbol{X})$. The central limit theorem implies the asymptotic convergence rate $\mathcal{O}(1/\sqrt{n})$. This convergence speed is independent of the dimension of d of the problem, which is a great advantage of MC methods; for instance, the product trapezoidal rule yields errors in $\mathcal{O}(n^{-2/d})$ for smooth integrands and is thus inferior to MC when the dimension of the problem is large. Assume that, after a suitable change of variable, we have $\mu = \mathbb{E}(g(\boldsymbol{U}))$ where $\boldsymbol{U} \sim \mathrm{U}(0,1)^d$ is a vector of iid $\mathrm{U}(0,1)$ random variables. The random nature of the sampled copies inevitably leads to some areas of the integration domain being over-sampled and some other areas being under-sampled. Quasi-Monte Carlo (QMC) methods tackle this issue employing a deterministic low-discrepancy point-set instead of a purely random one. Low-discrepancy point-sets aim at filling the unit hypercube in a more homogeneous way, and by doing so they can often reduce the integration error relative to MC significantly. Randomized quasi-Monte Carlo (RQMC) methods additionally randomize the deterministic low-discrepancy point set in a way that it does not lose the good "low-discrepancy" and additionally satisfies that any vector in the point set is uniformly distributed over the unit hypercube. This randomization not only can help improve the performance of QMC methods, but also gives rise to an easy and reliable estimation of the integration error. But do (R)QMC methods always perform better than their purely random counterparts? Despite their success, there is no theoretical result based on which one can uniformly answer this question with "yes". The performance of (R)QMC methods

heavily depends on the integrand. The famous Koksma-Hlawka inequality suggests that if the integrand is "sufficiently smooth" (to be made more precise in Chapter 2), the integration error when approximating μ with QMC from above by $\mathcal{O}(\log(n)^d/n)$ where n is the sample size; this is a faster convergence rate than MC. However, note that if d is large, this bound is only of little practical relevance, as for large d and moderate n, the logarithmic factor cannot be ignored. Also, many functions applied in practice are unbounded and do not have finite variation, and yet (R)QMC methods perform often well to integrate those. The question for which kind of problems (R)QMC methods outperform their MC counterparts has gained a lot of attention in the literature and is still an active area of research. Numerous examples suggest that (R)QMC performs well even for very high dimensional problems; see, for instance, [97] who successfully used RQMC methods for a 360-dimensional integration problem from finance. It is conjectured that (R)QMC methods can work well for (nominally) high dimensional problems when the effective dimension of the integrand is small; that is to say, that the integrand can be well approximated by lower dimensional functions. This was discussed, for instance, in [11], [95], [114] and [119]. A more rigorous discussion of (R)QMC methods will be given in Chapter 2.

The present thesis studies RQMC methods, along with applications thereof in quantitative risk management.

Chapter 3 focuses on multivariate normal variance mixture models and extensions; these are important classes of multivariate distributions widely used in risk management and include the multivariate normal and t distributions as prominent examples.

Evaluating multivariate distribution functions (such as the normal and the t) is a difficult, yet important problem that has gained much attention in the last couple of decades; see, for instance, [34], [50], [35], [36], [37] as well as references therein for a discussion of the estimation of multivariate normal and t probabilities and recent work in [9] for the evaluation of truncated multivariate t distributions. To further illustrate how challenging this problem is, we note that the R package `mvtnorm` (one of the most widely used packages according to reverse depends, see [28]) and other R packages do not even provide functionality for evaluating the distribution function of the well-known multivariate t distribution for non-integer degrees of freedom (dof) $\nu > 0$.

In Section 3.2, we provide an RQMC algorithm to efficiently evaluate the joint distribution function of a normal variance mixture which is obtained by generalizing methods by A. Genz and F. Bretz for evaluating the distribution function of the multivariate normal and t distribution. In particular, we generalize a variable reordering algorithm originally suggested by [39] and adapted by [36] which significantly reduces the variance of the integrand yielding fast convergence of our estimators. In addition to being able to handle

any normal variance mixture, the novelty of our approach is that within the integration routine required to evaluate the joint distribution function, we use RQMC methods in a way that better leverages the improved convergence properties of these methods compared to MC sampling. Furthermore, we explore the synergy between these methods and the variable reordering algorithm using the concept of Sobol' indices and effective dimension, thus providing new insight into why the reordering algorithm works so well.

An equally important task is the evaluation of the joint density function, which is often not available in closed form and requires numerical evaluation of an intractable one-dimensional integral; this is the case, for instance, when the mixing random variable W follows an inverse-Burr distribution. Since our goal is to provide algorithms that work for any normal variance mixture, an efficient algorithm to approximate the joint (log)-density function of $\boldsymbol{X}$ is needed. We tackle this by proposing in Section 3.3 a new adaptive RQMC algorithm that mostly samples in certain important subdomains of the range of the mixing variable to efficiently estimate the log-density of a multivariate normal variance mixture. Even log-densities around -100 can be estimated efficiently.

This flexible algorithm turns out to be a key ingredient for the task of parameter estimation. Here our contribution is to propose an algorithm that is general enough to handle any normal variance mixture with bounded density function, as explained in Section 3.4. More precisely, we employ an Expectation/Conditional Maximization Either (ECME) algorithm, which is a likelihood-based fitting procedure developed in [79]. This procedure requires repeated evaluations of the log-density function of $\boldsymbol{X}$, which is one of the reasons why efficient algorithms for the latter are important when this density does not have a closed form.

To the best of our knowledge, none of the three aforementioned tasks have been discussed in the literature in such generality where the only requirement is to have a computationally tractable quantile function for the mixing variable W. By specifying the latter, our methods can be used to perform standard modeling tasks for multivariate normal variance mixtures well beyond the case of a multivariate t distribution. To demonstrate this, a real financial data set is analyzed using an inverse-gamma, a Pareto and an inverse-Burr mixture at the end of Section 3.6. These contributions are published in [53]. All our algorithms are implemented in the R package `nvmix`; see [104], [59], [55].

A possible limitation of normal variance mixtures is their radial symmetry, which in a risk management context means that joint large losses are as likely as joint large gains. In the case of a multivariate t distribution, this can be overcome by allowing different margins to have different dof. On a copula level, this leads to the notion of grouped t copulas of [21] and generalized t-copulas of [82]. Motivated by these constructions, we

define in Section 3.7, grouped normal variance mixtures, a class of distributions which includes ungrouped normal variance mixtures. The grouped case is even more complicated than the ungrouped one, which can be seen by the fact that not even the density of a grouped t distribution is available in closed form. We extend our algorithms from the ungrouped case to the grouped case. In particular, we provide algorithms to estimate the joint density and distribution function, thereby filling a gap in the existing literature. These results have been published in [52].

An important task in risk management is the correct modelling of dependencies between different risk factors. Copulas have become increasingly popular to model dependencies; see, e.g., [85, Chapter 7], [30] and [90]. The t copula, which is the implicit copula of a multivariate t distribution, is a very popular copula to model dependencies, as, in contrast to the Gaussian copula, it is able to account for tail dependence. In Section 3.8, we address the important problem of fitting grouped t copulas to data. While the problem of parameter estimation for the grouped t copula was already studied in [21] and [82], the computation of the copula density which is required for the joint estimation of all dof parameters has not been investigated in full generality for arbitrary dimensions yet, which is one gap we fill in this section. Furthermore, unlike [21], we suggest joint rather than group-wise estimation of the degrees-of-freedom and show that this gives larger likelihood at the parameter estimates. We also provide an expectation maximization (EM)-like algorithm that works for many implicit copulas and apply it to t copulas. These results have been published in [54].

In Chapter 4, we study problems where all but one component in a model can be sampled via inversion and the remaining one, say W, only by calling a black box non-uniform random variate generator (NRVG) or via acceptance-rejection (AR). Many practical problems of interest, such as the output of normal mixture models with complicated mixing variable, fall under this umbrella.

The methods we propose for the black box setting in Section 4.1 are based on approximations of the quantile function of W. We consider the empirical quantile function, which when used instead of the true quantile function amounts to sorting the W samples obtained from the NRVG in a way that the ordering matches the ordering of the low-discrepancy point-set that would have been inverted. Another method in this black box setting we are considering is to estimate the quantile function in the tail based on a generalized Pareto distribution, which is justified by Pickands-Balkema-de-Haan Theorem ([29, Theorem 3.4.13]).

In the second setting, we explore different ways to feed a given AR algorithm with quasi-random numbers to sample W. AR algorithms are typically not popular in RQMC as we do

not know a-priori how many coordinates are needed to accept a point. [87] consider using smoothed rejection and weighted uniform sampling and show in their numerical results that these outperform AR sampling in terms of convergence speed. [116] show that the discrepancy with respect to the target distribution, i.e., the maximum difference between the target distribution function and the empirical distribution function computed from n observations obtained with their method has order $n^{-\alpha}$ for $1/2 \leq \alpha < 1$. They improve the error convergence rate by replacing the purely binary AR decision with weights, called extended smoothed rejection. This circumvents integration of an indicator function.

Rather than altering the AR scheme by including weights or smoothing the integrand, we focus on studying different ways to feed the AR algorithm with quasi-random numbers. [122] study using a purely deterministic low-discrepancy sequence with constant dimension, that is, points in the sequence are skipped until acceptance, and derive discrepancy properties. [91] consider RQMC and, similarly to [122], skip points in the randomized low-discrepancy sequence until acceptance. They give a convergence result, error bounds and a numerical study for AR with RQMC.

The two previous references have in common that they hold the dimension of the low-discrepancy sequence (LDS) constant and effectively use a subset of size n of the first N points in the sequence. In contrast, we also investigate the use of a point set of constant (target) size where the number of coordinates of each point is increased until acceptance. We prove unbiasedness results and show how to combine the methods from the two settings in such a way that the non-monotonicity inherent in AR is removed.

Finally, we perform a numerical study with all methods presented. In particular, we estimate the value of a Basket call option whose dependence is modelled with a normal variance mixture copula, in particular an inverse-gamma and generalized inverse gaussian mixture. This section is based on the submitted manuscript [56].

In Chapter 5, we consider the problem of rare event simulation in single index models, where the univariate output $f(\boldsymbol{X})$ depends on the random vector $\boldsymbol{X}$ mainly through some univariate random variable $T = T(\boldsymbol{X})$, called index. As we consider rare-event simulation, so the event $\{|f(\boldsymbol{X})| > 0\}$ has small probability, MC methods typically need to be combined with variance reduction techniques as otherwise a very large number of samples is required to obtain non-zero observations; the same holds for RQMC methods. Importance sampling (IS) is a variance reduction technique frequently applied to rare-event analysis in order to improve the reliability of MC estimators; see, e.g., [67] and [2]. The main idea of IS is to draw samples from a proposal distribution that puts more mass on the rare-event region of the sample space than the original distribution. As the efficiency of IS depends heavily on the choice of the proposal distribution, finding a good proposal

distribution is a crucial step in applying IS. Unfortunately, there is no single best strategy known for finding a good proposal distribution that works in every situation since the nature of the rare event and what constitutes a good proposal distribution depends on the problem at hand. As such, much of the existing work on IS in computational finance finds effective proposal distributions by exploiting the structure of specific problems: [41] develop IS methods to price path-dependent options under multivariate normal models; [42, 43] estimate the Value-at-Risk of a portfolio consisting of stocks and options under a normal and t-distribution; [109] estimate tail probabilities of equity portfolios under generalized hyperbolic marginals with a t-copula assumption; [44] estimate tail probabilities of credit portfolios under the Gaussian copula, [4, 13] consider t-copula models. As all these IS techniques are exploiting specific properties of the problem at hand, they can achieve substantial variance reduction but are typically specific techniques not applicable to other problems without major modifications.

In the joint work [57], our collaborator Y. Taniguchi derived expressions for optimal IS densities for sampling T in our single-index setting. Note that our framework is applicable to a wide range of problems, as we do not make any assumptions on the integrand or any distributional assumptions, other than assuming a single index model.

Furthermore, he showed that the estimators have zero variance when $f(\boldsymbol{X})$ is completely determined by T (to be made precise later) and explained how our conditional sampling step reduces the effective dimension of the problem and therefore makes RQMC particularly attractive in this setting. Further variance reduction is achieved by stratification of T. As the optimal IS densities involve conditional expectations that are not known in practice, the collaborator suggested using pilot runs to obtain a point-wise approximation of the optimal IS density, which can then be integrated and inverted numerically using the NINIGL algorithm developed in [64].

The author of this thesis takes up this issue and we give more details on how such pilot-runs can be implemented in practice. As numerically integrating and inverting the optimal density is typically too time-consuming and sometimes prone to numerical errors, we suggest finding an approximately optimal IS density in the same parametric family as the original density. We detail this calibration stage, i.e., the process of estimating T, the optimal density and a way to sample from it, in Section 5.2.

The numerical examples in Section 5.3 were implemented by us, we demonstrate that our methods are applicable to a wide range of problems and achieve substantial variance reduction. In this context, we consider the problem of tail probability estimation in Gaussian and t-copula credit portfolio problems and show that our methods outperform those of [44] and [13].

Finally, we note that the work closest to ours is the nonparametric partial importance sampling (NPIS) method of [89]. Our method is similar to NPIS with $u = 1$ (where u is defined as in [89]), in that both apply nonparametric IS only to the most important random variable variable. However, SIS allows for a more general form for identifying the important variable than NPIS, and the combination of IS and stratified sampling is not considered in [89].

In Chapter 6, we move away from RQMC methods on the unit cube $[0, 1)^d$ and instead consider the problem of integrating functions over triangular domains.

The most recent approaches for constructing points with small discrepancy on a triangle include the triangular van der Corput (vdC) and the triangular Kronecker lattice developed in [6]. Both methods can be used to sample deterministic points with vanishing parallelogram discrepancy, which is a discrepancy measure similar to the one used in [10]. [6] also show that the parallelogram discrepancy of their lattice approach achieves the optimal convergence order $\mathcal{O}(\log(n)/n)$. [45] generalize the triangular van der Corput sequence. Rather than focusing on discrepancy and Koksma–Hlawka like inequalities, they study the absolute worst case integration error over a set of functions, and show that their construction is almost optimal. Unlike the aforementioned references that attempt to directly construct points on the triangle, [100] studies various transformations to map a low-discrepancy sequence from the unit cube to any triangle.

A possible limitation of the lattice approach in [6] is that it is not extensible in the sample size. Our contribution here is the development of an extensible lattice construction, which, for certain sample sizes n, coincides with the non-extensible approach. Furthermore, we prove that their triangular vdC sequence with n points projects onto $2\sqrt{n}$ points on the x- and y-axis. Finally, we perform a numerical study comparing all methods, including some of the methods described in [100]. We note that a numerical study has not been done in either [6] or [45].

This project was in collaboration with G. Dong and the results will appear in [27]. All results presented in this chapter were contributed by the author of this thesis; the collaborator's work on using the triangular vdC sequence to construct a stratified sampling scheme is omitted.

Chapter 7 concludes this thesis.

Chapter 2

Background

2.1 Monte Carlo and randomized quasi-Monte Carlo methods

Quantities of interest in this thesis are (after a suitable transformation) expressed as intractable integrals over the unit hypercube $(0,1)^d$ for some $d \in \mathbb{N}$, i.e.,

$$\mu = \int_{(0,1)^d} g(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u}, \tag{2.1}$$

where $g : (0,1)^d \to \mathbb{R}$ is integrable. Monte Carlo (MC) methods approximate μ in (2.1) by the arithmetic average $\hat{\mu}_n^{\mathrm{MC}} = (1/n) \sum_{i=1}^n g(\boldsymbol{U}_i)$ where $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^d$. An asymptotic $(1-\alpha)$-confidence interval (CI) can be approximated for sufficiently large n by

$$\left[\hat{\mu}_n^{\mathrm{MC}} - z_{1-\alpha/2}\hat{\sigma}_g/\sqrt{n},\ \hat{\mu}_n^{\mathrm{MC}} + z_{1-\alpha/2}\hat{\sigma}_g/\sqrt{n}\right],$$

where $z_\alpha = \Phi^{-1}(\alpha)$ and $\hat{\sigma}_g^2 = \widehat{\mathrm{Var}}(g(\boldsymbol{U})) = \sum_{i=1}^n (g(\boldsymbol{U}_i) - \hat{\mu}_n^{\mathrm{MC}})^2/(n-1)$. One can choose n so that the length of this CI does not exceed a pre-determined absolute error tolerance.

Replacing the (pseudo-random) evaluation points $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n$ by a deterministic low-discrepancy point set which aims at filling the unit hypercube in a more homogeneous way, say $P_n = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_n\} \subset [0,1)^d$, leads to a *quasi-Monte Carlo* (QMC) estimator for μ, say $\hat{\mu}^{\mathrm{QMC}}$. QMC methods often provide better estimators than classical MC methods, the deterministic nature of the points in P_n however does not allow for simple error estimation via CIs as was done for the MC estimator $\hat{\mu}_n^{\mathrm{MC}}$. To overcome this, one can

randomize the point set P_n in a way such that the points in the resulting point set, say $\tilde{P}_n$, are uniformly distributed over $(0,1)^d$ without losing the low-discrepancy of the point set overall. This leads to randomized QMC (RQMC) methods. In our algorithms, we use a digitally-shifted Sobol' sequence ([111]) as implemented in the function `sobol(, randomize = "digital.shift")` of the R package `qrng`; see [62]. We remark that generating $\tilde{P}_n$ is slightly faster than generating $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^d$ using R's default pseudo random number generator, the Mersenne Twister ([84]).

Given B independently randomized copies of P_n, say $\tilde{P}_{n,b} = \{\boldsymbol{u}_{1,b}, \dots, \boldsymbol{u}_{n,b}\}$ for $b = 1, \dots, B$, one can construct B independent RQMC estimators of the form

$$\hat{\mu}_{b,n}^{\mathrm{RQMC}} = \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{u}_{i,b}), \quad b = 1, \dots, B, \tag{2.2}$$

and combine them to the RQMC estimator

$$\hat{\mu}_n^{\mathrm{RQMC}} = \frac{1}{B} \sum_{b=1}^{B} \hat{\mu}_{b,n}^{\mathrm{RQMC}} \tag{2.3}$$

of μ. An approximate $(1-\alpha)$-CI for μ can be estimated as

$$\left[\hat{\mu}_n^{\mathrm{RQMC}} - z_{1-\alpha/2}\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}/\sqrt{B},\ \hat{\mu}_n^{\mathrm{RQMC}} + z_{1-\alpha/2}\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}/\sqrt{B}\right], \tag{2.4}$$

where

$$\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}} = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\hat{\mu}_{b,n}^{\mathrm{RQMC}} - \hat{\mu}_n^{\mathrm{RQMC}})^2}. \tag{2.5}$$

One can compute $\hat{\mu}_n^{\mathrm{RQMC}}$ from (2.3) for some initial sample size n (e.g., $n = 2^7$) and iteratively increase the sample size of each $\hat{\mu}_{b,n}^{\mathrm{RQMC}}$ in (2.2) until the length of the CI in (2.4) satisfies a pre-specified error tolerance. In our implementations, we use $B = 15$, an absolute default error tolerance $\varepsilon = 0.001$ (which can be changed by the user) and $z_{1-\alpha/2} = 3.5$ (so $\alpha \approx 0.00047$); note that using t quantiles gives almost the same CIs. By using $\hat{\mu}_n^{\mathrm{RQMC}}$ as approximation for the true value of μ, one can also consider relative errors instead of absolute errors.

Function evaluations from iterations that did not meet the tolerance can be recycled as follows. Let $P_{n_1,n_2} = \{\boldsymbol{v}_{n_1+1}, \dots, \boldsymbol{v}_{n_1+n_2}\}$ be the point set consisting of the n_2 low-discrepancy points after skipping the first n_1-many points. Furthermore, let $\tilde{P}_{n_1,n_2,b} =$

$\{\boldsymbol{u}_{n_1+1,b}, \ldots, \boldsymbol{u}_{n_1+n_2,b}\}$ be the bth randomly shifted version of P_{n_1,n_2} and let

$$\hat{\mu}^{\text{RQMC}}_{b,n_1,n_2} = \frac{1}{n_2} \sum_{\boldsymbol{u} \in \tilde{P}_{n_1,n_2,b}} g(\boldsymbol{u}), \quad b = 1, \ldots, B.$$

If $\hat{\mu}^{\text{RQMC}}_{n_1}$ does not meet the error tolerance, an estimator based on $n_1 + n_2$ points can be calculated using only n_2 additional function evaluations based on

$$\hat{\mu}^{\text{RQMC}}_{b,1,n_2} = \frac{n_1 \times \hat{\mu}^{\text{RQMC}}_{b,1,n_1} + n_2 \times \hat{\mu}^{\text{RQMC}}_{b,n_1,n_2}}{n_1 + n_2}, \quad b = 1, \ldots, B.$$

In iteration i this update is being done with $n_1 = in_0$ and $n_2 = n_1 + n_0$ in Step 3a of our Algorithm 2.1.1 to estimate μ from (2.1). That is, we start with initial sample size n_0 and add another n_0 points in each iteration. We highlight that this update can be easily implemented for a Sobol' sequence, as one can generate P_{n_1,n_2} efficiently without having to generate P_{1,n_2}; in R, this can be achieved by calling `sobol(, skip = n1)`. We do not lose any low-discrepancy properties of the randomized Sobol' sequences as the resulting estimator is mathematically equivalent to $\hat{\mu}^{\text{RQMC}}_{n^*} = (1/B)\sum_{b=1}^{B} \hat{\mu}^{\text{RQMC}}_{b,1,n^*}$ where n^* is the total number of function evaluations in each randomization. We therefore leverage convergence properties in n of Sobol' sequence based estimators. It is important to point out that the reason why we can add points in this way without discarding previous function evaluations is because the Sobol' sequence is extensible in n. That is, it is constructed as a sequence in such a way that the first n points can be used as a low-discrepancy point set P_n for any n, with additional uniformity properties when n is a power of 2 (or a multiple of a power of 2).

The update in our algorithm is conceptually different from updates in RQMC methods suggested in the literature: For instance, the RQMC algorithm proposed in [36] to estimate the distribution function of a multivariate t distribution, therein referred to as QRSVN algorithm, is based on a randomized Korobov rule (which belong to the wider class of lattice rules; see [69] and [19]). The QRSVN algorithm also iteratively evaluates the integrand at low-discrepancy points until the estimated error is small enough; however, it does not move along the same sequence of low-discrepancy points from one iteration to another. In iteration i, their method computes an estimator based on a lattice of size p_i (a prime), and estimators from different iterations are combined as a variance-weighted average. Ultimately, the QRSVN algorithm outputs a weighted average of $B \cdot i^*$ different RQMC estimators based on different sample sizes (where i^* denotes the number of iterations needed until termination), whereas our algorithm outputs the average of B digitally-shifted RQMC estimators based on the first n^* points of a Sobol' sequence. Hence, our methods

leverage properties of the Sobol' sequence with growing n rather than combining more and more RQMC estimators of different sample sizes. Our proposed approach is thus superior because the variance of RQMC estimators can be shown to be in $O(n^{-\delta})$ with $\delta > 1$ (and the smoother f is, the larger δ is). Hence for a given fixed computing budget of Bn function evaluations that must be split between B and the size n for the point set P_n, it is best to try to take B just large enough so that we get a reasonable variance estimate, and then set n as large as possible in order to further reduce the variance thanks to its $O(n^{-\delta})$ behavior: this is precisely what our approach does. Numerical results in Section 3.6.2 illustrate how this leads to improved efficiency compared to the QRSVN algorithm.

Finally, our way of updating merely requires Bn_0 additional function evaluations in each iteration, rather than Bp_{i+1}. This typically leads to a smaller run-time, since only as many function evaluations as needed are computed.

Algorithm 2.1.1 (RQMC algorithm for estimating $\mu = \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}$.)
Given ε, B, n_0, $i_{\max}$, estimate $\mu = \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}$ via:

1. Set $n = n_0$, $i = 1$, and compute $\hat{\mu}_{b,n}^{\mathrm{RQMC}} = \hat{\mu}_{b,0,n_0}^{\mathrm{RQMC}}$ for $b = 1, \ldots, B$ and $\hat{\mu}_n^{\mathrm{RQMC}}$ from (2.2) and (2.3).
2. Set $\hat{\varepsilon} = 3.5\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}$ with $\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}$ from (2.5).
3. While $\hat{\varepsilon} > \varepsilon$ and $i \leq i_{\max}$ do:
   (a) Set $n = n + n_0$, compute $\hat{\mu}_{b,in_0,(i+1)n_0}^{\mathrm{RQMC}}$, $b = 1, \ldots, B$ and set $\hat{\mu}_{b,n}^{\mathrm{RQMC}} = (i\hat{\mu}_{b,n}^{\mathrm{RQMC}} + \hat{\mu}_{b,in_0,(i+1)n_0}^{\mathrm{RQMC}})/(i+1)$.
   (b) Update $\hat{\mu}_n^{\mathrm{RQMC}} = (1/B)\sum_{b=1}^{B} \hat{\mu}_{b,n}^{\mathrm{RQMC}}$ and update $\hat{\varepsilon} = 3.5\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}$ with $\hat{\sigma}_{\hat{\mu}_n^{\mathrm{RQMC}}}$ from (2.5).
   (c) Set $i = i + 1$.
4. Return $\hat{\mu}_n^{\mathrm{RQMC}}$.

Sometimes it is necessary to estimate $\log \mu$ rather than μ; in particular, when μ is small. For instance, if $\mu = f(\boldsymbol{x})$ where $f(\boldsymbol{x})$ is the density function of $\boldsymbol{X} \sim \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$ evaluated at $\boldsymbol{x} \in \mathbb{R}^d$, of interest may be $\log(\mu) = \log f(\boldsymbol{x})$ as this quantity is needed to compute the log-likelihood of a random sample (which then may be optimized over some parameter space). When μ is small, using $\log \mu \approx \log(\hat{\mu}_n^{\mathrm{RQMC}})$ directly should be avoided. One should

instead compute a numerically more robust estimator for $\log \mu$, a *proper logarithm*. To this end, define the function LSE (for Logarithmic Sum of Exponentials) as

$$\mathrm{LSE}(c_1, \ldots, c_n) = \log\left(\sum_{i=1}^{n} \exp(c_i)\right) = c_{\max} + \log\left(\sum_{i=1}^{n} \exp(c_i - c_{\max})\right),$$

where $c_1, \ldots, c_n \in \mathbb{R}$ and $c_{\max} = \max\{c_1, \ldots, c_n\}$. The right-hand side of this equation is numerically more stable than the left-hand side as the sum inside the logarithm is bounded between 1 and n, thereby avoiding overflow/underflow issues.

Let $c_{i,b} = \log g(u_{i,b})$ for $i = 1, \ldots, n$ and $b = 1, \ldots, B$. An estimator numerically superior (but mathematically equivalent) to $\log(\hat{\mu}_n^{\mathrm{RQMC}})$ is given by

$$\hat{\mu}_{n,\log}^{\mathrm{RQMC}} = -\log(B) + \mathrm{LSE}(\hat{\mu}_{1,n,\log}^{\mathrm{RQMC}}, \ldots, \hat{\mu}_{B,n,\log}^{\mathrm{RQMC}}), \tag{2.6}$$

where

$$\hat{\mu}_{b,n,\log}^{\mathrm{RQMC}} = -\log(n) + \mathrm{LSE}(c_{1,b}, \ldots, c_{n,b}), \quad b = 1, \ldots, B. \tag{2.7}$$

The standard deviation of $\hat{\mu}_{n,\log}^{\mathrm{RQMC}}$ is estimated in the usual way by computing the sample standard deviation of $\hat{\mu}_{1,n,\log}^{\mathrm{RQMC}}, \ldots, \hat{\mu}_{B,n,\log}^{\mathrm{RQMC}}$ so that, as before, the integration error can be estimated from the length of the CI in (2.4). A summary of the procedure to estimate $\log \mu$ with a proper logarithm via RQMC is given in Algorithm 2.1.2.

Algorithm 2.1.2 (RQMC algorithm to estimate $\log \mu$ where $\mu = \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}$.)
Given ε, B, n_0, $i_{\max}$, estimate $\log \mu = \log(\int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u})$ via:

1. Set $n = n_0$, $i = 1$, and compute $\hat{\mu}_{b,n,\log}^{\mathrm{RQMC}} = \hat{\mu}_{b,0,n_0,\log}^{\mathrm{RQMC}}$ for $b = 1, \ldots, B$ and $\hat{\mu}_{n,\log}^{\mathrm{RQMC}}$ from (2.6) and (2.7).
2. Set $\hat{\varepsilon} = 3.5\hat{\sigma}_{\hat{\mu}_{n,\log}^{\mathrm{RQMC}}}$ with $\hat{\sigma}_{\hat{\mu}_{n,\log}^{\mathrm{RQMC}}}$ as in (2.5).
3. While $\hat{\varepsilon} > \varepsilon$ and $i \le i_{\max}$ do:
   (a) Set $n = n + n_0$, compute $\hat{\mu}_{b,in_0,(i+1)n_0,\log}^{\mathrm{RQMC}}$, $b = 1, \ldots, B$ and update $\hat{\mu}_{b,n,\log}^{\mathrm{RQMC}} = -\log(i+1) + \mathrm{LSE}(i\hat{\mu}_{b,n}^{\mathrm{RQMC}}, \hat{\mu}_{b,in_0,(i+1)n_0}^{\mathrm{RQMC}})$ for $b = 1, \ldots, B$.
   (b) Update $\hat{\mu}_{n,\log}^{\mathrm{RQMC}} = -\log(B) + \mathrm{LSE}(\hat{\mu}_{1,n,\log}^{\mathrm{RQMC}}, \ldots, \hat{\mu}_{B,n,\log}^{\mathrm{RQMC}})$ and update $\hat{\varepsilon} = 3.5\hat{\sigma}_{\hat{\mu}_{n,\log}^{\mathrm{RQMC}}}$.
   (c) Set $i = i + 1$.
4. Return $\hat{\mu}_{n,\log}^{\mathrm{RQMC}}$.

Note that despite the fact that the problem under study here is a one-dimensional integral, we refer to our algorithm as being in the RQMC family. We do so because although the distinctive features of RQMC mostly have to do with how they design low-discrepancy point sets in dimension larger than 1, another distinctive feature they have is to make use of low-discrepancy sequences that are extensible in n, which is precisely what we are exploiting in Algorithm 2.1.2.

2.2 Discrepancies and Koksma Hlawka inequality

The preceding discussion explains how (R)QMC methods can be used in practice. This section shall give a brief overview of the main theoretical foundations of (R)QMC methods.

(R)QMC methods were motivated in the previous section as methods that fill the unit hypercube in a more homogeneous or uniform way. The question how to assess "homogeneity" or "uniformity" arises naturally. The notion of discrepancy, a concept originating in number theory, is crucial for (R)QMC methods. Consider an (a priori infinite) sequence of points $\{\boldsymbol{v}_i\}_{i=1,2,\ldots}$ whose uniformity is to be measured and denote by $P_n = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ the first n points of the sequence. For $\boldsymbol{a}, \boldsymbol{b} \in (0,1)^d$, the interval $[\boldsymbol{a}, \boldsymbol{b}) = \prod_{i=1}^d [a_i, b_i)$ denotes the d-dimensional hyperrectangle spanned by the lower left endpoint $\boldsymbol{a}$ and upper right endpoint $\boldsymbol{b}$. If the point set P_n is "uniform" the number of points in $P_n \cap [\boldsymbol{a}, \boldsymbol{b})$ divided by n should be close to $\lambda([\boldsymbol{a}, \boldsymbol{b}))$ where $\lambda(\cdot)$ denotes the Lebesgue measure. That is, the empirical probability that a point in P_n falls into $[\boldsymbol{a}, \boldsymbol{b})$ should be close to the Lebesgue volume of $[\boldsymbol{a}, \boldsymbol{b})$. This motivates the *star discrepancy* $D^*(P_n)$ of P_n:

$$D^*(P_n) = \sup_{\boldsymbol{b} \in (0,1)^d} \left| \frac{\#(P_n \cap [\boldsymbol{0}, \boldsymbol{b}))}{n} - \lambda([\boldsymbol{0}, \boldsymbol{b})) \right|; \tag{2.8}$$

here, $\#(A)$ denotes the cardinality of a countable set A. A different discrepancy measure arises when replacing $\boldsymbol{0}$ in the intervals $[\boldsymbol{0}, \boldsymbol{b})$ by a general $\boldsymbol{a} \in (0,1)^d$ with $\boldsymbol{a} \leq \boldsymbol{b}$ componentwise; that is, one can consider the *extreme discrepancy*

$$D(P_n) = \sup_{\boldsymbol{a}, \boldsymbol{b} \in (0,1)^d : \boldsymbol{a} \leq \boldsymbol{b}} \left| \frac{\#(P_n \cap [\boldsymbol{a}, \boldsymbol{b}))}{n} - \lambda([\boldsymbol{a}, \boldsymbol{b})) \right|. \tag{2.9}$$

In this case, all rectangular subintervals of $[0,1]^d$ are considered, not only those that are anchored at the origin as was the case in (2.8). There are explicit formulas for $D(P_n)$ and $D^*(P_n)$ for $d \in \{1, 2\}$, but for $d > 2$, calculating $D(P_n)$ and $D^*(P_n)$ is very difficult.

However, if the supremum norm in (2.8) and (2.9) is replaced by the L^2 norm, we get the L^2-star discrepancy and L^2-discrepancy given by

$$T^*(P_n)^2 = \int_{\boldsymbol{b}\in(0,1)^d} \left(\frac{\#(P_n \cap [\boldsymbol{0}, \boldsymbol{b}))}{n} - \lambda([\boldsymbol{0}, \boldsymbol{b})) \right)^2 \mathrm{d}\boldsymbol{b}, \tag{2.10}$$

$$T(P_n)^2 = \int_{\boldsymbol{a},\boldsymbol{b}\in(0,1)^d : \boldsymbol{a}\leq\boldsymbol{b}} \left(\frac{\#(P_n \cap [\boldsymbol{a}, \boldsymbol{b}))}{n} - \lambda([\boldsymbol{a}, \boldsymbol{b})) \right)^2 \mathrm{d}\boldsymbol{a}\, \mathrm{d}\boldsymbol{b}; \tag{2.11}$$

both $T^*(P_n)$ and $T(P_n)$ can be calculated using analytical formulas, see [76, Chapter 5] and references therein.

A sequence of points $\{\boldsymbol{v}_i\}_{i=1,2,\dots}$ is now called *low-discrepancy sequence* or *quasi-random sequence* if $D^*(P_n) = \mathcal{O}\left(\log(n)^d/n\right)$; a finite subset P_n of such a sequence is then called *low-discrepancy point set*. Note it is conjectured that for any d-dimensional deterministic point set P_n in $(0,1)^d$, there is a constant B_d such that $D^*(P_n) \geq B_d \log(n)^{d-1}/n$. If this is true, one cannot do better than low-discrepancy sequences in terms of discrepancy. It is also useful to know that if P_n is a random point set, $D^*(P_n) = \mathcal{O}(\sqrt{\log\log n}/\sqrt{n})$. Thus, low-discrepancy point sets do indeed "fill the hypercube in a more homogeneous way" than purely random point sets if homogeneity is measured via discrepancies.

The discrepancy of a point set is closely related to the integration error obtained when this point set is used to define a QMC rule. Consider again the estimator $\hat{\mu}^{\mathrm{QMC}}$. The famous *Koksma-Hlawka inequality* gives an upper bound on the integration error; see [58]:

$$|\hat{\mu}^{\mathrm{QMC}} - \mu| = \left| \frac{1}{n} \sum_{\boldsymbol{v}\in P_n} g(\boldsymbol{v}) - \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} \right| \leq V^{\mathrm{HK}}(g)\, D^*(P_n); \tag{2.12}$$

here, $V^{\mathrm{HK}}(g)$ denotes the variation of the function g in the sense of Hardy and Krause. A few comments about the Koskma-Hlawka inequality are in order: First, note that $V^{\mathrm{HK}}(g)$ is completely determined by the integrand g whereas $D^*(P_n)$ is completely determined by the point set P_n. Equation (2.12) then suggests that in order to reduce integration error a fruitful approach could be to employ a point set P_n with smallest possible discrepancy. Furthermore, if $V^{\mathrm{HK}}(g) < \infty$ and P_n is a low-discrepancy point set, Equation (2.12) implies that the integration error is at most $\mathcal{O}\left(\log(n)^d/n\right)$ which is a better order than $1/\sqrt{n}$ achieved by MC methods. However, note that if the dimension of the problem d is large, this bound is only of little practical relevance, as for large d and moderate n, the logarithmic factor cannot be ignored. Furthermore, $V^{\mathrm{HK}}(g) < \infty$ is a rather strong assumption often not satisfied by integrands in practice. For instance, if g models the payoff of an option,

g is unbounded and does not have finite variation in the sense of Hardy and Krause. Even the very simple function $g(\boldsymbol{x}) = \mathbf{1}_{\{x_1+x_2\leq 1/2\}}$ does not satisfy $V^{\text{HK}}(g) < \infty$. Lastly, the Koksma-Hlawka inequality (2.12) can, in general, not be used to estimate an upper bound of the integration error in practice, as neither $D^*(P_n)$ nor $V^{\text{HK}}(g)$ can be estimated in practice. For more information about RQMC methods and their applications in the financial literature, see, e.g., [92], [76] and [40].

2.3 The effective dimension and Sobol' indices

While Equation (2.12) and the discussion thereafter might suggest that for large d and moderate n, QMC may not lead to a better error behavior than MC in practice, numerous examples in the literature have shown that QMC can significantly outperform MC methods even in high dimensions; see, for instance, [97] who successfully used QMC methods for a 360-dimensional integration problem from finance or our own numerical results in Chapter 3, where a high-dimensional distribution function is estimated via RQMC methods.

How is this possible? (R)QMC methods often work better if only a small number of variables are important, see [118] and references therein for a discussion and examples. Let us now formalize what it means to "have a small number of variables that are important".

Sensitivity indices, such as Sobol' indices, can help understand the importance of different variables of an integrand. Following [76, Ch. 6.3] and [112], we consider the ANOVA decomposition of a (square integrable) function $g : (0,1)^d \to \mathbb{R}$ given by

$$g(\boldsymbol{u}) = \sum_{I\subseteq\{1,\dots,d\}} g_I(\boldsymbol{u}),$$

where

$$g_I(\boldsymbol{u}) = \int_{[0,1]^{d-k}} g(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}_{-I} - \sum_{J\subset I} g_J(\boldsymbol{u}), \quad g_\emptyset(\boldsymbol{u}) = \int_{[0,1]^d} g(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u};$$

here, $k = \#(I)$ and $\boldsymbol{u}_{-I}$ is the vector $\boldsymbol{u}$ with components $k \in I$ deleted. The g_I's only depend on variables $i \in I$ and are orthogonal: for $I \neq J$, $\int_{[0,1)^d} g_I(\boldsymbol{u}) g_J(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u} = 0$. If $I \neq \emptyset$, g_I has mean zero. The overall variance of the integrand can then be decomposed as

$$\sigma_g^2 = \mathrm{Var}(g(\boldsymbol{U})) = \sum_{I\subseteq\{1,\dots,d\}} \sigma_I^2$$

where $\sigma_I^2 = \mathrm{Var}(g_I(\boldsymbol{U})) = \int_{[0,1]^d} g_I(\boldsymbol{u})^2 \,\mathrm{d}\boldsymbol{u}$. The number

$$S_I = \frac{\sigma_I^2}{\sigma^2} \in [0,1] \tag{2.13}$$

is called *Sobol' index* of I. It explains the fraction of the overall variance of the integrand explained by the variables in I; if this number is close to 1, it means that most of the variance is explained by g_I and therefore by the variables in I. If $I = \{l\}$ is a singleton, $S_I = S_l$ is called a *first order index*.

Another useful sensitivity index is the *total effect index* of variable $l \in \{1, \dots, d\}$ given by

$$S_{T_l} = \frac{1}{\sigma^2} \sum_{I \subseteq \{1,\dots,d\}: l \in I} \sigma_I^2 \tag{2.14}$$

which measures the relative impact of the component l and all its interactions. Care must be taken when interpreting this value as $\sum_{i=1}^d S_{T_i} \geq 1$ in general since interactions are counted several times. For instance, $\sigma_{\{1,2\}}^2$ is contained in S_{T_1} as well as in S_{T_2}.

Finally, the *effective dimension in the superposition sense* in proportion $p \in (0,1]$ is the smallest integer d_S so that

$$\frac{1}{\sigma^2} \sum_{I: |I| \leq d_S} \sigma_I^2 \geq p.$$

If the effective dimension is d_S, the integrand can be well approximated by functions of at most d_S variables. Note that there are various different notions of effective dimension; see, for instance, [76, Ch. 6] and references therein for more details. In Chapter 3.6.2, an example with estimated Sobol' indices is given; there, a special 10-dimensional integrand g is analyzed.

Now, why would a function with low effective dimension be better integrated via QMC methods? The underlying intuition is the following: If g is mostly driven by d_S dimensional functions and the point set P_n has good projection properties onto d_S dimensions (so that we do well when estimating $\int g_I(\boldsymbol{u})\mathrm{d}\boldsymbol{u}$), the overall integration error is small despite the nominal dimension being large. More formally, the decomposition of g into orthogonal components gives rise to more general versions of the Koksma-Hlawka inequality in (2.12); we only consider a stylized form and follow up with a heuristic argument:

$$|\hat{\mu}^{\mathrm{QMC}} - \mu| = \left| \frac{1}{n} \sum_{\boldsymbol{v} \in P_n} g(\boldsymbol{v}) - \int_{(0,1)^d} g(\boldsymbol{u}) \,\mathrm{d}\boldsymbol{u} \right| \leq \sum_{I \subseteq \{1,\dots,d\}} D^*(P_{n,I}) V(g_I); \tag{2.15}$$

here, $P_{n,I}$ denotes the point set P_n projected onto coordinates in I and $V(\cdot)$ denotes a suitable norm; see, for instance, [49] for details. Heuristically, if the function g is such that $V(g_I)$ is negligible for $|I| > d_S$, we can estimate

$$|\hat{\mu}^{\text{QMC}} - \mu| \leq \sum_{I \subseteq \{1,\dots,d\}: |I| \leq d_S} D^*(P_{n,I}) V(g_I)$$

and obtain a small integration error if the projections of P_n onto components in I are low-discrepancy point sets. This can then lead to an order of convergence of $\mathcal{O}\left(\log(n)^{d_S}/n\right)$ (as opposed to the slower convergence rate $\mathcal{O}\left(\log(n)^d/n\right)$).

Whether or not having a small effective dimension is necessary or sufficient (or neither) for superiority of QMC methods over MC methods is still an open problem; see, for instance, [11], [95], [114] and [119]. Either way, the preceding discussion indicates that studying the effective dimension of an integrand g can help understand if (R)QMC methods perform well when integrating g.

Chapter 3

Multivariate normal variance mixtures and extensions

The multivariate normal and (Student) t distributions are among the most widely used multivariate distributions within applications in statistics, finance, insurance and risk management. Both belong to the class of *normal variance mixtures*, where we say that a random vector $\boldsymbol{X} = (X_1, \dots, X_d)$ belongs to this class, denoted $\boldsymbol{X} \sim \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$, if, in distribution,

$$\boldsymbol{X} = \boldsymbol{\mu} + \sqrt{W} A \boldsymbol{Z}, \tag{3.1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ denotes the *location (vector)*, $\Sigma = AA^\top$ for $A \in \mathbb{R}^{d \times k}$ is the *scale (matrix)* (a covariance matrix), and $W \sim F_W$ is a non-negative random variable independent of $\boldsymbol{Z} \sim \mathrm{N}_k(\boldsymbol{0}, I_k)$ (where $I_k \in \mathbb{R}^{k \times k}$ denotes the identity matrix), which we can think of as the mixing variable; see, for example, [85, Section 6.2]. Note that $(\boldsymbol{X} \mid W) \sim \mathrm{N}_d(\boldsymbol{\mu}, W\Sigma)$, hence the name of this class of distributions. This implies that if $\mathbb{E}(\sqrt{W}) < \infty$, then $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$, and if $\mathbb{E}(W) < \infty$, then $\mathrm{Cov}(\boldsymbol{X}) = \mathbb{E}(W)\Sigma$ and $\mathrm{Cor}(\boldsymbol{X}) = P$ (the correlation matrix corresponding to Σ). Furthermore, note that in the latter case with $A = I_d$ (so when the components of $\boldsymbol{X}$ are uncorrelated) the components of $\boldsymbol{X}$ are independent if and only if W is constant almost surely and thus $\boldsymbol{X}$ is multivariate normal; see [85, Lemma 6.5]. The multivariate t distribution is obtained by letting W have an inverse-gamma distribution. In what follows we focus on the case $k = d$ in which A is typically the Cholesky factor computed from a given Σ; other decompositions of Σ into $AA^\top$ for some $A \in \mathbb{R}^{d \times d}$ can be obtained from the eigendecomposition or singular-value decomposition.

Working with normal variance mixtures (as with any other multivariate distribution) often involves four tasks: sampling, computing the joint distribution function, computing

the joint density function as well as parameter estimation. Sampling is straightforward via (3.1) based on the Cholesky factor A of Σ.

As mentioned in the introduction, evaluating multivariate distribution functions (such as the normal and the t) is a challenging task. Furthermore, the joint density function of $\boldsymbol{X}$ may not be available in closed form, as is the case for the inverse-Burr mixture discussed later.

In the first six sections of this chapter, results of which are published in [53], we propose efficient algorithms for computing the joint distribution function and joint density function of a normal variance mixture, and also for estimating its parameters. The only requirement we have for the normal variance mixture is that we must have access to a function that, given a number between 0 and 1, returns the quantile of W for that input value. Note that this does not imply the quantile function is available in closed form: it could instead be that what we have access to is a numerical approximation for it. The assumption that such "black-box" procedure is available to evaluate the quantile is something we refer to as having a *computationally tractable* quantile function of W.

An extensive numerical study for all proposed algorithms is included in Section 3.6. This section also includes a detailed investigation of why the reordering algorithm works well with RQMC methods, as well as a data analysis with real-world financial data.

All presented algorithms are available in our R package `nvmix` (in particular, via `rnvmix()`, `pnvmix()`, `dnvmix()` and `fitnvmix()`); see [55], [59]. The following is an example of its usage:

```
library("nvmix") # load package
d <- 3 # dimension
scale <- diag(d) # scale matrix
loc <- rep(0, d) # location vector
n <- 100 # sample size
df <- 4.1 # degrees-of-freedom
x <- 1:d # evaluation point for density
qmix <- function(u, df) 1 / qgamma(1-u, shape = df/2, rate = df/2)
rt <- rnvmix(n, qmix = qmix, loc = loc, scale = scale, df = df)
pt <- pnvmix(x, qmix = qmix, loc = loc, scale = scale, df = df)
dt <- dnvmix(x, qmix = qmix, loc = loc, scale = scale, df = df)
fit_t <- fitnvmix(rt, qmix = qmix, mix.param.bounds = c(0.5, 10))
```

Here, `qmix` is the quantile function of an inverse-gamma distribution with shape and rate parameter `df/2`, so that the resulting mixture is multivariate t with df degrees of

freedom. Note that the functions `[r/p/d/fit]nvmix()` only have access to the quantile function of W in form of a "black box".

An important quantity in multivariate modelling is the squared Mahalanobis distance $D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma) = (\boldsymbol{X} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{X} - \boldsymbol{\mu})$. It can be used to perform graphical goodness-of-fit assessment. We provide methods to estimate the distribution and quantile function of D^2 in Section 3.5. These methods in turn can then be used to construct a Q-Q plot.

Normal variance mixtures have the following limitation: When $P = I_d$, all k-dimensional margins of $\boldsymbol{X}$ are identically distributed. In the case of a multivariate t distribution, this can be overcome by allowing different margins to have different dof. On a copula level, this leads to the notion of grouped t copulas of [21] and generalized t-copulas of [82]. Motivated by this, we define, more generally, grouped normal variance mixtures in Section 3.7, a class of distributions which includes the normal variance mixtures from (3.1). We extend the algorithms for the ungrouped case to the grouped case. In particular, we provide algorithms to estimate the joint density and distribution function, thereby filling a gap in the existing literature; these results have been published in [52].

An important task in risk management is the correct modelling of dependencies between different risk factors. Copulas have become increasingly popular to model dependencies. Consider a d-dimensional random vector $\boldsymbol{X}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with joint distribution function $F_{\boldsymbol{X}}(\boldsymbol{x}) = \mathbb{P}(X_1 \leq x_1, \dots, X_d \leq x_d)$ for $\boldsymbol{x} \in \mathbb{R}^d$ and continuous marginal distribution functions $F_j(x_j) = \mathbb{P}(X_j \leq x_j)$, $x_j \in \mathbb{R}$ for $j = 1, \dots, d$. Sklar's Theorem allows us to study the dependence among the components separately from the margins. More precisely, there is a d-dimensional *copula* C such that

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_1(x_1), \dots, F_d(x_d)), \quad \boldsymbol{x} \in \mathbb{R}^d,$$

where a copula C is defined to be a distribution function with standard uniform univariate margins. For more about copulas, see, for instance, [85, Chapter 7], [30] and [90]. The t copula, arising when $\boldsymbol{X}$ is multivariate normal, is a very popular copula to model dependencies, as, in contrast to the Gaussian copula, it is able to account for tail dependence. In Section 3.8, we address the important problem of fitting t and grouped t copulas to data. While the problem of parameter estimation for the grouped t copula was already studied in [21] and [82], the computation of the copula density which is required for the joint estimation of all dof parameters has not been investigated in full generality for arbitrary dimensions yet, which is one gap we fill in this section. Furthermore, unlike [21], we suggest joint rather than group-wise estimation of the degrees-of-freedom and show that this gives larger likelihood at the parameter estimates. We also provide an EM-like algorithm that can be applied to fit a wide range of copulas to data, and apply it to t copula and skew-t copula.

Section 3.9 concludes this chapter and gives some ideas for future research.

3.1 Normal variance mixture distribution function and density

We assume that Σ has full rank so that the density of $\boldsymbol{X} \sim \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$ exists. Denote by $D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ the (squared) Mahalanobis distance of $\boldsymbol{x} \in \mathbb{R}^d$ from $\boldsymbol{\mu}$ with respect to (wrt) Σ. By conditioning on W and substituting $w = F_W^{\leftarrow}(u)$ (where $F_W^{\leftarrow}(u) = \inf\{w \in [0,\infty) : F_W(w) \geq u\}$, $u \in (0,1)$, denotes the quantile function of F_W), the density of $\boldsymbol{X}$ can then be written as

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \int_0^\infty f_{\boldsymbol{X}|W}(\boldsymbol{x} \,|\, w) \,\mathrm{d}F_W(w) = \int_0^\infty \frac{1}{\sqrt{(2\pi w)^d |\Sigma|}} \exp\left(-\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2w}\right) \mathrm{d}F_W(w) \tag{3.2}$$

$$= \int_0^1 \frac{1}{\sqrt{(2\pi F_W^{\leftarrow}(u))^d |\Sigma|}} \exp\left(-\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2F_W^{\leftarrow}(u)}\right) \mathrm{d}u. \tag{3.3}$$

Note that this representation holds for the case when W is absolutely continuous, discrete or of mixed type. In the former case, (3.2) equals

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \int_0^\infty \frac{1}{\sqrt{(2\pi w)^d |\Sigma|}} \exp\left(-\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2w}\right) f_W(w) \,\mathrm{d}w, \tag{3.4}$$

where f_W denotes the density of W.

Furthermore, note that $f_{\boldsymbol{X}}(\boldsymbol{x})$ is decreasing in the Mahalanobis distance $D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$. Thus

$$f_{\boldsymbol{X}}(\boldsymbol{x}) \leq f_{\boldsymbol{X}}(\boldsymbol{\mu}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \mathbb{E}\left(\frac{1}{W^{d/2}}\right); \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{3.5}$$

so that $f_{\boldsymbol{X}}(\boldsymbol{x})$ is bounded if and only if $\mathbb{E}(W^{-d/2}) < \infty$.

Let $F_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{b})$ denote the probability that $\boldsymbol{X}$ falls into the hyperrectangle spanned by the lower-left endpoint $\boldsymbol{a}$ and upper-right endpoint $\boldsymbol{b}$, where $\boldsymbol{a}, \boldsymbol{b} \in \bar{\mathbb{R}}^d$ for $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ and $\boldsymbol{a} < \boldsymbol{b}$ (interpreted componentwise), where we interpret non-finite components as the corresponding limits. Note that the joint distribution function of $\boldsymbol{X}$ is a special case of $F_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{b})$ since $F_{\boldsymbol{X}}(\boldsymbol{x}) := \mathbb{P}(\boldsymbol{X} \leq \boldsymbol{x}) = F_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{x})$ for $\boldsymbol{a} = (-\infty, \ldots, -\infty)$. In what follows

we write $F(\boldsymbol{a}, \boldsymbol{b})$ instead of $F_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{b})$ to simplify notation. For computing $F(\boldsymbol{a}, \boldsymbol{b})$ assume (potentially after adjusting $\boldsymbol{a}, \boldsymbol{b}$) that $\boldsymbol{\mu} = \boldsymbol{0}$ and that Σ has full rank. By conditioning and the substitution $w = F_W^{\leftarrow}(u)$ we obtain that

$$\begin{aligned} F_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{b}) &= \mathbb{P}(\boldsymbol{a} < \boldsymbol{X} \leq \boldsymbol{b}) = \mathbb{P}(\boldsymbol{a} < \sqrt{W} A \boldsymbol{Z} \leq \boldsymbol{b}) = \mathbb{E}\left(\mathbb{P}(\boldsymbol{a}/\sqrt{W} < A\boldsymbol{Z} \leq \boldsymbol{b}/\sqrt{W} \mid W)\right) \\ &= \mathbb{E}\left(\Phi_\Sigma(\boldsymbol{a}/\sqrt{W}, \boldsymbol{b}/\sqrt{W})\right) = \int_0^\infty \Phi_\Sigma(\boldsymbol{a}/\sqrt{w}, \boldsymbol{b}/\sqrt{w}) \, \mathrm{d}F_W(w) \\ &= \int_0^1 \Phi_\Sigma\left(\boldsymbol{a}/\sqrt{F_W^{\leftarrow}(u)}, \boldsymbol{b}/\sqrt{F_W^{\leftarrow}(u)}\right) \mathrm{d}u, \end{aligned} \tag{3.6}$$

where $\Phi_\Sigma(\boldsymbol{a}, \boldsymbol{b}) = \mathbb{P}(\boldsymbol{a} < \boldsymbol{Y} \leq \boldsymbol{b})$ for $\boldsymbol{Y} \sim \mathrm{N}_d(\boldsymbol{0}, \Sigma)$.

3.2 Computing the distribution function

As mentioned in the introduction, we assume that the quantile function $F_W^{\leftarrow}$ of W is computationally tractable (possibly through an approximation). Assume furthermore that the scale matrix Σ has full rank; for the singular case, see [53, Appendix A].

One might be tempted to sample $U_i \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)$, $i = 1, \ldots, n$, and then approximate the integral in (3.6) by the conditional Monte Carlo estimator

$$F(\boldsymbol{a}, \boldsymbol{b}) \approx \hat{\mu}_F^{\mathrm{CMC}} = \frac{1}{n} \sum_{i=1}^n \Phi_\Sigma\left(\boldsymbol{a}/\sqrt{F_W^{\leftarrow}(U_i)}, \boldsymbol{b}/\sqrt{F_W^{\leftarrow}(U_i)}\right).$$

However, Φ_Σ itself is a d-dimensional integral typically evaluated by RQMC methods, so this approach would be time-consuming. Hence, the first step should be to approximate Φ_Σ. To this end, we follow [34] and start by expressing Φ_Σ (and then $F(\boldsymbol{a}, \boldsymbol{b})$) as integrals over the unit hypercube. In the second part of this section, we derive an efficient RQMC algorithm to approximate $F(\boldsymbol{a}, \boldsymbol{b})$ based on Algorithm 2.1.1. In particular, it details how a significant variance reduction (and hence decrease in run time) can be achieved through a variable reordering following an approach originally suggested by [39] for multivariate normal probabilities and later adapted by [36] to work for multivariate t probabilities.

The novelty of our approach for this problem is three-fold: first, our algorithm applies to any normal variance mixture; second, it uses RQMC methods in a way that better leverages their convergence properties, compared to previous work done for the multivariate normal and t distributions, and third, we include a detailed analysis (with our numerical results, in Section 3.6.2) of why the reordering algorithm works well with RQMC methods.

3.2.1 Reformulation of the integral

We now address Φ_Σ. Let $C = (C_{ij})_{i,j=1}^d$ be the Cholesky factor of Σ, i.e., a lower triangular matrix satisfying $CC^\top = \Sigma$. Denote by $C_k^\top$ the kth row of C for $k = 1, \dots, d$. [34] (see also [35], [36] and [37]) uses a series of transformations that rely on the lower triangular structure of C to produce a separation of variables as follows:

$$\begin{aligned}\Phi_\Sigma(\boldsymbol{a}, \boldsymbol{b}) &= \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}}{2}\right) \mathrm{d}\boldsymbol{x} \\ &= (\hat{e}_1 - \hat{d}_1) \int_0^1 (\hat{e}_2 - \hat{d}_2) \cdots \int_0^1 (\hat{e}_d - \hat{d}_d) \, \mathrm{d}u_{d-1} \dots \mathrm{d}u_1, \end{aligned} \tag{3.7}$$

where the $\hat{d}_i$ and $\hat{e}_i$ are recursively defined via

$$\hat{e}_1 = \Phi\left(\frac{b_1}{C_{11}}\right), \hat{e}_i = \hat{e}_i(u_1, \dots, u_{i-1}) = \Phi\left(\frac{b_i - \sum_{j=1}^{i-1} C_{ij} \Phi^{-1}\left(\hat{d}_j + u_j(\hat{e}_j - \hat{d}_j)\right)}{C_{ii}}\right),$$

and $\hat{d}_i$ is $\hat{e}_i$ with b_i replaced by a_i for $i = 1, \dots, d$. Note that the final integral in (3.7) is $(d-1)$-dimensional.

With this at hand, we can write (3.6) as

$$F(\boldsymbol{a}, \boldsymbol{b}) = \int_{(0,1)^d} g(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} = \int_0^1 g_1(u_0) \int_0^1 g_2(u_0, u_1) \cdots \int_0^1 g_d(u_0, \dots, u_{d-1}) \, \mathrm{d}u_{d-1} \dots \mathrm{d}u_0, \tag{3.8}$$

where

$$g(\boldsymbol{u}) = \prod_{i=1}^d g_i(u_0, \dots, u_{i-1}), \quad g_i(u_0, \dots, u_{i-1}) = e_i - d_i, \quad i = 1, \dots, d, \tag{3.9}$$

for $\boldsymbol{u} = (u_0, u_1, \dots, u_{d-1}) \in (0,1)^d$. The e_i are recursively defined by

$$\begin{aligned} e_1 &= e_1(u_0) = \Phi\left(\frac{b_1}{C_{11}\sqrt{F_W^{\leftarrow}(u_0)}}\right), \\ e_i &= e_i(u_0, \dots, u_{i-1}) = \Phi\left(\frac{1}{C_{ii}}\left(\frac{b_i}{\sqrt{F_W^{\leftarrow}(u_0)}} - \sum_{j=1}^{i-1} C_{ij} \Phi^{-1}(d_j + u_j(e_j - d_j))\right)\right), \end{aligned} \tag{3.10}$$

for $i = 2, \ldots, d$ and the d_i are e_i with b_i replaced by a_i for $i = 1, \ldots, d$. We remark that there is a typo (wrong bracket) in the corresponding formula for the special case of a multivariate t distribution in [36, p. 958].

Summarizing, the original $(d+1)$-dimensional integral is reduced to $F(\boldsymbol{a}, \boldsymbol{b}) = \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}$, with the function g defined in (3.9) so that RQMC methods from Section 2.1 could be applied directly to the problem in this form to estimate $F(\boldsymbol{a}, \boldsymbol{b})$. As pointed out in [37], the transformations undertaken in this section to produce a separation of variables essentially describe a Rosenblatt transform; see [106].

3.2.2 Variable reordering and RQMC estimation

Inspecting (3.8) and (3.10), we see that the sampled component u_j of $\boldsymbol{u}$ in the jth integral affects the ranges of all g_k with $k > j$. Observe that permuting the order in $\boldsymbol{a}$, $\boldsymbol{b}$ and Σ does not affect the value of $F(\boldsymbol{a}, \boldsymbol{b})$ as long as the same permutation is applied to $\boldsymbol{a}$, $\boldsymbol{b}$ and to both the rows and columns of Σ. It therefore seems to be a fruitful approach to choose a permutation of $\boldsymbol{a}$, $\boldsymbol{b}$ and Σ such that g_2 has, on average, the smallest range; g_3 the second smallest, and so on. This has been observed in [39] in the context of calculating multivariate normal probabilities and has been adapted by [36] to handle multivariate t integrals. As in the latter reference, one can sort the integration limits a priori according to their expected length of integration limits. This is more complicated than just ordering $\boldsymbol{a}$, $\boldsymbol{b}$ and Σ according to the lengths $b_i - a_i$ (assuming all of them are finite) as the latter does not take into account the dependence of the components in $\boldsymbol{X}$. We generalize the Gibson, Glasbey and Elston method for reordering according to expected ranges to work for normal variance mixture distribution functions in Algorithm 3.2.1.

Algorithm 3.2.1 (Variable reordering) 1. Start with given $\boldsymbol{a}, \boldsymbol{b}$ and Σ.

2. Calculate or approximate $\mu_{\sqrt{W}} = \mathbb{E}(\sqrt{W})$.

3. a) Choose the first integration variable as

$$i = \operatorname*{argmin}_{j \in \{1,\ldots,d\}} \left\{ \Phi\left(\frac{b_j}{\mu_{\sqrt{W}} \sqrt{\Sigma_{jj}}} \right) - \Phi\left(\frac{a_j}{\mu_{\sqrt{W}} \sqrt{\Sigma_{jj}}} \right) \right\}.$$

Swap components 1 and i of $\boldsymbol{a}$ and $\boldsymbol{b}$ and interchange both rows and columns of Σ corresponding to the variables i and 1.

b) Update $C_{11} = \sqrt{\Sigma_{11}}$ and $C_{j1} = \Sigma_{j1}/C_{11}$ for $j = 1, \ldots, d$. Set

$$y_1 = \frac{\int_{\hat{a}_1}^{\hat{b}_1} s\phi(s)ds}{\Phi(\hat{b}_1) - \Phi(\hat{a}_1)}$$

as expected value for u_1, where

$$\hat{a}_1 = \frac{a_1}{\mu_{\sqrt{W}} C_{11}} \quad \text{and} \quad \hat{b}_1 = \frac{b_1}{\mu_{\sqrt{W}} C_{11}}.$$

This is the same as $\mathbb{E}(Z \mid Z \in [\hat{a}_1, \hat{b}_1])$ for $Z \sim N(0,1)$.

4. For $j = 2, \ldots, d$,

a) Choose the jth integration variable as

$$i = \underset{l \in \{j,\ldots,d\}}{\operatorname{argmin}} \left\{ \Phi\left(\frac{\frac{b_l}{\mu_{\sqrt{W}}} - \sum_{k=1}^{j-1} C_{lk} y_k}{\sqrt{\Sigma_{l,l} - \sum_{k=1}^{j-1} C_{lk}^2}} \right) - \Phi\left(\frac{\frac{a_l}{\mu_{\sqrt{W}}} - \sum_{k=1}^{j-1} C_{lk} y_k}{\sqrt{\Sigma_{l,l} - \sum_{k=1}^{j-1} C_{lk}^2}} \right) \right\}.$$

Swap components i and j of $\boldsymbol{a}$ and $\boldsymbol{b}$ and interchange both rows and columns of Σ corresponding to variables i and j and interchange rows i and j in C.

b) Update $C_{jj} = \sqrt{\Sigma_{jj} - \sum_{k=1}^{j-1} C_{jk}^2}$ and $C_{lj} = \frac{1}{C_{jj}} \left(\Sigma_{lj} - \sum_{k=1}^{j-1} C_{jk} C_{lk} \right)$ for $l = j+1, \ldots, d$ and set

$$y_j = \frac{\int_{\hat{a}_j}^{\hat{b}_j} s\phi(s)ds}{\Phi(\hat{b}_j) - \Phi(\hat{a}_j)},$$

where

$$\hat{a}_j = \frac{\frac{a_j}{\mu_{\sqrt{W}}} - \sum_{k=1}^{j-1} C_{jk} y_k}{C_{jj}} \quad \text{and} \quad \hat{b}_j = \frac{\frac{b_j}{\mu_{\sqrt{W}}} - \sum_{k=1}^{j-1} C_{jk} y_k}{C_{jj}}.$$

From a simulation point of view, the particular value of u_1 will affect the ranges of all the remaining $d-2$ integrals. Indeed, each input $\boldsymbol{u} = (u_0, \ldots, u_{d-1}) \sim \mathrm{U}(0,1)^d$ is transformed to a product of conditional probabilities: The first component, u_0, is used to sample from the mixing variable via inversion; $g_1(u_0)$ is then the conditional probability of the first component of the random vector $\boldsymbol{X}$ falling into (a_1, b_1) given that $W = F_W^{\leftarrow}(u_0)$, that is $g_1(u_0) = \mathbb{P}(X_1 \in (a_1, b_1) \mid W = F_W^{\leftarrow}(u_0))$. Next, u_1 is transformed to $y_1 = \Phi^{-1}(d_1 + u_1(e_1 - d_1))$, which is a realization of the random variable $(X_1 \mid X_1 \in (a_1, b_1), W = F_W^{\leftarrow}(u_0))$. Then,

$g_2(u_0, u_1) = \mathbb{P}(X_2 \in (a_2, b_2) \mid X_1 = y_1, W = F_W^{\leftarrow}(u_0))$ and so on and so forth. As we are conditioning on events of the form $\{X_1 = y_1, \dots, X_l = y_l, W = F_W^{\leftarrow}(u_0)\}$ for all subsequent probabilities, this also explains why variable reordering can help decrease the variance: It is designed in a way so that X_1 has smallest (expected) range, X_2 second smallest and so on. In the explanation above, if $b_1 - a_1$ is small, there is only little variability in y_1 so that $g(u_0, u_1)$ should be close to $\mathbb{P}(X_2 \in (a_2, b_2) \mid X_1 \in (a_1, b_1), W = F_W^{\leftarrow}(u_0))$. We point out that if $F_W^{\leftarrow}(u)$ is a non-zero constant for all $u \in (0, 1)$ (corresponding to $\boldsymbol{X}$ being multivariate normal), this is the original derivation in [39] who independently developed a Monte Carlo procedure to approximate multivariate normal probabilities similar to [34].

Algorithm 3.2.1 is a greedy procedure that only reorders $\boldsymbol{a}, \boldsymbol{b}$, Σ (and updates the Cholesky factor C accordingly). Changing the order in $\boldsymbol{a}$, $\boldsymbol{b}$ and Σ does not introduce any bias so that one can use a rather crude approximation for $\mu_{\sqrt{W}}$ for $\mathbb{E}(\sqrt{W})$ if the true mean is not known. Note also that variable reordering needs to be performed only once before applying RQMC to the integrand g in (3.9) so that the cost of reordering is low compared to the overall cost of evaluating $F(\boldsymbol{a}, \boldsymbol{b})$.

Our method to estimate $F(\boldsymbol{a}, \boldsymbol{b})$ is summarized in Algorithm 3.2.2, where in Step 2 antithetic variates are employed as a simple variance reduction technique.

Algorithm 3.2.2
Given $\boldsymbol{a}, \boldsymbol{b}, \Sigma$, ε, B, n_0, $i_{\max}$, estimate $F(\boldsymbol{a}, \boldsymbol{b})$ as follows:

1. Apply the reordering Algorithm 3.2.1 to the inputs $\boldsymbol{a}, \boldsymbol{b}, \Sigma$.
2. Apply Algorithm 2.1.1 on the integrand $(g(\boldsymbol{u}) + g(\boldsymbol{1} - \boldsymbol{u}))/2$ with g from (3.9) and reordered inputs.

In Section 3.6.2 it is shown through a simulation study that this (rather cheap) variable reordering can yield a great variance reduction for the RQMC algorithm, Algorithm 3.2.2. A detailed study as to why this works so well is included in Section 3.6.2.

Note that parallelization of our methods, i.e., estimation of $F(\boldsymbol{a}_i, \boldsymbol{b}_i)$, $i = 1, \dots, n$, simultaneously is difficult for two reasons: Reordering needs to be performed for each input $\boldsymbol{a}_i, \boldsymbol{b}_i$ separately so that Algorithm 3.2.1 needs to be called n times. Furthermore, the structure of the integrand g from (3.9) (see also (3.10)) does not allow for an efficient implementation of common random numbers as all quantile evaluations $\Phi^{-1}(\cdot)$ depend on the limits $\boldsymbol{a}$, $\boldsymbol{b}$ so that they cannot be recycled.

3.3 Computing the (logarithmic) density

We now turn to the task of computing the (logarithmic) density function of a normal variance mixture. Let us first point out that the main reason why we need to be able to evaluate the density function is for the fitting procedure, which is likelihood-based and is explained in detail in Section 3.4. Now, since our goal is to be able to cover a variety of normal variance mixtures, we cannot assume that the density function of $\boldsymbol{X}$ is available in closed form. Indeed, a closed form for $f_{\boldsymbol{X}}(\boldsymbol{x})$ exists in some cases (e.g., when W is an inverse-gamma or Pareto), but not in all cases (e.g., when W follows an inverse-Burr distribution, a model actually used with success in Section 3.6.2). For those latter cases, an efficient approximation is needed, as there is likely to be a repeated need for evaluating the density (or log-density) within the fitting procedure. This also means that fitting algorithms proposed for the multivariate t cannot be directly adapted for the general normal variance mixture case, as they would not include functionalities able to deal with a density that does not exist in closed form. Below we propose an adaptive RQMC algorithm to deal with those cases, which is based on the ideas presented in Section 2.1.

From (3.3) it follows that computing the density at $\boldsymbol{x}$ requires the evaluation of the univariate integral $\mu := f_{\boldsymbol{X}}(\boldsymbol{x}) = \int_0^1 h(u)\,\mathrm{d}u$, where

$$h(u) = \frac{1}{\sqrt{(2\pi F_W^{\leftarrow}(u))^d |\Sigma|}} \exp\left(-\frac{D^2(\boldsymbol{x};\boldsymbol{\mu},\Sigma)}{2F_W^{\leftarrow}(u)}\right), \quad u \in (0,1). \tag{3.11}$$

To simplify notation, we write $f(\boldsymbol{x})$ instead of $f_{\boldsymbol{X}}(\boldsymbol{x})$ whenever confusion is not possible. Furthermore, we assume that $f(\boldsymbol{x})$ is finite; see also (3.5).

For likelihood-based methods one should compute the logarithmic density (or *log-density*) rather than the density. Since $f(\boldsymbol{x})$ is expressed as a univariate integral over $(0,1)$, Algorithm 2.1.2, that is, RQMC methods combined with a proper logarithm as described at the end of Section 2.1 on Page 11, can be applied directly to estimate $\log(\mu) = \log f(\boldsymbol{x})$ via RQMC. In fact, given inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-densities $\log f(\boldsymbol{x}_1), \ldots, \log f(\boldsymbol{x}_N)$ can be estimated simultaneously by using the same realizations of W, i.e., using the same $F_W^{\leftarrow}(u_{i,b})$ for all inputs $\boldsymbol{x}_k$, $k = 1, \ldots, N$, until the precision is reached for all inputs. This procedure, i.e. estimating $\log \mu$ directly based on (3.11) via RQMC, will be referred to as the *crude procedure*.

It turns out that the crude procedure works sufficiently well for inputs $\boldsymbol{x}$ with small to moderate Mahalanobis distances, but deteriorates for larger Mahalanobis distances. The reason is that the overall shape of the integrand h is heavily influenced by $D^2(\boldsymbol{x};\boldsymbol{\mu},\Sigma)$ and for large values, most of the mass is concentrated in a small domain of $(0,1)$. This is

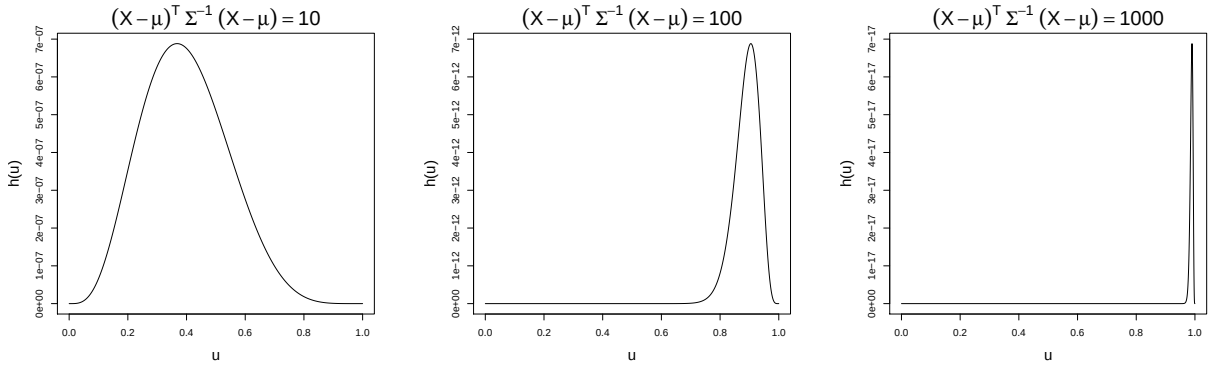

Figure 3.1: Integrand h for a 10-dimensional t distribution with 2 degrees of freedom.

illustrated in Figure 3.1 where the integrand $h(u)$ is plotted against u in the special case where $\boldsymbol{X}$ follows a multivariate t distribution in dimension 10 with 2 degrees of freedom. For instance, in the right-most plot, most of the mass is concentrated near 1. It thus seems to be a fruitful approach to tailor the integration routine in a way so that it samples mostly in this relevant domain around the maximum, giving rise to an *adaptive algorithm*. To this end, we summarize some properties of the function h in the following lemma which can be shown using elementary calculus.

Lemma 3.3.1

Let W have a continuous distribution supported on the whole positive real line. Then, the function h from Equation (3.11) is continuous on $(0,1)$, satisfies $h(0) = h(1) = 0$ and $h(u) > 0$ for $u \in (0,1)$. Furthermore, the maximum value of h on $(0,1)$, i.e., $h_{\max} = \max\{h(u) : u \in [0,1]\}$ is attained in the interior of $(0,1)$. The maximum is attained at

$$u^* = F_W\left(\frac{D^2(\boldsymbol{x};\boldsymbol{\mu},\Sigma)}{d}\right) \text{ with } h(u^*) = \left(\frac{2\pi|\Sigma|^{1/d}\cdot D^2(\boldsymbol{x};\boldsymbol{\mu},\Sigma)}{d}\right)^{-d/2}\exp\left(-\frac{d}{2}\right), \quad (3.12)$$

so that $h_{\max}$ is independent of the distribution of W. Finally, h is strictly increasing on $(0,u^*)$ and strictly decreasing on $(u^*,1)$.

Equation (3.12) is crucial for the adaptive algorithm we propose: The value $h_{\max}$, i.e., the height of the peak of the integrand h, is independent of the distribution of W as long as W is continuous and supported on the whole positive real line. If W is continuous but has bounded support, $h_{\max}$ may need to be replaced by $h(0)$ or $h(1)$. If the mixing variable W takes on finitely many values, the problem becomes trivial as the density becomes a finite sum.

The idea is now to apply RQMC to a relevant region around u^* from (3.12), which can be done as follows: Given a threshold ε_{th} with $0 < \varepsilon_{\text{th}} \ll h_{\max}$, the structure of the integrand h guarantees the existence of u_l and u_r (l for "left" and r for "right") with $0 < u_l < u^* < u_r < 1$ so that $h(u) > \varepsilon_{\text{th}}$ if and only if $u \in (u_l, u_r)$. For instance, take

$$\varepsilon_{\text{th}} = 10^{\log(h_{\max})/\log(10) - k_{\text{th}}}, \tag{3.13}$$

where $k_{\text{th}} = 10$ so that ε_{th} is 10 orders smaller than $h_{\max}$. RQMC can then be used in the region (u_l, u_r) by replacing every number $v \in (0,1)$ by $v' = u_l + (u_r - u_l)v \in (u_l, u_r)$ yielding an estimate for $\log \int_{u_l}^{u_r} h(u)\, \mathrm{d}u$. For the remaining regions $(0, u_l)$ and $(u_r, 1)$ we suggest using a crude trapezoidal rule: If $\varepsilon_{\text{th}} \ll h_{\max}$ those regions do not significantly contribute to the overall integral anyway, so a rather cheap and quick procedure is recommended here.

It remains to discuss how the numbers u_l, u^*, u_r can be computed. Recall that the only information available about W is its quantile function $F_W^{\leftarrow}$ in form of a "black box" so that u^* from (3.12) cannot be computed directly. We suggest using a bisection algorithm to solve the equivalent equation $F_W^{\leftarrow}(u) = \boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}/d$. Starting values can be found using a small number of pilot runs. Similarly, there is no direct formula for u_l and u_r. While those can be expressed using Lambert's W function, the lack of information about W does not allow a direct computation. A bisection can be used here as well. Clearly, all pilot runs and all quantile evaluations performed in the bisections should be stored so that those expensive evaluations can be re-used.

It is clear from Figure 3.1 that the shape of the integrand heavily depends on $\boldsymbol{x}$ through its Mahalanobis distance, and this holds true for u_l, u^*, u_r as well. As such, the adaptive procedure just described does not allow for simultaneous estimation of $\log f(\boldsymbol{x}_1), \dots, \log f(\boldsymbol{x}_N)$ directly, as the regions to which RQMC is applied differ from one input to another one. In order to reduce run time, we suggest using the crude procedure on all inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ with a small number of iterations (say, $i_{\max} = 4$) first and use the adaptive procedure only for those inputs $\boldsymbol{x}_j$ whose error estimates did not reach the tolerance. The advantage is that only little run time is spent on estimating "easy" integrals. Furthermore, if $i_{\max} = 4$, $B = 15$ and the initial sample size is $n_0 = 128$, such pilot run gives 7680 pairs $(u, F_W^{\leftarrow}(u))$. These can be used to determine starting values for the bisections to find u_l, u_r and u^* and they can also be used to estimate the integral in the regions $(0, u_l)$ and $(u_r, 1)$ using a trapezoidal rule with non-equidistant knots. The following algorithm summarizes our procedure, which is implemented in the R function `dnvmix(, log = TRUE)` of the R package `nvmix`.

Algorithm 3.3.2 (Adaptive RQMC Algorithm to estimate $\log f(\boldsymbol{x}_1), \dots, \log f(\boldsymbol{x}_N)$)
Given $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$, Σ, ε, $\varepsilon_{\text{bisec}}$, B, $i_{\max}$, k_{th}, estimate $\log f(\boldsymbol{x}_l)$, $l = 1, \dots, N$, via:

1. Apply Algorithm 2.1.2 with at most $i_{\max}$ iterations on all inputs $\boldsymbol{x}_l$, $l = 1, \dots, N$. Store all uniforms and corresponding quantiles $F_W^{\leftarrow}(\cdot)$ in a list, say $\mathcal{L}$.

2. If all estimates $\mu^{\mathrm{RQMC}}_{\log f(\boldsymbol{x}_l)}$, $l = 1, \dots, N$ meet the error tolerance ε, go to Step 4.
If not, we can assume wlog (after reordering) that $\boldsymbol{x}_s$, $s = 1, \dots, N'$ with $1 \le N' \le N$ are the inputs whose error estimates did not meet the error tolerance.

3. For each remaining input $\boldsymbol{x}_s$, $s = 1, \dots, N'$, do the following:

   (a) Determine $h_{\max}$ using (3.12) and $\varepsilon_{\mathrm{th}}$ using (3.13).

   (b) Find maximal $u^{*,\mathrm{l}}$ and minimal $u^{*,\mathrm{r}}$ in the list $\mathcal{L}$ so that $F_W^{\leftarrow}(u^{*,\mathrm{l}}) \le \boldsymbol{x}_s^\top \Sigma^{-1} \boldsymbol{x}_s / d \le F_W^{\leftarrow}(u^{*,\mathrm{r}})$ (which implies $u^{*,\mathrm{l}} \le u^* \le u^{*,\mathrm{r}}$). Use a bisection algorithm with starting values $u^{*,\mathrm{l}}$ and $u^{*,\mathrm{r}}$ and a tolerance of $\varepsilon_{\mathrm{bisec}}$ to find u^*. Add any additional u's and $F_W^{\leftarrow}(u)$'s computed in the bisection to the list $\mathcal{L}$.

   (c) Find the largest number $u_l^{(1)} \in \mathcal{L}$ and the smallest number $u_l^{(2)} \in \mathcal{L}$ such that $u_l^{(1)} \le u_l^{(2)} \le u^*$, $h(u_l^{(1)}) \le \varepsilon_{\mathrm{th}}$ and $h(u_l^{(2)}) \ge \varepsilon_{\mathrm{th}}$. Then $u_l^{(1)} \le u_l \le u_l^{(2)} \le u^*$.
   Similarly, find the largest number $u_r^{(1)} \in \mathcal{L}$ and the smallest number $u_r^{(2)} \in \mathcal{L}$ such that $u^* \le u_r^{(1)} \le u_r^{(2)}$, $h(u_r^{(1)}) \ge \varepsilon_{\mathrm{th}}$ and $h(u_r^{(2)}) \le \varepsilon_{\mathrm{th}}$. Then $u^* \le u_r^{(1)} \le u_r \le u_r^{(2)}$.
   Then use a bisection to find u_l (using starting values $u_l^{(1)}$ and $u_l^{(2)}$) and u_r (using starting values $u_r^{(1)}$ and $u_r^{(2)}$) with a tolerance of $\varepsilon_{\mathrm{bisec}}$. Add any additional u's and $F_W^{\leftarrow}(u)$'s computed in the bisection to the list $\mathcal{L}$.

   (d) Approximate $\log \int_0^{u_l} h(u)\,\mathrm{d}u$ using a trapezoidal rule with knots $u_1', \dots, u_m'$ where u_i' are those u's in $\mathcal{L}$ satisfying $u \le u_l$. Call the approximation $\hat{\mu}_{(0,u_l)}(\boldsymbol{x}_s)$.

   (e) Approximate $\log \int_{u_r}^1 h(u)\,\mathrm{d}u$ using a trapezoidal rule with knots $u_1'', \dots, u_p''$ where u_i'' are those u's in $\mathcal{L}$ satisfying $u \ge u_r$. Call the approximation $\hat{\mu}_{(u_r,1)}(\boldsymbol{x}_s)$.

   (f) Apply Algorithm 2.1.2 where all uniforms $v \in (0,1)$ are replaced by $v' = u_l + (u_r - u_l)v \in (u_l, u_r)$. Call the output $\widehat{\log \mu}$. Then set $\widehat{\log \mu}_{(u_l,u_r)} = \log(u_r - u_l) + \widehat{\log \mu}$ which estimates $\log \int_{u_l}^{u_r} h(u)\,\mathrm{d}u$.

   (g) Combine
   $$\hat{\mu}^{\mathrm{RQMC}}_{\log f(\boldsymbol{x}_s)} = \mathrm{LSE}\left(\hat{\mu}_{(0,u_l)}(\boldsymbol{x}_s), \hat{\mu}_{(u_l,u_r)}(\boldsymbol{x}_s), \hat{\mu}_{(u_r,1)}(\boldsymbol{x}_s)\right).$$

4. Return $\hat{\mu}^{\mathrm{RQMC}}_{\log f(\boldsymbol{x}_l)}$, $l = 1, \dots, N$.

Remark 3.3.3
Algorithm 3.3.2 can be applied to estimate a slightly larger class of integrals. Let

$$\mu = \int_0^\infty cw^{-k} \exp(m/w)\, \mathrm{d}F_W(w) = \int_0^\infty \tilde{h}(u)\, \mathrm{d}u;$$

here, $k, m > 0$ are constant and $\tilde{h}(u) = cF_W^{\leftarrow}(u)^{-k} \exp(m/F_W^{\leftarrow}(u))$ for $u \in (0,1)$. A result similar to Lemma 3.3.1 applies to $\tilde{h}$ (replace d by k in the formula for u^* in (3.12)). Thus, after only slight adjustments to Algorithm 3.3.2, the latter can be used to estimate $\log(\mu)$ efficiently. This will be useful in Section 3.4.

3.4 Fitting multivariate normal variance mixtures

In this section, we derive an expectation-maximization (EM)-like algorithm whose distinctive feature is that it can estimate the parameters of any given normal variance mixture, provided its density function is bounded. Its design is inspired by the ECME algorithm used for fitting multivariate t models, but is appropriately modified to allow for a general mixing variable W. This requirement means that our approach must be able to handle the case where the density $f_{\boldsymbol{X}}(\boldsymbol{x})$ may not exist in closed form, and must therefore be approximated. The fact that ECME-type algorithms break the optimization part into two steps—and thus handle the parameters $\boldsymbol{\nu}$ of W's distribution separately from $\boldsymbol{\mu}$ and Σ—meshes very well with our assumption that all we may know about W is through access to a "black-box" function for its quantile function. That is, since the step to find $\boldsymbol{\nu}$ is done separately, we can easily make it adaptable to whether or not W's distribution is such that $f_{\boldsymbol{X}}(\boldsymbol{x})$ exists in closed form. More precisely, in our R implementation, we assume the user either provides a "black-box" function for the quantile function of W—in which case $f_{\boldsymbol{X}}(\boldsymbol{x})$ is approximated using the algorithm described in the previous section—or specifies that W is constant, inverse-gamma, or Pareto, in which case $f_{\boldsymbol{X}}(\boldsymbol{x})$ is evaluated exactly. Examples provided in Section 3.6.2 demonstrate that the versatility of our algorithm, which we now explain, does not come at the cost of decreased accuracy.

Assume $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \overset{\text{ind.}}{\sim} \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$ with unknown location vector $\boldsymbol{\mu}$ and unknown scale matrix Σ where F_W has quantile function $F_W^{\leftarrow}(u; \boldsymbol{\nu})$ with unknown parameter vector $\boldsymbol{\nu} \in \widetilde{\mathcal{V}}$ of length $p_{\boldsymbol{\nu}}$, where the set $\mathcal{V}$ is such that $\mathbb{E}(W^{-d/2}) < \infty$ for all $\boldsymbol{\nu} \in \mathcal{V}$. This assumption assures that the likelihood function is bounded; see also (3.5). For notational convenience, let $\boldsymbol{\theta} = (\boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma^{-1})$ and denote by $\boldsymbol{\theta}_k$ the current value of $\boldsymbol{\theta}$ in iteration k.

Before deriving our algorithm, we need some notation. The original log-likelihood is given by

$$\log L^{\text{org}}(\boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n) = \sum_{i=1}^{n} \log f_{\boldsymbol{X}}(\boldsymbol{X}_i; \boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma)$$

and the complete log-likelihood $\log L^{\text{c}}$ can be written as

$$\begin{aligned} \log L^{\text{c}}(\boldsymbol{\theta}; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, W_1, \ldots, W_n) &= \sum_{i=1}^{n} \log f_{\boldsymbol{X},W}(\boldsymbol{X}_i, W_i; \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \log f_{\boldsymbol{X}|W}(\boldsymbol{X}_i \,|\, W_i; \boldsymbol{\mu}, \Sigma) + \sum_{i=1}^{n} \log f_W(W_i; \boldsymbol{\nu}), \end{aligned} \tag{3.14}$$

where $W_1, \ldots, W_n$ are (unobserved) iid copies of W. Note that the first sum in (3.14) contains the log-likelihood contributions of $\mathrm{N}_d(\boldsymbol{\mu}, W_i\Sigma)$ and thus is almost the log-likelihood of a normal distribution apart from potentially different W_i (expected, for example, if W is continuously distributed on the whole positive real line). The expected value of the complete log-likelihood given the (observed) data $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ and current estimate $\boldsymbol{\theta}_k$ is then

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k) = \mathbb{E}(\log L^{\text{c}}(\boldsymbol{\theta}; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, W_1, \ldots, W_n) \,|\, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n; \boldsymbol{\theta}_k). \tag{3.15}$$

As mentioned earlier, rather than trying to maximize $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ over $\boldsymbol{\theta}$ as a classical EM algorithm would do, we instead employ an ECME algorithm as developed in [79]; see also references therein for more details on variations of the EM algorithm. In this way, and as explained below, optimization is broken into two steps, which respectively deal with $(\boldsymbol{\mu}, \Sigma)$ and $\boldsymbol{\nu}$.

The basic structure of our algorithm is as follows:

Algorithm 3.4.1 (ECME Algorithm for fitting normal variance mixtures: Main idea) Given iid data $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$, estimate $\boldsymbol{\mu}, \Sigma, \boldsymbol{\nu}$ via:

1. Obtain an initial estimate $\boldsymbol{\theta}_0 = (\boldsymbol{\nu}_0, \boldsymbol{\mu}_0, \Sigma_0^{-1})$.
2. For $k = 1, \ldots,$ repeat until convergence:
   (a) Update $\boldsymbol{\mu}_k$ and Σ_k by maximizing $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ with respect to $\boldsymbol{\mu}$ and Σ with $\boldsymbol{\nu} = \boldsymbol{\nu}_{k-1}$ held fixed.

(b) Update $\boldsymbol{\nu}_k$ by maximizing $\log L^{\text{org}}(\boldsymbol{\nu}, \boldsymbol{\mu}_k, \Sigma_k; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ with respect to $\boldsymbol{\nu}$.

That is, in the k'th iteration, we first update $\boldsymbol{\mu}$ and Σ by maximizing the expected complete log-likelihood conditional on the observed data and then update $\boldsymbol{\nu}$ by maximizing the original likelihood with respect to $\boldsymbol{\nu}$ with $\boldsymbol{\mu}$ and Σ set to their current estimates. This is an ECME algorithm as we either maximize the expected complete log-likelihood or the original likelihood; see also [80] for a discussion of an ECME algorithm for the multivariate t distribution.

Let

$$\xi_{ki} = \mathbb{E}(\log W_i \,|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k) \quad \text{and} \quad \delta_{ki} = \mathbb{E}(1/W_i \,|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k), \quad i = 1, \ldots, n.$$

We calculate $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ from (3.15) in the following lemma:

Lemma 3.4.2
$Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ from (3.15) allows for the decomposition $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k) = Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) + Q_W(\boldsymbol{\nu}; \boldsymbol{\theta}_k)$, where

$$Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) = -\frac{1}{2}\left(nd \log(2\pi) - n \log(\det(\Sigma^{-1})) + \sum_{i=1}^{n} (D^2(\boldsymbol{X}_i; \boldsymbol{\mu}, \Sigma)\delta_{ki} + d\xi_{ki}) \right),$$

$$Q_W(\boldsymbol{\nu}; \boldsymbol{\theta}_k) = \sum_{i=1}^{n} \mathbb{E}(\log f_W(W_i; \boldsymbol{\nu}) \,|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k).$$

Proof. Starting from (3.15) and using (3.14) we obtain

$$\begin{aligned}
Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k) &= \mathbb{E}(\log L^{\text{c}}(\boldsymbol{\theta}; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n, W_1, \ldots, W_n) \,|\, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n; \boldsymbol{\theta}_k) \\
&= \sum_{i=1}^{n} \mathbb{E}(\log f_{\boldsymbol{X}|W}(\boldsymbol{X}_i \,|\, W_i; \boldsymbol{\mu}, \Sigma) \,|\, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n; \boldsymbol{\theta}_k) \\
&\quad + \sum_{i=1}^{n} \mathbb{E}(\log f_W(W_i; \boldsymbol{\nu}) \,|\, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n; \boldsymbol{\theta}_k) \\
&= \sum_{i=1}^{n} \mathbb{E}(\log f_{\boldsymbol{X}|W}(\boldsymbol{X}_i \,|\, W_i; \boldsymbol{\mu}, \Sigma) \,|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k) + \sum_{i=1}^{n} \mathbb{E}(\log f_W(W_i; \boldsymbol{\nu}) \,|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k) \\
&= Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) + Q_W(\boldsymbol{\nu}; \boldsymbol{\theta}_k),
\end{aligned}$$

where the first expectation is taken with respect to $W_1, \ldots, W_n$ for given $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ and $\boldsymbol{\theta}_k$, and the last line of the displayed equation is understood as the definition of $Q_{\boldsymbol{X}|W}$ and

Q_W. Using the fact that $\boldsymbol{X} \mid W \sim \mathrm{N}_d(\boldsymbol{\mu}, W\Sigma)$, it is easily verified that

$$\begin{aligned} Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) &= \sum_{i=1}^{n} \mathbb{E}(\log f_{\boldsymbol{X}|W}(\boldsymbol{X}_i \mid W_i; \boldsymbol{\mu}, \Sigma) \mid \boldsymbol{X}_i; \boldsymbol{\theta}_k) \\ &= -\frac{1}{2}\left(nd\log(2\pi) - n\log(\det(\Sigma^{-1})) + \sum_{i=1}^{n}(D^2(\boldsymbol{X}_i; \boldsymbol{\mu}, \Sigma)\delta_{ki} + d\xi_{ki}) \right). \end{aligned}$$

□

With $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ at hand, we show in the following lemma how $\boldsymbol{\mu}$ and Σ are updated in Step 2a of Algorithm 3.4.1.

Lemma 3.4.3
Maximizing $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k)$ with respect to $\boldsymbol{\mu}$ and Σ in Step 2a of Algorithm 3.4.1 gives the next iterates

$$\boldsymbol{\mu}_{k+1} = \frac{\sum_{i=1}^{n} \delta_{ki}\boldsymbol{X}_i}{\sum_{i=1}^{n} \delta_{ki}} \quad \text{and} \quad \Sigma_{k+1} = \frac{1}{n}\sum_{i=1}^{n} \delta_{ki}(\boldsymbol{X}_i - \boldsymbol{\mu}_k)(\boldsymbol{X}_i - \boldsymbol{\mu}_k)^\top. \tag{3.16}$$

Proof. By Lemma 3.4.2, $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_k) = Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) + Q_W(\boldsymbol{\nu}; \boldsymbol{\theta}_k)$ and $\boldsymbol{\mu}$ and Σ do not appear in $Q_W(\boldsymbol{\nu}; \boldsymbol{\theta}_k)$ so that we only need to maximize $Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k)$.
The necessary conditions are $\frac{\partial}{\partial \boldsymbol{\mu}} Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) = 0$ and $\frac{\partial}{\partial \Sigma^{-1}} Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) = 0$. Using $\frac{\partial}{\partial \boldsymbol{\mu}} D^2(\boldsymbol{X}_i; \boldsymbol{\mu}, \Sigma) = -2\Sigma^{-1}(\boldsymbol{X}_i - \boldsymbol{\mu})$ one obtains that $\frac{\partial}{\partial \boldsymbol{\mu}} Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) = 0$ if and only if $\sum_{i=1}^{n} \delta_{ki}\Sigma^{-1}(\boldsymbol{X}_i - \boldsymbol{\mu}) = 0$. Solving for $\boldsymbol{\mu}$ gives $\boldsymbol{\mu}_{k+1}$ as given in the lemma.For full rank Σ, it holds that $\frac{\partial}{\partial \Sigma^{-1}} \log\det(\Sigma^{-1}) = \Sigma$. Since $\frac{\partial}{\partial \Sigma^{-1}} D^2(\boldsymbol{X}_i; \boldsymbol{\mu}, \Sigma) = (\boldsymbol{X}_i - \boldsymbol{\mu})(\boldsymbol{X}_i - \boldsymbol{\mu})^\top$ one gets $\frac{\partial}{\partial \Sigma^{-1}} Q_{\boldsymbol{X}|W}(\boldsymbol{\mu}, \Sigma^{-1}; \boldsymbol{\theta}_k) = 0$ if and only if $n\Sigma - \sum_{i=1}^{n} \delta_{ki}(\boldsymbol{X}_i - \boldsymbol{\mu})(\boldsymbol{X}_i - \boldsymbol{\mu})^\top = 0$ which, after solving for Σ, gives the formula for Σ_{k+1} as given in the statement. □

Lemma 3.4.3 indicates that we need to approximate the weights δ_{ki}, $i = 1, \ldots, n$, in Step 2a of Algorithm 3.4.1. Note that

$$\mathrm{d}F_{W|\boldsymbol{X}}(w \mid \boldsymbol{x}) = \frac{f_{\boldsymbol{X}|W}(\boldsymbol{x} \mid w)\,\mathrm{d}F_W(w)}{f_{\boldsymbol{X}}(\boldsymbol{x})} = \frac{\phi(\boldsymbol{x}; \boldsymbol{\mu}, w\Sigma)}{f_{\boldsymbol{X}}(\boldsymbol{x})}\,\mathrm{d}F_W(w), \quad w > 0,$$

where $\phi(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$ denotes the density of $\mathrm{N}_d(\boldsymbol{\mu}, \Sigma)$ so that

$$\begin{aligned} \delta_{ki} = \mathbb{E}\left(\frac{1}{W_i} \,\middle|\, \boldsymbol{X}_i; \boldsymbol{\theta}_k \right) &= \int_0^\infty \frac{1}{w}\,\mathrm{d}F_{W|\boldsymbol{X}}(w \mid \boldsymbol{X}_i) \\ &= \frac{1}{f_{\boldsymbol{X}}(\boldsymbol{X}_i; \boldsymbol{\mu}_k, \Sigma_k, \boldsymbol{\nu}_k)} \int_0^\infty \frac{\phi(\boldsymbol{X}_i; \boldsymbol{\mu}_k, w\Sigma_k)}{w}\,\mathrm{d}F_W(w; \boldsymbol{\nu}_k). \end{aligned}$$

This yields

$$\begin{aligned}\log(\delta_{ki}) &= \log\left(\int_0^\infty \frac{\phi(\boldsymbol{X}_i;\boldsymbol{\mu}_k, w\Sigma_k)}{w}\,\mathrm{d}F_W(w;\boldsymbol{\nu}_k)\right) - \log f_{\boldsymbol{X}}(\boldsymbol{X}_i;\boldsymbol{\mu}_k,\Sigma_k,\boldsymbol{\nu}_k)\\ &= \log\left(\int_0^1 \frac{1}{\sqrt{(2\pi)^d F_W^{\leftarrow}(u;\boldsymbol{\nu}_k)^{d+2}|\Sigma_k|}}\exp\left(-\frac{D^2(\boldsymbol{X}_i;\boldsymbol{\mu}_k,\Sigma_k)}{2F_W^{\leftarrow}(u;\boldsymbol{\nu}_k)}\right)\mathrm{d}u\right)\\ &\quad - \log\left(\int_0^1 \frac{1}{\sqrt{(2\pi)^d F_W^{\leftarrow}(u;\boldsymbol{\nu}_k)^{d}|\Sigma_k|}}\exp\left(-\frac{D^2(\boldsymbol{X}_i;\boldsymbol{\mu}_k,\Sigma_k)}{2F_W^{\leftarrow}(u;\boldsymbol{\nu}_k)}\right)\mathrm{d}u\right). \qquad (3.17)\end{aligned}$$

Estimation of the latter integral (corresponding to $\log f_{\boldsymbol{X}}(\boldsymbol{x})$) was discussed in Algorithm 3.3.2; the former integral differs from the latter only by a factor of $F_W^{\leftarrow}(u)^{-1}$, and can be estimated similarly; see Remark 3.3.3 for details.

Summarizing, the k'th iteration of the algorithm consists of approximating the weights δ_{ki}, $i = 1,\dots,n$ with $\boldsymbol{\nu} = \boldsymbol{\nu}_k$ held fixed (which are then used to update $\boldsymbol{\mu}$ and Σ as in (3.16)) and then updating $\boldsymbol{\nu}$ by maximizing the original likelihood $\log L^{\mathrm{org}}(\boldsymbol{\theta};\boldsymbol{X}_1,\dots,\boldsymbol{X}_n)$ as a function of $\boldsymbol{\nu}$ with $\boldsymbol{\mu}$ and Σ set to their current estimates, i.e., we set

$$\boldsymbol{\nu}_{k+1} = \operatorname*{argmax}_{\boldsymbol{\nu}} \log L^{\mathrm{org}}(\boldsymbol{\nu},\boldsymbol{\mu}_{k+1},\Sigma_{k+1};\boldsymbol{X}_1,\dots,\boldsymbol{X}_n) \qquad (3.18)$$

and solve this $p_{\boldsymbol{\nu}}$-dimensional optimization problem numerically. This optimization problem is the same optimization problem one would solve if $\boldsymbol{\mu}$ and Σ were known (and given by $\boldsymbol{\mu}_{k+1}$ and Σ_{k+1}) and is a classical ingredient in ECME algorithms; for more details on rates of convergence of the proposed ECME scheme, see [79, Section 4].

Solving the optimization problem in (3.18) is the most costly part of our algorithm, as it involves estimating the likelihood using Algorithm 3.3.2 multiple times: Each call to the likelihood function requires the approximation of n integrals, each of which results in some random integration error. This results in an estimated log-likelihood function which is itself random and can be "bumpy", so having multiple local maxima. As such, typically fast derivative-based methods can (but do not necessarily) fail to detect a global optimum. This is why in our implementation, we use the R optimizer `optim()` which by default only relies on function evaluations and works for non-differentiable functions. Note that the dimension $p_{\boldsymbol{\nu}}$ of $\boldsymbol{\nu}$ is typically small so that this optimization problem is also numerically feasible.

It turns out that estimating the weights δ_{ki} is faster than solving (3.18) so that it seems to be fruitful to first update $\boldsymbol{\mu}$ and Σ until convergence (with $\boldsymbol{\nu} = \boldsymbol{\nu}_k$ held fixed) and then update $\boldsymbol{\nu}$. In fact, this can be done efficiently: The weights δ_{ki} depend on $\boldsymbol{X}_i$,

$\boldsymbol{\mu}_k$ and Σ_k only through the Mahalanobis distances $D^2(\boldsymbol{X}_i; \boldsymbol{\mu}_k, \Sigma_k)$. Once $\boldsymbol{\mu}_k$ and Σ_k are updated to, say, $\boldsymbol{\mu}_k'$ and Σ_k', (some of) the new weights δ_{ki}' for the new Mahalanobis distances $D^2(\boldsymbol{X}_i; \boldsymbol{\mu}_k', \Sigma_k')$ can be obtained by interpolating the already calculated weights δ_{ki} corresponding to the (old) Mahalanobis distances $D^2(\boldsymbol{X}_i; \boldsymbol{\mu}_k, \Sigma_k)$.

It remains to discuss how a starting value $\boldsymbol{\theta}_0$ can be found. We suggest using $\boldsymbol{\mu}_0 = \bar{\boldsymbol{X}}_n$, the sample mean vector, as an unbiased estimator for $\boldsymbol{\mu}$. Denote by S_n the sample covariance matrix (Wishart matrix) of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$. Since S_n is unbiased for $\mathrm{Cov}(\boldsymbol{X})$ it follows that $\mathbb{E}(S_n) = \mathbb{E}(W)\Sigma$. The idea is now to maximize the likelihood given $\boldsymbol{\mu} = \boldsymbol{\mu}_0$ and given $\Sigma = c \cdot S_n$ with respect to $\boldsymbol{\nu}$ and c (restricted to $c > 0$) which is a $(p_{\boldsymbol{\nu}} + 1)$ dimensional optimization problem. That is, we find

$$(\boldsymbol{\nu}^*, c^*) = \operatorname*{argmax}_{\boldsymbol{\nu}, c>0} L^{\mathrm{org}}(\boldsymbol{\nu}, \boldsymbol{\mu}_0, cS_n; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n) \tag{3.19}$$

numerically (again via R's `optim()`) and set $\boldsymbol{\nu}_0 = \boldsymbol{\nu}^*$ and $\Sigma_0 = c^* S_n$ which is just a multiple of the Wishart matrix. As this step is merely needed to obtain a starting value for $\boldsymbol{\nu}$ and Σ, this optimization can be done over a subset of the sample $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$ to save run time.

The complete procedure is summarized in Algorithm 3.4.4. As convergence criterion we suggest stopping once the maximal relative difference in parameter estimates is smaller than a given threshold. We define the maximal relative difference by

$$d(\boldsymbol{\nu}_k, \boldsymbol{\nu}_{k+1}) = \max_{i=1,\ldots,p_{\boldsymbol{\nu}}} \frac{|\boldsymbol{\nu}_{k,i} - \boldsymbol{\nu}_{k+1,i}|}{|\boldsymbol{\nu}_{k,i}|}, \quad \boldsymbol{\nu}_k = (\boldsymbol{\nu}_{k,1}, \ldots, \boldsymbol{\nu}_{k,p_{\boldsymbol{\nu}}}),$$

and similarly for $\boldsymbol{\mu}$ and Σ.

Algorithm 3.4.4 (ECME algorithm for fitting normal variance mixtures)
Given iid input data $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ and convergence criteria $\varepsilon_{\boldsymbol{\mu}}$, ε_Σ and $\varepsilon_{\boldsymbol{\nu}}$, estimate $\boldsymbol{\mu}, \Sigma, \boldsymbol{\nu}$ via:

1. *Starting value.*
   Set $\boldsymbol{\mu}_0 = \bar{\boldsymbol{X}}_n$ and solve the optimization problem (3.19) numerically to obtain $\boldsymbol{\nu}^*$ and c^*. Set $\boldsymbol{\nu}_0 = \boldsymbol{\nu}^*$ and $\Sigma_0 = c^* S_n$.

2. *ECME iteration.*
   For $k = 0, 1, \ldots,$ do:

   (a) *Update $\boldsymbol{\mu}$ and Σ.*
   Set $\boldsymbol{\mu}_k^{(1)} = \boldsymbol{\mu}_k$ and $\Sigma_k^{(1)} = \Sigma_k$.
   For $l = 1, \ldots,$ do:

i. Estimate new weights $\delta_{ki}^{(l+1)} = \mathbb{E}(1/W_i \mid \boldsymbol{X}_i; \boldsymbol{\mu}_k^{(l)}, \Sigma_k^{(l)}, \boldsymbol{\nu}_k)$, $i = 1, \ldots, n$ using (3.17) and Algorithm 3.3.2.

ii. Calculate the new iterates $\boldsymbol{\mu}_k^{(l+1)}$ and $\Sigma_k^{(l+1)}$ using (3.16) with weights $\delta_{ki}^{(l+1)}$, $i = 1, \ldots, n$.

iii. If $d(\boldsymbol{\mu}_k^{(l)}, \boldsymbol{\mu}_k^{(l+1)}) < \varepsilon_{\boldsymbol{\mu}}$ and $d(\Sigma_k^{(l)}, \Sigma_k^{(l+1)}) < \varepsilon_{\Sigma}$, set $\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k^{(l+1)}$, $\Sigma_{k+1} = \Sigma_k^{(l+1)}$ and go to Step 2b.

(b) *Update* $\boldsymbol{\nu}$.
Numerically solve the optimization problem (3.18) to obtain $\boldsymbol{\nu}_{k+1}$.

(c) If $d(\boldsymbol{\nu}_k, \boldsymbol{\nu}_{k+1}) < \varepsilon_{\boldsymbol{\nu}}$, return the MLEs $\boldsymbol{\mu}^* = \boldsymbol{\mu}_{k+1}$, $\Sigma^* = \Sigma_{k+1}$ and $\boldsymbol{\nu}^* = \boldsymbol{\nu}_{k+1}$.

Algorithm 3.4.4 is implemented in the function `fitnvmix()` of our R package `nvmix`. The mixing variable is specified by providing a function to the argument `qmix`. In the special case where W follows an inverse-gamma or Pareto distribution, the density function is known in closed form which is used by `fitnvmix()` when called with argument `qmix = "inverse.gamma"` or `qmix = "pareto"`.

Remark 3.4.5
We remark that by our assumption on the set $\mathcal{V}$, Algorithm 3.4.4 is only applicable to normal variance mixtures with bounded density. This is a limitation, since some widely used normal mixture models do have unbounded density at the location.

The "unbounded likelihood problem" has been addressed, for instance, in [81]. To overcome the problem of an unbounded likelihood function, one can use the *correct likelihood*, defined by

$$L^{\text{corr}}(\boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n) = \prod_{i=1}^{n} \int_{\boldsymbol{X}_i - \Delta_i}^{\boldsymbol{X}_i + \Delta_i} f_{\boldsymbol{X}}(\boldsymbol{x}; \boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma) \, \mathrm{d}\boldsymbol{x}, \tag{3.20}$$

where the (componentwise positive) $\Delta_i \in \mathbb{R}^d$ represent the roundoff error when measuring $\boldsymbol{X}_i$. Being a product of probabilities, L^{corr} is bounded by 1. The integrals in (3.20) can be estimated using the methods from Section 3.2. However, moving from the original likelihood to the correct likelihood means that for each call to the likelihood function, n d-dimensional integrals instead of n 1-dimensional integrals need to be approximated. Thus, if n and d are reasonably large, this approach might be computationally too expensive.

Numerical issues can arise even when the likelihood function is bounded. Indeed, in Step 2(a)i, weights $\delta_{ki} = \mathbb{E}(1/W_i \mid \boldsymbol{X}_i, \boldsymbol{\theta}_k)$ are estimated and these can be quite large and suffer from numerical estimation problems when $\boldsymbol{X}_i$ is close to $\boldsymbol{\mu}$.

In our implementation of Algorithm 3.4.4 in the R function `fitnvmix()`, we avoid evaluating the density function at a potential singularity by "capping" the Mahalanobis distance, that is, by replacing any appearing $D^2(\boldsymbol{X}_i, \boldsymbol{\mu}_k, \Sigma_k)$ which is smaller than Δ by Δ, where $\Delta > 0$ is small (e.g., $\Delta = 10^{-16}$). The same "capping" approach was successfully used in [93] in the context of estimating multivariate skewed variance gamma model parameters. We do not claim that this "capping" works for all normal variance mixtures with bounded or unbounded density, and leave further investigation of this important problem for future research.

3.5 Gamma-mixture models

For statistical purposes it is often interesting to study the distribution of the squared Mahalanobis distance of $\boldsymbol{X} \sim \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$ given by $D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma) = (\boldsymbol{X} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{X} - \boldsymbol{\mu})$. We write $D^2 := D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma)$ if there is no confusion.

It follows readily from the stochastic representation (3.1) of $\boldsymbol{X}$ that, in distribution,

$$D^2 = W\, X^2,$$

where $X^2 \sim \chi^2_d$. This immediately gives rise to a sampling algorithm to generate random variates from D^2. Since a χ^2 distribution is a special case of a gamma distribution, it follows that $D^2 \mid W \sim \Gamma(d/2, 2W)$ where $\Gamma(\alpha, \beta)$ denotes a gamma distribution with shape $\alpha > 0$ and scale $\beta > 0$ which admits the density $f_{\Gamma(\alpha,\beta)}(x) = (\beta^\alpha \Gamma(\alpha))^{-1} x^{\alpha-1} e^{-x/\beta}$, $x > 0$, and distribution function $F_\Gamma(x; \alpha, \beta) = \int_0^x f_{\Gamma(\alpha,\beta)}(t)\,\mathrm{d}t$ for $x > 0$. The function $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\,\mathrm{d}t$, $z > 0$ denotes the gamma function.

In the special case where $W = 1$ almost surely, $D^2 \sim \chi^2_d$; if W follows an inverse-gamma distribution so that $\boldsymbol{X}$ follows a multivariate t with $\nu > 0$ degrees of freedom, it can be easily seen that $D^2/d \sim \mathrm{F}(d, \nu)$. For the general case where only $F^{\leftarrow}_W$ is available, we can use methods similar to the ones developed so far to approximate the density and the distribution function of D^2.

3.5.1 Distribution, density and quantile function of D^2

Using a conditioning argument similar to the normal variance mixture case, we obtain that

$$F_{D^2}(x) = \mathbb{P}(D^2 \le x) = \mathbb{E}\left(F_{\Gamma(d/2,2)}\left(\frac{x}{W}\right)\right), \quad x \ge 0.$$

This univariate integral can be approximated directly using an RQMC approach similar to Algorithm 3.2.2. An implementation can be found in the function `pgammamix()` in the R package `nvmix`.

In a similar fashion as in the derivation of Equation (3.3), the density of D^2 can be calculated as $f_{D^2}(x) = \int_0^1 \tilde{h}(u)\,\mathrm{d}u$ for $x > 0$, where

$$\tilde{h}(u) = \frac{1}{\Gamma(d/2)(2F_W^{\leftarrow}(u))^{d/2}} x^{d/2-1} \exp\left(-\frac{x}{2F_W^{\leftarrow}(u)}\right), \quad u \in (0,1).$$

The functions $\tilde{h}$ and h from Equation (3.11) differ only in constants with respect to u, the functional form is identical. Algorithm 3.3.2 can then, with some slight modifications, be used to estimate the density $f_{D^2}(x)$ (or $\log f_{D^2}(x)$); see also Remark 3.3.3. This is implemented in the function `dgammamix()` in the R package `nvmix`.

Many applications, such as graphical goodness-of-fit assessment or random variate generation, rely on the quantile function of D^2. Note that both the density and the distribution function of D^2 can be estimated as discussed above; the quantile function can then be estimated by numerically solving the equation $F_{D^2}(q_u) - u = 0$ for q_u where $u \in (0,1)$ is given. We suggest using Newton's method: In iteration $k \geq 1$, given a current iterate $q_u^{(k)}$, the next iterate is given by

$$\begin{aligned} q_u^{(k+1)} &= q_u^{(k)} - \frac{F_{D^2}(q_u^{(k)}) - u}{f_{D^2}(q_u^{(k)})} \\ &= q_u^{(k)} - \operatorname{sign}(F_{D^2}(q_u^{(k)}) - u_i) \exp\left\{\log\left(|F_{D^2}(q_u^{(k)}) - u_i|\right) - \log f_{D^2}(q_u^{(k)})\right\}. \end{aligned}$$

The second line is a numerically more stable version of the first. We remark that (potentially) many calls to $F_{D^2}(\cdot)$ and $f_{D^2}(\cdot)$ are necessary until convergence takes place. We also note that in most applications, the quantile function has to be evaluated at multiple inputs, say $u_1, \dots, u_n$. In order to reduce run time, one can sort the inputs u_i in increasing order and also store all calls to $F_{D^2}(\cdot)$ and $f_{D^2}(\cdot)$. These values can be used as starting values for the next quantile calculation. If they are reasonably close to the true quantile, the procedure enjoys local quadratic convergence so that only a few calls to $F_{D^2}(\cdot)$ and $f_{D^2}(\cdot)$ are needed. We note that Newton's method can suffer from convergence problems; in our numerical study, however, we have not seen such cases. Furthermore, $F_{D^2}(\cdot)$ and $f_{D^2}(\cdot)$ can be estimated simultaneously using the same realizations of W, and all those realizations can also be stored so that they do not need to be generated more often than necessary. This is implemented in the function `qgammamix()` in the R package `nvmix`; the same idea can be exploited to estimate the quantile function of univariate normal variance mixtures which is implemented in the function `qnvmix()`.

3.5.2 Graphical goodness-of-fit assessment

With an (estimated) quantile function of D^2 at hand, we can produce QQ plots of the observed mahalanobis distances versus their theoretical counterparts; this is implemented in the function `qqplot_maha()` of the R package `nvmix`. Besides approximating the theoretical quantiles, the function also computes asymptotic standard errors as in [33, p. 35-36] and a Bootstrap confidence interval for the empirical quantiles. Including these confidence intervals helps address the problem of large variations in the ordered samples. As QQ plots can merely be used as a first graphical assessment, the function additionally performs statistical GoF tests. In particular, a Kolmogorov–Smirnov GoF test is performed via `ks.test()` on the univariate Mahalanobis distances as well as an Anderson–Darling GoF test in which case `qqplot_maha()` calls `ad.test()` from the R package `ADGofTest`; see [8]. The p-values for testing the null hypothesis of having specified the correct distribution and the values of the test-statistic are displayed on the second y-axis.

As an example, we sample 100 realizations from $\mathrm{PNVM}_5(1.5, \mathbf{0}, \Sigma)$ for a randomly sampled correlation matrix Σ and fit the corresponding mixture via `fitnvmix()`:

```
set.seed(42)
d  <- 4
n  <- 100
nu. <- 1.5
scale <- cov2cor(tcrossprod(matrix(runif(d * d), ncol = d)))
x <- rnvmix(n, qmix = "pareto", alpha = nu., scale = scale)
m.p.b <- c(0.1, 50)
(fit.par1 <- fitnvmix(x, qmix = qmix = function(u, nu) (1-u)^(-1/nu), mix.param.
    bounds = m.p.b))
```

```
Call: fitnvmix(x = x, qmix = qmix., mix.param.bounds = m.p.b)
Input data: 100 4-dimensional observations.
Normal variance mixture specified through quantile function of the mixing
    variable
     function (u, nu)  (1 - u)^(-1/nu)
with unknown 'loc' vector and unknown 'scale' matrix.
Approximated log-likelihood at reported parameter estimates: -410.292900
Termination after 16 iterations, convergence detected.
Estimated mixing parameter(s) 'nu':
[1] 1.412
Estimated 'loc' vector:
[1]  0.03800 -0.11296  0.02966  0.01760
Estimated 'scale' matrix:
```

```
       [,1]   [,2]   [,3]   [,4]
[1,] 0.8930 0.7701 0.7399 0.8916
[2,] 0.7701 0.8360 0.6491 0.7532
[3,] 0.7399 0.6491 0.8283 0.6019
[4,] 0.8916 0.7532 0.6019 1.0194
```

Next, we call `qqplot_maha()` and produce Figure 3.2. Note that the return value of this is an object of class `"qqplot_maha"` for which the methods `plot()` and `print()` are defined. This object contains, among others, the theoretical quantiles computed via `qgammamix()`, i.e., the quantiles of the hypothesized distribution of the Mahalanobis distance.

```
set.seed(1)
qq.par <- qqplot_maha(x, qmix = "pareto", alpha = fit.par1$nu,
   loc = fit.par1$loc, scale = fit.par1$scale, plot = FALSE)
plot(qq.par)
plot(qq.par, plot.pars = list(log = "xy"))
print(qq.par)
```

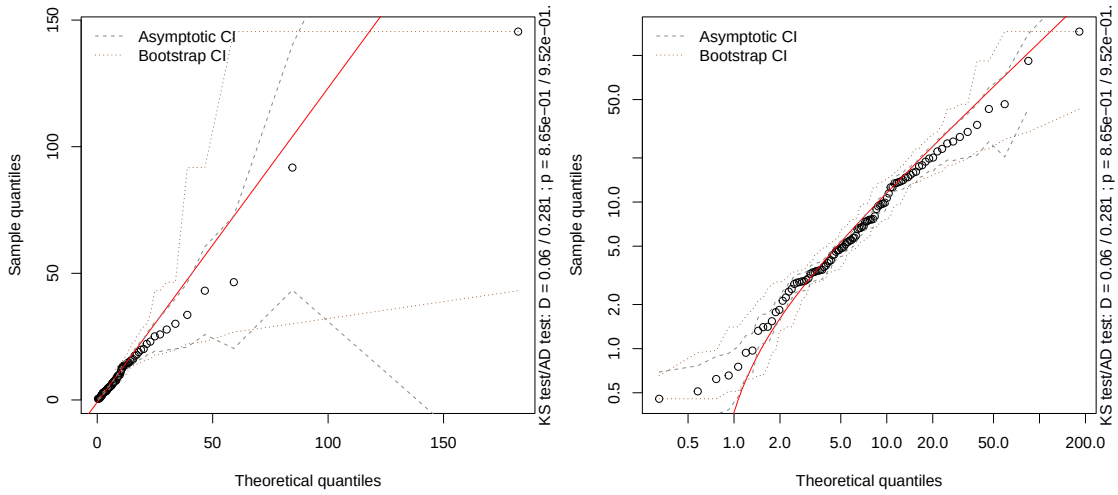

Figure 3.2: Q-Q plot of the empirical quantiles $D^2(\boldsymbol{x}_i; \boldsymbol{\mu}, \Sigma)$ versus their theoretical counterparts on ordinary scale (left) and log-log scale (right).

```
Call: qqplot_maha(x = x, qmix = "pareto", loc = fit.par1$loc, scale = fit.par1$
    scale
,     plot = FALSE, alpha = fit.par1$nu)
Input: 100 squared Mahalanobis distances.
KS test: D = 0.06, p = 8.65e-01
```

```
AD test: D = 0.281, p = 9.52e-01.
The R Package nvmix
Computed results stored in the object:
- theoretical quantiles in $theo_quant;
- sorted, squared Mahalanobis distances in $maha2;
- estimated, asymptotic standard errors in $asymptSE;
- Bootstrap CIs (estimated from 500 resamples) in $boot_CI;
- GoF test results in $testout;
```

3.6 Numerical examples

In this section we provide a careful numerical analysis of all algorithms presented. The first part discusses the type of mixing distributions used; the second, third and fourth part detail numerical examples for estimating the distribution function using Algorithm 3.2.2 with variable reordering as in Algorithm 3.2.1, estimating the log-density function using Algorithm 3.3.2, and estimating parameters ν, $\boldsymbol{\mu}$ and Σ given a random sample using Algorithm 3.4.4, respectively. The last part provides an application of our methods to a multivariate financial data set.

3.6.1 Test distributions

For our numerical examples in Sections 3.6.2 to 3.6.4, we consider two distributions for the mixing variable W, an inverse-gamma distribution (so that $\boldsymbol{X}$ is multivariate t) and a Pareto distribution.

Inverse-gamma mixture Here W follows an inverse-gamma distribution with shape and scale parameter $\nu/2$. The resulting distribution is the multivariate t distribution, $\boldsymbol{X} \sim \mathrm{MVT}_d(\nu, \boldsymbol{\mu}, \Sigma)$ with positive degrees of freedom ν; see, for instance, [71, Chapter 1]. Note that if $\nu > 1$, $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ and if $\nu > 2$, $\mathrm{Cov}(\boldsymbol{X}) = \frac{\nu}{\nu-2}\Sigma$. The multivariate t distribution has the density

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{\Gamma((\nu+d)/2)}{\Gamma(\nu/2)\sqrt{(\nu\pi)^d|\Sigma|}} \left(1 + D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)/\nu\right)^{-\frac{\nu+d}{2}}, \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{3.21}$$

For the ECME procedure it is useful to calculate the weight $\mathbb{E}(1/W \mid \boldsymbol{X})$. Since

$$f_{W|\boldsymbol{X}}(w \mid \boldsymbol{x}) \propto f_{\boldsymbol{X}|W}(\boldsymbol{x} \mid w) f_W(w) \propto w^{-\frac{d+\nu}{2}-1} \exp\left(-\frac{(D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) + \nu)/2}{w}\right), \quad w > 0,$$

$W \mid \boldsymbol{X}$ follows an Inverse gamma (IG) distribution, i.e., $W \mid \boldsymbol{X} \sim \mathrm{IG}((d+\nu)/2, (D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma)+\nu)/2)$. This implies

$$\mathbb{E}(1/W \mid \boldsymbol{X}) = \frac{\nu + d}{\nu + D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma)},$$

so that the weights δ_{ki} in Step 2(a)i of Algorithm 3.4.4 can be calculated analytically in this case.

Pareto mixture In order to test our algorithms for a normal variance mixture distribution that has not been studied as extensively as the multivariate t distribution we consider $W \sim \mathrm{Par}(\alpha, x_m)$ with density

$$f_W(w) = \alpha \frac{x_m^\alpha}{w^{\alpha+1}}, \quad w \geq x_m.$$

One can calculate that $\mathbb{E}(W^k)$ exists with $\mathbb{E}(W^k) = \alpha/(\alpha - k)$ if $k < \alpha$. This implies for the resulting normal variance mixture $\boldsymbol{X} = \boldsymbol{\mu} + \sqrt{W} A \boldsymbol{Z}$ that $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ for $\alpha > 1/2$ and $\mathrm{Cov}(\boldsymbol{X}) = \frac{\alpha}{\alpha-1}\Sigma$ for $\alpha > 1$. The density $f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{X}}(\boldsymbol{x}; \mu, \Sigma, \alpha, x_m)$ can be determined using (3.4):

$$\begin{aligned}
f_{\boldsymbol{X}}(\boldsymbol{x}) &= \frac{\alpha x_m^\alpha}{\sqrt{(2\pi)^d |\Sigma|}} \int_{x_m}^\infty w^{-d/2-\alpha-1} \exp\left(-\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2w}\right) \mathrm{d}w \\
&= \frac{\alpha x_m^\alpha}{\sqrt{(2\pi)^d |\Sigma|}} \left(\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2}\right)^{-d/2-\alpha} \int_0^{\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2x_m}} u^{d/2+\alpha-1} \exp(-u) \, \mathrm{d}u \\
&= \frac{\alpha x_m^\alpha}{\sqrt{(2\pi)^d |\Sigma|}} \left(\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2}\right)^{-d/2-\alpha} \gamma\left(\alpha + \frac{d}{2}; \frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2x_m}\right), \quad \boldsymbol{x} \in \mathbb{R}^d,
\end{aligned}$$

where $\gamma(z; x) = \int_0^x t^{z-1} e^{-t} \, \mathrm{d}t$ for $z, x > 0$ denotes the (lower) incomplete gamma function. Note that $f_{\boldsymbol{X}}(\boldsymbol{x}; \mu, \Sigma, \alpha, x_m) = f_{\boldsymbol{X}}(\boldsymbol{x}; \mu, x_m\Sigma, \alpha, 1)$ so that the scale parameter x_m is redundant as the scaling can be achieved via scaling Σ. We can thus set $x_m = 1$ and obtain

$$f_{\boldsymbol{X}}(\boldsymbol{x}; \mu, \Sigma, \alpha) = \frac{\alpha}{\sqrt{(2\pi)^d |\Sigma|}} \left(\frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2}\right)^{-d/2-\alpha} \gamma\left(\alpha + \frac{d}{2}; \frac{D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)}{2}\right), \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{3.22}$$

We use the notation $\boldsymbol{X} \sim \text{PNVM}(\alpha, \boldsymbol{\mu}, \Sigma)$ ("Pareto normal variance mixture") for a random vector $\boldsymbol{X}$ with density (3.22).

As in the case of an inverse-gamma mixture, it is possible to derive an expression for $\mathbb{E}(1/W \mid \boldsymbol{X})$ in the Pareto setting. Note that

$$f_{W|\boldsymbol{X}}(w \mid \boldsymbol{x}) \propto f_{\boldsymbol{X}|W}(\boldsymbol{x} \mid w) f_W(w) \propto w^{-(\alpha+d/2+1)} \exp\left(-D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)/(2w)\right), \quad w > 1,$$

so that using the density transformation formula we obtain for $\tilde{W} = 1/W$ that

$$f_{\tilde{W}|\boldsymbol{X}}(\tilde{w} \mid \boldsymbol{x}) \propto \tilde{w}^{\alpha+d/2-1} \exp(-\tilde{w} D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)/2), \quad \tilde{w} \in (0, 1).$$

Therefore, $W^{-1} \mid \boldsymbol{X}$ follows a $(0, 1)$ truncated gamma distribution with shape $\alpha + d/2$ and scale $2/D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma)$. For more details on truncated gamma distributions, see [15]; Equation (2.12) therein implies that

$$\mathbb{E}(1/W \mid \boldsymbol{X}) = \frac{F_\Gamma(1; \alpha + d/2 + 1, 2/D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma))}{F_\Gamma(1; \alpha + d/2, 2/D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma))} \frac{2\alpha + d}{D^2(\boldsymbol{X}; \boldsymbol{\mu}, \Sigma)}.$$

3.6.2 Estimating the distribution function

In the case where $\boldsymbol{X}$ follows a Multivariate t (MVT) distribution, denoted $\boldsymbol{X} \sim \text{MVT}_d(\nu, \boldsymbol{\mu}, \Sigma)$, Algorithm 3.2.2 combined with the variable reordering Algorithm 3.2.1 can be used to estimate $F(\boldsymbol{a}, \boldsymbol{b})$, and is implemented in the function `pStudent()` in the R package `nvmix`. In this case, one can also use the QRSVN algorithm from [36], which is implemented in the function `pmvt()` of the R package `mvtnorm` ([38]). The differences between these two algorithms was explained in Section 2.1. Furthermore, our implementation relies on C code, whereas `pmvt()` internally calls Fortran code.

We also note that the most natural competitor to our estimation procedure is the crude estimator $(1/n) \sum_{i=1}^n \mathbb{1}_{\{\boldsymbol{a} \le \boldsymbol{X}_i \le \boldsymbol{b}\}}$ where $\boldsymbol{X}_i \overset{\text{ind.}}{\sim} \text{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$. It was shown in [36] that this estimator has a larger variance than the one based on the integrand g derived earlier.

Error behaviour as a function of the sample size

In order to assess the performance of our algorithm let us first consider estimated absolute errors as a function of the number of function evaluations. Four settings are considered: (pure) MC with and without reordering and RQMC (using a randomized Sobol' sequence) with and without reordering. In Figures 3.3 and 3.4, estimated absolute errors (estimated

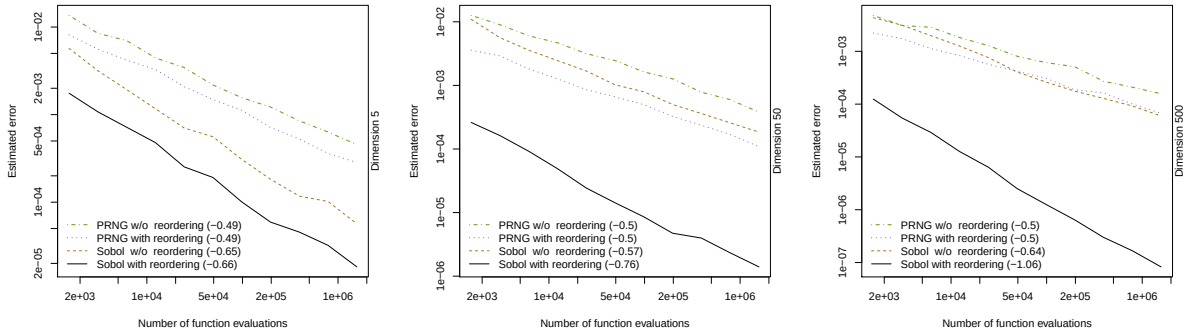

Figure 3.3: Average absolute errors of different estimators for $F_{\boldsymbol{X}}(\boldsymbol{x})$ as a function of n for $\boldsymbol{X} \sim \mathrm{MVT}_d(2, \mathbf{0}, \Sigma)$, where for each n, 15 different settings for Σ and $\boldsymbol{x}$ are randomly chosen. Regression coefficients are in parentheses in the legends.

as in Algorithm 3.2.2 via $\hat{\varepsilon}$ in Step 4.3)) are reported for different sample sizes n (which refer to the total number of function evaluations) in different dimensions using the four aforementioned methods for the multivariate t case and the Pareto mixture. For each dimension and for each n we report the average estimated absolute error for 15 different parameter settings. In each parameter setting, an upper limit is randomly chosen via $\boldsymbol{b} \sim \mathrm{U}(0, 3\sqrt{d})^d$ and a correlation matrix R is sampled as a standardized Wishart matrix via the function `rWishart()` in R. The lower limit is set to $\boldsymbol{a} = (-\infty, \dots, -\infty)$. The degrees of freedom ν in the MVT setting and the shape parameter α in the PNVM setting are set to 2.

It is evident that RQMC methods yield lower errors than their MC counterparts. We also report the convergence speed (as measured by the regression coefficient α of $\log \hat{\varepsilon} = \alpha \log n + c$ displayed in the legend): Variable reordering does not have an influence on the convergence speed $1/\sqrt{n}$ of MC methods; however, it does speed up the RQMC methods. A possible explanation is that variable reordering can reduce the effective dimension. This is discussed below in more detail.

The effect of variable reordering

Investigating the variance of the integrand It is interesting to further investigate the effect of variable reordering as detailed in Section 3.2.2. To this end, the variance of

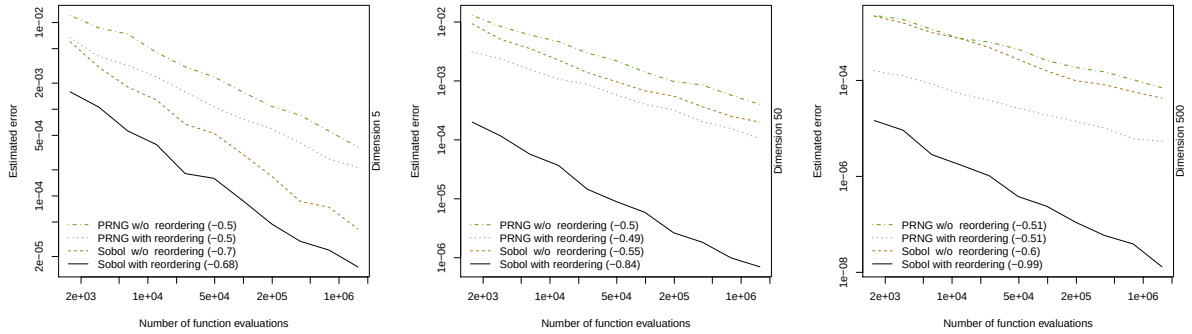

Figure 3.4: Average absolute errors of different estimators for $F_{\boldsymbol{X}}(\boldsymbol{x})$ as a function of n for $\boldsymbol{X} \sim \mathrm{PNVM}_d(2, \mathbf{0}, \Sigma)$, where for each n, 15 different settings for Σ and $\boldsymbol{x}$ are randomly chosen. Regression coefficients are in parentheses in the legends.

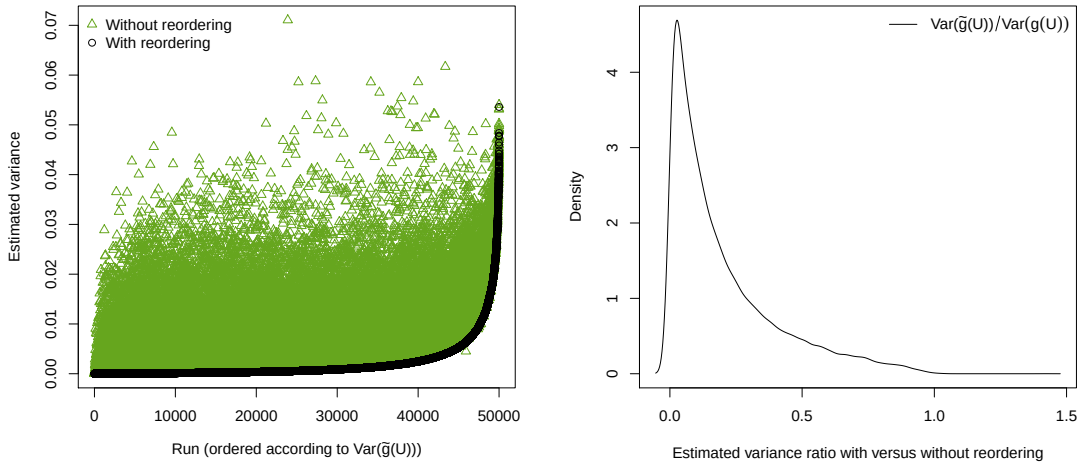

Figure 3.5: Left: Variance of the integrand $\mathrm{Var}(g(\boldsymbol{U}))$ with and without variable reordering. Right: Density plot of estimated variance ratios.

the integrand g from (3.9) given by

$$\mathrm{Var}(g(\boldsymbol{U})) = \int_{[0,1]^d} g^2(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} - \left(\int_{[0,1]^d} g(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} \right)^2$$

is estimated, once with the original g without reordering, and once with $\tilde{g}$ which is the integrand g after applying Algorithm 3.2.1 to the inputs $\boldsymbol{a}, \boldsymbol{b}, \Sigma$. We use a randomized experiment and do the following 50 000 times for an inverse-gamma mixture: Sample $d \sim \mathrm{U}(\{5, \dots, 500\})$, $\nu \sim \mathrm{U}(0.1, 5)$ and $\boldsymbol{a}$, $\boldsymbol{b}$, Σ are randomly chosen as in the previous section. The variance of the integrand is then estimated via the sample variance of $g(\boldsymbol{U}_1), \dots, g(\boldsymbol{U}_N)$ for $N = 10\,000$. Results can be found in Figure 3.5: The left plot displays estimated variances of the integrand with and without variable reordering in each run. To make the effect of the preconditioning step more visible, the runs have been permuted so that the variance of the integrand when reordering was employed is increasing with the index of the run. On the right, a density plot of the non-negative ratios $\mathrm{Var}(\tilde{g}(\boldsymbol{U}))/\mathrm{Var}(g(\boldsymbol{U}))$ is shown; it can be seen that the re-ordering gives typically a substantial variance reduction. It can be confirmed that in the vast majority of cases, variable reordering substantially decreases the variance of the integrand. In only 12 of the 50 000 runs did the estimated variance after reordering exceed the variance without reordering.

Effective dimension of the integrand As was seen in Figures 3.3 and 3.4, reordering improves both MC and RQMC methods; the effect is however stronger for RQMC methods. A possible explanation for this is that the variable reordering not only reduces the overall variance of the integrand, $\sigma^2 = \mathrm{Var}(g(\boldsymbol{U}))$, as seen in the previous part, but also the *effective dimension* of the integrand; see Chapter 2. (R)QMC methods often work better if only a small number of variables are important, see [118] and references therein for a discussion and examples. Variable reordering, as explained in Section 3.2.2, was derived in a way such that the first components are the most important ones.

The first order indices $S_{\{l\}}$ and total effect indices S_{T_l} for $l \in \{1, \dots, d\}$ can be estimated using the method of [96] which is implemented in the function `sobolowen()` in the R package `sensitivity`; see [103]. Figure 3.6 shows estimated Sobol' indices in two settings: In each setting, $W \sim \mathrm{IG}(1/2, 1/2)$ (so that $\boldsymbol{X}$ follows a multivariate t distribution with 1 degree of freedom) and $d = 10$. The upper limit $\boldsymbol{b}$ and the scale matrix Σ were found by trial & error so that there is either a substantial variance reduction (top figure) achieved by reordering or an increase in variance (bottom figure). In order to be consistent with the definition of the integrand g in (3.9), variables are called $0, \dots, d-1$ so that they correspond to $u_0, \dots, u_{d-1}$. For instance, in the top figure, one can read that $S_{(0)} \approx 0.52$ after reordering so that 52% of the variance of g can be explained by a function $g_{\{0\}}(u_0)$.

Inspecting the top figures where variable reordering led to a decrease in variance of approximately 99% reveals that both first order and total effect indices are decreasing in the dimension after variable reordering was performed. Also, the figure label includes the

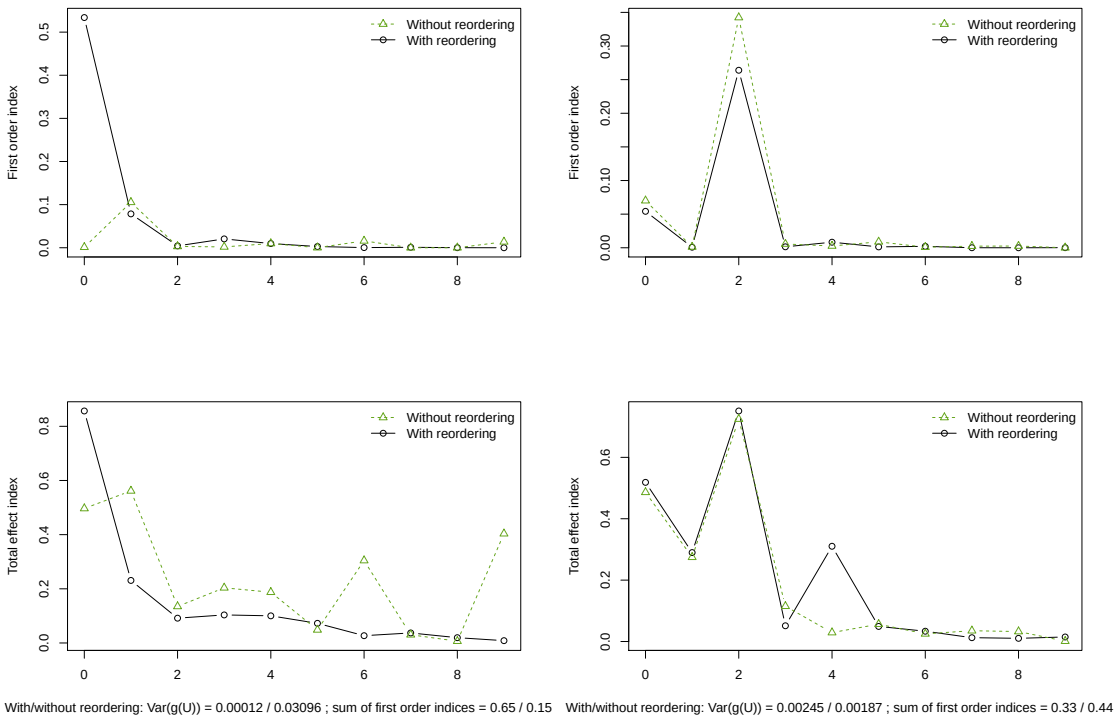

Figure 3.6: Estimated first order and total effect indices with and without reordering for an inverse-gamma mixture in a setting with high variance reduction (top) and increase in variance (bottom).

sum of the first order indices. After reordering, 65% (as opposed to 15%) of the overall variance of the integrand is explained by components g_I of g of exactly one variable, hinting at the fact that the effective dimension decreased: The effective dimension in the superposition sense in proportion 65% decreased to 1 after reordering.

There are rare cases when variable reordering leads to an increase in variance: In the bottom figures, the relative increase is about 31%. Here, the new ordering is clearly not optimal and indices are not decreasing with the dimension. Given the nature of the greedy procedure it is expected that in some cases, no improvement is achieved.

Run times

In this part we take a brief look at the run-times of Algorithm 3.2.2 combined with the variable reordering Algorithm 3.2.1. We restrict our attention to the important multivariate t case and compare run times of our implementation in `pStudent()` with the run times of the above mentioned QRSVN algorithm described in [36] and provided by the function `pmvt()` in the R package `mvtnorm`.

In order to get meaningful estimates of the CPU time, for each dimension d, the following is done 15 times: Sample $\boldsymbol{b}$ and Σ as before when estimating $\mathrm{Var}(g(\boldsymbol{U}))$, set $\boldsymbol{a} = (-\infty, \dots, -\infty)$ and $\nu = 2$. Then call `pmvt()` and `pStudent()` three times each and average their CPU times obtained using the package `microbenchmark` of [86]. The above procedure is done for an absolute error tolerance $\varepsilon = 0.001$ and the maximum number of function evaluations is chosen such that both algorithms always terminate with the correct precision.

Figure 3.7 shows the run times obtained. The symbols represent the corresponding means whereas the lines show the largest/smallest CPU time measured for that dimension. Note that `pmvt()` only works for dimensions up to 1 000. Figure 3.7 shows that our implementation significantly outperforms the existing standard which takes up to 8 times more run time.

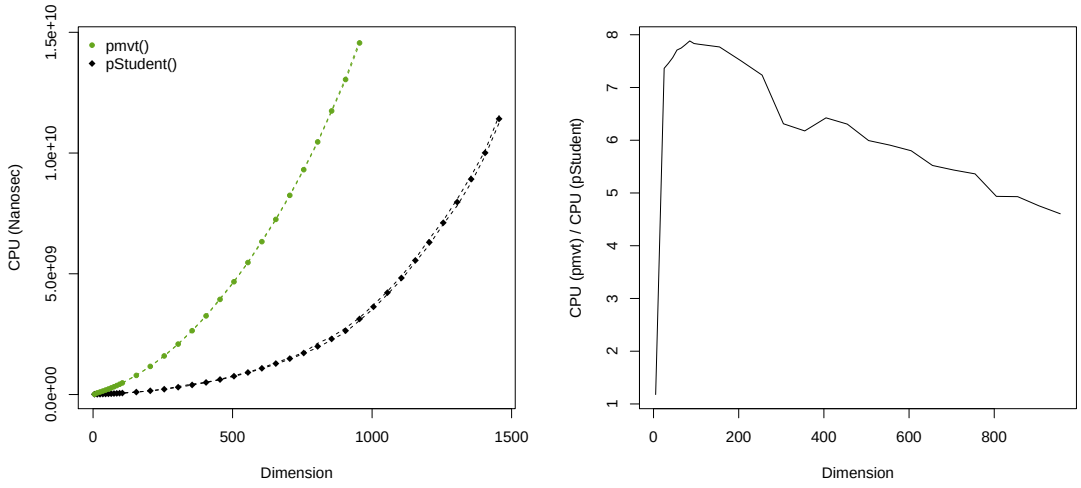

Figure 3.7: Run times based on three replications of 15 randomly chosen inputs $\boldsymbol{b}$ and Σ in each dimension (left); run-time ratios relative to `pStudent()` (right).

3.6.3 Estimating the density function

In this section we test the performance of Algorithm 3.3.2 to estimate the log-density of $\boldsymbol{X} \sim \mathrm{MVT}_d(\nu, \boldsymbol{\mu}, \Sigma)$ and $\boldsymbol{X} \sim \mathrm{PNVM}_d(\alpha, \boldsymbol{\mu}, \Sigma)$. Note that the density is known in either case and given in (3.21) and (3.22) so that estimated and true log-density values can be compared.

We sample $n = 1\,000$ points from $\boldsymbol{X} \sim \mathrm{MVT}_d(\nu = 1, \mathbf{0}, I_d)$ in dimension $d = 10$ and evaluate the density of $\mathrm{MVT}_d(\nu = 4, \mathbf{0}, I_d)$ at the sampled points. The Pareto case is done similarly. Figure 3.8 displays results obtained by the adaptive algorithm (Algorithm 3.3.2) and by the crude (non-adaptive) Algorithm 2.1.2; the true log-density and the probability $\mathbb{P}(D^2(\boldsymbol{X}, \mathbf{0}, I_d) > m^2)$ are also plotted. The latter probability gives an idea of how likely it is to see a sample point $\boldsymbol{x}$ with Mahalanobis distance greater than m. For small Mahalanobis distances, both algorithms perform well. For larger ones the problem becomes harder as the underlying integrand becomes more difficult to integrate (recall Figure 3.1 and the discussion thereafter) and the crude, non-adaptive version gives highly biased results. The adaptive version, however, is able to accurately estimate the log-density for any Mahalanobis distance and is furthermore much faster (it takes only approximately 1 second for a total of $n = 1\,000$ log-density estimations).

By inspecting the axes in Figure 3.8, one can see that our procedure performs well even for very large Mahalanobis distances that would rarely be observed. For likelihood-based methods, such as Algorithm 3.4.4, it is, however, crucial to be able to evaluate the density function for a wide range of inputs. For instance, consider the problem where a sample $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n \overset{\text{ind.}}{\sim} \mathrm{MVT}_d(\nu, \mathbf{0}, I_d)$ for unknown ν is given. It is then necessary to evaluate the log-density of $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n$ at a range of values of ν in order to find the Maximum likelihood estimator (MLE). In fact, this was the motivation for performing the experiments undertaken to produce Figure 3.8: The sample is coming from a heavy-tailed multivariate t distribution and the log-density function of a less heavy tailed multivariate t distribution is evaluated at that sample. The same intuition lies behind the experiment to produce the plot on the right of Figure 3.8.

3.6.4 Fitting normal variance mixture distributions

In this section we provide examples for our fitting procedure Algorithm 3.4.4. While in the special case where W follows an inverse-gamma distribution (i.e., $\boldsymbol{X} \sim \mathrm{MVT}_d(\nu, \boldsymbol{\mu}, \Sigma)$ for which the joint density function is available in closed form), ECME methods described in [80] and [88] can be applied directly (implemented, for instance, in the function `fit.mst()`

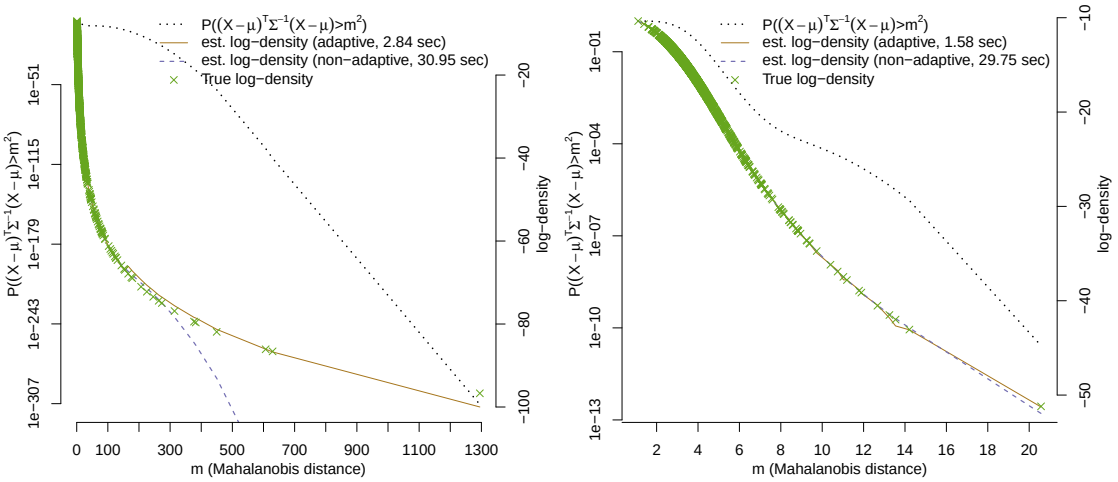

Figure 3.8: Estimated log-density of $\mathrm{MVT}_d(\nu = 4, \mathbf{0}, I_d)$ (left) and $\mathrm{PNVM}_d(\alpha = 6, \mathbf{0}, I_d)$ (right) in $d = 10$ evaluated at $n = 1\,000$ points sampled from $\mathrm{MVT}_d(\nu = 1, \mathbf{0}, I_d)$ (left) and $\mathrm{PNVM}(\alpha = 2, \mathbf{0}, I_d)$ (right).

in the R package `QRM`; see [98]), this is not the case for a general normal variance mixture distribution where the density function may not be available in closed form. In the latter case, we do rely on Algorithm 3.4.4 in combination with our adaptive procedure described in Algorithm 3.3.2 to estimate the log-density function. This is all done automatically in the function `fitnvmix()` which merely needs a specification of the mixing distribution in the form of its quantile function.

As in the previous sections, we consider an inverse-gamma and a Pareto mixture as test cases. We chose these two distributions where the density function is known in closed form so that we are able to investigate if optimizing the log-likelihood estimated via Algorithm 3.3.2 (as opposed to using a closed formula for the log-likelihood) has a significant effect on parameter estimates. In a practical setting where the density function is not known in closed form (as is the case for the inverse-Burr mixture considered in the data analysis done in Section 3.6.5) such comparison is not possible.

Our algorithm is tested in dimensions $d \in \{10, 50\}$ for sample sizes n between 250 and 5 000. In each setting, n random vectors $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n \overset{\text{ind.}}{\sim} \mathrm{MVT}_d(\nu = 2.5, \mathbf{0}, \Sigma)$ are sampled and then Algorithm 3.4.4 is used to estimate the parameters. We randomly choose Σ as DRD where R is a random Wishart matrix and D is diagonal with entries $D_{ii} \overset{\text{ind.}}{\sim} \mathrm{U}(2, 5)$

for $i = 1, \dots, d$. Results are displayed in Figure 3.9 where the estimate $\hat{\nu}$ of ν is plotted as a function of the number of ECME iterations (see Step 2 of Algorithm 3.4.4). The optimizations in Steps 1 and 2b of Algorithm 3.4.4 are based on the estimated log-likelihood function via Algorithm 3.3.2.

As mentioned earlier, an ECME procedure for estimating parameters of a multivariate t distribution is available in the function `fit.mst()`. The symbols at the end of the curves in Figure 3.9 denote estimates obtained from this function. It can be confirmed that not only does our procedure converge to the correct maximum likelihood estimate in the given examples, but also that run times are reasonably small for this challenging problem. Note that only few iterations are needed until convergence is detected.

A similar experiment is performed for the Pareto-mixture case, see Figure 3.10. Here, the symbols at the end of each line display results obtained from Algorithm 3.4.4 using analytical weights and densities, obtained by calling our function `fitnvmix()` with `qmix = "pareto"`.

The run times displayed in the legends of Figures 3.9 and 3.10 may seem counter-intuitive; however, several factors influence run time: The larger the sample size n, the more integrals need to be approximated and the higher the probability of observing extreme Mahalanobis distances. Furthermore, the problem of estimating the log-density and the weights becomes harder the larger the Mahalanobis distance of the input. However, larger sample sizes can also lead to a quicker convergence of the weights in Step 2a of Algorithm 3.4.4 and also to faster convergence of the estimates of the mixing variable in Step 2b of Algorithm 3.4.4. Overall, as there are numerical approximations involved at many levels, it will depend on the sample at hand how long the algorithm takes. This explains why run times are not monotone in the sample size n.

3.6.5 Application to financial data

This section demonstrates an application of many of our methods and software to a real financial data set. We consider daily return data from the 15 real estate investment trusts (REITs) which are constituents of the SP500 index between 2010 and 2012 ($n = 753$ data points in $d = 15$). The dataset `SP500` is obtained from the R package `qrmdata`, see [60]. We first fit marginal ARMA$(1,1)$ − GARCH$(1,1)$ models and then fit normal variance mixture models to the standardized residuals ("innovations"):

```
library("qrmdata") # for the data-set
library("qrmtools") # for returns()
library("rugarch") # for fit.ARMA.GARCH()
```

```
set.seed(123) # reproducibility
data("SP500_const") # load negative return data of the REITs in the SP500 index
time <- c("2010-01-01", "2012-12-31")
x <- SP500_const[paste0(time, collapse = "/"),
                 SP500_const_info$Subsector == "REITs"]
X <- -returns(x)
## deGARCHing
uspec <- rep(list(ugarchspec(distribution.model = "std")), ncol(X))
fit.ARMA.GARCH <- fit_ARMA_GARCH(X, ugarchspec.list = uspec, verbose = FALSE)
fits <- fit.ARMA.GARCH$fit
resi <- lapply(fits, residuals, standardize = TRUE)
X <- as.matrix(do.call(merge, resi))
colnames(X) <- colnames(x)
n <- nrow(X) # sample size
```

Four normal variance mixture models are considered: The multivariate t (an inverse-gamma mixture), a Pareto-mixture, an inverse-Burr mixture and the multivariate normal, where $\boldsymbol{X}$ follows an inverse-Burr mixture if $F_W^{\leftarrow}(u, \boldsymbol{\nu}) = (u^{-1/\nu_2} - 1)^{-1/\nu_1}$ (which is the quantile function of $1/\tilde{W}$ where $\tilde{W} \sim \text{Burr}(\nu_1, \nu_2)$ has distribution function $F_{\tilde{W}}(\tilde{w}) = 1 - (1 + \tilde{w}^{\nu_1})^{-\nu_2}$ for $\tilde{w} > 0$ and $\nu_1, \nu_2 > 0$). We highlight that in the inverse-Burr mixture case, neither the density of the resulting mixture nor weights for our estimation procedure are available in closed form, so that in this case, we indeed rely on our adaptive estimation procedure Algorithm 3.3.2 to estimate the log-density function. Note that, with the exception of the inverse-Burr mixture, `qmix` can be provided as a string so that `fitnvmix()` uses closed formulas for densities and weights as opposed to estimating them internally. Bounds on the mixing parameter also need to be provided via the argument `mix.param.bounds`.

```
qmix_ <- list(constant = "constant",
              inverse.gamma = "inverse.gamma",
              inverse.burr  = function(u, nu) (u^(-1/nu[2])-1)^(-1/nu[1]),
              pareto = "pareto")
m.p.b_ <- list(constant = c(0, 1e8), # irrelevant
                          inverse.gamma = c(1, 8),
                          inverse.burr = matrix(c(0.1, 0.1, 8, 8), ncol = 2),
                          pareto = c(1, 8))
```

We remark that the multivariate normal case is trivial from an estimation point of view, as the maximum likelihood estimators for $\boldsymbol{\mu}$ and Σ are merely the sample mean and the sample variance, respectively; this case is included for the sake of comparison.

We fit the aforementioned distributions to the stock data using Algorithm 3.4.4, imple-

mented in the function `fitnvmix()`.

```
fit.results <- lapply(1:4, function(i)
   fitnvmix(X, qmix = qmix_[[i]], mix.param.bounds = m.p.b_[[i]]))
```

For the inverse-gamma and Pareto-mixtures we find $\hat{\nu} = 5.65$ and $\hat{\nu} = 1.64$, respectively, when using the closed form densities and weights; if weights and densities are estimated, we found $\hat{\nu} = 5.62$ (20 sec) and $\hat{\nu} = 1.61$ (13 sec), respectively. Overall it is reassuring that the estimates obtained from analytical and estimated weights and densities only differ slightly; given the difficulty of the problem the run times also seem reasonable. For the inverse-Burr mixture, we found $\hat{\boldsymbol{\nu}} = (2.15, 3.61)$ after 30 seconds run-time.

To assess the fit of the different models, we construct QQ plots via `qqplot_maha()`; see Figure 3.11.

```
qq.results <- lapply(1:4, function(i)
    qqplot_maha(fitnvmix_object = fit.results[[i]]))
```

Clearly, the multivariate normal distribution (corresponding to constant W) provides a poor fit to the data as the tail is heavily underestimated. Both the inverse-gamma mixture and the inverse-Burr mixture provide an excellent fit to the data; the Pareto-mixture however shows too heavy tails. These plots confirm our main motivation outlined in the introduction: The multivariate normal is poorly suited for heavy-tailed return-data; normal variance mixtures, however, are more flexible in that they allow for heavier joint tails, often giving a better fit.

Next, we plot the fitted log-densities (computed using Algorithm 3.3.2 via the function `dnvmix`) as functions of the Mahalanobis distances $D^2(\boldsymbol{x}; \hat{\mu}, \hat{\Sigma})$; see Figure 3.12.

```
l.dens <- matrix(NA, ncol = 4, nrow = n)
mahas  <- matrix(NA, ncol = 4, nrow = n)
for(i in 1:4) {
   mahas[, i] <- sqrt(mahalanobis(X, center = fit.results[[i]]$loc,
                                  cov = fit.results[[i]]$scale))
   order.maha <- order(mahas[, i])
   mahas[, i] <- mahas[order.maha, i] # sorted for plotting
   l.dens[, i] <- dnvmix(X[order.maha, ], qmix = qmix_[[i]],
                         loc = fit.results[[i]]$loc,
                         scale = fit.results[[i]]$scale,
                         nu = fit.results[[i]]$nu, log = TRUE)
}
```

Next, we use Algorithm 3.2.2 to estimate the joint quantile shortfall probability

$$Q(u) := \mathbb{P}(X_1 > F_{X_1}^{\leftarrow}(u), \dots, X_d > F_{X_d}^{\leftarrow}(u)), \quad u \in (0,1).$$

In our context this is the probability that each of the 15 stocks yields a negative return larger than their respective u quantile; for large u, $Q(u)$ is the probability of a joint large loss and a rare event. This quantity is often considered in risk management to quantify the risk associated with joint extreme events. By radial symmetry of $\boldsymbol{X} \sim \mathrm{NVM}_d(\boldsymbol{\mu}, \Sigma, F_W)$ and continuity of the marginal distribution functions F_{X_j}, $j = 1, \dots, d$, it follows that for any $u \in (0,1)$,

$$\begin{aligned} Q(u) &= \mathbb{P}(X_1 > F_{X_1}^{\leftarrow}(u), \dots, X_d > F_{X_d}^{\leftarrow}(u)) = \mathbb{P}(X_1 \leq F_{X_1}^{\leftarrow}(1-u), \dots, X_d \leq F_{X_d}^{\leftarrow}(1-u)) \\ &= \mathbb{P}(F_{X_1}(X_1) \leq 1-u, \dots, F_{X_d}(X_d) \leq 1-u) = C(1-u, \dots, 1-u) \end{aligned}$$

where $C : [0,1]^d \to [0,1]$ with $C(\boldsymbol{u}) = \mathbb{P}(F_{X_1}(X_1) \leq u_1, \dots, F_{X_d}(X_d) \leq u_d)$ is the copula of $\boldsymbol{X}$. Such copula can be evaluated with the function `pnvmixcop()`, which first calls `qnvmix()` to estimate the quantiles $F_{X_j}^{\leftarrow}(u)$ and then calls `pnvmix()` with argument `upper` set to the corresponding quantile estimates.

```
n. <- 50
u. <- seq(0.95, to = 0.999, length.out = n.)
u.matrix <- matrix(u., nrow = n., ncol = ncol(X))
set.seed(2)
tailprobs <- sapply(1:4, function(i) pnvmixcopula(
   1 - u.matrix, qmix = qmix_[[i]], scale = cov2cor(fit.results[[i]]$scale),
   nu = fit.results[[i]]$nu, control = list(pnvmix.abstol = 1e-5)))
```

In Figure 3.13 we plot the estimated quantile shortfall probability $Q(u)$ for a range of values of u for each fitted model separately. The figure on the right shows the same probabilities $Q(u)$ standardized by the corresponding normal probability. The plots show again that the Pareto-mixture is significantly more heavy tailed than the multivariate t distribution: It yields significantly higher shortfall probabilities. Furthermore these plots exemplify that our Algorithm 3.2.2 is also capable of estimating small probabilities despite the increasing numerical difficulty when moving outwards in the joint tail.

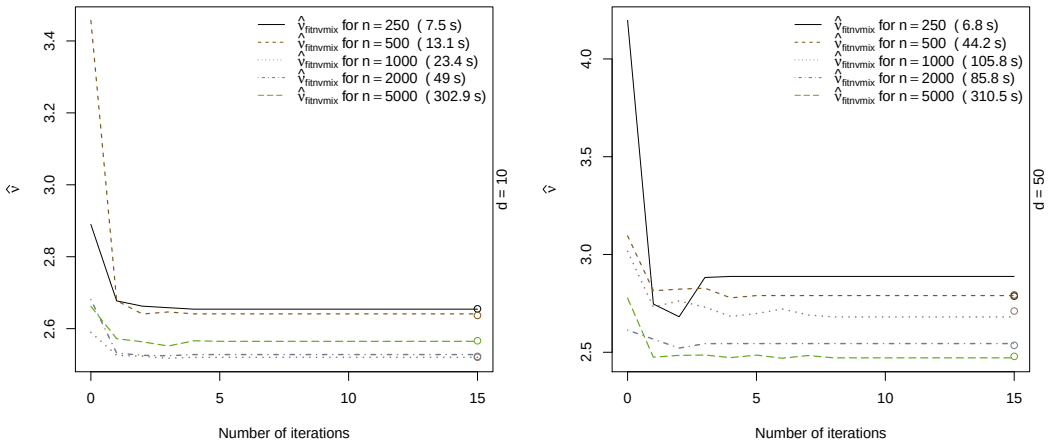

Figure 3.9: Estimates $\hat{\nu}$ computed by Algorithm 3.4.4 as a function of the number of ECME iterations for multivariate t distributions of different sample sizes and dimensions. The symbols at the end of each curve denote the maximum likelihood estimator of ν as found by the ECME algorithm with analytical weights and densities.

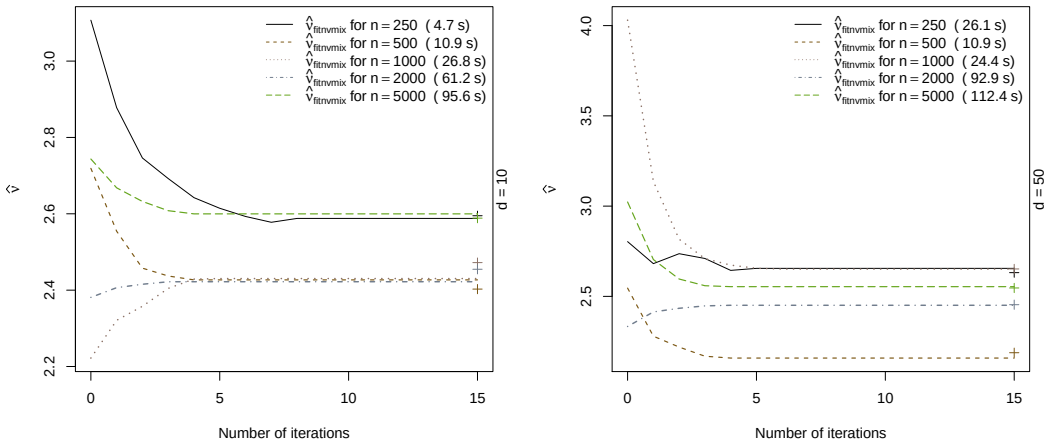

Figure 3.10: Estimates $\hat{\nu}$ computed by Algorithm 3.4.4 as a function of the number of ECME iterations for Pareto mixture distributions of different sample sizes and dimensions. The symbols at the end of each curve denote the maximum likelihood estimator of ν as found by the ECME algorithm with analytical weights and densities.

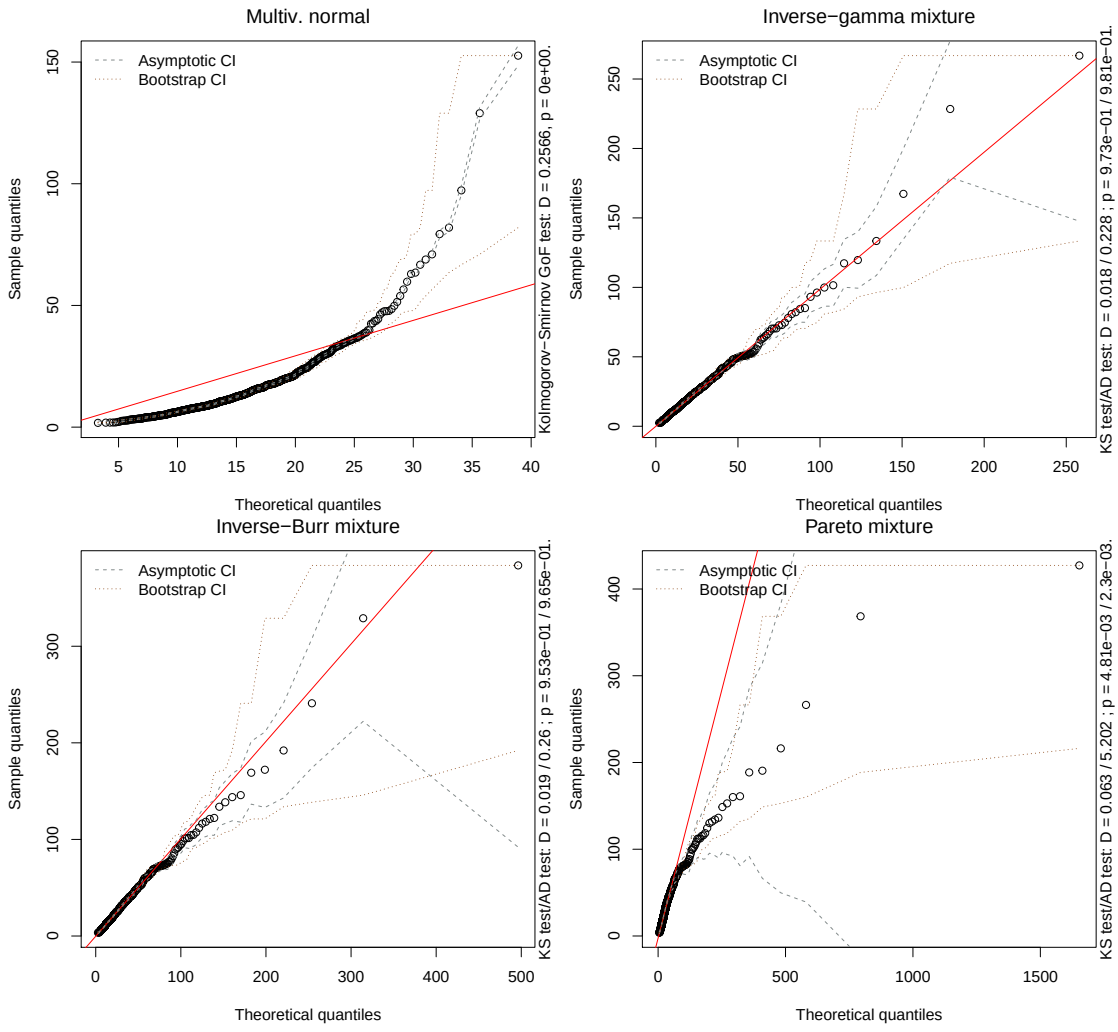

Figure 3.11: Q-Q Plots of the empirical quantiles of the Mahalanobis distances $D^2(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}, \hat{\Sigma})$, $i = 1, \ldots, n$, versus their theoretical quantiles for different models using a 5 stock portfolio with data from the `SP500` data set.

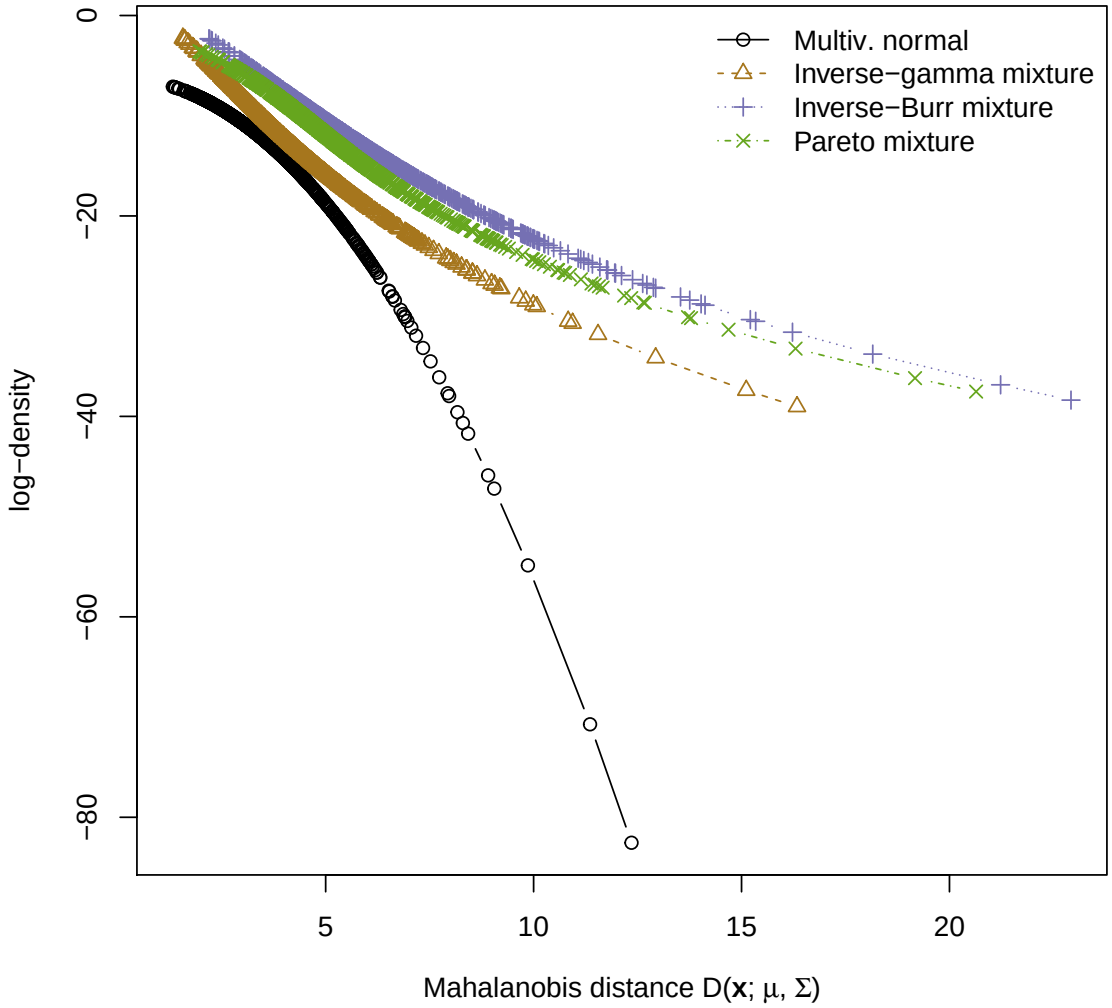

Figure 3.12: Log-densities as functions of the Mahalanobis distance for four fitted normal variance mixture models using a 15 stock REIT portfolio with data from the SP500 data set from 2010-01-01 to 2012-12-31 after deGARCHing.

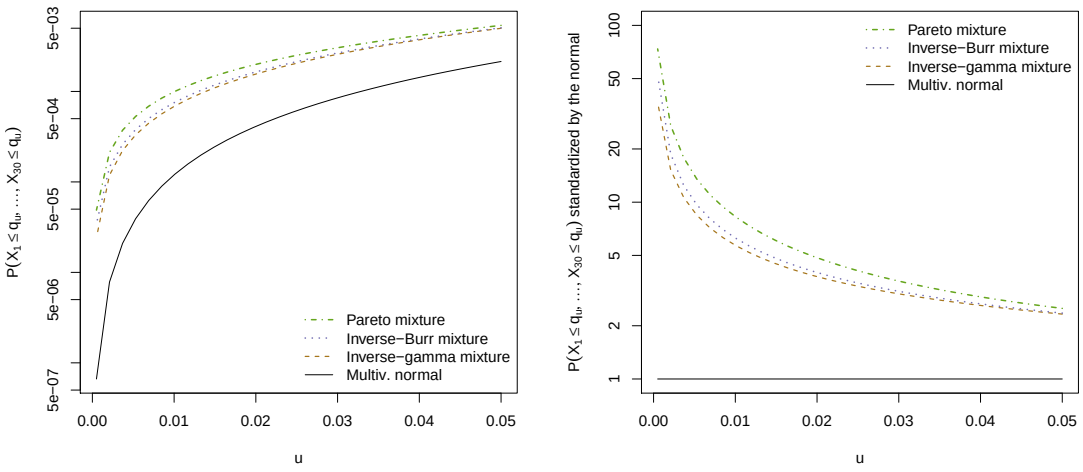

Figure 3.13: Estimated shortfall probabilities for different models for a 5 stock portfolio with data from the `SP500` data set (left); same probabilities standardized by the normal case (right).

3.7 Grouped normal variance mixtures

We define *grouped normal variance mixtures* via the stochastic representation

$$\boldsymbol{X} = \boldsymbol{\mu} + \operatorname{diag}(\sqrt{\boldsymbol{W}})A\boldsymbol{Z}, \tag{3.23}$$

where $\boldsymbol{W} = (W_1, \dots, W_d)$ is a d-dimensional non-negative and comonotone random vector with $W_j \sim F_{W_j}$ that is independent of $\boldsymbol{Z}$. Denote by $F_W^{\leftarrow}(u) = \inf\{w \geq 0 : F_W(w) \geq u\}$ the quantile function of a random variable W. Comonotonicity of the W_j implies the stochastic representation

$$\boldsymbol{W} = (W_1, \dots, W_d) = (F_{W_1}^{\leftarrow}(U), \dots, F_{W_d}^{\leftarrow}(U)), \quad U \sim \mathrm{U}(0,1). \tag{3.24}$$

If a d-dimensional random vector $\boldsymbol{X}$ satisfies (3.23) with $\boldsymbol{W}$ given as in (3.24), we use the notation $\boldsymbol{X} \sim \mathrm{gNVM}(\boldsymbol{\mu}, \Sigma, F_{\boldsymbol{W}})$ where $F_{\boldsymbol{W}}(\boldsymbol{w}) = \mathbb{P}(\boldsymbol{W} \leq \boldsymbol{w})$ for $\boldsymbol{w} \in \mathbb{R}^d$ and the inequality is understood component-wise. By moving from a scalar mixing rv to a comonotone random vector, one obtains non-elliptical distributions well beyond the classical multivariate t case, giving rise to flexible modelling of joint and marginal body and tail behaviours. The price to pay for this generalization are significant computational challenges: Not even the density of a grouped t distribution is available in closed form.

At first glance, the definition given in (3.23) does not indicate any "grouping" yet. However, Equation (3.24) allows one to group components of the random vector $\boldsymbol{X}$ such that all components within a group have the same mixing distribution. More precisely, let $\boldsymbol{W}$ be split into S sub-vectors, i.e., $\boldsymbol{W} = (\boldsymbol{W}_1, \dots, \boldsymbol{W}_S)$ where $\boldsymbol{W}_k$ has dimension d_k for $k = 1, \dots, S$ and $\sum_{k=1}^S d_k = d$. Now let each $\boldsymbol{W}_k$ have stochastic representation $\boldsymbol{W}_k = (F_{W_k}^{\leftarrow}(U), \dots, F_{W_k}^{\leftarrow}(U))$. Hence, all univariate margins of the subvector $\boldsymbol{W}_k$ are identically distributed. This implies that all margins of the corresponding subvector $\boldsymbol{X}_k$ are of the same type.

An example is the copula derived from $\boldsymbol{X}$ in (3.23) when $F_{W_k} = \mathrm{IG}(\nu_k/2, \nu_k/2)$ for $k = 1, \dots, S$; this is the aforementioned grouped t copula. Here, different margins of the copula follow (potentially) different t-copulas with different dof, allowing for more flexibility in modelling pairwise dependencies. A grouped t-copula with $S = d$, that is when each component has their own mixing distribution, was proposed in [115] (therein called "individuated t copula") and studied in more detail in [82] (therein called "t copula with multiple dof"). If $S = 1$, the classical t-copula with exactly one dof parameter is recovered.

For notational convenience, derivations in this section are often done for the case $S = d$, so that the F_{W_j} are all different; the case $S < d$, that is when grouping is present, is merely

a special case where some of the F_{W_j} are identical. That being said, we chose to keep the name "grouped" to refer to this class of models so as to reflect the original motivation for this type of model, e.g., as in [21], where it is used to model the components of a portfolio in which there are subgroups representing different business sectors.

Previous work on grouped t-copulas and their corresponding distributions includes some algorithms for the tasks needed to handle these models, but were mostly focused on demonstrating the superiority of this class of models over special cases such as the multivariate normal or t distribution. More precisely, in [21] the grouped t copula was introduced and applied to model an internationally diversified credit portfolio of 92 risk factors split into 8 subgroups. It was demonstrated that the grouped t copula is superior to both the Gaussian and t copula in regards to modelling the tail dependence present in the data. [82] also study the grouped t copula and, unlike in [21], allow group sizes of 1 (corresponding to $S = d$ in our definition). They provide calibration methods to fit the copula to data and furthermore study bivariate characteristics of the grouped t copula, including symmetry properties and tail dependence.

We extend the algorithms for the ungrouped case to grouped case in the first subsection. In particular, we provide algorithms to estimate the joint density and distribution function, thereby filling a gap in the existing literature; these results have been published in [52] and are implemented in the R package `nvmix`. Finally, we illustrate how the R pacakge `nvmix` can be used to sample from grouped normal variance mixtures.

3.7.1 Estimating the distribution and density function

To simplify the notation, we use the shorthand notations $F^{\leftarrow}_{\boldsymbol{W}}(U) = (F^{\leftarrow}_{W_1}(U), \dots, F^{\leftarrow}_{W_d}(U))$, $\boldsymbol{W}^{\mathrm{D}} = \operatorname{diag}(\boldsymbol{W})$, $\boldsymbol{W}^{\mathrm{D}}(U) = \operatorname{diag}(F^{\leftarrow}_{\boldsymbol{W}}(U))$; $\sqrt{\boldsymbol{W}}^{\mathrm{D}}$, $(1/\boldsymbol{w})^{\mathrm{D}}$ as well as $(1/\sqrt{\boldsymbol{w}})^{\mathrm{D}}$ are defined similarly. Many properties of grouped normal variance mixtures are derived by conditioning on the d-dimensional random vector $\boldsymbol{W}$, or equivalently by conditioning on the underlying univariate uniform rv U. Indeed,

$$\boldsymbol{X} \mid \boldsymbol{W} \sim \mathrm{N}_d\left(\boldsymbol{\mu}, \sqrt{\boldsymbol{W}}^{\mathrm{D}} \Sigma \sqrt{\boldsymbol{W}}^{\mathrm{D}}\right) \quad \text{or equivalently} \quad \boldsymbol{X} \mid U \sim \mathrm{N}_d\left(\boldsymbol{\mu}, \sqrt{\boldsymbol{W}}^{\mathrm{D}}(U) \Sigma \sqrt{\boldsymbol{W}}^{\mathrm{D}}(U)\right).$$

One can see that $\boldsymbol{W}$ "mixes" the covariance matrix of a multivariate normal and can be regarded as a shock affecting all components of $\boldsymbol{X}$.

Let $-\boldsymbol{\infty} \leq \boldsymbol{a} < \boldsymbol{b} \leq \boldsymbol{\infty}$ componentwise (entries $\pm\infty$ to be interpreted as the corresponding limits). Then $F(\boldsymbol{a}, \boldsymbol{b}) = \mathbb{P}(\boldsymbol{a} < \boldsymbol{X} \leq \boldsymbol{b})$ is the probability that the random vector $\boldsymbol{X}$ falls in the hyper-rectangle spanned by the lower-left and upper-right endpoints $\boldsymbol{a}$ and

$\boldsymbol{b}$, respectively. If $\boldsymbol{a} = (-\infty, \dots, -\infty)$, we recover $F(\boldsymbol{a}, \boldsymbol{x}) = F(\boldsymbol{x}) = \mathbb{P}(X_1 \le x_1, \dots, X_d \le x_d)$ which is the (cumulative) distribution function of $\boldsymbol{X}$.

Assume wlog that $\boldsymbol{\mu} = \boldsymbol{0}$, otherwise adjust $\boldsymbol{a}$, $\boldsymbol{b}$ accordingly. Then

$$\begin{aligned} F(\boldsymbol{a}, \boldsymbol{b}) &= \mathbb{P}(\boldsymbol{a} < \sqrt{\boldsymbol{W}}^{\mathrm{D}} A\boldsymbol{Z} \le \boldsymbol{b}) = \mathbb{E}\Big[\mathbb{P}\big((1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(U)\, \boldsymbol{a} < A\boldsymbol{Z} \le (1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(U)\, \boldsymbol{b} \,|\, U\big)\Big] \\ &= \mathbb{E}\Big[\Phi_\Sigma\big((1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(U)\, \boldsymbol{a}, (1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(U)\, \boldsymbol{b}\big)\Big] = \int_0^1 \Phi_\Sigma\Big((1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(u)\, \boldsymbol{a}, (1/\sqrt{\boldsymbol{W}})^{\mathrm{D}}(u)\, \boldsymbol{b}\Big)\, \mathrm{d}u, \end{aligned} \tag{3.25}$$

where $\Phi_\Sigma(\boldsymbol{a}, \boldsymbol{b}) = \mathbb{P}(\boldsymbol{a} < \boldsymbol{Y} \le \boldsymbol{b})$ for $\boldsymbol{Y} \sim \mathrm{N}_d(\boldsymbol{0}, \Sigma)$ and we recall that $AA^\top = \Sigma$. Comonotonicity of the W_j allowed us to write $F(\boldsymbol{a}, \boldsymbol{b})$ as a $(d+1)$-dimensional integral; had the W_j a different dependence structure, this convenience would be lost and the resulting integral in (3.25) could be up to $2d$-dimensional (e.g., when all W_j are independent).

Using the same sequence of transformations as in Section 3.2.1, we find that

$$F(\boldsymbol{a}, \boldsymbol{b}) = \int_{(0,1)^d} g(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} = \int_0^1 g_1(u_0) \int_0^1 g_2(u_0, u_1) \cdots \int_0^1 g_d(u_0, \dots, u_{d-1})\, \mathrm{d}u_{d-1} \dots \mathrm{d}u_0,$$

where

$$g(\boldsymbol{u}) = \prod_{i=1}^d g_i(u_0, \dots, u_{i-1}), \quad g_i(u_0, \dots, u_{i-1}) = e_i - d_i, \quad i = 1, \dots, d, \tag{3.26}$$

for $\boldsymbol{u} = (u_0, u_1, \dots, u_{d-1}) \in (0,1)^d$. The e_i are recursively defined by

$$\begin{aligned} e_1 &= e_1(u_0) = \Phi\left(\frac{b_1}{C_{11}\sqrt{F^{\leftarrow}_{W_1}(u_0)}}\right), \\ e_i &= e_i(u_0, \dots, u_{i-1}) = \Phi\left(\frac{1}{C_{ii}}\left(\frac{b_i}{\sqrt{F^{\leftarrow}_{W_i}(u_0)}} - \sum_{j=1}^{i-1} C_{ij}\Phi^{-1}(d_j + u_j(e_j - d_j))\right)\right), \end{aligned}$$

for $i = 2, \dots, d$ and the d_i are e_i with b_i replaced by a_i for $i = 1, \dots, d$. With the integrand g at hand, we can proceed as in Section 3.2.2 to estimate $F(a, b)$. In particular, first, a greedy re-ordering algorithm is applied to the inputs $\boldsymbol{a}$, $\boldsymbol{b}$, Σ. It re-orders the components $1, \dots, d$ of $\boldsymbol{a}$ and $\boldsymbol{b}$ as well as the corresponding rows and columns in Σ in a way that the expected ranges of g_i in (3.26) are increasing with the index i for $i = 1, \dots, d$,

just as in the normal variance mixture case, by applying Alg. 3.2.1 with $a_j/\mu_{\sqrt{W}}$ replaced by $a_j/\mu_{\sqrt{W_j}}$ and similarly for b_j for $j = 1, \dots, d$ to account for the generalization.

Let us now focus on the density of $\boldsymbol{X} \sim \text{gNVM}(\boldsymbol{\mu}, \Sigma, F_{\boldsymbol{W}})$, where we assume that Σ has full rank in order for the density to exist. The same conditioning argument used to derive (3.25) yields that the density of $\boldsymbol{X} \sim \text{gNVM}_d(\boldsymbol{\mu}, \Sigma, F_{\boldsymbol{W}})$ evaluated at $\boldsymbol{x} \in \mathbb{R}^d$ can be written as

$$\begin{aligned} f_{\boldsymbol{X}}(\boldsymbol{x}) &= \mathbb{E}\left(\frac{1}{\sqrt{(2\pi)^d |\sqrt{\boldsymbol{W}}^{\mathrm{D}}(U)\Sigma\sqrt{\boldsymbol{W}}^{\mathrm{D}}(U)|}} \exp\left(-\frac{D^2\left(\boldsymbol{x}; \boldsymbol{\mu}, \sqrt{\boldsymbol{W}}^{\mathrm{D}}(U)\Sigma\sqrt{\boldsymbol{W}}^{\mathrm{D}}(U)\right)}{2} \right) \right) \\ &= \int_0^1 \frac{1}{\sqrt{(2\pi)^d |\Sigma| \prod_{i=1}^d F^{\leftarrow}_{W_i}(u)}} \exp\left(-\frac{D^2\left(\boldsymbol{x}; \boldsymbol{\mu}, \sqrt{\boldsymbol{W}}^{\mathrm{D}}(u)\Sigma\sqrt{\boldsymbol{W}}^{\mathrm{D}}(u)\right)}{2} \right) \mathrm{d}u \\ &= \int_0^1 h(u)\, \mathrm{d}u, \end{aligned} \tag{3.27}$$

where $D^2(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = (\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$ and the integrand $h(u)$ is defined in an obvious manner. Except for some special cases (e.g., when all W_j are inverse-gamma with the same parameters), this integral cannot be computed explicitly, so that we rely on numerical approximation thereof.

From (3.27) we find that computing the density $f(\boldsymbol{x})$ of $\boldsymbol{X} \sim \text{gNVM}_d(\boldsymbol{\mu}, \Sigma, F_{\boldsymbol{W}})$ evaluated at $\boldsymbol{x} \in \mathbb{R}^d$ requires the estimation of a univariate integral just like in the ungrouped case. We generalize Algorithm 3.3.2 to the grouped case, which is more complicated because the distribution is not elliptical, hence the density does not only depend on $\boldsymbol{x}$ through $D^2(\boldsymbol{x}, \boldsymbol{\mu}, \Sigma)$. Furthermore, the height of the (unique) maximum of h in the ungrouped case can be easily computed without simulation, which helps the adaptive procedure find the relevant region; in the grouped case, the value of the maximum is usually not available. Lastly, S (as opposed to 1) quantile evaluations are needed to obtain one function value $h(u)$; from a run time perspective, evaluating these quantile functions is the most expensive part.

Summarizing, we propose the following method to estimate $\log(f(\boldsymbol{x}_i))$, $i = 1, \dots, N$, for given inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ and error tolerance ε.

Algorithm 3.7.1 (Adaptive RQMC Algorithm to estimate $\log(f(\boldsymbol{x}_1)), \dots, \log(f(\boldsymbol{x}_N))$.) Given $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$, Σ, ε, ε_{th}, n_0, estimate $\log(f(\boldsymbol{x}_l))$, $l = 1, \dots, N$, via:

1. Compute $\hat{\mu}^{\text{RQMC}}_{\log f(\boldsymbol{x}_i),n_0}$ with sample size n_0 using the same random numbers for all input $\boldsymbol{x}_i$, $i = 1, \dots, N$. Store all uniforms with corresponding quantile evaluations $F^{\leftarrow}_{\boldsymbol{W}}$ in a list $\mathcal{L}$.

2. If all estimates $\hat{\mu}^{\text{RQMC}}_{\log f(\boldsymbol{x}_i),n_0}$, $i = 1, \dots, N$, meet the error tolerance ε, go to Step 4. Otherwise let $\boldsymbol{x}_s$, $s = 1, \dots, N'$ with $1 \le N' \le N$ be the inputs whose error estimates exceed the error tolerance.

3. For each remaining input $\boldsymbol{x}_s$, $s = 1, \dots, N'$, do:

   (a) Use all pairs $(u, F^{\leftarrow}_{\boldsymbol{W}}(u))$ in $\mathcal{L}$ to compute values of $h(u)$ and set $\hat{h}_{\max} = \max_{u \in \mathcal{L}} h(u)$. If the largest value of h is obtained for the largest (smallest) u in the list $\mathcal{L}$, set $u^* = 1$ ($u^* = 0$).

   (b) If $u^* = 1$, set $u_r = 1$ and if $u^* = 0$, set $u_l = 0$. Unless already specified, use bisections to find u_l and u_r such that $u_l < u^* < u_r$ and u_l (u_r) is the smallest (largest) u such that $h(u) > \varepsilon_{\text{th}}$ from (3.13) with $h_{\max}$ replaced by $\hat{h}_{\max}$. Starting intervals for the bisections can be found from the values in $\mathcal{L}$.

   (c) If $u_l > 0$, approximate $\log(\int_0^{u_l} h(u)\,\mathrm{d}u)$ using a trapezoidal rule with proper logarithm and knots $u'_1, \dots, u'_m$ where u'_i are those u's in $\mathcal{L}$ satisfying $u \le u_l$. Call the approximation $\hat{\mu}_{(0,u_l)}(\boldsymbol{x}_s)$. If $u_l = 0$, set $\hat{\mu}_{(0,u_l)} = -\infty$.

   (d) If $u_r < 1$, approximate $\log(\int_{u_r}^1 h(u)\,\mathrm{d}u)$ using a trapezoidal rule with proper logarithm and knots $u''_1, \dots, u''_p$ where u''_i are those u's in $\mathcal{L}$ satisfying $u \ge u_r$. Call the approximation $\hat{\mu}_{(u_r,1)}(\boldsymbol{x}_s)$. If $u_r = 0$, set $\hat{\mu}_{(u_r,1)}(\boldsymbol{x}_s) = -\infty$.

   (e) Estimate $\log(\int_{u_l}^{u_r} h(u)\,\mathrm{d}u)$ via RQMC. That is, compute $\hat{\mu}^{\text{RQMC}}_{\log f,n}$ via Algorithm 2.1.2 where every $u_{i,b} \in (0,1)$ is replaced by $u'_{i,b} = u_l + (u_r - u_l)u_{i,b} \in (u_l, u_r)$. Increase n until the error tolerance ε is met. Then set $\hat{\mu}_{(u_l,u_r)} = \log(u_r - u_l) + \hat{\mu}^{\text{RQMC}}_{\log f,n}$ which estimates $\log(\int_{u_l}^{u_r} h(u)\,\mathrm{d}u)$.

   (f) Combine

$$\hat{\mu}^{\text{RQMC}}_{\log f(\boldsymbol{x}_s)} = \text{LSE}\left(\hat{\mu}_{(0,u_l)}(\boldsymbol{x}_s), \hat{\mu}_{(u_l,u_r)}(\boldsymbol{x}_s), \hat{\mu}_{(u_r,1)}(\boldsymbol{x}_s)\right)$$

4. Return $\hat{\mu}^{\text{RQMC}}_{\log f(\boldsymbol{x}_l)}$, $l = 1, \dots, N$.

This algorithm is implemented in the function `dgnvmix(, log = TRUE)` in the R package `nvmix`, which by default uses a Relative error (RE) tolerance.

The advantage of the proposed algorithm is that only little run time is spent on estimating "easy" integrals, thanks to the pilot run in Step 1. If $n_0 = 2^{10}$ and $B = 15$ (the

current default in the `nvmix` package), this step gives 15 360 pairs $(u, F^{\leftarrow}_{\boldsymbol{W}}(u))$. These pairs give good starting values for the bisections to find u_l, u_r. Note that no additional quantile evaluations are needed to estimate the less important regions $(0, u_l)$ and $(u_r, 1)$.

[82] are faced with almost the same integration problem when estimating the density of a bivariate grouped t-copula. They use a globally adaptive integration scheme from [99] to integrate h. While this procedure works well for a range of inputs, it deteriorates for input $\boldsymbol{x}$ with large components.

We consider a grouped inverse-gamma mixture model, so a grouped t distribution, and let $\boldsymbol{X} \sim \mathrm{gt}_d(\boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma)$. The density $f^{\mathrm{gt}}_{\boldsymbol{\nu},\boldsymbol{\mu},P}$ of $\boldsymbol{X} \sim \mathrm{gt}_d(\boldsymbol{\nu}, \boldsymbol{\mu}, \Sigma)$ is not available in closed form, so that here we indeed need to rely on estimation of the latter. The following experiment is performed for $\boldsymbol{X} \sim \mathrm{gt}_2(\boldsymbol{\nu}, \mathbf{0}, I_2)$ with $\boldsymbol{\nu} = (3, 6)$ and for $\boldsymbol{X} \sim \mathrm{gt}_{10}(\boldsymbol{\nu}, \mathbf{0}, I_{10})$ where $\boldsymbol{\nu} = (3, \dots, 3, 6, \dots, 6)$ (corresponding to two groups of size 5 each). First, a sample from a more heavy tailed grouped t distribution of size 2500 is sampled (with degrees of freedom $\boldsymbol{\nu}' = (1, 2)$ and $\boldsymbol{\nu}' = (1, \dots, 1, 2, \dots, 2)$, respectively) and then the log-density function of $\boldsymbol{X} \sim \mathrm{gt}_d(\boldsymbol{\nu}, \mathbf{0}, I_d)$ is evaluated at the sample. The results are shown in Figure 3.14.

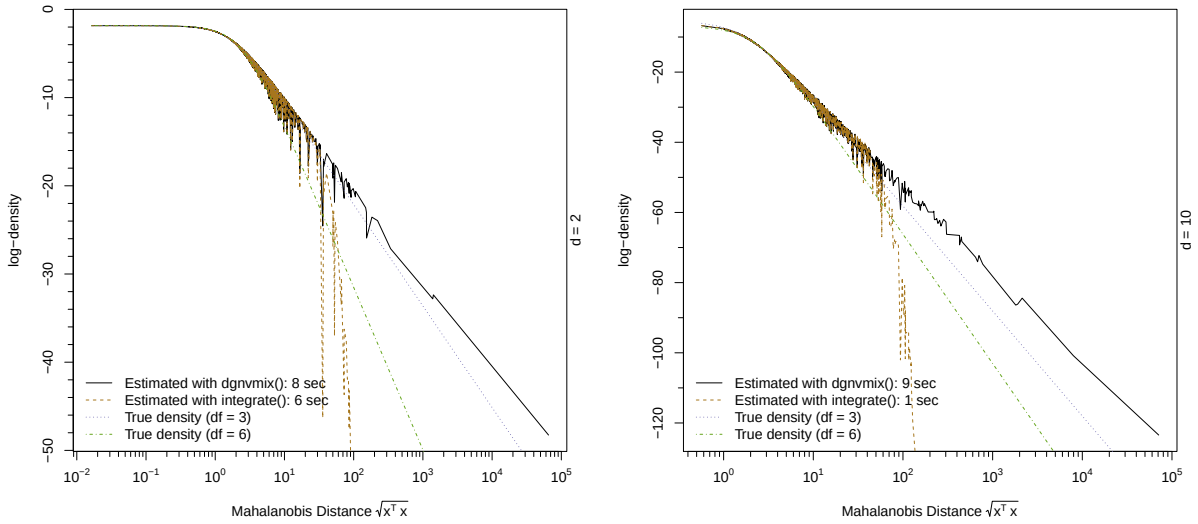

Figure 3.14: Estimated log-density of a grouped t distribution with $\boldsymbol{\nu} = (3, 6)$ in $d = 2$ (left) and $\boldsymbol{\nu} = (3, \dots, 3, 6, \dots, 6)$ in $d = 10$ (right). Estimation with `dgnvmix()` was carried out using a relative error tolerance of 0.01. The plot also shows the log-density function of $t_d(3, \mathbf{0}, I_d)$ and $t_d(6, \mathbf{0}, I_d)$ for comparison.

It is clear from the plots that `integrate()` again gives wrong approximations to $f(\boldsymbol{x})$ for input $\boldsymbol{x}$ far out in the tail; for small input $\boldsymbol{x}$, the results from `integrate()` and from `dgnvmix()` coincide. Furthermore, it can be seen that the density function is not monotonic in the Mahalanobis distance (as grouped normal mixtures are not elliptical anymore). The plot also includes the log-density functions of an ungrouped d-dimensional t distribution with degrees of freedom 3 and 6, respectively. The log-density function of the grouped mixture with $\boldsymbol{\nu} = (3, 6)$ is not bounded by either; in fact, the grouped mixture shows heavier tails than both the t distribution with 3 and with 6 dof.

3.7.2 Sampling grouped normal variance mixtures

To illustrate how the R package `nvmix` can be used to sample any grouped normal variance mixture, assume $\boldsymbol{X} \sim \text{gNVM}_5(0, \Sigma, F_{\boldsymbol{W}})$ where $W_1 \sim \text{IG}(1, 1)$, $W_2 = 1$ a.s., $W_3 \sim \text{Exp}(1)$, $W_4 \sim \text{Par}(2, 1)$ and $W_5 = W_1$. That is, marginally, $X_1, X_5 \sim t_2$, $X_2 \sim \text{N}(0, 1)$, and X_3 and X_4 are Exponential and Pareto mixtures. The following code samples 1000 iid copies of $\boldsymbol{X}$; see Figure 3.15 for a pairs plot of the sample. Note how different bivariate margins are of quite different types; this cannot be the case for normal variance mixtures which are elliptical; see [85, Chapter 6].

```
d <- 5 # dimension
df <- 2 # dof for margins 1, 5
n <- 1e3 # sample size
A <- matrix(runif(d * d), ncol = d)
scale <- cov2cor(A %*% t(A)) # (random) scale matrix
## Group structure (here): Components 1+5 same group, all others individual
groupings <- c(1, 2, 3, 4, 1)
## Specify mixing distribution for each group
qmix <- list(function(u, df) 1 / qgamma(1-u, shape = df/2, rate = df/2),
             function(u) rep(1, length(u)),
             list("exp", rate = 1), function(u) (1-u)^(-1/2))
## Sample
r.gnvm <- rgnvmix(n, groupings = groupings, qmix = qmix, scale = scale, df = df)
```

3.8 Fitting t and grouped t copulas

In this section, we address the problem of fitting t and grouped t copulas to data. In the former case, we provide an EM like algorithm and compare it with three existing methods

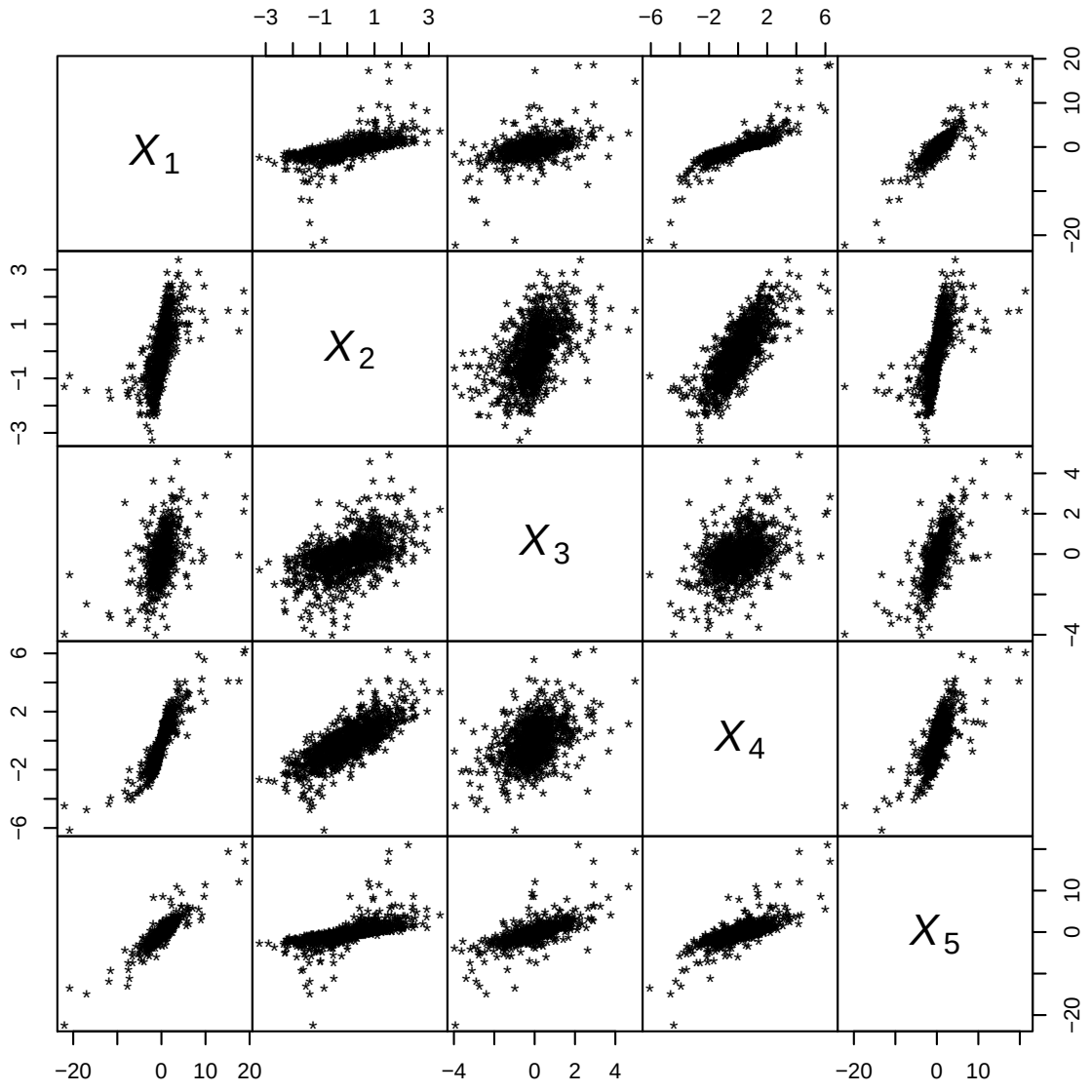

Figure 3.15: Plot of the samples `r.gnvm` from a 5-dimensional grouped normal variance mixture.

to tackle this problem. For the grouped t copula, we argue for joint maximization of the copula log-likelihood and show that this gives higher likelihood than the group-wise fitting suggested in [23]. Most of the results in this section are published in [54]; the fitting algorithm for the grouped t copula was derived in [52].

3.8.1 Notations

Let F be the joint distribution function of a d-dimensional random vector $\boldsymbol{X}$ with absolutely continuous marginal distribution functions F_j for $j = 1, \dots, d$. Denote by f and f_j the joint and marginal density functions. By Sklar's theorem, the copula of $\boldsymbol{X}$, that is, the joint distribution function of the marginally uniform random vector $(F_1(X_1), \dots, F_d(X_d))$ is unique and given by

$$C(\boldsymbol{u}) = F(F_1^{\leftarrow}(u_1), \dots, F_d^{\leftarrow}(u_d)), \quad \boldsymbol{u} = (u_1, \dots, u_d) \in (0,1)^d.$$

The copula density follows to be

$$c(\boldsymbol{u}) = \frac{f(F_1^{\leftarrow}(u_1), \dots, F_d^{\leftarrow}(u_d))}{\prod_{j=1}^d f_j(F_j^{\leftarrow}(u_j))}, \quad \boldsymbol{u} \in (0,1)^d.$$

If $\boldsymbol{X} \sim \mathrm{MVT}_d(\nu, \mathbf{0}, P)$ for a correlation matrix P, the copula of $\boldsymbol{X}$ is the t-copula, denoted by $C^t_{\nu,P}$. If $\boldsymbol{X} \sim \mathrm{gt}_d(\boldsymbol{\nu}, \mathbf{0}, P)$, the copula of $\boldsymbol{X}$ is the grouped t copula, denoted by $C^{gt}_{\boldsymbol{\nu},P}$. The copula densities are denoted with smaller case letters as $c^t_{\nu,P}$ and $c^{gt}_{\boldsymbol{\nu},P}$, respectively.

3.8.2 Fitting the t copula: An EM-like algorithm

Given $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n \overset{\text{ind.}}{\sim} C^t_{\nu,P}$, the copula log-likelihood function is given by

$$\log L(\nu, P; \boldsymbol{U}_1, \dots, \boldsymbol{U}_n) = \sum_{i=1}^n \log c^t_{\nu,P}(\boldsymbol{U}_i) = \sum_{i=1}^n \log f(\boldsymbol{X}_i; \nu, P) - \sum_{i=1}^n \sum_{j=1}^d \log f_j(\boldsymbol{X}_{ij}; \nu), \tag{3.28}$$

where $X_{ij} = t_\nu^{-1}(U_{ij})$ for $i = 1, \dots, n$ and $j = 1, \dots, d$, f is the joint density of $\mathrm{MVT}(\nu, \mathbf{0}, P)$ and the f_j are univariate t_ν densities. Fitting the t copula to data by means of Maximum likelihood (ML) estimation, i.e., maximizing $\log L$, requires optimization of a $(d(d-$

$1)/2 + 1)$-dimensional log-likelihood function. This optimization problem has non-linear constraints (as positive-semidefiniteness of P must be ensured), making this problem particularly hard. The R package `copula` (see [120], [70], [63] and [61]) currently uses such ML procedure via the function `fitCopula()`, which searches for the optimum in the space of all square matrices and rejects those that are not positive definite.

In the context of estimating skew-t copulas, [121] instead represent the Cholesky factor L of P (a lower triangular matrix L such that $LL^\top = P$) using angles $\theta_{ij} \in [0, \pi)$ for $j = 1, \dots, i-2$ and $\theta_{i,i-1} \in [0, 2\pi)$ for $i = 2, \dots, d$; see [121, Equation 12] for details. The benefit of this transformation is that one can proceed by maximizing the copula log-likelihood without imposing non-linear constraints (or rejecting non-positive definite matrices). We refer to this method as "Full-MLE". Note that the optimization problem still has $\mathcal{O}(d^2)$ parameters, making this procedure ill-suited for higher dimensions d.

If ν is known, we see from (3.28) that maximizing $\log L$ is equivalent to maximizing the term $\sum_{i=1}^n \log f(\boldsymbol{X}_i; \nu, P)$. As $\boldsymbol{X}_i \overset{\text{ind.}}{\sim} \text{MVT}_d(\nu, \boldsymbol{0}, P)$, this is the same finding the ML estimator of the scale matrix of a multivariate t distribution, which can be done efficiently using the EM algorithm; see [24], [102], and [85, Chapter 15.1]. This motivates maximizing the profile log-likelihood, given by

$$\log L^*(\nu; \boldsymbol{U}_1, \dots, \boldsymbol{U}_n) = \log L(\nu, \hat{P}(\nu); \boldsymbol{U}_1, \dots, \boldsymbol{U}_n),$$

where $\hat{P}(\nu)$ is the ML estimator of P based on the samples $\boldsymbol{X}_i$, $i = 1, \dots, n$. Note that $\log L^*$ is only a function of ν and thus univariate. As such, we were able to drastically reduce the dimensionality of the optimization problem. We refer to this method as "EM-MLE"

Another popular estimation method for t copulas explained in [83, Appendix C] (see also [23]) is to empirically estimate all pairwise Kendall's tau ρ_{ij}^τ, $1 \le i < j \le d$, and then map these estimates to a correlation matrix P using $P_{ij} = \sin(\pi \rho_{ij}^\tau / 2)$. With an estimate of P at hand, the degrees-of-freedom ν can be estimated by optimizing a univariate log-likelihood function. We refer to this method as "Moment-MLE".

The function `fitStudentcopula()` from the R package `nvmix` provides all previously mentioned estimation methods. One can supply initial estimates for the degrees-of-freedom ν and bounds for it via the arguments `df.init` and `df.bounds`. In the following, we use the function `fitStudentcopula()` to perform a simulation study to investigate the performance of the three methods. For comparison, we also include the `fitCopula()` function of the R package `copula` as a fourth method.

```
1 set.seed(1)
```

```
methods <- c("Moment-MLE", "EM-MLE", "Full-MLE", "Full-MLE ('copula')")
reps <- 50 # number of replications for each method
res <- array(, dim = c(length(methods), reps, 4),
             dimnames = list(method = methods, rep = 1:reps,
                             c("loglik", "cpu", "df", "rho12")))
for(i in 1:reps) {
   U <- rStudentcopula(n, df = df, scale = scale) # sample
   t.mm   <-  system.time(fit.mm  <- fitStudentcopula(U, fit.method = "Moment-
      MLE"))[1]
   t.em   <-  system.time(fit.em  <- fitStudentcopula(U, fit.method = "EM-MLE"))
      [1]
   t.fl <- system.time(fit.fl <- fitStudentcopula(U, fit.method = "Full-MLE"))
      [1]
   t.fl.cop <- system.time(fit.fl.cop <- fitCopula(
      tCopula(dim = d, dispstr = "un"), U, method = "ml", estimate.variance =
         FALSE,
      start = c(fit.mm$scale[upper.tri(fit.mm$scale)], 5)))[1]
   res[, i, "loglik"] <- c(fit.mm$max.ll, fit.em$max.ll, fit.fl$max.ll,
                           fit.fl.cop@loglik)
   res[, i, "cpu"] <- c(t.mm, t.em, t.fl, t.fl.cop)
   res[, i, "df"] <- c(fit.mm$df, fit.em$df, fit.fl$df,
                       fit.fl.cop@estimate[d*(d-1)/2+1])
   res[, i, "rho12"] <- c(fit.mm$scale[1, 2], fit.em$scale[1, 2], fit.fl$scale
      [1, 2],
                          fit.fl.cop@estimate[1])
}
```

Figure 3.16 confirms that all methods give reasonable estimates and that all methods perform similarly but differ significantly in run-time. Furthermore, the function `fitCopula()` from `copula` is substantially slower than the "Full-MLE" method, while giving almost identical results. The "EM-MLE" method is not much slower than "Moment-MLE".

3.8.3 Fitting the grouped t copula

Moving from a t copula to a grouped t copula means moving from a model with known density to a model where the density needs to be estimated. Indeed, given $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n \overset{\text{ind.}}{\sim}$

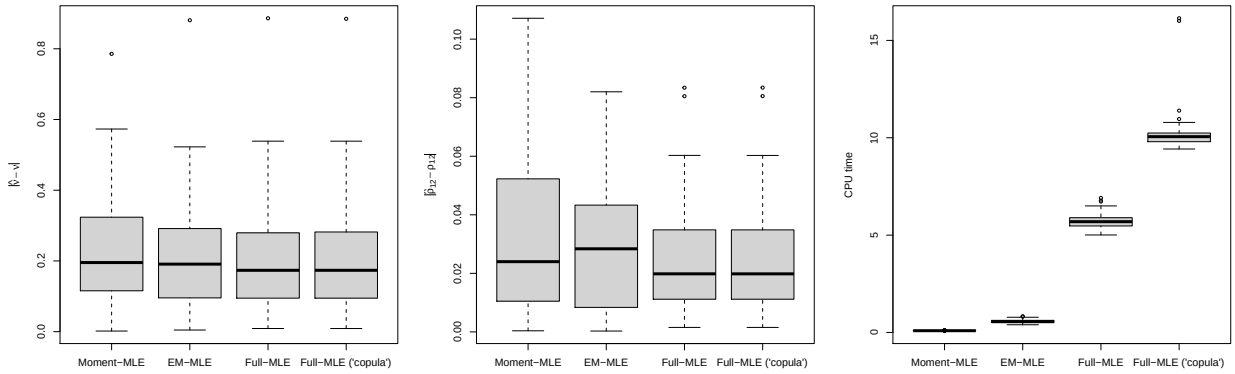

Figure 3.16: Boxplots of absolute errors for the degrees-of-freedom (left) and correlation parameter (middle) and of the run times (right) for a 7-dimensional t copula.

$C^{gt}_{\boldsymbol{\nu},P}$, the copula log-likelihood function is given by

$$\log L(\nu, P; \boldsymbol{U}_1, \dots, \boldsymbol{U}_n) = \sum_{i=1}^{n} \log f(\boldsymbol{X}_i; \boldsymbol{\nu}, P) - \sum_{i=1}^{n} \sum_{j=1}^{d} \log f_j(\boldsymbol{X}_{ij}; \nu), \tag{3.29}$$

where $X_{ij} = t^{-1}_{\nu_j}(U_{ij})$ for $i = 1, \dots, n$ and $j = 1, \dots, d$, f is the joint density of $\mathrm{gt}_d(\boldsymbol{\nu}, \mathbf{0}, P)$ and the f_j are univariate t_{ν_j} densities.

[21] consider a grouped t copula where each group has size at least 2, so that all subgroups are t copulas. The authors suggest to estimate the degrees-of-freedom separately in each group. [82] consider the grouped t copula with d groups (each group belongs to its own group of size 1) and suggest to jointly estimate the d degrees-of-freedom parameters by maximizing the copula log-likelihood. In both references, the matrix P is estimated by estimating pairwise Kendall's tau and using the approximate identity $\rho^{\tau}_{ij} \approx 2 \arcsin(\rho_{ij})/\pi$ for $i \neq j$. Note that [82] relies on the integration method in [99], which we saw in the previous section deteriorates when $\boldsymbol{x}$ has large components.

With Algorithm 3.7.1 at hand, we can evaluate (3.29). Our method to estimate the grouped t copula parameters $(\boldsymbol{\nu}, P)$ works as follows:

1. Estimate all pairwise Kendall's tau and use the approximate identity $\rho^{\tau}_{ij} \approx 2 \arcsin(\rho_{ij})/\pi$ to form a correlation matrix $\hat{P}$.

2. Find initial parameters $\hat{\nu}_k$ in all subgroups k with $d_k \geq 2$ by maximizing the marginal t-copula log-likelihoods. For groups with $d_k = 1$, choose the initial estimate from prior/expert experience.

3. With initial estimates $\hat{\nu}_k$, $k = 1, \dots, S$, where S is the number of groups, maximize the copula log-likelihood $\log L$ from (3.29) over $(\nu_1, \dots, \nu_S)$ numerically, where the joint density f of $\mathrm{gt}_d(\boldsymbol{\nu}, \mathbf{0}, P)$ is computed using Algorithm 3.7.1.

[21] stop after the second step, which means that their procedure fails to consider the dependence between the groups correctly. We demonstrate this in the following simulation using the function `fitgStudentcopula()` where we simulate, for each sample size, 10 realizations of the estimators in [21] (initial estimates) and the estimates produced by our method (MLEs). Note that, in order to control for the effect of an estimated scale matrix, we suppress its estimation by supplying it as argument `scale`.

```
ns <- c(50, 250, 500, 750, 1000) # sample sizes
reps <- 10 # number of repetitions for each sample size
d <- 4 # dimension
df <- c(3, 8) # degrees-of-freedom for each group
grp <- rep(1:2, each = 2) # 2 components in each group
set.seed(1)
scale <- cov2cor(rWishart(1, d, diag(d))[,,1]) # same known scale for all reps
fit.res <- array(, dim = c(length(ns), reps, length(df), 2),
                 dimnames = list(n = ns, rep = 1:reps, df = c("df1", "df2"),
                                 est = c("init", "MLE")))
for(j in 1:reps) {
   set.seed(j)
   sample <- rgStudentcopula(max(ns), groupings = groupings, scale = scale, df =
       df)
   for(i in seq_along(ns)){
      fit <- fitgStudentcopula(u = sample[1:ns[i], ], groupings = grp,
                               scale = scale, verbose = FALSE)
      fit.res[i, j, , "init"] <- fit$df.init
      fit.res[i, j, , "MLE"]  <- fit$df
   }
}
```

Figure 3.17 displays initial estimates on the left and MLEs on the right. As can be seen from Figure 3.17, maximization of the copula log-likelihood jointly over all degrees-of-freedom parameters improves the precision. For instance, even when $n = 1000$, the initial estimates for `df2` are much more fluctuating than the MLEs for `df2`. The price to pay is a substantially longer run time, as the underlying procedure optimizes an estimated log-likelihood.

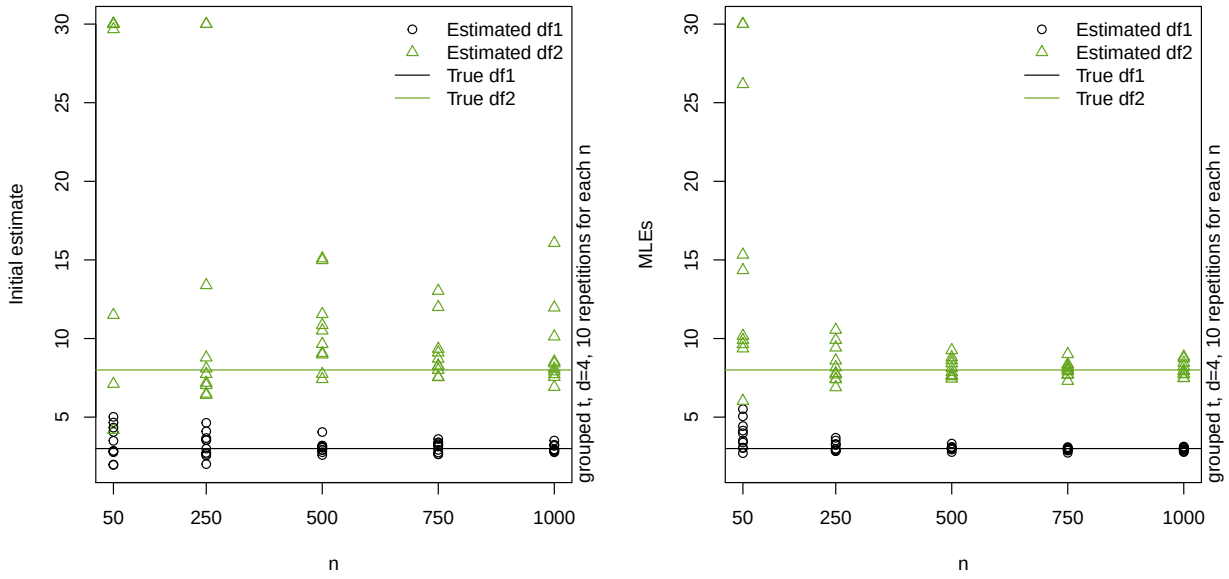

Figure 3.17: Initial estimates (left) and MLEs (right) for the degrees-of-freedom parameters of a grouped t copula with 2 groups.

3.9 Discussion

We introduced efficient algorithms to tackle important tasks for multivariate normal variance mixtures, such as estimating the distribution and log-density function as well as parameter estimation. Furthermore, we extended the algorithms to work for grouped normal variance mixtures, and finally also addressed the problem of fitting t and grouped t copulas to data. Due to the importance of multivariate normal variance mixtures for disciplines such as actuarial science or quantitative risk management, these algorithms along with the provided software are also widely applicable in practice. The results also exemplify the superiority of RQMC methods even in very high dimensions over MC methods for this class of problem.

A possible limitation of our methods is the assumption of a computationally tractable quantile function of the mixing variable W. For more complicated distributions such quantile function may not be available so that an avenue for future research could be to modify our methods so that they work with a non-uniform random variate generator (NRVG) for W (for instance, based on acceptance-rejection (AR) algorithms). While sampling and estimating the distribution function is possible when instead of the quantile function of W a NRVG for W is provided, this is not the case for estimating the log-

density (and thus for the fitting procedure) as our methods are adaptive and thus require sampling in certain low-probability subregions of the support of W. We take up the issue of combining NRVGs, including AR methods in Chapter 4.

The proposed "EM-MLE" method can be applied to a range of implicit copulas well beyond the t-copula. All we need is a decomposition of the log-likelihood as in (3.28) and that the first term involving the joint parameter matrix P can be maximized easily. This is the case for normal variance mixture copulas, but also for more complicated models, such as the skew-t copula. We plan on applying our "EM-MLE" method in the skew-t case in future research; note that substantial complication arises when introducing moving the skew-t copula, as there is no easy way to evaluate the quantile function of the univariate skew-t distribution.

Chapter 4

Quasi-random sampling with black box or acceptance-rejection inputs

Consider the problem of estimating the quantity

$$\mu = \mathbb{E}(g(\boldsymbol{Y}, W)) \tag{4.1}$$

where $g : \mathbb{R}^{d+1} \to \mathbb{R}$ is integrable and $\boldsymbol{Y} \sim F_{\boldsymbol{Y}}$ is a d-dimensional random vector independent of the random variable $W \sim F_W$. For instance, if $\boldsymbol{Y}$ is multivariate normal and W follows a Generalized inverse Gaussian (GIG) distribution (see, e.g., [65] for an AR algorithm to sample from GIG distributions), we could be estimating the expected shortfall of a generalized hyperbolic distribution, a normal variance mixture.

We assume that there is an easy way to sample from $F_{\boldsymbol{Y}}$ based on uniforms; e.g., based on the Rosenblatt transform ([106]). That is to say, assume there is a transformation $T_{\boldsymbol{Y}} : (0,1)^{d+k} \to \mathbb{R}^d$ such that $T_{\boldsymbol{Y}}(\boldsymbol{U}) \sim F_{\boldsymbol{Y}}$ for $\boldsymbol{U} \sim \mathrm{U}^{d+k}$ for constant $k \geq 0$; if, e.g., $\boldsymbol{Y} \sim \mathrm{N}_d(\boldsymbol{\mu}, \Sigma)$ then $k = 0$ and the function $T_{\boldsymbol{Y}}(\boldsymbol{u})$ is given by $T_{\boldsymbol{Y}}(\boldsymbol{u}) = \boldsymbol{\mu} + A(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))^\top$ where A is such that $AA^\top = \Sigma$.

In this section, we investigate how one can construct a RQMC estimator for μ when W cannot be sampled by inversion. More precisely, we assume that the (always existing) quantile function $F_W^{\leftarrow}(u) = \inf\{x : F_W(x) \geq u\}$ is intractable and instead we rely on other methods for non-uniform random variate generation (NRVG), such as AR algorithms, where at first glance it may seem hard to directly apply RQMC methods.

We investigate the above question under two sets of assumptions on what we mean by the existence of a "NRVG" method for W.

1. *Black-box case.* Here, we assume that we have a (random) function $R_W : \mathrm{N} \to \mathsf{R}^n$ such that if $R_W(n) = \boldsymbol{W}$ for $\boldsymbol{W} = (W_1, \ldots, W_n)$ then $W_i \overset{\text{ind.}}{\sim} F_W$ for $i = 1, \ldots, n$. As such, we have a "black box" function that returns samples from F_W of any size. The underlying sampling method could be based on MCMC, machine learning techniques or methods based on a stochastic representation (SR), among others. In Section 4.1, we propose methods that estimate the quantile function $F_W^{\leftarrow}$, as well as re-ordering strategies that make the output of R_W mimick the behavior of the underlying LDS. We also perform a numerical study comparing our methods. We highlight the assumption that we have only access to R_W, irrespective of whether or not W admits a tractable density. If W does have a density that can be efficiently computed, other methods that approximate the quantile function using this additional information may be better suited; see [25] for a popular method. There are, however, examples where this is not the case: if W follows a stable distribution, sampling is easy based on the stochastic representation derived in [12], but not even the density function can be computed without numerical integration.

2. *AR algorithms for W,* where the proposal (or envelope) distribution and the acceptance decision can be sampled by inversion of uniforms. The main difference to the black-box setting is that here, we do have access to the underlying sampling mechanism and can feed the AR sampler with a randomized low-discrepancy sequence (LDS). AR algorithms are typically not popular in RQMC as it is possibly infinite-dimensional. Smoothed rejection and weighted uniform sampling is considered in [87], along with numerical results showing that these outperform AR sampling in terms of convergence speed. It is shown in [116] that the F discrepancy, i.e., $\sup_x |F_n(x) - F(x)|$, where F_n and F denote the empirical and theoretical distribution function, of a sample obtained via AR is in $\mathcal{O}(n^{-\alpha})$ for $1/2 \leq \alpha < 1$. The error convergence rate is improved by replacing the purely binary AR decision with weights, called extended smoothed rejection. This circumvents integration of an indicator function. Discrepancy properties of points produced by totally deterministic AR methods, i.e., AR with a (non-randomized) Sobol' sequence are derived in [122]. A convergence result, error bounds and a numerical study for AR with RQMC is given in [91]. What all previous references have in common is that they hold the dimension of the LDS constant and effectively use a subset of size n of the first N points in the sequence. We investigate, among other things, whether there is a difference between holding d constant (and thereby skipping points in the sequence) or holding n constant (thereby thinking of the first n points having potentially unbounded dimension). This is the topic of Section 4.2.

To be clear, AR could even be an algorithm used within the black-box setting, but given its prevalence, we choose to treat AR separately. We revisit this point at the end of Section 4.2, where we combine ideas from both settings.

Section 4.3 applies the methods presented in Sections 4.1 and 4.2 to the problem of estimating the price of a basket call option under a normal variance mixture copula dependence. As mixing distributions, we use the inverse-gamma distribution (as its known quantile function can be used as a benchmark) and the GIG distribution. In the latter case, we also include the method based on numerical inversion of the density in [25], which was shown to be efficient for the GIG in [77]. We perform the same experiment with the aforementioned stable mixture, a model where the method in [25] cannot be easily applied for sampling due to the lack of a tractable density. This section is based on [56].

4.1 Methods for the black box setting

Recall that the classical MC estimator $\hat{\mu}_n^{\mathrm{MC}}$ based on n samples for (4.1) can be written as

$$\hat{\mu}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{U}_i), W_i), \tag{4.2}$$

where $\boldsymbol{U}_i \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^{d+k}$ is independent of $W_1, \dots, W_n \overset{\text{ind.}}{\sim} F_W$ obtained by calling $R_W(n)$. To simplify the notation, we henceforth assume $k = 0$; the case $k > 0$ is handled by replacing d by $d + k$ in what follows. In order to be able to apply RQMC to the problem, we first rewrite (4.1) as an integral over the unit hypercube. With a change of variable, we obtain

$$\mu = \int_{(0,1)^{d+1}} g(T_{\boldsymbol{Y}}(u_{1:d}), Q(u_{d+1}))\,\mathrm{d}\boldsymbol{u}, \tag{4.3}$$

where $\boldsymbol{u} = (u_{1:d}, u_{d+1})$ with $u_{1:d} = (u_1, \dots, u_d)$, and we use the function $Q : [0,1] \to \mathbb{R}$ as a shorthand notation for the quantile function $F_W^{\leftarrow}$ for the remainder of this section.

If we were able to sample W via inversion, then RQMC sampling could be used to estimate μ using the following approach: Let $\tilde{P}_{b,n} = \{\boldsymbol{u}_{b,1}, \dots, \boldsymbol{u}_{b,n}\} \subseteq [0,1)^{d+1}$, where $\boldsymbol{u}_{b,i} = (u_{b,i,1}, \dots, u_{b,i,d+1})$ for $b = 1, \dots, B$, denote B independent randomizations of the first n points of the low-discrepancy sequence (LDS) used; here we assume that the randomization is such that each $\boldsymbol{u}_{b,i} \sim U(0,1)^{d+1}$. Then

$$\hat{\mu}_{b,n}^{\mathrm{RQMC}} = \frac{1}{n}\sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), Q(u_{b,i,d+1})), \quad b = 1, \dots, B, \tag{4.4}$$

and an RQMC estimator for μ based on a total of nB points would be given by

$$\hat{\mu}_{B,n}^{\text{RQMC}} = \frac{1}{B}\sum_{b=1}^{B} \hat{\mu}_{b,n}^{\text{RQMC}}.$$

The variance/error of $\hat{\mu}_{B,n}^{\text{RQMC}}$ could then be estimated in the usual way.

However, we do not know Q, so the estimators $\hat{\mu}_{b,n}^{\text{RQMC}}$ in (4.4) cannot be computed. In this section, we propose two different methods to *approximate* $\hat{\mu}_{b,n}^{\text{RQMC}}$ for $b = 1, \ldots, B$. Both methods essentially replace Q by an estimate thereof.

4.1.1 Methods based on the empirical quantile function

A simple ad-hoc method to approximate $\hat{\mu}_{b,n}^{\text{RQMC}}$ could be to replace the Q values by a random sample of F_W obtained by calling $R_W(Bn)$. More precisely, let $W_{b,i}$ for $b = 1, \ldots, B$, $i = 1, \ldots, n$, denote the Bn iid samples from F_W obtained by calling $R_W(Bn)$. Replacing $Q(u_{b,i,d+1})$ by $W_{b,i}$ for $b = 1, \ldots, B, i = 1, \ldots, n$, is then equivalent to replacing the last coordinate of the n points in $\tilde{P}_{b,n}$ by independent $\text{U}(0,1)$ variates. With $W_{b,i} = Q(U_{b,i})$ where $U_{b,i} \overset{\text{ind.}}{\sim} \text{U}(0,1)$, $b = 1, \ldots, B$, $i = 1, \ldots, n$, $b = 1, \ldots, B$, we can write

$$\hat{\mu}_{b,n}^{\text{mc-rqmc}} = \frac{1}{n}\sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), Q(U_{b,i}))), \quad b = 1, \ldots, B. \tag{4.5}$$

From the inverse probability integral transform (see, e.g., [26, Theorem 2.1]), we know that $Q(U)$ for $U \sim \text{U}(0,1)$ and $R_n(1)$ have the same distribution, namely F_W. As such, unbiasedness of $\hat{\mu}_{b,n}^{\text{mc-rqmc}}$ (and therefore of $(1/B)\sum_{b=1}^{B} \hat{\mu}_{b,n}^{\text{mc-rqmc}}$) for μ follows immediately.

Note that only the first d coordinates of $\tilde{P}_{b,n}$ enter the estimation, so that the good projection properties of coordinate $d+1$ (and its interactions) are lost. Loosely speaking, the last coordinate of the point set we are effectively using to integrate the function g is unrelated with the first d. A better approach is to use the sampled $W_{b,i}$ to construct B empirical quantile functions $\hat{Q}_{n,b}$, $b = 1, \ldots, B$, and replace $Q(U_{b,i})$ by $\hat{Q}_{n,b}(u_{b,i,d+1}) = W_{b,(\lceil nu_{b,i,d+1} \rceil)}$, where, for $b = 1, \ldots, B$, we denote by $W_{b,(i)}$, $i = 1, \ldots, n$, the order statistics of $W_{b,1}, \ldots, W_{b,n}$, so $W_{b,(1)} \leq \cdots \leq W_{b,(n)}$. We define

$$\hat{\mu}_{b,n}^{\text{b-eqf}} = \frac{1}{n}\sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), W_{b,(\lceil nu_{b,i,d+1} \rceil)}), \quad b = 1, \ldots, B,$$

where superscript "b-eqf" indicates that in each randomization b, the empirical quantile function obtained in that randomization based on n samples is used (instead of Q). That is, in each of the B randomizations (each of which requires n function evaluations), estimate Q by its empirical quantile function $\hat{Q}_{n,b}$ obtained from n independent samples from F_W via a call to the black-box function $R_W(n)$.

Note that as long as $\tilde{P}_{b,n,d+1} = \{u_{b,i,d+1} : i = 1, \ldots, n\}$ is *properly stratified*, i.e., has exactly one point in each interval of the form $[j/n, (j+1)/n)$ for $j \in \{0, \ldots, n-1\}$, each $W_{b,i}$, $i = 1, \ldots, n$ will be sampled exactly once when using $\tilde{P}_{b,n,d+1}$ to sample the empirical quantile function $\hat{Q}_{b,n}$. Hence an alternative way to describe the estimator $\hat{\mu}_{b,n}^{\text{b-eqf}}$ that is useful from an implementation perspective is to realize that if the last coordinate of a given point $\boldsymbol{u}_{b,i}$ is the jth smallest value among those n last coordinates, we "stitch" $W_{b,(j)}$ to that ith point. Hence the last coordinate of $\tilde{P}_{b,n}$ is used to order the sample $W_{b,1}, \ldots, W_{b,n}$. Also note that if $\tilde{P}_{b,n}$ is a digitally shifted or scrambled Sobol' point set with $n = b^k$ points or a randomly shifted rank-1 lattice, then $\tilde{P}_{b,n,d+1}$ is properly stratified; see [75].

The estimators $\hat{\mu}_{b,n}^{\text{b-eqf}}$ for $b = 1, \ldots, B$ are independent and as long as $\tilde{P}_{b,n,d+1}$ is properly stratified, they are also unbiased, see Proposition 4.1.1.

This alternative description gives rise to a slightly different estimator: Let $r^n(u_{b,i,d+1})$ be the rank of $u_{b,i,d+1}$ among $u_{b,1,d+1}, \ldots, u_{b,n,d+1}$. We then define the rank-based estimator as

$$\hat{\mu}_{b,n}^{\text{b-rk}} = \frac{1}{n} \sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), W_{b,(r^n(u_{b,i,d+1}))}), \quad b = 1, \ldots, B. \tag{4.6}$$

If $\tilde{P}_{b,n,d+1}$ is properly stratified, then $\hat{\mu}_{b,n}^{\text{b-rk}}$ and $\hat{\mu}_{b,n}^{\text{b-eqf}}$ coincide, and each sample $W_{b,i}$ is used exactly once. Otherwise, unlike $\hat{\mu}_{b,n}^{\text{b-eqf}}$, $\hat{\mu}_{b,n}^{\text{b-rk}}$ still uses every $W_{b,i}$ exactly once.

Proposition 4.1.1
Let $b \in \{1, \ldots, B\}$ and let $\tilde{P}_{b,n,d+1}$ be properly stratified. Then $\hat{\mu}_{b,n}^{\text{b-rk}}$ (and therefore $\hat{\mu}_{b,n}^{\text{b-eqf}}$) is unbiased for μ.

Proof. Let $i \in \{1, \ldots, n\}$. We show that $\mathbb{E}(g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), W_{b,(r^n(u_{b,i,d+1})),n})) = \mu$. By definition, $(u_{b,i,1}, \ldots, u_{b,i,d+1}) \sim \mathrm{U}(0,1)^{d+1}$, in particular, $\boldsymbol{Y} := T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}) \sim F_{\boldsymbol{Y}}$ is independent of $u_{b,i,d+1}$. Let $r^n(u_{b,i,d+1}) = K(i)$ (a random variable) and note that $(K(1), \ldots, K(n))$ is a permutation of $(1, \ldots, n)$ chosen according to some distribution (which may not be uniform because of the low-discrepancy properties of $\tilde{P}_{b,n}$). Then $W_{b,K(i)}$ is an element chosen from the list $W_{b,1}, \ldots, W_{b,n}$ according to some distribution, and the latter is an independent

random sample from F_W. Hence, $W_{b,K(i)}$ and $\boldsymbol{Y}$ are independent, $(\boldsymbol{Y}, W_{b,K(i)}) \sim F_{\boldsymbol{Y}} \times F_W$ and the main claim follows by linearity of the expectation. □

The previous methods can be thought of as approximating the quantile function B times, each based on n samples obtained from the black box. In order to base our simulation on a sampling mechanism closer to inversion and thereby mimicking more closely the estimator in (4.4), we could instead construct a single rank-based quantile function estimator based on the Bn outputs $W_{b,i}, b = 1, \dots, B, i = 1, \dots, n$. That is, instead of reordering the n samples $W_{b,i}$, $i = 1, \dots, n$ according to $u_{b,i,d+1}$ in each randomization $b = 1, \dots, B$, separately, we reorder the Bn realizations $W_{b,i}$, $i = 1, \dots, n$, $b = 1, \dots, B$, according to the ranks of the $u_{b,i,d+1}$. That is, we construct the estimator

$$\hat{\mu}_{b,n}^{\text{1:B-rk}} = \frac{1}{n} \sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), W_{(r^{Bn}(u_{b,i,1}))}), \quad b = 1, \dots, B, \tag{4.7}$$

where $r^{Bn}(u_{b,i,d+1}) = k$ if $u_{b,i,d+1}$ is the kth smallest among the Bn uniforms $u_{1,1,d+1}, \dots, u_{1,n,d+1}$, $\dots, u_{B,1,d+1}, \dots, u_{B,n,d+1}$.

We can replace the ranks by the empirical quantile function computed from $R_W(Bn)$, and obtain as an analog of $\hat{\mu}^{\text{b-eqf}}$ the estimator

$$\hat{\mu}_{b,n}^{\text{1:B-eqf}} = \frac{1}{n} \sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), W_{(\lceil nBu_{b,i,1} \rceil)}), \quad b = 1, \dots, B,$$

for a sample $W_1, \dots, W_{nB} \overset{\text{ind.}}{\sim} F_W$ obtained by calling $R_W(Bn)$. The superscript "1:B-eqf" shall indicate that in all randomizations $1, \dots, B$, the same quantile function estimator is used. Note that $\hat{\mu}_{b,n}^{\text{1:B-eqf}}$ are not independent anymore for $b = 1, \dots, B$, the same applies to $\hat{\mu}_{b,n}^{\text{1:B-rk}}$.

4.1.2 Methods based on a generalized Pareto approximation in the tail

The methods presented in the previous section are purely nonparametric and amount to replacing the true quantile function Q by an empirical estimate thereof. Empirical quantile functions typically estimate quantiles away from the tail with reasonable accuracy; this does not hold for the tails if W is unbounded. However, approximating the tail of Q well is crucial for an effective RQMC procedure to outperform MC.

In the following, assume that W is supported on $[0, \infty)$ so that only the upper tail needs to be estimated. Since this is typically the case in practice, this is a rather weak assumption. If W is instead supported on $\mathbb{R}$, the methods described here can be applied to the positive and negative real line separately.

The main idea behind the methods presented in this section is the following: Given a random sample from F_W, estimate Q in the body (say, for $u \in (0, 0.9)$) by interpolation of the empirical quantile function and in the (right) tail based on a fitted generalized Pareto Distribution (GPD), which has a cumulative distribution function (cdf)

$$G_{\xi,\beta}(x) = \begin{cases} 1 - \left(1 + \frac{\xi x}{\beta}\right)^{-\frac{1}{\xi}}, & \text{if } \xi \neq 0, \\ 1 - \exp\left(-\frac{x}{\beta}\right), & \text{if } \xi = 0, \end{cases}$$

where $\beta > 0$ and the support is $x \in [0, \infty)$ when $\xi \geq 0$ and $x \in [0, -\beta/\xi]$ when $\xi < 0$.

Let F be any cdf and let $X \sim F$. Denote by $F_u(x) = \mathbb{P}(X - u \leq x \mid X > u)$ the excess distribution over the threshold u. Under weak assumptions, the Pickands-Balkema-de-Haan Theorem (see [29, Theorem 3.4.13]) implies that for large enough u one can approximate F_u by $G_{\xi,\beta}$.

In practice, ξ and β are estimated from given data. With estimates of ξ, μ at hand, we can compute $G^{-1}_{\xi,\beta}$ analytically, which, appropriately scaled, provides us with an estimate of F^{-1}. In what follows, assume F_W fulfills the assumptions underlying the Pickands-Balkema-de-Haan Theorem, and denote by $g_{\xi,\beta}$ the density of $G_{\xi,\beta}$. The following algorithm returns a quantile function estimator $\hat{Q}$ of Q.

Algorithm 4.1.2
Given $W_1, \ldots, W_{n'} \overset{\text{ind.}}{\sim} F_W$ and $\alpha \in (0, 1)$, construct an estimator $\hat{Q}$ for Q as follows:

1. Denote by $W_{(1)}, \ldots, W_{(n')}$ the order statistics of $W_1, \ldots, W_{n'}$.
2. Let $T = W_{(\lceil n'\alpha \rceil)}$ and denote by $N = |\{i \in \{1, \ldots, n'\} : W_i > T\}|$ the number of exceedances over T. Let $\tilde{W}_i = W_{(\lceil n'\alpha \rceil + i)} - T$ for $i = 1, \ldots, N$ be the excesses. Then maximize the log-likelihood function

$$l(\xi, \beta; \tilde{W}_1, \ldots, \tilde{W}_N) = \sum_{k=1}^{N} \log g_{\xi,\beta}(\tilde{W}_k)$$

with respect to ξ and β numerically over their ranges to obtain the MLEs $\hat{\xi}$ and $\hat{\beta}$.

3. Return the function

$$\hat{Q}(u) = \begin{cases} (1-\kappa)W_{(\lfloor (n'+1)u \rfloor)} + \kappa W_{(\lfloor (n'+1)u \rfloor + 1)}, & \text{if } u \le \alpha, \\ T + \frac{\hat{\beta}}{\hat{\xi}} \left(\left(\frac{1-u}{1-\alpha} \right)^{-\xi} - 1 \right), & \text{otherwise,} \end{cases}$$

where $\kappa = (n'+1)u - \lfloor (n'+1)u \rfloor$.

Algorithm 4.1.2 does not give any error estimates, nor do we have an a-priori guess of how large n should be. In order to obtain error estimates, one could use Algorithm 4.1.2 to obtain M independent estimators $\hat{Q}_m$, $m = 1, \ldots, M$, and estimate the error using a CLT argument. That is, the (absolute) error of $\hat{Q}(u) = (1/M)\sum_{m=1}^{M} \hat{Q}_m(u)$ for some fixed $u \in (0,1)$ may be estimated via $3.5/\sqrt{M} \times \hat{\sigma}$, where $\hat{\sigma} = \mathrm{sd}(\hat{Q}_1(u), \ldots, \hat{Q}_M(u))$. As $\hat{Q}_m(u) - Q(u)$ follows approximately a $\mathrm{N}(0, \hat{\sigma}^2/M)$ distribution, we can be 99.95% confident that the error is within $\pm 3.5/\sqrt{M} \times \hat{\sigma}$.

With an error estimation procedure at hand, one can now construct the quantile function iteratively until a pre-specified error tolerance for the estimated absolute error is met. That is, one can specify knots $u'_1, \ldots, u'_N \in (0,1)$ and error tolerances $\varepsilon_1, \ldots, \varepsilon_N > 0$ and construct the quantile function with more and more points until the error tolerance at all knots is met. The choice of knots and error tolerances can be guided from the function g so that important subdomains have little error, or one can put most of the knots uniformly between 0 and 1 and the remaining ones in the tails. The main idea is summarized in the following algorithm.

Algorithm 4.1.3
Given $n_0 \in \mathrm{N}$, $\alpha \in (0,1)$, NRVG R_W, knots $u'_1, \ldots, u'_N \in (0,1)$, error tolerances $\varepsilon_1, \ldots, \varepsilon_N > 0$, maximum number of iterations $i_{\max}$, $B \in \mathbb{N}$, construct an estimator $\hat{Q}$ for Q as follows:

1. Set $i = 1$, and $S_b = \{\}$ for $k = 1, \ldots, B$.
2. Repeat
   (a) For $b = 1, \ldots, B$,
      i. Set $S_b = S_b \cup \{R_W(n_0)\}$.
      ii. Call Algorithm 4.1.2 with input sample S_b to construct an estimated quantile function $\hat{Q}_b$.
   (b) For $k = 1, \ldots, N$ set $e_k = 3.5/\sqrt{B} \times \mathrm{sd}(\hat{Q}_1(u'_k), \ldots, \hat{Q}_B(u'_k))$ as the estimated error at knot u'_k.

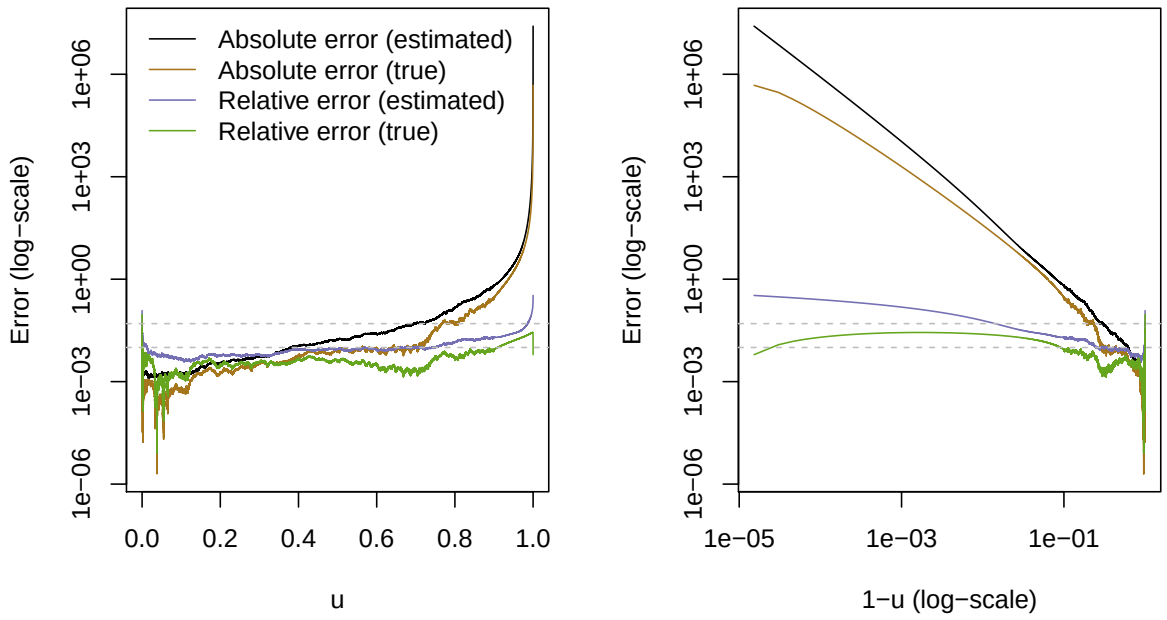

Figure 4.1: Estimated and realized absolute and relative errors when estimating the quantile function of IG$(1.2, 1.2)$ using Algorithm 4.1.3 with $n_0 = 7500$, $B = 20$.

(c) Set $i = i + 1$.

Until $e_k \leq \varepsilon_k$ for $k = 1, \ldots, N$ or $i > i_{\max}$.

3. Return the estimated quantile function $\hat{Q}^{\text{eqf-gpd}}(u) = (1/B) \sum_{b=1}^{B} \hat{Q}_b(u)$.

The input argument $i_{\max}$ determines the maximum number of iterations allowed in case convergence cannot be achieved. Note that the superscript "eqf-gpd" shall indicate that the (interpolated) empirical quantile function is used in the body and a GPD approximation in the tail. For an implementation, in any iteration $i > 1$, results from the previous iterations should be reused; for instance, the MLE $(\hat{\xi}, \hat{\beta})$ from a previous iteration can be used as a starting value for the maximization of the log-likelihood function in the next iteration. In practice one could also return the $\hat{Q}_b$, $b = 1, \ldots, B$, so that for any $u \in (0, 1)$ one can compute $\hat{Q}(u)$ along with an error estimate.

Given an estimated quantile function, say $\hat{Q}^{\text{eqf-gpd}}$, an RQMC estimator for μ from (4.1) is given by

$$\hat{\mu}_{B,n}^{\text{eqf-gpd}} = \frac{1}{B}\sum_{b=1}^{B} \hat{\mu}_{b,n}^{\text{eqf-gpd}}, \tag{4.8}$$

where

$$\hat{\mu}_{b,n}^{\text{eqf-gpd}} = \frac{1}{n}\sum_{i=1}^{n} g(T_{\boldsymbol{Y}}(\boldsymbol{u}_{b,i,1:d}), \hat{Q}^{\text{eqf-gpd}}(u_{b,i,d+1})), \quad b = 1, \dots, B,$$

and the inputs $u_{b,i,d+1}$ and $\boldsymbol{u}_{b,i,1:d}$ are as in the previous section. In contrast to the estimators from Section 4.1.1, computing this estimator requires a two-stage procedure: First, Algorithm 4.1.3 needs to be applied to compute the estimated quantile function $\hat{Q}^{\text{eqf-gpd}}$, which will then, in the second stage, be treated as the "true quantile function" when computing the estimator $\hat{\mu}^{\text{eqf-gpd}}$.

Example 4.1.4 (Inverse-gamma example)
Consider $W \sim \text{IG}(1.2, 1)$. We use $n_0 = 7500$, $B = 20$, and uniform knots between 0.01 and 0.95 with relative error tolerance 0.025, one knot at 0.99 with relative error tolerance 0.075 and and another knot at 0.999 with relative error tolerance 0.1. The algorithm needed 10 iterations until convergence, so a total of 1 350 000 realizations of W. The approximation is very accurate and the true quantile lies within the approximated error bounds. This can be seen from Figure 4.1, which displays realized and estimated absolute and relative errors.

Example 4.1.5 (Expected shortfall of portfolio under a multivariate t distribution)
The multivariate t distribution is a normal variance mixture distribution and falls into the general framework of this section, if we assume that the quantile function of an inverse-gamma distribution is not available. We do this to compare our methods with the "best possible" estimator from (4.4). Let $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\Sigma = AA^\top$ for some covariance matrix Σ. Recall that $\boldsymbol{X} \sim t_d(\nu, \boldsymbol{\mu}, \Sigma)$ has stochastic representation

$$\boldsymbol{X} = \boldsymbol{\mu} + \sqrt{W}\boldsymbol{Y}, \tag{4.9}$$

where $W \sim \text{IG}(\nu/2, \nu/2)$ independent of $\boldsymbol{Y} \sim \text{N}_d(\boldsymbol{0}, \Sigma)$. For a continuous random variable $L \sim F$ with $\mathbb{E}(|L|) < \infty$ and level $\alpha \in (0, 1)$ small, expected shortfall is the mean conditional loss $\text{ES}_\alpha(L) = \mathbb{E}(L \mid L > F_L^{-1}(\alpha))$. In our simulation, we assume that $L = \boldsymbol{1}^\top \boldsymbol{X}$ where $\boldsymbol{X} \sim t_d(\nu, \boldsymbol{0}, \Sigma)$; it follows from the closedness of normal variance mixtures that $L \sim t_1(\nu, 0, \boldsymbol{1}^\top \Sigma \boldsymbol{1})$. The value of $\mu = \text{ES}_\alpha(L) := \mathbb{E}(g(\boldsymbol{Y}, W))$ is known in closed-form; see [85, Example 2.15]. This allows us to estimate the Mean squared error (MSE) and compare

it with the variance. For a range of values of the total number of function evaluations, we report in Figure 4.2 the MSE and variance for various methods, each estimated by using $M = 50$ independent copies of the estimators, each of which is based on $B = 20$ repetitions. Here and in what follows, we use a digitally shifted Sobol' sequence as implemented in the R package `qrng`; see [62]. All RQMC based estimators, including MC-RQMC from (4.5), outperform MC, though MC-RQMC gives only a moderate variance reduction. This is in contrast to b-rk, which for small n gives MSE similar to inversion, which we recall would not be available in a realistic setting where Q is unknown.

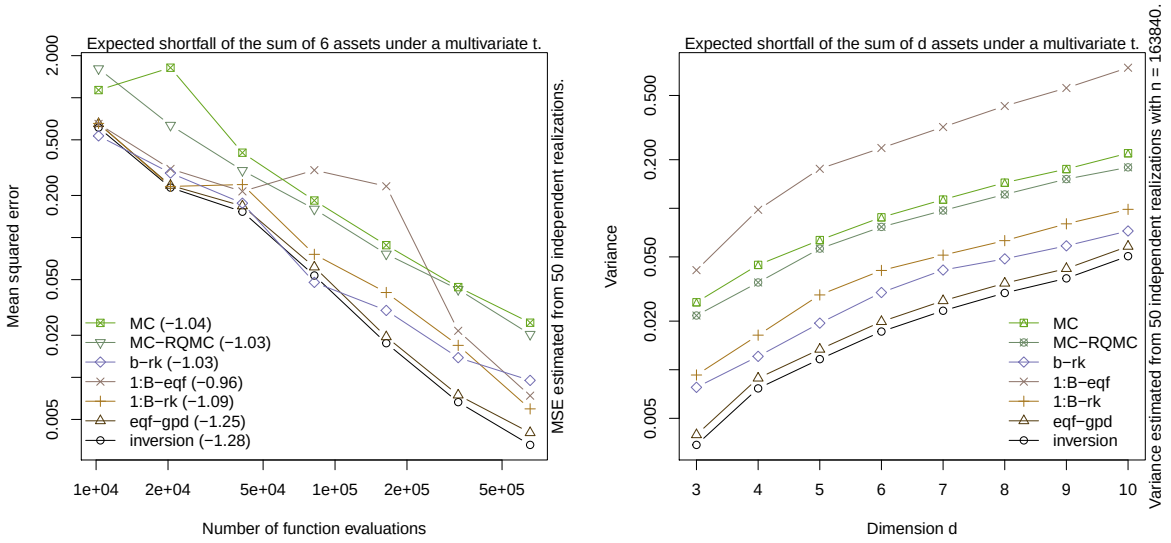

Figure 4.2: Mean squared errors as a function of n (left) and variances as a function of d (right) when estimating $\mathrm{ES}_{0.95}(L)$ for $L = \mathbf{1}^\top \boldsymbol{X}$ where $\boldsymbol{X} \sim t_d(\nu, \mathbf{0}, \Sigma)$.

4.2 Combining AR with RQMC

Rather than working with a "black-box" NRVG R_W, we assume in this section that W can be sampled using AR and explore how we can apply RQMC in this setting. Recall from (4.3) that we are interested in estimating $\mu = \mathbb{E}(g(\boldsymbol{Y}, W))$, so we need n samples $(\boldsymbol{Y}_i, W_i)$ where $W_i \sim F_W$. When using AR, there is no a-priori bound on how many uniforms are needed, so we have an a priori infinite-dimensional integration problem: If T_{AR} denotes

i	Sample $\boldsymbol{Y}$				F^{-1}	AR
1	$u_{1,1}$	$u_{1,2}$	$\dots$	$u_{1,d}$	$u_{1,d+1}$	$u_{1,d+2}$
2	$u_{2,1}$	$u_{2,2}$	$\dots$	$u_{2,d}$	$u_{2,d+1}$	$u_{2,d+2}$
3	$u_{3,1}$	$u_{3,2}$	$\dots$	$u_{3,d}$	$u_{3,d+1}$	$u_{3,d+2}$
4	$u_{4,1}$	$u_{4,2}$	$\dots$	$u_{4,d}$	$u_{4,d+1}$	$u_{4,d+2}$
5	$u_{5,1}$	$u_{5,2}$	$\dots$	$u_{5,d}$	$u_{5,d+1}$	$u_{5,d+2}$
6	$u_{6,1}$	$u_{6,2}$	$\dots$	$u_{6,d}$	$u_{6,d+1}$	$u_{6,d+2}$
$\vdots$	$\vdots$				$\vdots$	$\vdots$

Figure 4.3: Schematic description of AR-n. Gray coordinates in the same row correspond to rejected coordinates.

i	Sample $\boldsymbol{Y}$				F^{-1}	AR	F^{-1}	AR	F^{-1}	AR	$\dots$
1	$u_{1,1}$	$u_{1,2}$	$\dots$	$u_{1,d}$	$u_{1,d+1}$	$u_{1,d+2}$	$u_{1,d+3}$	$u_{1,d+4}$	$u_{1,d+5}$	$u_{1,d+6}$	
2	$u_{2,1}$	$u_{2,2}$	$\dots$	$u_{2,d}$	$u_{2,d+1}$	$u_{2,d+2}$					
3	$u_{3,1}$	$u_{3,2}$	$\dots$	$u_{3,d}$	$u_{3,d+1}$	$u_{3,d+2}$					
4	$u_{4,1}$	$u_{4,2}$	$\dots$	$u_{4,d}$	$u_{4,d+1}$	$u_{4,d+2}$	$u_{4,d+3}$	$u_{4,d+4}$			

i	Sample $\boldsymbol{Y}$				F^{-1}	F^{-1}	F^{-1}	$\dots$	AR	AR	AR	$\dots$
1	$u_{1,1}$	$u_{1,2}$	$\dots$	$u_{1,d}$	$u_{1,d+1}$				$u_{1,d+M+1}$			
2	$u_{2,1}$	$u_{2,2}$	$\dots$	$u_{2,d}$	$u_{2,d+1}$	$u_{2,d+2}$	$u_{2,d+3}$		$u_{2,d+M+1}$	$u_{2,d+M+2}$	$u_{2,d+M+3}$	
3	$u_{3,1}$	$u_{3,2}$	$\dots$	$u_{3,d}$	$u_{3,d+1}$				$u_{3,d+M+1}$			
4	$u_{4,1}$	$u_{4,2}$	$\dots$	$u_{4,d}$	$u_{4,d+1}$	$u_{4,d+2}$			$u_{4,d+M+1}$	$u_{4,d+M+2}$		

Figure 4.4: Schematic description of AR-d with consecutive (top) and blockwise (bottom) coordinate assignment. Gray coordinates in the same row correspond to rejected coordinates.

the AR transformation, we can write $\mu = \mathbb{E}(h(\boldsymbol{U})) = \mathbb{E}(g(T_{\boldsymbol{Y}}(\boldsymbol{U}_{1:d}), T_{\text{AR}}(\boldsymbol{U}_{(d+1):\infty}))$ with $\boldsymbol{U} \sim \text{U}(0,1)^\infty$ and h appropriately defined. The integrand h is a non-monotone and discontinuous function of its input uniforms, a result from the acceptance decision. This can diminish the variance reduction effect of RQMC over MC.

We assume that W has density f_W over $(a, b) \subseteq \mathbb{R}$, we use the proposal density f having the same support (a, b) with quantile function F^{-1}, and that $c = \sup_{x\in(a,b)} f_W(x)/f(x) < \infty$.

A major difference between the application of RQMC and MC is that with the former, we need to carefully assign which coordinate of the points is used to sample which random

variable, and there is typically more than one way to do so. As in the previous section, we assume that the first d coordinates $\boldsymbol{u}_{1:d}$ of $\boldsymbol{u} \in (0,1)^\infty$ are used to sample from $F_{\boldsymbol{Y}}$. Algorithms 4.2.1 and 4.2.2 describe two AR methods to sample n copies of $(\boldsymbol{Y}, W)$; a schematic description is given in Figures 4.3 and 4.4. The former method, henceforth referred to as AR-n, always uses coordinates $\{d+1, d+2\}$ in the AR part, and moves along the index i. If a point is rejected, just like the point in row $i = 1$ in Figure 4.3, the algorithm tries again with point $i+1$. That is, when sampling n points we move along the index of a randomized LDS with constant dimension $d+2$. In contrast, Algorithm 4.2.2 (AR-d) samples the ith point by moving along the coordinates $\{d+1, d+2, d+3, \dots\}$ of the ith point in the sequence until it is accepted; see the top of Figure 4.4, where we assume that coordinates $d+2j-1$ and $d+2j$ for $j = 1, 2, \dots$ are used for sampling from the proposal and sampling from the AR decision, respectively. Another possibility to assign the coordinates for the AR part is to consider two blocks of size M (chosen so that, with high probability, M trials are sufficient to accept a point), where the coordinates in the first block are used for the sampling in Step 2(a)i and the coordinates in the second block determine the acceptance decision in Step 2(a)ii. This version of AR-d is illustrated at the bottom of Figure 4.4.

Algorithm 4.2.1 (AR-n)
Let $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \} \subset (0,1)^{d+2}$ be a randomized LDS. Sample n copies of $(\boldsymbol{Y}, W)$ as follows.

1. Set $j = 1$, $O_n = \{\}$.
2. For $i = 1, \dots, n$,
   (a) Repeat
      i. Compute $W = F^{-1}(u_{j,d+1})$ and set $U = u_{j,d+2}$.
      ii. If $U > f_W(W)/(cf(W))$ set $j = j+1$ Else
         Set $O_n = O_n \cup \{(T_{\boldsymbol{Y}}(\boldsymbol{u}_{j,1:d}, W)\}$
         Set $j = j+1$ and break;
3. Return O_n.

The main difference between AR-n and AR-d is that in the former approach, points in the sequence are skipped, and, effectively, a subset of size n of the first $N > n$ points in the sequence is used to integrate g, whereas in AR-d we always use the first n points in the sequence and move along the coordinates.

Algorithm 4.2.2 (AR-d)
Let $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\} \subset (0,1)^\infty$ be a randomized low discrepancy point-set. Sample n copies of $(\boldsymbol{Y}, W)$ as follows.

1. Set $O_n = \{\}$.
2. For $i = 1, \ldots, n$,
   (a) For $j = 1, 2, \ldots,$
      i. Compute $W = F^{-1}(u_{i,d+2j-1})$.
      ii. If $u_{i,d+2j} \leq f_W(W)/(cf(W))$:
         A. Set $O_n = O_n \cup \{(T_{\boldsymbol{Y}}(\boldsymbol{u}_{i,1:d}, W)\}$
         B. Break.
3. Return O_n.

A potential advantage of AR-d over AR-n for numerical integration is that it really only uses the first n points of the LDS rather than a subset of the first $N > n$ points in

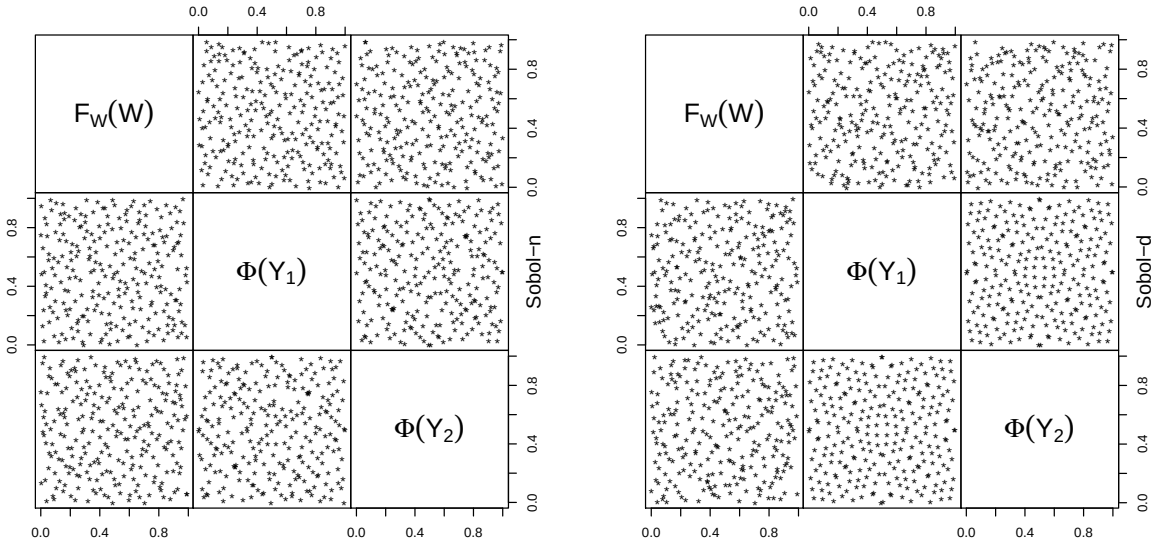

Figure 4.5: Pairs plot of $(F_W(W_i), \Phi(Z_{i1}), \Phi(Z_{i2})) \sim \mathrm{U}(0,1)^3$, where the trivariate points were sampled with AR-n (left) and with AR-d (right) for $W \sim \Gamma(1.2, 1)$ and $n = 2^8$.

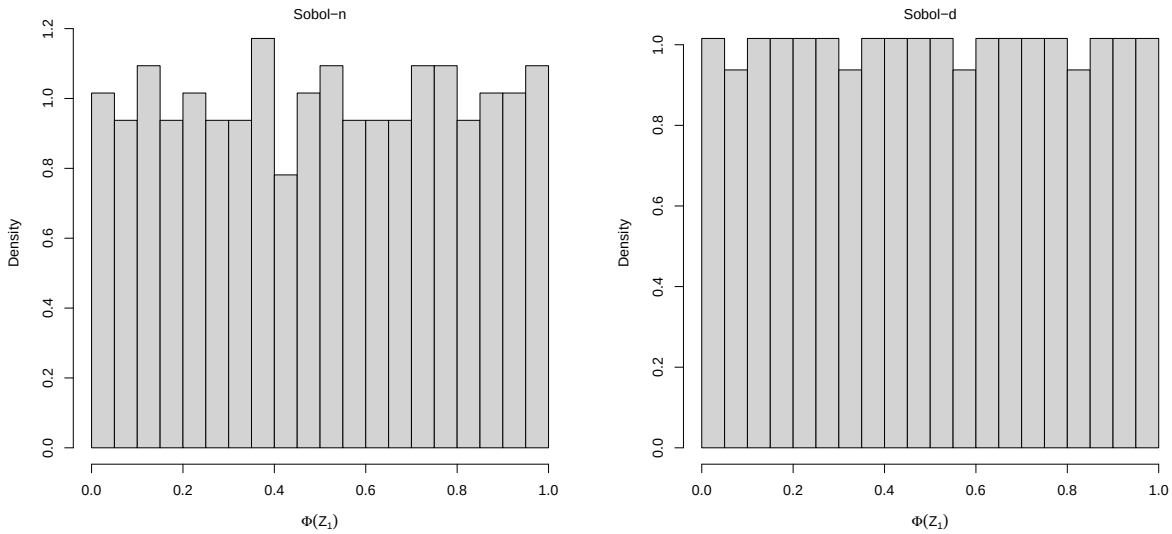

Figure 4.6: Histogram of U_1 when constructed with AR-n (left) and AR-d (right).

the sequence. In order to highlight this point, assume that our integrand does not depend on W and that $n = 2^k$. When estimating μ based on AR-d, we will then use the first 2^k points of the underlying LDS and keep all its good projection properties. In contrast, using AR-n, we only use a subset of size 2^k of the first $N > 2^k$ points, thereby potentially loosing some of the good projection properties of the LDS. This point is illustrated in Figure 4.5, where we first sample (W_i, Y_{i1}, Y_{i2}) where $Y_{ij} \sim \mathrm{N}(0,1)$, $j = 1, 2$, and $W_i \sim \Gamma(1.2, 1)$ for $i = 1, \ldots, n = 2^7$, and then set $\boldsymbol{U}_i = (F_W(W_i), \Phi(Y_{i1}), \Phi(Y_{i2}))$ for $i = 1, \ldots, n$. By the probability integral transformation, $\boldsymbol{U}_i \sim \mathrm{U}(0,1)^3$. Note that if we had used inversion to sample the W_i, the points would be exactly the original LDS. Note how Sobol'-d gives a point set with better marginal uniformity than Sobol'-n, which is also confirmed in the histogram of the first standardized coordinate in Figure 4.6. Note that if we had 2^k bins with $k \leq 7$ we would see a flat histogram on the right-hand side of this figure; here and in what follows, we use the AR samplers for the Gamma distribution from [14] and [72] for $\nu > 1$ and $\nu < 1$, respectively.

Next, we show in Propositions 4.2.3 and 4.2.4 that both algorithms produce point sets with the correct distribution.

Proposition 4.2.3
Each $\boldsymbol{x} \in O_n$ produced by Algorithm 4.2.1 has distribution $F_{\boldsymbol{Y}} \times F_W$.

Proof. It suffices to show that the two numbers used to sample from the proposal and the acceptance decision are independent $\mathrm{U}(0,1)$ random variables. The rest follows from the correctness of the AR algorithm; see, e.g., [32] for a proof. Let $\boldsymbol{x} = (\boldsymbol{Y}, W) \in O_n$. Then there is a $j \in \{1, 2, \dots\}$ such that $W = F^{-1}(U_1)$ and $U_2 \leq f_W(W)/(cf(W))$ where $U_1 = u_{j,d+1}$ and $U_2 = u_{j,d+2}$ satisfy $U_1, U_2 \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)$ by the randomization of the LDS. □

Proposition 4.2.4
Each $\boldsymbol{x} \in O_n$ produced by Algorithm 4.2.2 has distribution $F_{\boldsymbol{Y}} \times F_W$.

Proof. Since we assumed that the chosen LDS is randomized so that each $\boldsymbol{u}_{b,i} \sim U(0,1)^{d+1}$, the coordinates $u_{i,d+j}$ used in Step 2(a)i and 2(a)ii are independent $\mathrm{U}(0,1)$ for $j \geq 1$. The claim follows from the correctness of the AR algorithm. □

Our investigation of AR-d was motivated by the argument that AR corresponds to infinite-dimensional integration; see [40, p. 62–63], who also notes that "potential drawback of AR methods, compared with the inverse transform method, is that their outputs are generally neither continuous nor monotone functions of the input uniforms." We can address the monotonicity by using the rank transformations from the black box setting in Section 4.1: that is, we re-order the outputs $W_1, \dots, W_n$ so that their order matches the ordering of $u_{1,d+1}, \dots, u_{n,d+1}$. If $n = 2^k$, this is exactly the b-rk method from Section 4.1 applied with the output of AR-d as a "black box". Note that this makes the AR-d output monotone in coordinate $d+1$ of the underlying LDS. Note that with AR-n, we always use $u_{i,d+1}$ for some i to sample from the envelope via inversion, so that the monotonicity in this coordinate is already given.

Example 4.2.5 (Expected shortfall example continued)
We perform the same example as on page 84, but this time, using the AR based methods instead of the black-box setting. See Figure 4.7. All AR based methods outperform pure MC and MC-RQMC, and the convergence speed of AR-n and AR-d, 1:B-rk are almost as high as for the method "inversion", which we recall would not be available in a realistic setting.

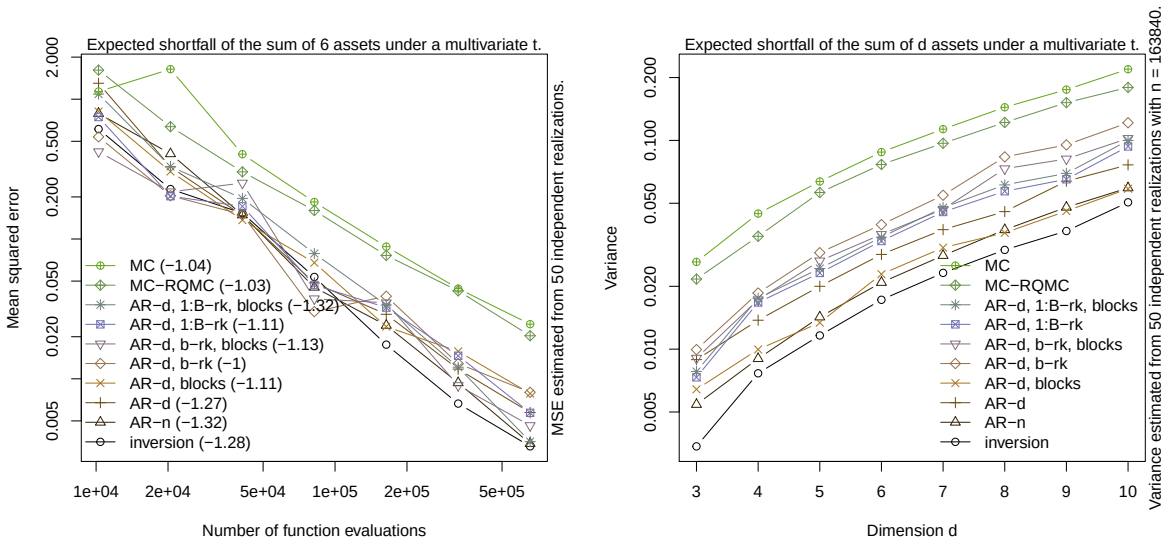

Figure 4.7: Mean squared errors as a function of n (left) and variances as a function of d (right) when estimating $\mathrm{ES}_{0.95}(L)$ for $L = \mathbf{1}^\top \boldsymbol{X}$ where $\boldsymbol{X} \sim t_d(\nu, \mathbf{0}, \Sigma)$.

4.3 Application: Basket option pricing

Consider the problem of estimating the value of a Basket call option with strike K, whose payoff with maturity $T = 1$, can be expressed as

$$\mu_{\mathrm{bskt}} = e^{-r}\mathbb{E}\left(\max\left\{\frac{1}{d}\sum_{j=1}^{d} S_j - K, 0\right\}\right);$$

we assume that the dependence of the log-normal assets S_j, $j = 1, \ldots, d$, is modelled via a t-copula. As such, the assets S_j have stochastic representation

$$S_j = F_{\mathrm{LN}}^{-1}(U_j), \quad U_j = F_{t_\nu}(X_j), \quad j = 1, \ldots, d, \quad \boldsymbol{X} \sim t_d(\nu, \mathbf{0}, \Sigma);$$

here, Σ is a correlation matrix. The t copula is one of the most widely used copulas in risk management; see, e.g., [23] for more. Pricing basket options is a popular problem to perform RQMC experiments; see, e.g., [74]. The value of μ_{bskt} is not known, so we look at the estimated variances for the following methods:

- `MC`: Use MC for W and $\boldsymbol{Y}$;

- `MC-RQMC`: use MC for W and RQMC for $\boldsymbol{Y}$, i.e., compute $\hat{\mu}^{\text{mc-rqmc}}$ in (4.5).
- `AR-d`: Use Algorithm 4.2.2, i.e., sample W based on AR whilst moving along the coordinates of a point in the LDS until acceptance.
- `AR-n`: Use Algorithm 4.2.1, i.e., sample W based on AR whilst moving along the index of the point in the LDS until acceptance.
- `AR-d, b-rk`: Use AR-d in each repetition b and additionally reorder the n samples $W_{1,b}, \dots, W_{n,b}$ according to $u_{1,b,d+1}, \dots, u_{n,b,d+1}$ for $b = 1, \dots, B$.
- `AR-d, 1:B-rk`: Use AR-d and sort all the nB sample points $W_{1,1}, \dots, W_{n,B}$ according to $u_{1,1,d+1}, \dots, u_{n,B,d+1}$.
- `b-rk`: Treat R_W as black-box and compute $\hat{\mu}_{b,n}^{\text{b-rk}}$ from (4.6) for $b = 1, \dots, B$.
- `1:B-rk`: Treat R_W as black-box and compute $\hat{\mu}_{b,n}^{\text{1:B-rk}}$ from (4.7) for $b = 1, \dots, B$.
- `eqf-gpd`: First, build gpd based estimate $\hat{Q}$ using samples obtained from the black box R_W, then treat it as true Q and proceed with inversion; see $\hat{\mu}^{\text{eqf-gpd}}$ in (4.8).
- `inversion`: Compute the inversion based estimator $\hat{\mu}_{b,n}^{\text{RQMC}}$ from (4.4) for $b = 1, \dots, B$ using the true quantile function.

The last method "inversion" is not available in a realistic setting like the stable example at the end of this section, but is included here to compare our methods with the best possible one. All methods (except for MC) sample the multivariate normal random vector $\boldsymbol{Y}$ based on inversion of a digitally shifted Sobol' sequence.

The results in Figure 4.8 indicate that using RQMC for sampling $\boldsymbol{Y}$ gives at least a modest variance reduction. Furthermore, treating the sampler as a black box and reordering the W samples as described in Section 4.1 gives further variance reduction. On the right hand side we see the AR methods from Section 4.2 which all give lower variance than the black-box methods; this makes sense as we are directly manipulating the sampler with some rank reordering methods. AR-d, combined with the re-ordering methods, outperforms AR-n.

Next, we alter this example so that we end up with a model where the quantile function of W is not as easily available as the quantile function of the inverse-gamma distribution (via `qgamma()`). To this end, we replace the t copula with a GIG-mixture copula. A random vector $\boldsymbol{X}$ has a GIG-mixture distribution if it follows the stochastic representation (4.9) with $W \sim \text{GIG}(\beta, \lambda)$; see [65] for a definition and an AR algorithm.

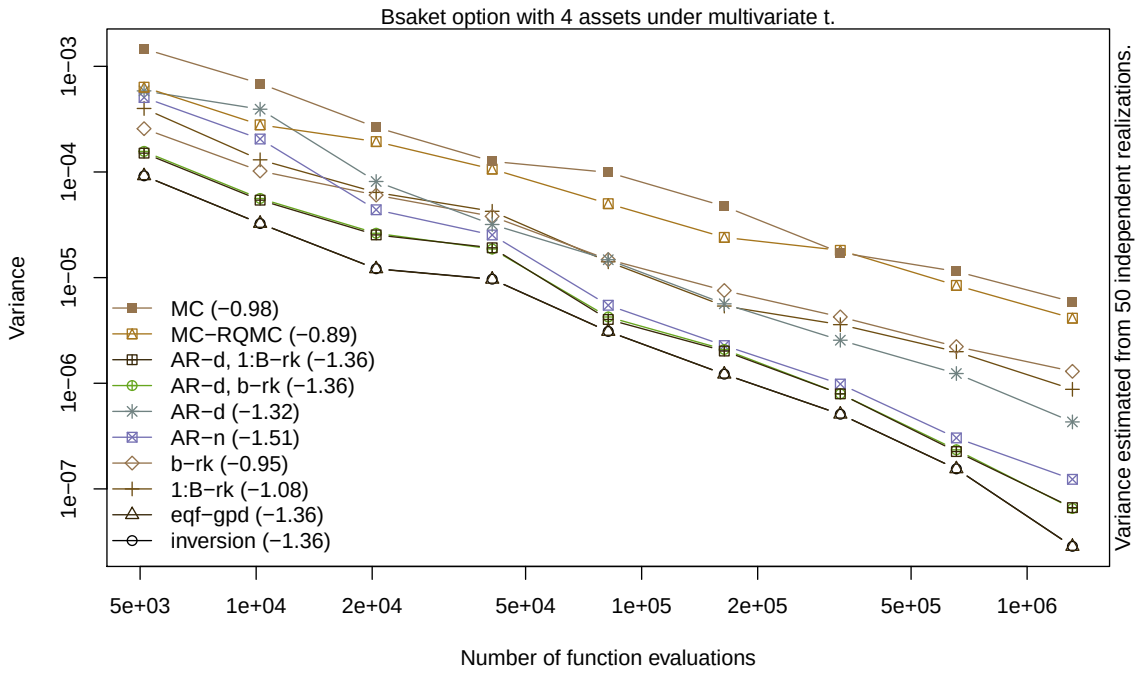

Figure 4.8: Variances when estimating μ_{bskt} under a t copula with $\nu = 2.2$ dof, $r = 0.01$, $\sigma = 0.2$ (volatility for all stocks) as a function of n.

The marginal distribution functions F_j of X_j needed to compute the copula sample are not known, so we denote by $\hat{F}_j(x) = (n+1)^{-1} \sum_{i=1}^{n} \mathbf{1}_{\{X_{ij} \le x\}}$ the empirical distribution function of X_j, and instead compute the pseudo observations $\boldsymbol{U}_i = (\hat{F}_1(X_{i1}), \ldots, \hat{F}_d(X_{id}))$ for $i = 1, \ldots, n$.

In this example, we also include the method `runuran`, implemented in the R package with the same name [78]. By using the density function as input, it approximates the distribution function numerically and builds an approximation of the quantile function using splines; see [25]. It was demonstrated in [77] that this method works well for the GIG distribution.

Figure 4.9 shows estimated variances as a function of n (left) and the number of assets d (right); the lines for the methods runuran, eqf-gpd, 1:B-rk and b-rk are overlapping. These are the best methods in terms of estimated variance. All RQMC based methods outperform MC and MC-RQMC gives the smallest variance reductions while AR-d and AR-n yield a modest variance reduction.

	runuran	eqf-gpd	1:B-rk	b-rk	AR-n	AR-d	AR-d, b-rk	AR-d, 1:B-rk	MC-RQMC	MC
CPU	0.342	3.965	0.545	0.545	0.541	0.530	0.532	0.553	0.548	0.542
REff	7.32	0.63	4.60	4.62	2.70	2.54	4.72	4.52	1.61	1.00

Table 4.1: Average run times in seconds (top) and estimated efficiencies (bottom) when computing various estimators with sample size $n = 20 \times 2^{12}$ to estimate μ_{bskt} under a 9-dimensional GIG mixture copula with $\beta = 0.3$ and $\lambda = 0.5$.

In the first row of Table 4.1, we show average run-times (in seconds) when computing various estimators when $n = 20 \times 2^{12}$ and $d = 9$. All methods, with the exception of eqf-gpd and runuran, take roughly the same time. Recall that with eqf-gpd, the idea is to estimate the quantile function Q once only using R_W, and then use it as a true quantile function for all subsequent simulations.The runuran method is the fastest. The second row of Table 4.1 shows relative efficiencies. The efficiency of a method is defined by $(\text{CPU} \cdot \text{Var})^{-1}$ and we prefer methods with large efficiency. We estimate these efficiencies and standardize them by the efficiency of pure MC. Method "eqf-gpd", due to its long run time, is the least efficient method, while the runuran method is most efficient, though we remark that it is the only method displayed that had access to the density function of W. The black box methods 1:B-rk and b-rk substantially outperform MC-RQMC, giving support for this simple re-ordering scheme. The re-ordering also helps the AR-based methods.

Finally, we repeat the same experiment with the only change being that now we assume that W follows a stable distribution; see [12] for a sampling algorithm for the stable distribution. Note that not even the density of a stable distribution can be easily computed,

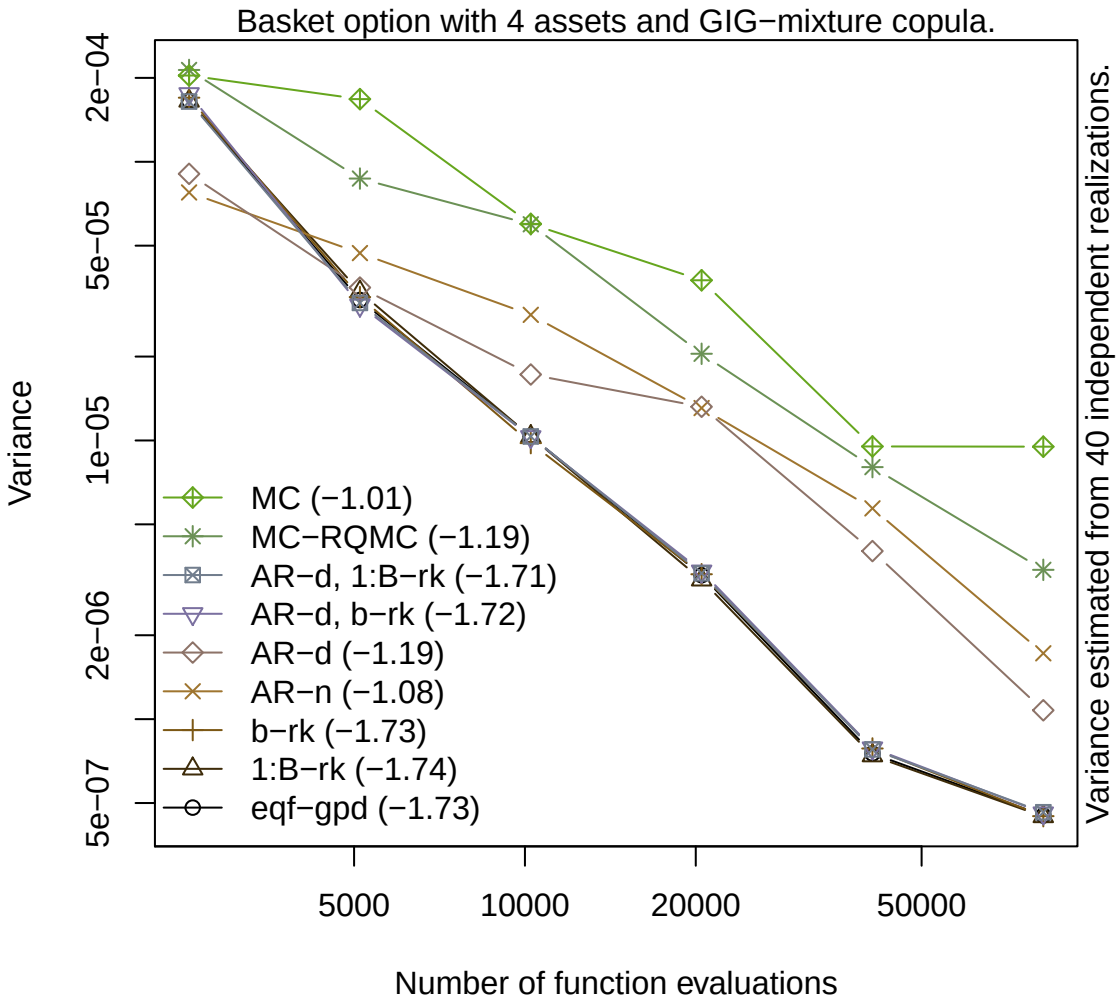

Figure 4.9: Variances when estimating μ_{bskt} under a GIG mixture copula with $\lambda = 0.5$, $\beta = 0.3$, $r = 0.01$, $\sigma = 0.2$ (volatility for all stocks) as a function of n.

hindering the application of numerical integration schemes to approximate the quantile function. In our simulation, we use the R package `stabledist`; see [20]. We use the function `rstable()` as a "black box" NRVG and the function `qstable()` to compare against the inversion method; note that it relies on numerical integration. The results are displayed in Figure 4.10, where we used the parameters $\alpha = 0.6$, $\beta = 1$ and $\gamma = \cos((\pi/2)\alpha)^{1/\alpha}$ for the stable distribution so that the support is $[0, \infty)$. Due to numerical problems with `qstable()` it was only used for sample sizes up to 2×10^4. See Table 4.2 for run times: For the chosen sample size, even our eqf-gpd method is faster than inversion. Our rank based methods are the most efficient methods in this example.

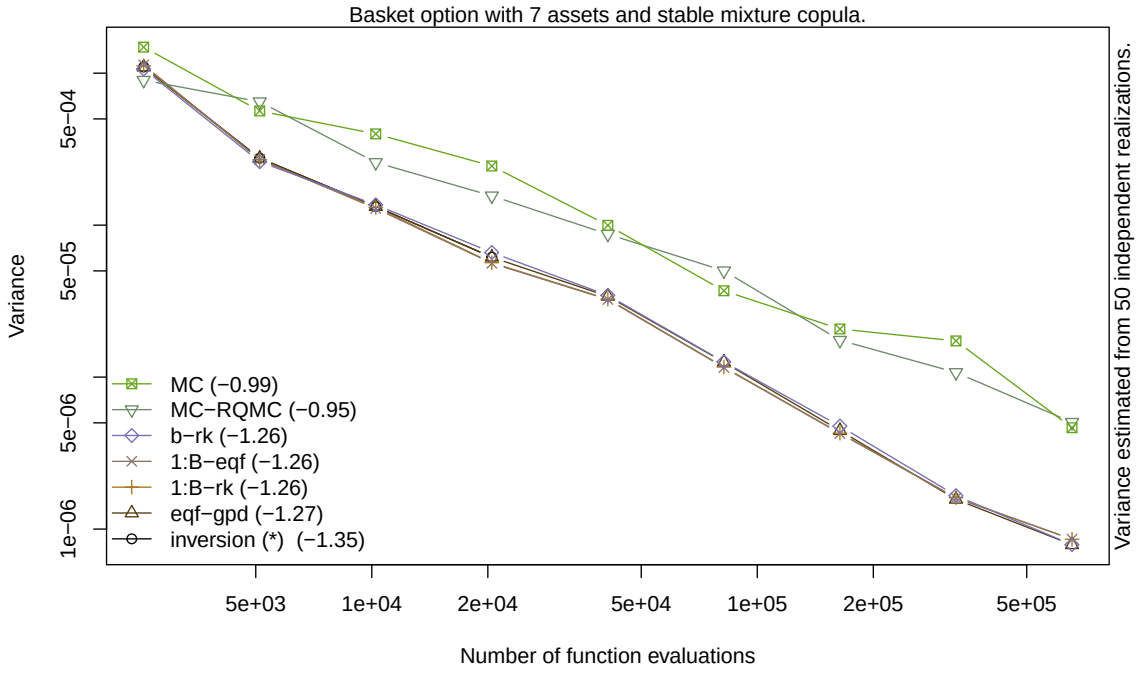

Figure 4.10: Variances when estimating μ_{bskt} under a stable mixture copula with $\alpha = 0.6$, $\beta = 1$ and $\gamma = \cos((\pi/2)\alpha)^{1/\alpha}$, $r = 0.01$, $\sigma = 0.2$ (volatility for all stocks) as a function of n. (*) The experiment for "inversion" was only performed up to $n = 2 \times 10^4$, so the regression coefficient was computed using a smaller sample than the other coefficients.

	inversion	eqf-gpd	1:B-rk	b-rk	MC-RQMC	MC
CPU	23.8	4.4	0.4	0.4	0.4	0.4
REff	0.06	0.33	3.64	3.59	1.26	1.00

Table 4.2: Average run times in seconds (top) and estimated efficiencies (bottom) when computing various estimators with total sample size $n = 20 \times 2^{10}$ to estimate μ_{bskt} under a 7-dimensional stable mixture copula with $\alpha = 0.9$, $\beta = 1$ and $\gamma = 1$.

4.4 Discussion

We explored the question how RQMC can be applied to estimate $\mu = \mathbb{E}(g(\boldsymbol{Y}, W))$ when all components but one can be sampled via inversion, and the remaining one W by calling a NRVG only. Our proposed algorithms in the black box setting were motivated by the fact that RQMC works best when combined with inversion, so that our methods aim at mimicking this observation by exploiting the sample to estimate the quantile function. In Section 4.2, we assumed the existence of an AR algorithm, and motivated an AR-d algorithm that samples along the coordinates rather than moving along the sequence. Our numerical results indicate that RQMC can still provide a substantial variance reduction when combined with a NRVG. In particular we saw that the re-ordering methods outperform MC-RQMC (where we merely combine RQMC with pseudo-random sampling of W). Furthermore, we saw that moving along the coordinates as we do in AR-d can give better results than the previously proposed AR-n methods. With the methods in this section at hand, we could extend the algorithms in Chapter 3 to estimate various quantities related to multivariate normal variance mixture distributions, such as the distribution function. Furthermore, we plan to address some questions of computational nature, such as exploring efficient implementations of AR-d based on point sets that are easily extensible in the number of coordinates, such as Korobov rules based on well-chosen generators a; see [75]. Finally, this section mostly focused on numerical comparisons of different RQMC-based algorithms based on digitally shifted Sobol' sequences. In the near future we plan to study settings under which it might be possible to obtain theoretical results demonstrating the superiority of our proposed RQMC-based methods (perhaps based on scramblings rather than shifts) over Monte Carlo. Here, the work [48] which focuses on quantile estimation via RQMC may be a starting point.

Chapter 5

Stratified single index importance sampling for rare event simulation

In this section, we consider the problem of rare event simulation: The goal is to estimate the existing $\mu = \mathbb{E}(\Psi(\boldsymbol{X}))$ where $\Psi : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{X} \sim F_{\boldsymbol{X}}$ is such that $\mathbb{P}(|\Psi(\boldsymbol{X})| > 0)$ is small. As mentioned in the introduction, plain MC or RQMC is often inefficient in this setting, as a large number of simulations is required to obtain estimates with small variance.

In many stochastic problems, the output of interest depends on an input random vector mainly through a single random variable (or index) via an appropriate univariate transformation of the input. We exploit this feature by proposing an importance sampling method that makes rare events more likely by changing the distribution of the chosen index. Let $T = T(\boldsymbol{X})$ be some univariate random variable, such as $\boldsymbol{\beta}^\top \boldsymbol{X}$ for some (well chosen) $\boldsymbol{\beta} \in \mathbb{R}^d$, and assume sampling from $\boldsymbol{X} \mid T$ is feasible. If T has density f (resp., g) under the original (resp., proposal) distribution (both distributions assumed to have the same support Ω_T for now), let

$$\hat{\mu}_n^{\mathrm{IS}} = (1/n) \sum_{i=1}^{n} \Psi(\boldsymbol{X}_i) f(T_i)/g(T_i), \quad T_i \overset{\text{ind.}}{\sim} g, \quad \boldsymbol{X}_i \overset{\text{ind.}}{\sim} F_{\boldsymbol{X}|T}(\cdot \mid T = T_i), \quad i = 1, \ldots, n.$$

If T explains much of the variability of the output, so if $R^2 := \mathrm{Var}(\mathbb{E}(\Psi(\boldsymbol{X}) \mid T))/\mathrm{Var}(\Psi(\boldsymbol{X}))$ is large, we can choose g optimally and make the rare event more likely by changing the distribution of $\boldsymbol{X}$ through changing the distribution of the univariate T. Many high dimensional financial problems are of this nature; see, e.g., [11, 118, 119, 117].

In order to analyze our estimator, we work with the semi-parametric model

$$\Psi(\boldsymbol{X}) = m(T) + \varepsilon_{\boldsymbol{X},T}$$

for some (unknown) transformation $T : \mathbb{R}^d \to \mathbb{R}$, where $m^{(k)}(t) = \mathbb{E}(\Psi(\boldsymbol{X})^k \mid T)$ for $k \in \mathbb{N}$ and $\varepsilon_{\boldsymbol{X},T}$ is a random error so that $\varepsilon_{\boldsymbol{X},T} \mid T$ has mean 0 and variance $v^2(t) = \mathrm{Var}(\Psi(\boldsymbol{X}) \mid T = t)$. We say that $\Psi(\boldsymbol{X})$ has a strong single index structure if R^2 is large (say, $R^2 > 0.9$), and the resulting estimator is referred to as Single-index importance sampling (SIS) estimator. If the proposal distribution g allows for a simple way to evaluate the quantile function G_T^{-1} of g, we can further reduce the variance by applying equal stratification to the support of T, i.e., instead of sampling $T_1, \dots, T_n \overset{\text{ind.}}{\sim} g$, we can set $T_i = G_T^{-1}(U_i)$ where $U_k \overset{\text{ind.}}{\sim} \mathrm{U}(k/n, (k+1)/n)$ for $k = 0, \dots, n-1$ and $G_T^{\leftarrow}(u) = \inf\{t \in \mathbb{R} : G_T(t) \geq u\}$ is the quantile function of T under g. The resulting method is referred to as Stratified single-index importance sampling (SSIS).

The performance of our procedure heavily depends on the choice of the transformation T, which must be chosen such that i) sampling from $\boldsymbol{X} \mid T$ is feasible and ii) T explains a lot of the variability of $\Psi(\boldsymbol{X})$, i.e., R^2 is as close to 1 as possible. The choice of the transformation is clearly not unique. In our numerical examples, we typically assume that T is a linear function of $\boldsymbol{X}$, whose coefficients can be estimated via the average derivative method of [113], which does not require the form of the function $m(t)$ to be known.

In the joint work , the collaborator Y. Taniguchi derived expressions for the optimal densities under SIS and SSIS, showed that the estimators have zero variance when $R^2 = 1$ and explained how our conditional sampling step reduces the effective dimension of the problem and therefore makes RQMC particularly attractive in this setting. These results are reviewed in Section 5.1. There, we will see that the optimal proposal distribution for g under SIS is proportional to $\sqrt{m^{(2)}(t)} f(t)$ and this choice results in an estimator with variance no larger than the plain MC estimator. The collaborator proposed using pilot runs to estimate the optimal (S)SIS density, which can then be integrated and inverted numerically using the NINIGL algorithm developed in [64].

The author of this thesis considers the problem of implementing (S)SIS in practice in more detail. As mentioned in the introduction, relying on numerical routines like NINIGL can be time-consuming and prone to numerical errors, as the input density is merely an estimate of the true optimal density. We give more details on how to estimate the conditional moment function using pilot-runs. Rather than feeding the approximated (S)SIS densities to the NINIGL algorithm, we propose finding an approximately optimal g in the same parametric family as f (e.g., a location-scale transform of the original density). We

detail this calibration stage, i.e., the process of estimating T, the optimal density and a way to sample from it, in Section 5.2. In the numerical examples in Section 5.3 performed by the author of this thesis, we demonstrate that our methods are applicable to a wide range of problems and achieve substantial variance reduction. After investigating a simple linear model example, we consider the problem of tail probability estimation in Gaussian and t-copula credit portfolio problems and show that our methods outperform those of [44] and [13].

As our formulation of (S)SIS does not assume a specific Ψ or $F_{\boldsymbol{X}}$, it is applicable to a wide range of problems and is efficient as long as the problem of interest has a strong enough single-index structure. It also adapts to the problem through the design of the one-dimensional transformation revealing the single-index structure and through the choice of the proposal distribution. Besides its applicability to a wide range of problems, our proposed method has the following advantages: First, as it applies IS only to the univariate transformation variable, SIS is less susceptible to the dimensionality problem of IS, which is discussed in [3, 68, 110]. This also simplifies the task of finding an optimal proposal distribution. Second, SIS has a dimension reduction feature, so it enhances the effectiveness of RQMC sampling methods. Third, by applying IS to a transformation of the input random vector $\boldsymbol{X}$, our proposal distribution amounts to changing the dependence structure of the problem under study, which can have a significant advantage over methods that only change the marginal distributions.

Concluding remarks are given in Section 5.4.

5.1 Variance analysis

To fix notation, recall we estimate $\mu = \mathbb{E}(\Psi(\boldsymbol{X}))$ via

$$\hat{\mu}_n^{\mathrm{SIS}} = (1/n)\sum_{i=1}^{n} \Psi(\boldsymbol{X}_i)w(T_i), \quad T_i \overset{\text{ind.}}{\sim} g_T, \quad \boldsymbol{X}_i \overset{\text{ind.}}{\sim} F_{\boldsymbol{X}|T}(\cdot \mid T = T_i), \quad i = 1, \dots, n,$$

where f_T and g_T denote the original and proposal densities for T with supports $\Omega_f = (t_{\inf}, t_{\sup})$ (with possibly $t_{\inf}, t_{\sup} \in \{\pm\infty\}$) and Ω_g and $w(t) = f_T(t)/g_T(t)$ is the IS weight function.

Furthermore, we model the output $\Psi(\boldsymbol{X})$ as $\Psi(\boldsymbol{X}) = m(T) + \varepsilon_{\boldsymbol{X},T}$, where

$$m^{(k)}(t) = \mathbb{E}(\Psi(\boldsymbol{X})^k \mid T), \quad \mathbb{E}(\varepsilon_{\boldsymbol{X},T} \mid T) = 0, \quad \mathrm{Var}(\varepsilon_{\boldsymbol{X},T} \mid T) = v^2(t) = \mathrm{Var}(\Psi(\boldsymbol{X}) \mid T).$$

We already introduced the coefficient of determination $R^2 = \mathrm{Var}(m(T))/\mathrm{Var}(\Psi(\boldsymbol{X}))$ (see, e.g., [73]) and said that $\Psi(\boldsymbol{X})$ is a strong single-index model if R^2 is large. This can be true for any model $\Psi(\boldsymbol{X})$, as we allow $\varepsilon_{\boldsymbol{X},T}$ to depend on $\boldsymbol{X}$. However, a pure single index model is a situation where $\varepsilon_{\boldsymbol{X},T} = \varepsilon_T$ only depends on $\boldsymbol{X}$ through T. In that case, it is easy to see that $\mathbb{E}(\Psi(\boldsymbol{X}) \mid T) = m(T)$, so that overall the random variable $\Psi(\boldsymbol{X})$ depends on $\boldsymbol{X}$ only through T. However, we do not impose the assumption of a pure single index model. Readers are referred to [101], [46] and [66] for more information on single-index models.

Based on the representation of $\Psi(\boldsymbol{X})$ and using the law of total variance, we can write

$$\mathrm{Var}(\Psi(\boldsymbol{X})) = \mathrm{Var}(m(T)) + \mathbb{E}(v^2(T)) = \mathrm{Var}(m(T)) + \mathrm{Var}(\varepsilon_{\boldsymbol{X},T}), \tag{5.1}$$

since $\mathbb{E}(v^2(T)) = \mathrm{Var}(\varepsilon_{\boldsymbol{X},T}) - \mathrm{Var}(\mathbb{E}(\varepsilon_{\boldsymbol{X},T} \mid T)) = \mathrm{Var}(\varepsilon_{\boldsymbol{X},T})$. We see that (5.1) decomposes the variance of $\Psi(\boldsymbol{X})$ into two pieces: the one of the (random) systematic part, $m(T)$ and the unsystematic error $\varepsilon_{\boldsymbol{X},T}$ of the model. Note that (5.1) holds irrespective of whether we have a pure single index model or not.

In addition to applying IS on T, we also propose to use stratification on T to further reduce the variance; it will turn out that this essentially "stratifies away" $\mathrm{Var}(m(T))$, the variance of the systematic part of the model. More precisely, let $\Omega_f = (t_{\inf}, t_{\sup})$ where possibly $t_{\inf} = -\infty$ and $t_{\sup} = \infty$. The SSIS scheme splits Ω_f into n strata of equal probability under g and draws one sample of T from each stratum. Our estimator becomes

$$\hat{\mu}_n^{\mathrm{SSIS}} = (1/n)\sum_{i=1}^{n} \Psi(\boldsymbol{X}_i) w(T_i), \quad T_i = G_T^{\leftarrow}(U_i), \quad U_i \sim \mathrm{U}((i-1)/n, i/n),$$

and, as before, $\boldsymbol{X}_i \overset{\text{ind.}}{\sim} F_{\boldsymbol{X}|T}(\cdot \mid T = T_i)$ for $i = 1, \ldots, n$.

For our variance analysis below, it is useful to find an expression for $\mathrm{Var}(\hat{\mu}_n^{\mathrm{MC}})$. Note that the conditional moment functions $m^{(k)}$ do not depend on whether we sample from f_T or g_T. From (5.1) and the fact that $\mathrm{Var}(m(T)) = \mathbb{E}(m(T))^2 - \mu^2$ as well as $\mathbb{E}(v^2(T)) = \mathbb{E}(m^{(2)}(T)) - \mathbb{E}(m(T))^2$, we find

$$n\,\mathrm{Var}(\hat{\mu}_n^{\mathrm{MC}}) = \mathrm{Var}(m(T)) + \mathbb{E}(v^2(T)) = \mathbb{E}(m^{(2)}(T)) - \mu^2. \tag{5.2}$$

As should be clear from the form of our estimators, their bias depends on the support Ω_g of g_T. We define

$$\mu_{\mathrm{SIS}} = \int_{\Omega_g} m(t) f_T(t)\,\mathrm{d}t, \quad \sigma^2_{\mathrm{SIS}} = \int_{\Omega_g} m^{(2)}(t)\frac{f_T^2(t)}{g_T(t)}\,\mathrm{d}t - \mu^2_{\mathrm{SIS}}, \quad \sigma^2_{\mathrm{SSIS}} = \int_{\Omega_g} v^2(t)\frac{f_T^2(t)}{g_T(t)}\,\mathrm{d}t.$$

Notice that μ_{SIS} depends on g_T through the region Ω_g. The SIS and SSIS estimators are unbiased only if g_T is such that $g_T(t) > 0$ whenever $m(t)f_T(t) > 0$, but we do not impose this unbiasedness assumption on g_T here and will get back to this point later.

5.1.1 Optimal proposal densities under (S)SIS

We are now able to give properties of the (S)SIS estimators and the optimal (variance-minimizing) proposal distribution of g_T. As the objective of our IS techniques is variance reduction, we call the practice of setting g_T to its optimal density or their approximation as *optimal calibration*, and the resulting methods SIS* and SSIS*. The following is [57, Prop. 2.1].

Proposition 5.1.1 (Variance-optimal SIS)
We have $\mathbb{E}(\hat{\mu}_n^{\text{SIS}}) = \mu_{\text{SIS}}$ and $\text{Var}(\hat{\mu}_n^{\text{SIS}}) = \sigma_{\text{SIS}}^2/n$. If $\mathbb{E}_g(m^2(T)w^2(T)) < \infty$, then $\sqrt{n}(\hat{\mu}_n^{\text{SIS}} - \mu_{\text{SIS}}) \xrightarrow{\text{d}} \text{N}(0, \sigma_{\text{SIS}}^2)$ as $n \to \infty$.

Suppose that $\Psi(\boldsymbol{x}) \geq 0$ or $\Psi(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \in \Omega_{\boldsymbol{X}}$. The density g_T that gives an unbiased SIS estimator with the smallest variance is

$$g_T^{\text{opt}}(t) = c^{-1}\sqrt{m^{(2)}(t)}f_T(t), \quad t \in (t_{\text{inf}}, t_{\text{sup}}), \quad c = \int_{t_{\text{inf}}}^{t_{\text{sup}}} \sqrt{m^{(2)}(t)}f_T(t)\,\text{d}t. \tag{5.3}$$

The variance of the optimal SIS estimator, denoted by $\hat{\mu}_n^{\text{SIS,opt}}$, is $\text{Var}(\hat{\mu}_n^{\text{SIS,opt}}) = (c^2 - \mu^2)/n$.

Remark 5.1.2
1. Proposition 5.1.1 implies that using optimal SIS gives variance no larger than MC. Indeed, by Jensen's inequality, $n\,\text{Var}(\hat{\mu}_n^{\text{SIS,opt}}) \leq \mathbb{E}(m^{(2)}(T)) - \mu^2$, which is equal to $\text{Var}(\hat{\mu}_n^{\text{MC}})$ using (5.2). This inequality holds as an equality only when $m^{(2)}(t)$ is constant for all $t \in \Omega_T$.

2. If $R^2 = 1$ (corresponding to the strongest possible single index structure), then $\text{Var}(\hat{\mu}_n^{\text{SIS,opt}}) = 0$: SIS provides a zero-variance estimator if $m^{(2)}(t) = (m(t))^2$ for all t, which is equivalent to having $v^2(t) = 0$ for all t, or equivalently, to having $\mathbb{E}(v^2(T)) = 0$ since $v^2(t) \geq 0$ for all t. This is the same as asking $\text{Var}(m(T))/\,\text{Var}(\Psi(\boldsymbol{X})) = R^2 = 1$. This is why choosing a function T such that the model is an as good fit as possible is important for the SIS method to achieve significant variance reduction.

The following proposition gives the properties of the SSIS estimator and the optimal (variance-minimizing) proposal distribution of g_T; see [57, Prop. 2.2].

Proposition 5.1.3 (Variance-optimal SSIS)
It holds that $\mathbb{E}(\hat{\mu}_n^{\text{SSIS}}) = \mu_{\text{SIS}}$ and, for large enough n, $\text{Var}(\hat{\mu}_n^{\text{SSIS}}) = \sigma_{\text{SIS}}^2/n + o(1/n)$. If

$\mathbb{E}_g\left(|m(T)w(T)|^{2+\delta}\right) < \infty$ for some $\delta > 0$, $\hat{\mu}_n^{\text{SSIS}}$ is asymptotically normal as $\sqrt{n}(\hat{\mu}_n^{\text{SSIS}} - \mu_{\text{SIS}}) \xrightarrow{\text{d}} \text{N}(0, \sigma_{\text{SIS}}^2)$ for $n \to \infty$. Suppose that $\Psi(\boldsymbol{x}) \geq 0$ or $\Psi(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \in \Omega_{\boldsymbol{X}}$ and that $\mathbb{P}_f(v^2(T) = 0,\ m(T) \neq 0) = 0$. The density g_T that gives an unbiased SSIS estimator with the smallest variance is

$$g_T^{\text{opt,s}}(t) = c^{-1} v(t) f_T(t), \quad t \in (t_{\text{inf}}, t_{\text{sup}}), \quad c = \int_{t_{\text{inf}}}^{t_{\text{sup}}} v(t) f_T(t)\, \mathrm{d}t. \tag{5.4}$$

The variance of the optimal SSIS estimator $\hat{\mu}_n^{\text{SSIS,opt}}$ is $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}}) = c^2/n + o(1/n)$. If $\mathbb{P}_f(v^2(T) = 0,\ m(T) \neq 0) > 0$, then $\hat{\mu}_n^{\text{SSIS,opt}}$ is biased.

Remark 5.1.4 1. Proposition 5.1.3 implies that using optimal SSIS gives asymptotically a variance no larger than MC. Indeed, Jensen's inequality implies that we have $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}}) \leq (1/n)\mathbb{E}(v^2(T)) + o(1/n)$ with equality only if $v(t)$ is constant for all $t \in \Omega_T$. From (5.2) (and ignoring the $o(1/n)$ term), this means $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}}) \leq \text{Var}(\hat{\mu}_n^{\text{MC}})$, with equality only if $v(t)$ is constant for all $t \in \Omega_T$ and $\text{Var}(m(T)) = 0$, which is unlikely to be the case since $m(T)$ has been chosen specifically such that $R^2 \approx 1$.

2. If $R^2 = 1$ (strongest possible single index structure), then $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}}) = 0$, since $\text{Var}(\hat{\mu}_n^{\text{SSIS}}) = 0$ iff $m^{(2)}(t) = (m(t))^2$ for all t, or equivalently $v^2(t) = 0$ for all t and thus $\mathbb{E}(v^2(T)) = 0$, which means $R^2 = 1$.

3. Unless $m(t) = 0$, SSIS achieves variance reduction compared to SIS, as $\text{Var}(\hat{\mu}_n^{\text{SSIS}}) \leq \text{Var}(\hat{\mu}_n^{\text{SIS}})$ for the same choice of g_T. This in turn implies that $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}}) \leq \text{Var}(\hat{\mu}_n^{\text{SSIS,opt}})$. The proposal densities g_T^{opt} and $g_T^{\text{opt,s}}$ defined in (5.3) and (5.4) give estimators with smallest variance if $\Psi(\boldsymbol{x}) \lesseqgtr 0$ for all $\boldsymbol{x} \in \Omega$, which holds for many applications in finance (e.g., when Ψ is an indicator and thus μ a probability or when Ψ is the payoff of an option). If Ψ takes both positive and negative values, $m(t)$ could be 0 for some values of t. We can then improve the optimal calibration by setting $g_T(t) = 0$ whenever $m(t) = 0$. Since it is generally unknown and hard to estimate which values of t give $m(t) = 0$, this improvement may not be implementable.

4. The expression for $\text{Var}(\hat{\mu}_n^{\text{SSIS,opt}})$ implies that SSIS* "stratifies away" the variance captured by the systematic part $m(T)$ of the single-index model, so the variance of the SSIS* estimator comes only from the error term $\varepsilon_{\boldsymbol{X},T}$ via $v(t)$. If g_T is not chosen optimally, then $\text{Var}(\hat{\mu}_n^{\text{SSIS}}) = \sigma_{\text{SIS}}^2/n + o(1/n)$ shows that we still make $\text{Var}(m(T))$ vanish by using stratification, but the contribution from $v^2(T)$ might be amplified (compared to how it contributes to the MC estimator's variance) if we do not choose

a good proposal density. Irrespective of the choice of g_T it is true that the stronger the fit of the single index model, the better (S)SIS works.

5. These results show that as long as the problem at hand has a strong single-index structure and sampling from T and $\boldsymbol{X} \mid T$ is feasible, SIS and SSIS can be applied and should give large variance reduction. As those conditions do not assume a specific form for Ψ or for the distribution of $\boldsymbol{X}$, SIS and SSIS are applicable to a wide range of problems.

5.1.2 SIS in multivariate normal models

Suppose that $\boldsymbol{X} \sim \mathrm{N}_d(\boldsymbol{0}, I_d)$. A popular strategy for constructing a proposal distribution under the Multivariate normal (MVN) model is to shift its mean vector of $\boldsymbol{X}$, that is, letting $\boldsymbol{X} \sim \mathrm{N}_d(\boldsymbol{\eta}, I_d)$ under the IS distribution for some $\boldsymbol{0} \neq \boldsymbol{\eta} \in \mathbb{R}^d$. The following proposition states that this type of IS can be achieved within our SIS framework by using $T(\boldsymbol{X}) = \boldsymbol{\theta}^\top \boldsymbol{X}$ where $\boldsymbol{\theta}$ is the normalized version of $\boldsymbol{\eta}$. Based on Proposition 5.1.1 and Remark 5.1.2, this result thus implies that this popular mean-shifting strategy for MVN models works well if the problem has a strong linear single-index structure based on the specific choice of shift vector $\boldsymbol{\eta}$.

Proposition 5.1.5 (SIS in MVN models)
Let $\boldsymbol{X} \sim \mathrm{N}_d(\boldsymbol{0}, I_d)$ under the original distribution. Fix $\boldsymbol{0} \neq \boldsymbol{\beta} \in \mathbb{R}^d$ with $\boldsymbol{\beta}^\top \boldsymbol{\beta} = 1$. Consider SIS with $T(\boldsymbol{X}) = \boldsymbol{\beta}^\top \boldsymbol{X}$. If g_T is the density of $\mathrm{N}(c, \sigma^2)$, then $\boldsymbol{X} \sim \mathrm{N}_d(c\boldsymbol{\beta}, I_d + (\sigma^2 - 1)\boldsymbol{\beta}\boldsymbol{\beta}^\top)$ in the IS scheme.

Proof. We use that $(\boldsymbol{X} \mid T = t) \sim \mathrm{N}_d(\boldsymbol{\beta} t, I_d - \boldsymbol{\beta}\boldsymbol{\beta}^\top)$ (see [47, Theorem 1]) to compute the moment generating function of $\boldsymbol{X}$. For $\boldsymbol{a} \in \mathbb{R}^d$,

$$\mathbb{E}_g(\exp(\boldsymbol{a}^\top \boldsymbol{X})) = \mathbb{E}_g\left(\mathbb{E}(\exp(\boldsymbol{a}^\top \boldsymbol{X}) \mid T)\right) = \mathbb{E}_g\left(\exp\left(\boldsymbol{a}^\top \boldsymbol{\beta} T + \frac{1}{2}\boldsymbol{a}^\top (I_d - \boldsymbol{\beta}\boldsymbol{\beta}^\top)\boldsymbol{a}\right)\right)$$
$$= \mathbb{E}_g(\exp(\boldsymbol{a}^\top \boldsymbol{\beta} T)) \exp\left(\frac{1}{2}\boldsymbol{a}^\top (I_d - \boldsymbol{\beta}\boldsymbol{\beta}^\top)\boldsymbol{a}\right) = \exp\left(c\boldsymbol{a}^\top \boldsymbol{\beta} + \frac{1}{2}(\boldsymbol{a}^\top \boldsymbol{\beta})^2 \sigma^2\right) \times$$
$$\times \exp\left(\frac{1}{2}\boldsymbol{a}^\top (I_d - \boldsymbol{\beta}\boldsymbol{\beta}^\top)\boldsymbol{a}\right) = \exp\left(\boldsymbol{a}^\top (c\boldsymbol{\beta}) + \frac{1}{2}\boldsymbol{a}^\top (I_d + (\sigma^2 - 1)\boldsymbol{\beta}\boldsymbol{\beta}^\top \boldsymbol{a}\right).$$

By uniqueness of the moment generating function, $\boldsymbol{X} \sim \mathrm{N}_d(c\boldsymbol{\beta}, I_d + (\sigma^2 - 1)\boldsymbol{\beta}\boldsymbol{\beta}^\top)$. □

Proposition 5.1.5 implies that $\boldsymbol{X} \sim \mathrm{N}_d(c\boldsymbol{\beta}, I_d)$ if g_T is chosen as the $\mathrm{N}(c, 1)$ density (where we recall that the original distribution f_T is $\mathrm{N}(0, 1)$), so that the previously mentioned mean-shifting strategy is a special case of IS (namely, by merely shifting the mean of T instead of applying SIS*). If $\mathrm{Var}(T) \neq 1$ under g, the dependence structure of the components in $\boldsymbol{X}$ does change in the IS scheme.

5.1.3 SIS and RQMC

It is widely accepted that the performance of RQMC is largely influenced by the effective dimension of the problem. We saw in Section 3.6.2 the notion of the effective dimension in the superposition sense (in proportion p). Another notion of effective dimension is the truncation dimension; see [119]. Essentially, a problem has a low truncation dimension when only a small number of leading input variables are important. Recall that $\boldsymbol{X}$ is sampled indirectly in SIS, that is, T is generated first then $\boldsymbol{X}$ is drawn from $F_{\boldsymbol{X}|T}$. Assuming T is generated using the inversion method and via the first coordinate u_1 of $\boldsymbol{u} \in [0, 1)^{k+d}$ (where $k \geq 0$ is problem dependent), the indirect sampling step of SIS transforms the problem in such a way that the first input variable accounts for $R^2 \cdot 100\%$ of the variance of $\Psi(\boldsymbol{X})$, where $R^2 = \mathrm{Var}(m(T))/\mathrm{Var}(\Psi(\boldsymbol{X}))$. That is, the problem has a truncation dimension of 1 in proportion R^2 under SIS. Therefore, if the fit of the single-index model is good, say $R^2 > 0.9$, the indirect sampling step via T serves as a dimension reduction technique and enhances the efficiency of RQMC.

5.2 Calibration in practice

To apply (S)SIS in practice, we must estimate the optimal transformation function $T = T(\boldsymbol{X})$ and construct an approximation $\hat{g}_T^{\mathrm{opt}}$ for the optimal density g_T^{opt}. We call the stage in which these two tasks are performed the *calibration stage*. Furthermore, the calibrations in (5.3) and (5.4) require the knowledge of the conditional mean function and variance function, respectively. As these are rarely known in practice, they must be estimated in the calibration stage as well.

5.2.1 Estimating the optimal transformation T

In what follows, we assume that T is a linear function of the components in $\boldsymbol{X}$, i.e., $T = \boldsymbol{\beta}^\top \boldsymbol{X}$ for some $\boldsymbol{\beta} \in \mathbb{R}^d$; note that if $\boldsymbol{X}$ is multivariate normal, then T is univariate

normal and sampling from $\boldsymbol{X} \mid T$ is straightforward. To find $\boldsymbol{\beta}$ that maximizes R^2, we use the average derivative method of [113], which essentially allows us to estimate $\boldsymbol{\beta}$ as if we met the assumptions of a linear regression. That is, we sample independent realizations $\Psi(\boldsymbol{X}_i) =: \Psi_i$ for $i = 1, \dots, n_1$ (say, $n_1 = 1000$) and compute the sample covariance matrix $\Sigma_{\boldsymbol{X},\boldsymbol{X}}$ as well as the sample cross covariance of $\boldsymbol{X}_1, \dots, \boldsymbol{X}_{n_1}$ and $(\Psi_1, \dots, \Psi_{n_1})$, say $\Sigma_{\boldsymbol{X},\Psi}$ to obtain

$$\hat{\boldsymbol{\beta}} = \Sigma_{\boldsymbol{X},\boldsymbol{X}}^{-1} \Sigma_{\boldsymbol{X},\Psi}.$$

In some applications, we may use only a subset of the components in $\boldsymbol{X}$; in later examples, for instance, we only use the systematic risk factors in a credit model to build our transformation T. Sometimes one may even not need to estimate $\boldsymbol{\beta}$, for instance, if it is clear that the d components in $\boldsymbol{X}$ are equally important, one can simply set $\boldsymbol{\beta} = (1/\sqrt{d}, \dots, 1/\sqrt{d})$.

5.2.2 Finding the optimal density

The calibration in (5.3) requires the knowledge of the conditional second moment function $m^{(2)}(t) = \mathbb{E}(\Psi^2(\boldsymbol{X}) \mid T = t)$ for all $t \in \Omega_T$, which, of course, is not known; similarly, the conditional variance function v^2 required for the calibration in (5.4) is not known either. We now describe how to calibrate (5.3) in practice; the calibration of (5.4) can be done similarly.

Our first ingredient is the construction of an estimate of $m^{(2)}(t) = \mathbb{E}(\Psi^2(\boldsymbol{X}) \mid T = t)$ for all $t \in \Omega_T$; we suggest using plain MC for this purpose. To this end, let $t_1 < \cdots < t_M$ be knots at which the function $m^{(2)}$ is to be estimated (e.g., $M = 20$ equally spaced points in the relevant range). Choose some small pilot sample size n_{pilot} (for example, 5% of the total sample size n). For each t_j, sample n_{pilot}-many realizations from $\boldsymbol{X} \mid T = t_j$ and estimate $m^{(2)}(t_j)$ by its empirical equivalent for $j = 1, \dots, M$. Then utilize smoothing splines (see, for example, [105]) and only those t_j associated with a positive estimate to construct an estimate $\hat{m}^{(2)}$ for all $t \in \Omega_T$; for those t where $\hat{m}^{(2)}(t) \leq 0$, one can either leave them as $\hat{m}^{(2)}(t) = 0$ (which may lead to bias as discussed below) or set $\hat{m}^{(2)}$ to be some positive function (e.g., the error function) that resembles the lower tail of ε.

Having constructed an estimate for $m^{(2)}$, we can set $\hat{g}_T^{\text{opt}} \propto \sqrt{\hat{m}^{(2)}(t)} f(t)$ for $t \in \mathbb{R}$. However, $\hat{g}_T^{\text{opt}}$ rarely belongs to known parametric families of distributions that are easily sampled from. One can use numerical techniques such as the NINIGL algorithm to approximate the quantile function of a distribution given its unnormalized density; see [64].

This approach, however, has three drawbacks: i) sampling from a numerically constructed density is time-consuming and can be prone to numerical problems; ii) the normalizing constant needs to be estimated, and iii) bias can occur when $\hat{g}_T^{\text{opt,s}}$ does not have the same support as $g_T^{\text{opt,s}}$, which in turn happens when $\hat{m}^{(2)}(t) = 0$ even though $m^{(2)}(t) \neq 0$ for some set D with $\int_D f(t)\,\mathrm{d}t > 0$.

The third drawback can be alleviated if we can define $\hat{m}^{(2)}(t)$ to be positive whenever $m^{(2)}(t)$ is (for example, by assuming some lower and upper tail behaviour). Furthermore, recall from Proposition 5.1.3 that (5.4) gives a biased estimator if $\mathbb{P}_f(v^2(T) = 0,\ m(T) \neq 0) > 0$ which in some cases can be debiased. For instance, if $v(t) > 0$ for all $t \in \Omega_t$, but the estimated $\hat{v}(t) = 0$ for $t \geq t_{\max}$ for some $t_{\max} \in \mathbb{R}$ and $m(t) = c$ for some constant c for $t \geq t_{\max}$ (for instance, if μ is a probability, then typically $m(t) = c = 1$ for $t \geq t_{\max}$). If $\hat{\mu}_n^{\text{SSIS}}$ is constructed using $\hat{v}$, we find

$$\mathbb{E}(\hat{\mu}_n^{\text{SSIS}}) = \int_{-\infty}^{t_{\max}} m(t) f_T(t)\,\mathrm{d}t = \mu - \int_{t_{\max}}^{\infty} m(t) f_T(t)\,\mathrm{d}t \approx \mu - c\mathbb{P}_f(T > t_{\max});$$

$\hat{\mu}_n^{\text{SSIS}}$ can therefore be debiased by adding $c\mathbb{P}_f(T > t_{\max})$.

The second drawback can be addressed by using weighted IS (so that the normalizing constant cancels out); see [76, Section 4.5]. Alternatively, the normalizing constant can be estimated as follows: Let $\hat{g}_{T,u}^{\text{opt}}(t) = \sqrt{\hat{m}^{(2)}} g(t)$ denote the unnormalized density, and $T_1, \ldots, T_n \overset{\text{ind.}}{\sim} \hat{g}_T^{\text{opt}}$ (obtained, for instance, using the NINIGL algorithm). Now construct an estimate of the density of $T_1, \ldots, T_n$, such as the kernel density estimator, and denote this estimated density by $\hat{h}$; note that $\hat{h}$ is normalized and that each of $\hat{h}(T_i)/\hat{g}_{T,u}^{\text{opt}}(T_i)$ for $i = 1, \ldots, n$ is an estimator for the normalizing constant. As such, we suggest using the sample median of $\{\hat{h}(T_1)/\hat{g}_{T,u}^{\text{opt}}(T_1), \ldots, \hat{h}(T_n)/\hat{g}_{T,u}^{\text{opt}}(T_n)\}$ as an estimator for the normalizing constant.

The first drawback, that is, the construction of an approximation to the quantile function of $\hat{g}_T^{\text{opt}}$ being both slow and potentially prone to numerical problems, is most severe. Below, we propose an alternative method, namely by setting $\hat{g}_T^{\text{opt}}(t) = 1/\sigma f((t-k)/\sigma)$ for carefully chosen $k \in \mathbb{R}$ and $\sigma > 0$. In other words, we suggest using a location-scale transform of the original density as proposal density and will therefore call this method $\text{SIS}^{c,\sigma}$. While this procedure does require estimation of k and σ, it does not suffer from any of the three aforementioned problems: i) if we can sample from f, we can also sample from $f((t-k)/\sigma)/\sigma$; ii) there is no normalizing constant or density to be estimated; iii) if f is supported on $\mathbb{R}$, f and $f((t-k)/\sigma)/\sigma$ have the same support, so that the resulting estimator is unbiased.

The idea behind using a location-scale transform arises from the observation that in many practical examples (as will be seen later) the optimal density has roughly the same shape as the original density. As such, we try to find k and σ so that $1/\sigma f((t-k)/\sigma)$ is approximately $g_T^{\text{opt}}(t)$. Denote again by $\hat{g}_{T,u}^{\text{opt}}(t) = \sqrt{\hat{m}^{(2)}(t)} f(t)$ the unnormalized, estimated optimal density and assume that the mode of f_T is at zero (otherwise, shift accordingly). Now find $k^* = \text{argmax}_t \hat{g}_{T,u}^{\text{opt}}(t)$ numerically; this makes sure that the theoretical and approximated densities have (roughly) the same mode, thereby both sample from the "important region". Having estimated k^*, the next step is to compute σ such that it minimizes the variance of the resulting estimator. More precisely, given a sample $T_1, \dots, T_{n_{\text{pilot}}}$ from f, we can estimate the variance of the estimator for a given σ as follows: Set $\tilde{T}_i = k^* + \sigma T_i$ and $w_i = \frac{f(\tilde{T}_i)}{f((\tilde{T}_i - k^*)/\sigma)/\sigma}$ and sample $\boldsymbol{X}_i \mid \tilde{T}_i$ for $i = 1, \dots, n_{\text{pilot}}$. The second moment of the IS estimator (written as a function of the scale σ) is then

$$V(\sigma) = \sum_{i=1}^{n_{\text{pilot}}} \Psi(\boldsymbol{X}_i) w_i^2, \quad \sigma > 0. \tag{5.5}$$

We can now solve $\sigma^* = \text{argmin}_{\sigma>0} V(\sigma)$ numerically. Note that due to the nature of a location-scale transform, we only need to sample $T_1, \dots, T_{n_{\text{pilot}}}$ once. Intuitively, k^* shifts the density to the important region, while σ^* scales it appropriately. If computing $V(\sigma)$ is very time consuming (for example, when the sampling of $\boldsymbol{X} \mid T$ is complicated), one can set $\sigma^* = 1$; the resulting method is then called SIS^{μ} instead of $\text{SIS}^{\mu,\sigma}$.

Algorithm 5.2.1 (Calibration and estimation stage for estimating μ via $\text{SIS}^{\mu,\sigma}$)
Given knots $t_1, \dots, t_{n_{\text{pilot}}}$, a total pilot budget n_{tot} and knot-sample size n_{knot}, target sample size n, estimate μ via:

1. *Estimation of the direction vector.*
   (a) Sample $\boldsymbol{X}_i \overset{\text{ind.}}{\sim} f_{\boldsymbol{X}}$ for $i = 1, \dots, n_{\text{pilot}}$ and compute $\Psi(\boldsymbol{X}_i) =: \Psi_i$, $i = 1, \dots, n_{\text{pilot}}$.
   (b) Compute $\Sigma_{\boldsymbol{X},\boldsymbol{X}}$ and $\Sigma_{\boldsymbol{X},\boldsymbol{\Psi}}$and set $\hat{\boldsymbol{\beta}} = \Sigma_{\boldsymbol{X},\boldsymbol{X}}^{-1} \Sigma_{\boldsymbol{X},\Psi}$.
2. *Estimation of c^* and σ^*.*
   (a) For each $k = 1, \dots, n_{\text{pilot}}$, sample $\boldsymbol{X}_{j,k} \overset{\text{ind.}}{\sim} f_{\boldsymbol{X}|T}(\cdot \mid t_k)$ for $j = 1, \dots, n_{\text{knot}}$.
   (b) Utilize smoothing splines[1] through $(t_k, (1/n_{\text{knot}}) \sum_{j=1}^{n_{\text{knot}}} \Psi(\boldsymbol{X}_{j,k})^2)$, $k = 1, \dots, n_{\text{pilot}}$, to construct an estimate for $\hat{m}^{(2)}(t)$ for $t \in \mathbb{R}$.

[1] In the case when Ψ is an indicator, use a logistic regression (available, for instance, via the R function `glm()`) with $n_{\text{knot}} = 1$ instead.

(c) Find $c^* = \text{argmax}_t \sqrt{\hat{m}^{(2)}(t)} f(t)$ numerically.

(d) Sample $T_1, \ldots, T_{n_{\text{pilot}}} \overset{\text{ind.}}{\sim} f$ and find $\sigma^* = \text{argmin}_{\sigma>0} V(\sigma)$ with the function V from (5.5) numerically.

3. *Estimation of μ.*

(a) Sample $T_1', \ldots, T_n' \overset{\text{ind.}}{\sim} f$, set $T_i = c^* + \sigma^* T_i$ and compute $w_i = \frac{f(T_i)}{f((T_i - c^*)/\sigma^*)/\sigma^*}$ for $i = 1, \ldots, n$.

(b) Sample $\boldsymbol{X}_i \overset{\text{ind.}}{\sim} f_{\boldsymbol{X}|T}(\cdot \mid T_i)$ for $i = 1, \ldots, n$.

(c) Return $\hat{\mu}_n^{\text{SIS}} = (1/n) \sum_{i=1}^n \Psi(\boldsymbol{X}_i) w_i$.

Remark 5.2.2 1. Algorithm 5.2.1 can be easily adapted to accommodate quasi-random numbers and stratification.

2. The effort for the conditional sampling needed in Steps 2a, 2d and 3b is problem specific – for some problems, samples of $\boldsymbol{X} \mid T = t_1$ can be easily transformed to samples from $\boldsymbol{X} \mid T = t_2$ for $t_1 \neq t_2$, making these steps very fast; in some other problems, the conditional sampling is more involved.

3. Our proposed SIS method can also be combined with other VRTs. For instance, in Section 5.3, we combine conditional MC (CMC) and SIS to estimate loss probabilities of a credit portfolio whose dependence is governed by a t-copula.

5.3 Numerical examples

In this subsection, we perform an extensive numerical study to demonstrate the effectiveness of our proposed methods. We start with a simplistic linear model example, in which case calibration of the optimal densities can be done easily. This allows us to investigate the effect of replacing g_T^{opt} by $\hat{g}_T^{\text{opt}}$. Next, we apply our SIS and SSIS schemes to a credit portfolio problem under the Gaussian copula model studied by [44]. The same financial problem but this time using a more complicated t-copula model is studied at the end of this subsection.

5.3.1 Linear Model Example

Let $L = \alpha T + \varepsilon_T$ where $T \sim \mathrm{N}(0,1)$, $\varepsilon_T \mid T \sim \mathrm{N}(0, s^2)$ and $\alpha^2 + s^2 = 1$. L has a single index structure when $\alpha^2 \approx 1$ since $R^2 = \mathrm{Var}(m(T))/\mathrm{Var}(L) = \mathrm{Var}(\alpha T) = \alpha^2$. Assume interest lies in estimating the probability $p_l = \mathbb{P}(L > l) = \bar{\Phi}(l) = \mathbb{E}(\mathbb{1}_{\{L>l\}})$ for some large l; note that we can approximate the true value of p_l efficiently with high precision since $L \sim \mathrm{N}(0,1)$. Furthermore it is easily seen that $p_l(t) = \mathbb{P}(L > l \mid T = t) = \bar{\Phi}\left((l - \alpha t)/s\right)$ for $l, t \in \mathbb{R}$. Since the integrand Ψ in this setting is an indicator, we find from Proposition 5.1.1 that $g_T^{\mathrm{opt}}(t) \propto \sqrt{p_l(t)} f_T(t)$.

Unlike in this simplistic setting, $p_l(t)$ for $t \in \mathbb{R}$ is unknown in practice as discussed in Section 5.2; thus, this setting serves as an excellent example to also compare whether approximating g_T^{opt} by $\hat{g}_T^{\mathrm{opt}}$ has a significant effect on the accuracy of the estimators. Sampling from the true optimal densities is performed using the R package `Runuran` of [78]. We consider the methods SIS** (constructed using known $p_l(t)$), SIS* (approximated $p_l(t)$ and NINGL), SIS$^{\mu}$ and SIS$^{\mu,\sigma}$

For the two settings of $\alpha^2 \in \{0.7, 0.99\}$ (corresponding to a weaker and stronger single index structure), we estimate p_l for $l \in \{3, 4, \ldots, 7\}$ using the five aforementioned methods. For each value of l, the optimal density is calibrated separately. In all examples, we use a sample size of $n = 10^6$ and a pilot sample size of 5×10^4. We repeat the experiment 200 times.

Figure 5.1 displays on the left the optimally calibrated and approximated IS densities. The true optimal density is bell shaped, so it is well approximated by a normal density. It can be confirmed from the plot that in this case, all IS densities seem to cover the important range. The right of Figure 5.1 displays a boxplot of run-times needed to estimate p_l; note that the run-time does not depend on α or l. This plot, however, should be interpreted with caution as it highly depends on how the pilot runs are implemented.

Figure 5.2 displays mean relative errors and Figure 5.3 displays estimated variances; recall that we know p_l here. The relative errors for the different methods are similar, though SIS$^{\mu,\sigma}$ seems to give smallest errors. A possible explanation might be that the simplicity of that method (e.g., in terms of the support) relative to numerically constructing the optimal density via NINGL might outweigh the benefit of the latter having slightly more theoretical support. Furthermore, note that the IS methods perform much better when $R^2 = \alpha^2$ is larger, i.e., when the single index structure is strong, as expected.

5.3.2 Tail probabilities of a Gaussian copula credit portfolio

In this section, we study the effectiveness of the proposed methods for a credit portfolio problem studied in [44], where the goal is to estimate the probability of large portfolio losses under a normal copula model. We compare our proposed methods to the IS technique of Glasserman and Li, to which we refer to as G&L IS.

Problem formulation

Suppose that Y_k denotes the default indicator of the kth obligor with exposure c_k and a default probability of p_k for $k = 1, \dots, h$. The incurred loss is then $L = \sum_{k=1}^{h} c_k Y_k$. Let

$$Y_k = \mathbb{1}_{\{X_k > \Phi^{-1}(1-p_k)\}}, \quad X_k = a_{k1}Z_1 + \cdots + a_{kd}Z_d + b_k\varepsilon_k \sim \mathrm{N}(0,1), \quad k = 1, \dots, h,$$

where

$$(Z_1, \dots, Z_d) \sim \mathrm{N}_d(\mathbf{0}, I_d), \quad \varepsilon_1, \dots, \varepsilon_h \overset{\text{ind.}}{\sim} \mathrm{N}(0,1), \quad \sum_{j=1}^{h} a_{kj}^2 \leq 1, \quad b_k = \sqrt{1 - \sum_{j=1}^{h} a_{kj}^2}.$$

The a_{kj} represent the kth obligor's factor loadings for the d risky systematic factors; the choice of b_k ensures $X_k \sim \mathrm{N}(0,1)$. Our goal is to estimate $P(L > l)$ for small $l > 0$.

As in [44], we consider a portfolio with $h = 1\ 000$ obligors in a 10-factor model (i.e. $d = 10$). The marginal default probabilities and exposures are $p_k = 0.01 \cdot (1 + \sin(16\pi k/h))$ and $c_k = (\lceil 5k/h \rceil)^2$ for $k = 1, \dots, h$, respectively. The marginal default probabilities vary between 0% and 2% and the possible exposures are 1, 4, 9, 16 and 25, with 200 obligors at each level. The factor loadings a_{kj}'s are independently generated from a $\mathrm{U}(0, 1/\sqrt{d})$. Letting $\boldsymbol{Z} = (Z_1, \dots, Z_d)^\top$ and $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_h)^\top$, we write $L = L(\boldsymbol{Z}, \boldsymbol{\varepsilon})$, i.e., the vector $\boldsymbol{X}$ to which we have referred throughout this work is given by $\boldsymbol{X} = (\boldsymbol{Z}, \boldsymbol{\varepsilon})$ for this example. We investigate whether or not L has a single-index structure. Let $T = \boldsymbol{\theta}^\top \boldsymbol{Z}$ where $\boldsymbol{\theta} \in \mathbb{R}^d$ such that $\boldsymbol{\theta}^\top \boldsymbol{\theta} = 1$, so $T \sim \mathrm{N}(0,1)$. We estimate $\boldsymbol{\theta}$ that maximize the fit by using the average derivative method of [113]. The estimated $\boldsymbol{\theta}$ has almost equal entries close to $\sqrt{1/d}$. This makes intuitive sense, as each component of $\boldsymbol{Z}$ is likely to be equally important because the factor loadings are generated randomly. The left side of Figure 5.4 shows the scatter plot of (T, L). The figure reveals the single-index model fits L well even in the extreme tail, implying SIS based on this choice of T will give substantial variance reduction. The right side of Figure 5.4 displays the original density of T, the optimally calibrated SIS* density as well as the estimated function $p_l(t)$. Note that the optimally calibrated density's mode substantially differs from the original one.

Proposed estimators

The method of [44] consists of a two-step procedure. In a calibration stage, an optimal mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ is found by solving an optimization problem minimizing the variance of the resulting IS estimator. Next, one samples $\boldsymbol{Z} \sim \mathrm{N}_d(\boldsymbol{\mu}, I_d)$ and computes the conditional default probabilities $p_k(\boldsymbol{Z}) = \mathbb{P}(Y_k = 1 \mid \boldsymbol{Z}) = \Phi((\boldsymbol{a}_k^\top \boldsymbol{Z} - x_k)/b_k)$, which enter another optimization problem used to find a number $\theta \in \mathbb{R}$ so that $q_k(\theta, \boldsymbol{Z})$ are variance minimizing default probabilities. Given $\boldsymbol{Z}$, we know that $Y_1, \dots, Y_h$ are independent and can therefore easily sample the loss via $L = \sum_{k=1}^h c_k \mathbb{1}_{\{U_k \le q_k(\theta(\boldsymbol{Z})\}}$ where $(U_1, \dots, U_h) \sim \mathrm{U}(0,1)^h$. Finally, the estimator $\mathbb{1}_{\{L>l\}} \cdot w(\boldsymbol{Z}, L)$ where w denotes the IS weight function is an unbiased estimator.

Our method $\mathrm{SIS}^{\mu,\sigma}$ proceeds as described in 5.2.1; SIS^{μ} sets the scale to unity while the SSIS methods also stratify. Once $\boldsymbol{Z} \mid T$ is sampled, we sample Y_k from $p_k(Z)$ independently. We also include SIS^* and SSIS^*, where the function $p_l(t)$ is estimated as before and the quantile function of the optimal distribution is estimated via the NINIGL algorithm, in our experiments; see also Figure 5.4.

Comparison

We compare SIS and SSIS to G&L IS by computing estimates, standard errors and computation times for $l \in \{100, 1000, 2000, 3000, 4000\}$. All methods require a calibration stage. For this comparison, we optimize the proposal distributions at each loss level of l separately and estimate the corresponding loss probability. Table 5.1 shows the estimated probabilities along with half-widths of estimated confidence intervals (CI) in brackets. The last column shows the average computational time of each method over all loss levels l. All examples used $n = 5000$ samples and 1000 samples for the calibration.

We see that all our methods lead standard errors smaller than G&L IS, while the estimated CIs for both methods are typically overlapping, supporting the correctness of both approaches. Given the small run-time, unbiasedness and small estimated errors, we can conclude that $\mathrm{SSIS}^{\mu,\sigma}$ is the best estimator for this problem. This supports our claim that the optimal density of T can be quickly and accurately approximated by a location scale transform of f_T. Note that SIS^* and SSIS^* are particularly slow, as it involves numerically approximation the quantile function corresponding to the optimal g_T.

l	100	1000	2000	3000	4000	Avg run-time (sec)
G&L IS	0.28	0.0079	0.00077	9.2e-05	1.1e-05	2.45
	(0.0078)	(0.00036)	(4.1e-05)	(6.3e-06)	(8.8e-07)	
SIS*	0.28	0.0081	0.00076	9.2e-05	1.1e-05	6.62
	(0.0068)	(0.00021)	(2.1e-05)	(2.4e-06)	(3.5e-07)	
SSIS*	0.28	0.0082	0.00077	9.5e-05	1.1e-05	12.56
	(0.0046)	(0.00014)	(1.4e-05)	(1.7e-06)	(2.5e-07)	
SIS$^\mu$	0.28	0.0077	0.00074	8.6e-05	1e-05	1.41
	(0.0086)	(0.00039)	(4.2e-05)	(5.5e-06)	(6.8e-07)	
SSIS$^\mu$	0.28	0.008	0.00075	9.1e-05	1.1e-05	1.45
	(0.0062)	(0.00028)	(2.9e-05)	(4e-06)	(5.1e-07)	
SIS$^{\mu,\sigma}$	0.28	0.0082	0.00077	9.4e-05	1.1e-05	2.45
	(0.0077)	(0.00034)	(3.3e-05)	(5.2e-06)	(4.6e-07)	
SSIS$^{\mu,\sigma}$	0.28	0.0081	0.00075	8.9e-05	1.1e-05	2.2
	(0.0059)	(2e-04)	(1.9e-05)	(2.3e-06)	(3e-07)	

Table 5.1: Estimates and CI halfwidths when estimating p_l in the Gaussian Credit Portfolio problem with $h = 1000$ obligors and $d = 10$ factors for various l and methods. The last column displays average run-times.

5.3.3 Tail probabilities of a t-copula credit portfolio

In this section, we apply SIS to a credit portfolio problem under a t-copula model, which is the model just studied with a multiplicative shock variable included. This t-copula model is a special case of the models with extremal dependence studied in [4]. Unlike the Gaussian copula, the t-copula is able to model tail dependence of latent variables, so simultaneous defaults of many obligors are more likely under the t-copula model than under its Gaussian

l	100	1000	2000	3000	4000
G&L IS	1.5	1.5	1.5	1.5	1.6
SIS*	1.3	1.3	1.2	1.7	1.5
SIS$^{\mu,\sigma}$	1.6	1.5	1.9	1.7	1.6
SSIS$^{\mu,\sigma}$	1	1.1	1	1.1	0.9

Table 5.2: Relative error reduction factors RE(MC)/RE(RQMC) for the Gaussian credit portfolio with $h = 1000$ obligors and $d = 10$ factors for various l and methods.

copula counterpart.

Problem formulation

In the t-copula model, the latent variables $\boldsymbol{X} = (X_1, \ldots, X_d)$ follow a multivariate t distribution, more precisely,

$$X_k = \sqrt{W}(a_{k1}Z_1 + \cdots + a_{kd}Z_d + b_k\varepsilon_k), \quad k = 1, \ldots, h,$$

where $W \sim \mathrm{IG}(\nu/2, \nu/2)$ is independent of $Z_1, \ldots, Z_d, \varepsilon_k \overset{\text{ind.}}{\sim} \mathrm{N}(0,1)$. Accordingly, we define $Y_k = \mathbb{1}_{\{X_k > t_\nu^{-1}(1-p_k)\}}$. We assume the same parameters as in the previous subsection, except that now we have $h = 50$ obligors, and the two settings for the degrees-of freedom $\nu \in \{5, 12\}$. Let $\boldsymbol{Z} = (Z_1, \ldots, Z_d)^\top$ and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_h)$. We consider two transformations. For the first transformation, let $Z_W = \Phi^{-1}(F_W(W))$ and

$$T_1(W, \boldsymbol{Z}, \boldsymbol{\epsilon}) = \beta_W Z_W + \boldsymbol{\beta}_L^\top \boldsymbol{Z},$$

where $\beta_W \in \mathbb{R}$ and $\boldsymbol{\beta}_L \in \mathbb{R}^d$ are such that $\beta_W^2 + \boldsymbol{\beta}_L^\top \boldsymbol{\beta}_L = 1$. Then, $T_1 \sim \mathrm{N}(0,1)$ since $Z_W \sim \mathrm{N}(0,1)$ is independent of $\boldsymbol{Z}$.

Our second transformation relies on the random variable $S_l(\boldsymbol{Z}, \boldsymbol{\varepsilon}) = \mathbb{P}(L > l \mid \boldsymbol{Z}, \boldsymbol{\varepsilon})$ where we note that $\mathbb{P}(L > l) = \mathbb{E}(S_l(\boldsymbol{Z}, \boldsymbol{\varepsilon}))$. Based on this and the fact that, given a sample $\boldsymbol{Z}, \boldsymbol{\varepsilon}$, the function S_l can be computed analytically, [13] propose to use CMC, i.e., estimating $\mathbb{P}(L > l)$ by the sample mean of $S_l(\boldsymbol{Z}_i, \boldsymbol{\varepsilon}_i)$ for independent $\boldsymbol{Z}_i, \boldsymbol{\varepsilon}_i$ for $i = 1, \ldots, n$. We propose to use this CMC idea combined with SIS by using the transformation

$$T_2 = \boldsymbol{\beta}_S^\top \boldsymbol{Z}$$

with $\boldsymbol{\beta}_S$ such that $\boldsymbol{\beta}_S^\top \boldsymbol{\beta}_S = 1$, which implies $T_2 \sim \mathrm{N}(0,1)$.

The second method based on CMC, is very effective as the variable W which accounts for a large portion of the variance of L, is integrated out. Furthermore, [13] additionally employ IS on $(\boldsymbol{Z}, \boldsymbol{\varepsilon})$ to make the event $\{L > l\}$ more frequent using the cross-entropy method; see [22, 107, 108]. We refer to Chan and Kroese's method as C&K CMC+IS. The numerical study in [13] demonstrates that C&K CMC+IS achieves substantial variance reduction. We will show in our numerical examples below that combining their CMC idea with our proposed single index IS method gives even greater variance reduction.

Fit of single-index models with and without conditional MC

We first investigate whether or not L and S_l have single-index structures. As before, the coefficients $\boldsymbol{\beta}$ that maximize the fit of the single-index model are estimated using the average derivative method of Stoker [113].

Figure 5.5 shows scatter plots of (T_1, L) and (T_2, S_l) for $\nu = 12$ and $\nu = 5$. The figures show that there is a strong association between T_1 and L but the dependence is stronger when $\nu = 12$ than when $\nu = 4$. When $\nu = 4$, there is a significant variation of L that cannot be captured by the single-index model based on T_1 in the right-tail. This observation holds more generally; the smaller ν (i.e., the stronger the dependence between the $\boldsymbol{X}_i$), the worse the fit of the single-index model becomes in the right-tail. When investigating the fit of (T_2, S_l), recall that the main advantage of CMC is that W is integrated out; the resulting estimators should be less sensitive to the degrees-of-freedom ν, which is the case in the plot. We can see that the fit of T_2 is excellent even in the outer right-tail for all settings of ν and l.

Comparison

We compare the original C&K CMC+IS from [13] with SIS with and without CMC. We additionally investigate whether employing RQMC yields a variance reduction. To this end, we estimate p_l for $l = 100$ for various n and methods; see Figures 5.7 and 5.8. Variances are estimated as the sample variance of $B = 20$ repetitions.

Note that for fixed ν, the data for C&K CMC+IS are identical independent of which transformation is used, so these lines can be used as reference. As expected, variances with the CMC idea are smaller than without the CMC idea. Note further that all our (S)SIS methods combined with T_1 (which does not integrate out W) give smaller variances than C&K CMC+IS, which does integrate out W.

5.4 Concluding remarks

In this chapter, we developed importance sampling and stratification techniques that are designed to work well for problems with a single-index structure, i.e., where the response variable depends on input variables mostly through some one-dimensional transformation. The main theme of our approach is to exploit the low-dimensional structure of a given problem in rare-event simulation by introducing a conditional sampling step on this important

transformed random variable and using optimal IS. Expressions for optimal densities of said one-dimensional transformation which achieve minimum variance were derived by Y. Taniguchi. The author of this thesis then provided a detailed description of a calibration stage in practice, as the optimal density expressions cannot be computed in practice.

The theoretical framework and numerical examples suggest substantial variance reduction for problems having strong single-index structures. Our numerical experiments revealed that the proposed methods outperform existing methods that were specifically tailored to the Gaussian and t-copula credit portfolio problem. The success of our method in this framework highlights the flexibility and wide applicability of our approach.

By combining our single-index framework with RQMC methods, we achieve even more precise estimation results, thanks to the dimension reduction feature of our conditional sampling step.

While our framework allows for non-linear transformations, these are typically difficult to find and sample from, which is why the numerical examples assumed T to be linear. The work in [94] could be helpful to find non-linear indices. We also note that there exist many other low-dimensional structures studied in the literature and they may provide a better fit than single-index models do. For instance, the structure assumed by the sufficient dimension reduction can be seen as a multi-index extension of the linear single-index model; see [16, 17, 1]. We would like to develop importance sampling techniques for problems based on other low-dimensional structures in future research.

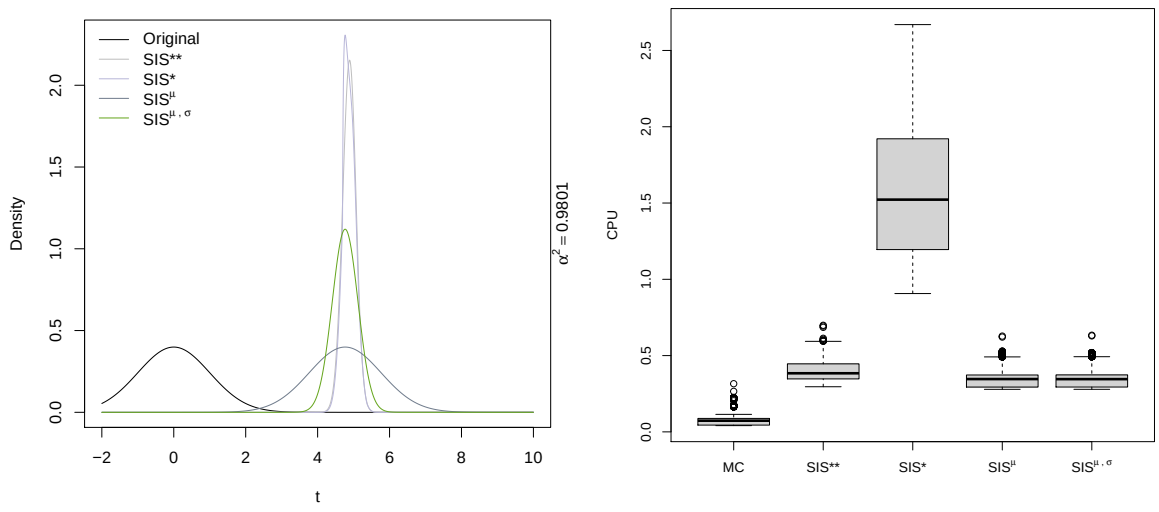

Figure 5.1: Left: Calibrated densities for $\alpha^2 = 0.99$, $l = 5$. Right: Run-times for each method including pilot runs.

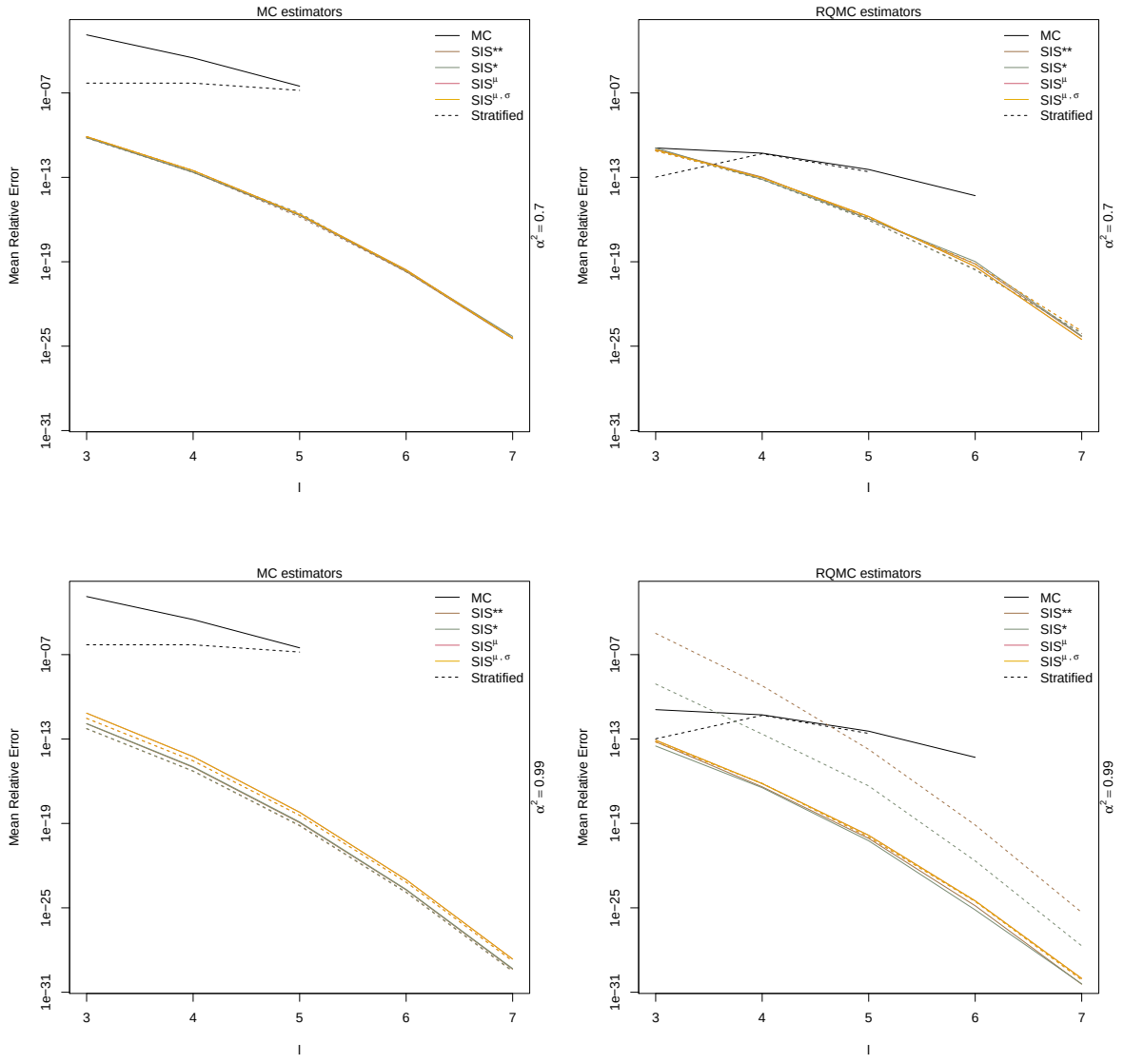

Figure 5.2: Mean relative errors when using pseudo-random numbers (left) and quasi-random numbers (right) for $\alpha^2 = 0.7$ (top) and $\alpha^2 = 0.99$ (bottom).

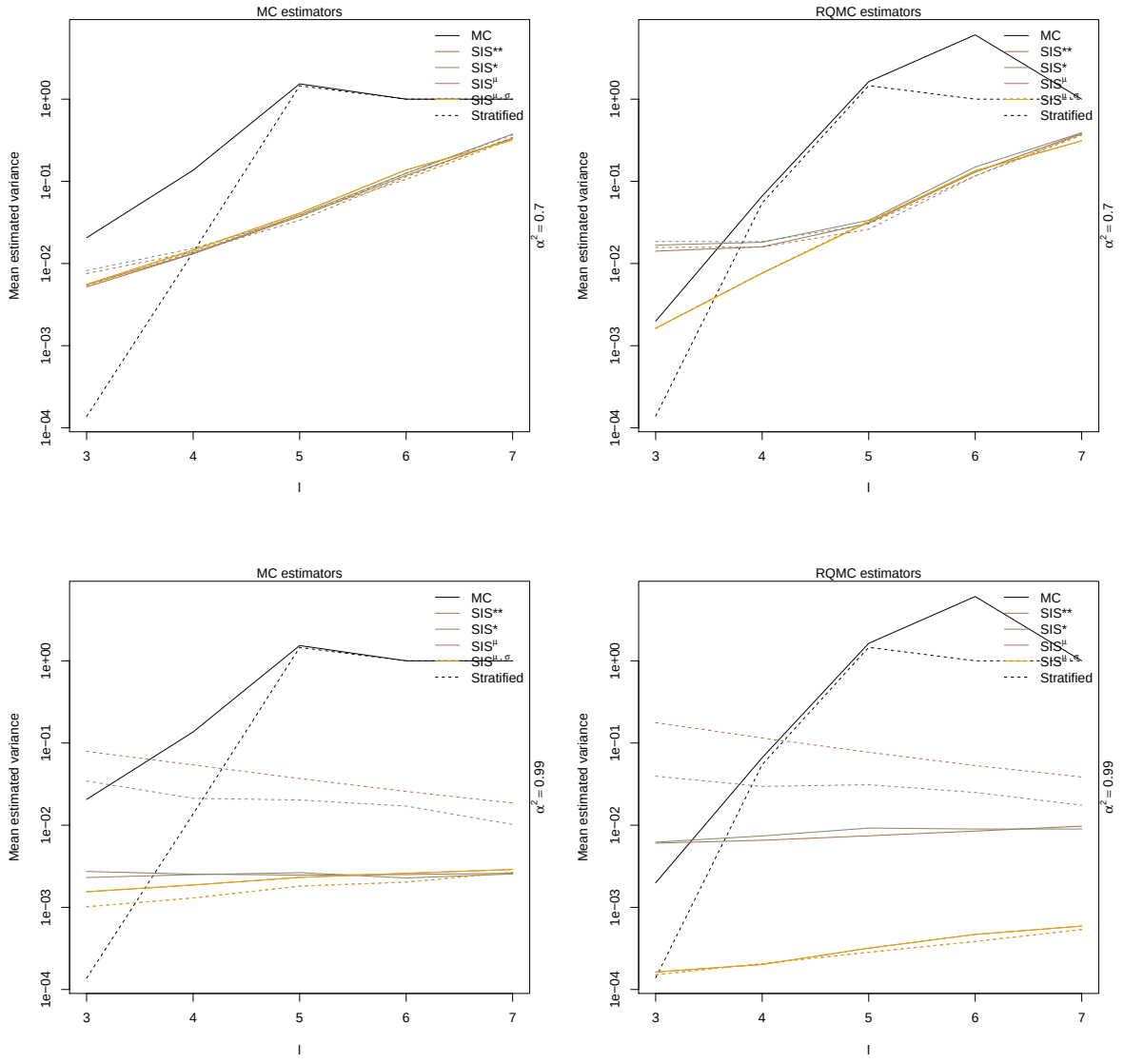

Figure 5.3: Mean estimated variance when using pseudo-random numbers (left) and quasi-random numbers (right) for $\alpha^2 = 0.7$ (top) and $\alpha^2 = 0.99$ (bottom).

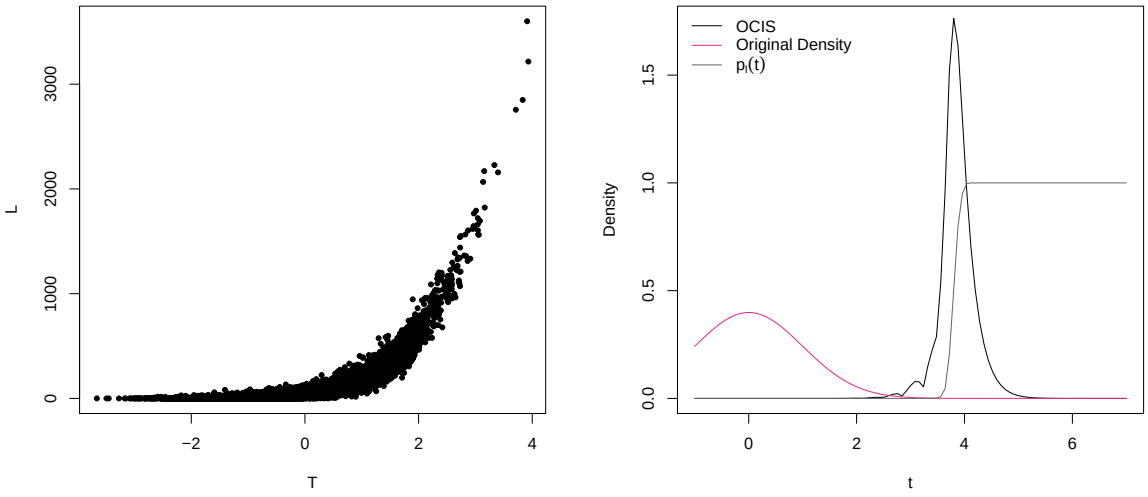

Figure 5.4: Plot of Transformed variable (T) vs Portfolio Loss (L) based on 10 000 observations (left) and OCIS density calibrated to $l = 3000$ (right).

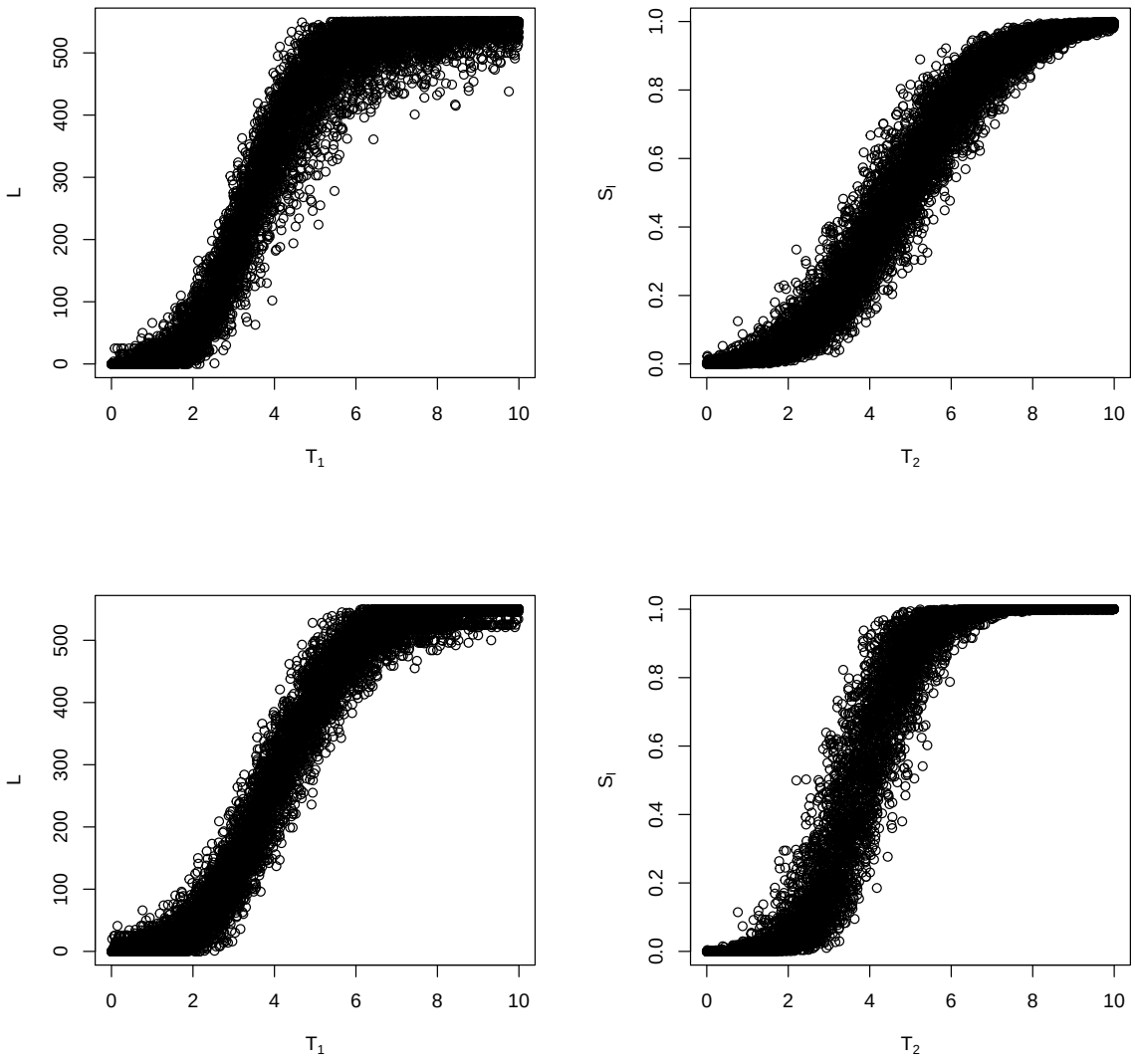

Figure 5.5: Scatter plots of L vs T_1 (left) and S_l vs T_2 (right) where $l = 500$ and $\nu = 5$ (top) and $\nu = 12$ (bottom).

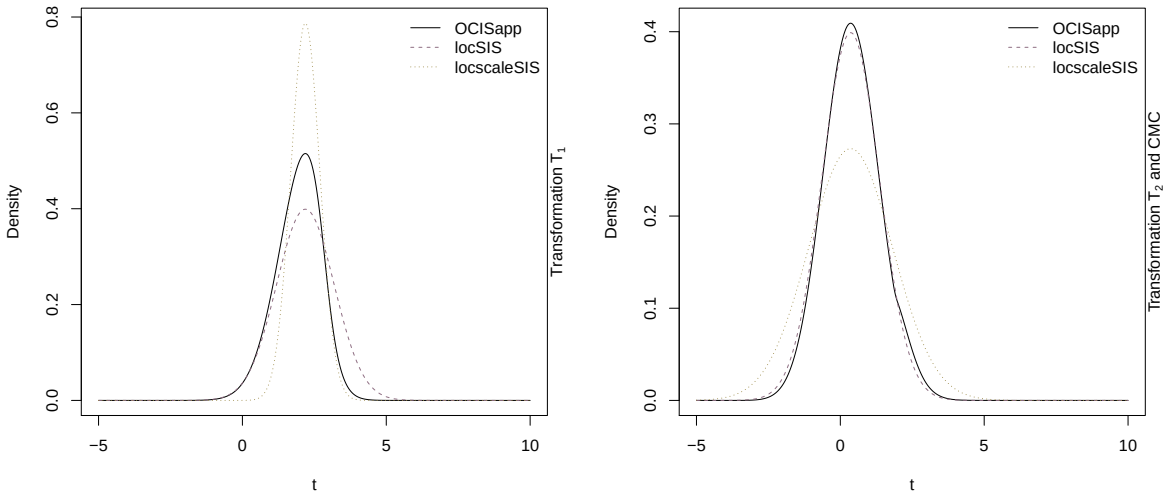

Figure 5.6: Optimally calibrated densities for $l = 100$ and the transformations T_1 (left) and T_2 (right).

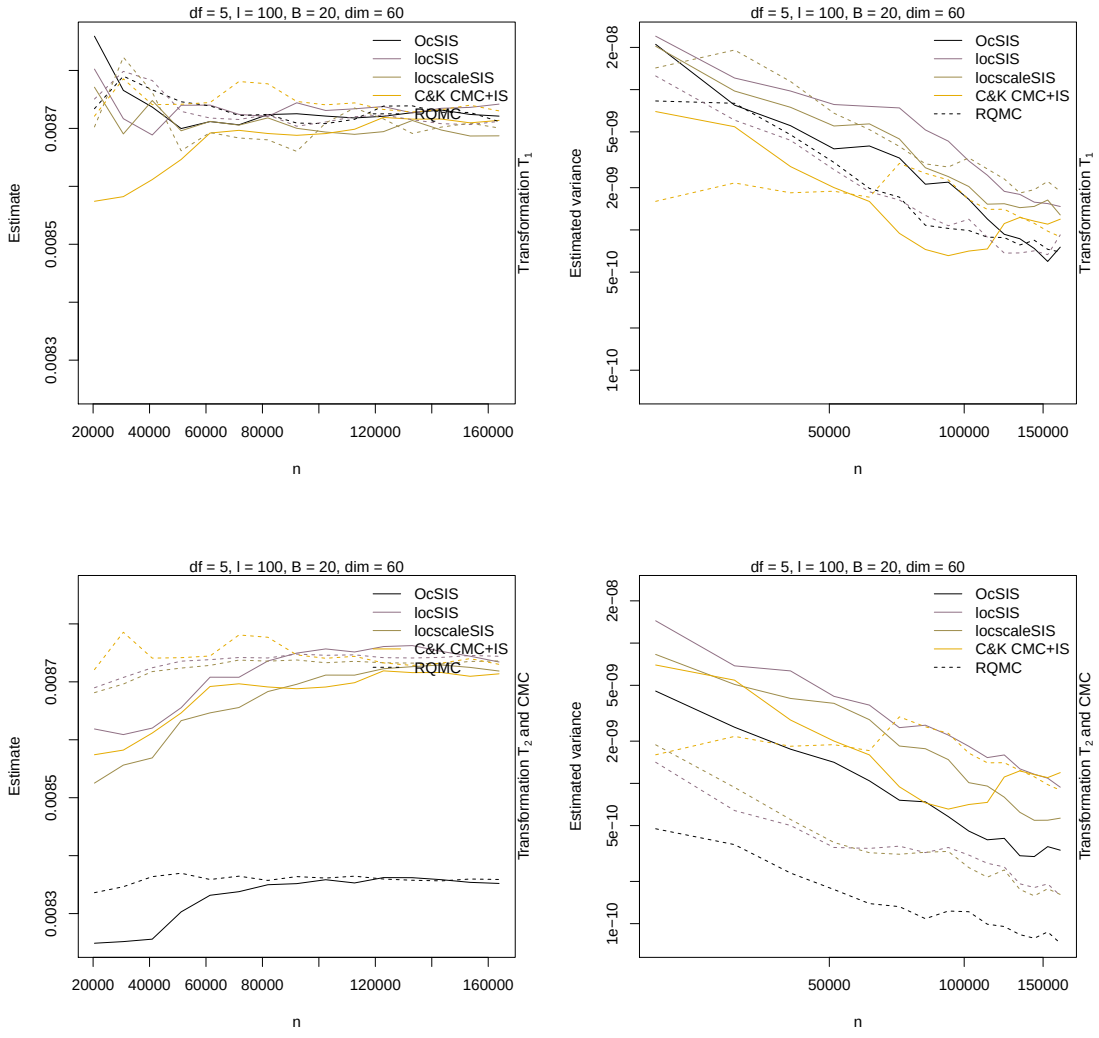

Figure 5.7: Estimates (left) and estimated variances (right) as a function of n for $\nu = 5$.

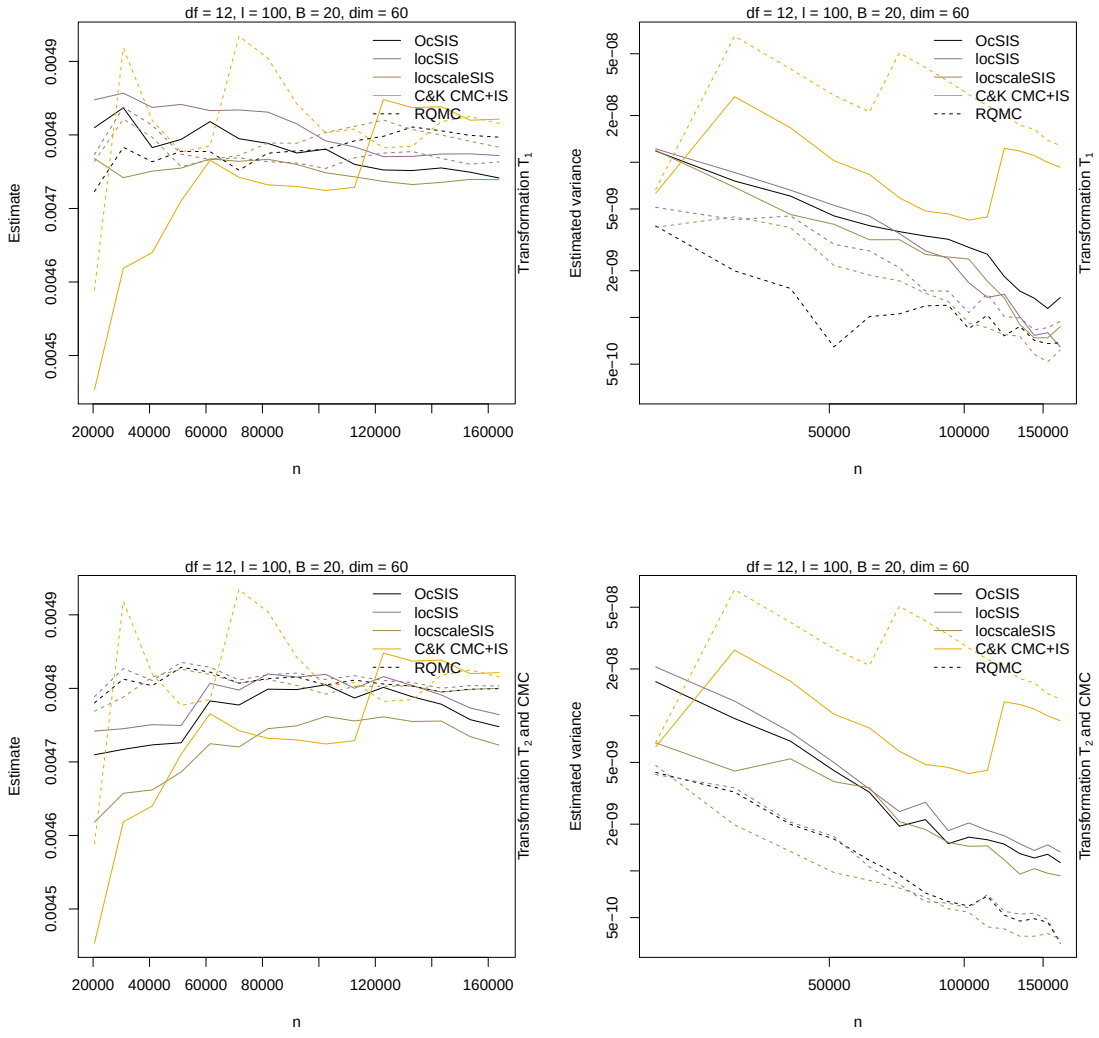

Figure 5.8: Estimates (left) and estimated variances (right) as a function of n for $\nu = 12$.

Chapter 6

RQMC on triangles

So far we have considered RQMC methods where the low discrepancy sequence was constructed on the unit cube $[0,1)^d$. There are, however, also constructions over non-cubical spaces, such as triangles and spheres, that enjoy wide applicability; see [5]. In this chapter, we focus on the problem of integrating functions over triangles, a problem appearing in computer graphics.

Recall from the introduction that [6] recently provided two constructions of points with low discrepancy on the triangle, the triangular vdC sequence and the triangular Kronecker lattice. Both their methods can be used to sample deterministic points with vanishing parallelogram discrepancy, though only the lattice approach achieves the optimal order $\mathcal{O}(\log(n)/n)$; the parallelogram discrepancy measure is similar to the one used in [10]. A potential limitation of their lattice approach is that it is not extensible: If an estimator is constructed based on n_1 points and it is found that a larger sample size, say $n_2 > n_1$ is needed to obtain an error small enough, one needs to generate a new point-set of size n_2, rather than just adding $n_2 - n_1$ points to the already constructed lattice.

The contributions of this chapter are the following: i) We provide an extensible rank-1 lattice construction for points in the triangle; ii) We show that the triangular vdC with n points projects onto $2\sqrt{n}$ points on the x- and y-axis; and iii) we perform a numerical study comparing all methods, including some of the transformation methods in [100], which has not been done in either [6] or [45]. These results will be published in [27].

6.1 Background

Triangles

Let $A, B, C \in \mathbb{R}^d$ not on one line. We define the triangle spanned by A, B and C as

$$\triangle(A, B, C) = \left\{ \lambda_1 A + \lambda_2 B + \lambda_3 C \mid \min\{\lambda_j\} \geq 0, \sum_{j=1}^{3} \lambda_j = 1 \right\}.$$

We can without loss of generality consider $A, B, C \in \mathbb{R}^2$. We often construct the point sets on special triangles, such as the equilateral triangle

$$\triangle_E = \triangle\left((0,0), (1,0), (1/2, \sqrt{3}/2)\right)$$

or the right-angle triangle

$$\triangle_R = \triangle\left((0,0), (0,1), (1,0)\right).$$

Assume we constructed points on the triangle $\triangle = \triangle(A, B, C)$ and would like to map them to the triangle $\triangle' = \triangle(A', B', C')$. This can be done as follows: Define the matrices

$$M(\triangle) = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad M(\triangle') = \begin{pmatrix} a_1' & b_1' & c_1' \\ a_2' & b_2' & c_2' \\ 1 & 1 & 1 \end{pmatrix},$$

and let $M(\triangle, \triangle') = M(\triangle')M(\triangle)^{-1}$. If the point $\boldsymbol{x} = (x, y) \in \triangle$, then the point $\boldsymbol{x}' = (x', y') \in \triangle'$ where (x', y') are the first two components of the matrix-vector product

$$(x', y', z') = M(\triangle, \triangle') \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}. \tag{6.1}$$

Discrepancy

We follow [6] and describe the discrepancy measures used. Let $\Omega \in \mathbb{R}^d$ be bounded and denote by λ the Lebesgue measure. Let Ω be such that $\lambda(\Omega) > 0$; if Ω lies on a flat subset, define λ as the Lebesgue measure with respect to the lowest-dimensional flat. Next, define the normalized restriction of λ to Ω via $\lambda_\Omega(S) = \lambda(S \cap \Omega)/\lambda(\Omega)$ for $S \in \mathcal{B}(\mathbb{R}^d)$. For a point

set $P_n = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subset \Omega$ and set $S \subset \Omega$, let $\alpha(P_n, S) = \sum_{\boldsymbol{x} \in \mathbb{P}_n} \mathbb{1}_{\{\boldsymbol{x} \in S\}}$ be the number of points of P_n that fall into S and denote by $\delta_n(S; P_n, \Omega) = \lambda_\Omega(S) - \alpha(P_n; S)/n$ the *signed* discrepancy of P_n at S. Now let $\Omega = \triangle(A, B, C)$ where $A, B, C \in \mathbb{R}^2$. For $\lambda_1, \lambda_2 > 0$, let $T_{\lambda_1,\lambda_2,C}$ be the parallelogram with point C and vectors $\lambda_1(A - C)$ and vector $\lambda_2(B - C)$. Next, denote by

$$\mathcal{S}_C = \{T_{\lambda_1,\lambda_2,C} \mid 0 \le \lambda_1 \le \|A - C\|, 0 < \lambda_2 < \|B - C\|\}$$

the set of all such triangles anchored at C. Define $\mathcal{S}_A$ and $\mathcal{S}_B$ analogously. The *parallelogram discrepancy* is defined as

$$D_n^P(P_n; \Omega) = D_n(\mathcal{S}_A \cup \mathcal{S}_B \cup \mathcal{S}_C, P_n, \Omega),$$

where

$$D_n(\mathcal{S}, P_n, \Omega) = \sup_{S \in \mathcal{S}} |\delta_n(S; P_n, \Omega)|.$$

The optimal discrepancy bound is $\mathcal{O}(\log n/n)$. If we replace the set $\mathcal{S}_A \cup \mathcal{S}_B \cup \mathcal{S}_C$ by $\mathcal{S} = \{[0, \boldsymbol{x}) \mid \boldsymbol{x} \in [0, 1)^2\}$ we obtain the discrepancy measure defined in [100], say D_n^{PC}, which resembles the star discrepancy typically considered in the classical QMC literature over the cube. [6, Lemma 2.1] shows that if $\Omega = \triangle((0, 0), (0, 1), (1, 1))$, then $D_n^{PC}(P_n, \Omega) \le 2D_n^P(P_n, \Omega)$; that is, if the parallelogram discrepancy vanishes, so does the one defined [100] with at least the same rate.

Transforming a low-discrepancy point set from $[0, 1)^2$ to $\triangle(A, B, C)$

A natural and popular approach to sample points from some domain Ω is to find a mapping $\phi : [0, 1)^d \to \Omega$ so that $\phi(\boldsymbol{U}) \sim \mathrm{U}(\Omega)$ for $\boldsymbol{U} \sim \mathrm{U}(0, 1)^d$, where $d \in \mathbb{N}$ is fixed. That is, for some integrable function $f : \Omega \to \mathbb{R}$, we find

$$\mu = \int_\Omega f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \lambda(\Omega) \int_{[0,1)^d} f(g(\boldsymbol{u})) \, \mathrm{d}\boldsymbol{u},$$

where λ denotes the Lebesgue measure. Let $P_n = \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\} \subset [0, 1)^d$. The Koksma–Hlawka inequality originally appearing in [58] implies that

$$\left| \frac{\lambda(\Omega)}{n} \sum_{i=1}^n f(\phi(\boldsymbol{u}_i)) - \mu \right| \le D_n^*(P_n) V^{\mathrm{HK}}(f \circ \phi),$$

where D_n^* denotes the star-discrepancy of a point set and V^{HK} the variation in the sense of Hardy and Krause. Note that $V^{\text{HK}}(\phi) < \infty$ and $V^{\text{HK}}(f) < \infty$ do not imply that $V^{\text{HK}}(f \circ \phi) < \infty$; see [7] for conditions on ϕ and f.

For $\Omega = \triangle_R$ being a triangle, [100] give six possible transformations ϕ to map $[0,1)^2$ to $\triangle_R$. We mention two of them.

1. Transformation *mirror*. Take the point set on $[0,1)^2$, leave the points that are already in $\triangle$ and reflect the other ones at $(1/2, 1/2)$. The resulting transformation is fast, but discontinuous. As such, $V^{\text{HK}}(f \circ \phi^{\text{mirror}}) < \infty$ cannot be guaranteed.

2. Transformation *root*. This transformation is due to [31] and given by

$$\phi(u_1, u_2) = (1 - \sqrt{u_1}, \sqrt{u_1} u_2).$$

   This transformation is smooth and [7] show that $V^{\text{HK}}(f \circ \phi) < \infty$ for all functions $f \in C^2(\triangle)$.

A method to sample from any multivariate distribution is the inverse Rosenblatt transform (also known as the conditional distribution method); see [106]. In order to sample $(U_1, U_2) \sim \text{U}(\triangle_E)$, the idea is to first sample U_1 and then $U_2 \mid U_1$; see Algorithm 6.1.1.

Algorithm 6.1.1 (Inverse Rosenblatt transform to sample from $\text{U}(\triangle_E)$)
Sample $(U_1, U_2) \sim \text{U}(\triangle_E)$ via:

1. Sample $V_1, V_2 \overset{\text{ind.}}{\sim} \text{U}(0,1)$.

2. Set
$$U_1 = \begin{cases} \sqrt{V_1/2}, & \text{if } V_1 \leq 1/2, \\ 1 - \sqrt{(1-V_1)/2}, & \text{otherwise.} \end{cases}$$

3. Set
$$U_2 = \begin{cases} V_2 U_1 \tan \pi/3, & \text{if } U_1 \leq 1/2, \\ V_2 (1 - U_1) \tan \pi/3, & \text{otherwise.} \end{cases}$$

4. Return (U_1, U_2).

Proposition 6.1.2
The point (U_1, U_2) returned by Algorithm 6.1.1 satisfies $(U_1, U_2) \sim \text{U}(\triangle_E)$.

Proof. We need to sample from $(U_1, U_2) \sim f$ where

$$f(u_1, u_2) = \begin{cases} \frac{2}{\sin \pi/3} & \text{if } 0 \le u_1 \le 1/2 \text{ and } u_2 \le u_1 \tan \pi/3 \text{ or } 1/2 \le u_1 \le 1 \text{ and } (1-u_1)\tan \pi/3 \\ 0 & \text{otherwise.} \end{cases}$$

The marginal pdf of U_1 is given by

$$f_{U_1}(u_1) = \begin{cases} \int_0^{u_1 \tan \pi/3} \frac{2}{\sin \pi/3} \mathrm{d}u_2 = \frac{2u_1}{\cos \pi/3} = 4u_1 & \text{if } 0 \le u_1 \le 1/2 \\ \int_0^{(1-u_1)\tan \pi/3} \frac{2}{\sin \pi/3} \mathrm{d}u_2 = \frac{2(1-u_1)}{\cos \pi/3} = 4(1-u_1) & \text{if } 1/2 < u_1 \le 1; \end{cases}$$

the corresponding distribution function is then

$$F(u_1) = \begin{cases} 2u_1^2 & \text{if } 0 \le u_1 \le 1/2 \\ 1 - 2(1-u_1)^2 & \text{if } 1/2 < u_1 \le 1. \end{cases}$$

We find $F^{-1}(v_1) = \sqrt{v_1/2}$ if $v_1 \in [0, 1/2]$ and $F^{-1}(v_1) = 1 - \sqrt{(1-v_1)/2}$ otherwise. As $U_1 = F^{-1}(V_1)$ for $V_1 \sim \mathrm{U}(0,1)$ in Algorithm 6.1.1, we find that U_1 has the correct distribution.

Next, the conditional density function $f_{U_2|U_1}(u_2 \mid u_1)$ is given by

$$\begin{cases} \frac{2/\sin \pi/3}{2u_1/\cos \pi/3} = \frac{1}{u_1 \tan \pi/3} & \text{if } 0 \le u_1 \le 1/2 \text{ and } u_2 \le u_1 \tan \pi/3, \\ \frac{2/\sin \pi/3}{2(1-u_1)\cos \pi/3} = \frac{1}{(1-u_1)\tan \pi/3} & \text{if } 1/2 < u_1 \le 1 \text{ and } u_2 \le (1-u_1)\tan \pi/3. \end{cases}$$

It is easy to see that $U_2 \mid U_1 = u_1 \sim \mathrm{U}(0, b(u_1))$ where $b(u_1) = u_1 \tan \pi/3$ if $0 \le u_1 \le 1/2$ and $b(u_1) = (1-u_1)\tan \pi/3$ otherwise. Thus, U_2 in Algorithm 6.1.1 has the correct distribution and the result follows from the inverse Rosenblatt transform; see [106]. □

6.2 Lattice constructions

6.2.1 Triangular lattice construction of Basu and Owen

K. Basu and A. Owen give a lattice construction for points on the triangle with optimal discrepancy. To this end, let $\alpha \in (0, 2\pi)$ be such that $\tan(\alpha)$ is a quadratic irrational number, i.e., $\tan(\alpha) = (a + b\sqrt{c})/d$ for $b, d \neq 0$ and $c > 0$ not a perfect square. The point set P_n obtained by rotating the lattice $(2n)^{-1/2}\mathbb{Z}^2$ counterclockwise by α and intersecting with $\triangle$ satisfies $D_n^P(P_n, \triangle_R) \le C \log(n)/n$. The following algorithm produces such a point set.

Algorithm 6.2.1 (Basu and Owen Lattice)
Given target sample size n, α such that $\tan(\alpha)$ is badly approximable (e.g., $\alpha = 3\pi/8$), a random vector $\boldsymbol{U} \sim \mathrm{U}(0,1)^2$ (optional), an integer N and a target triangle $\triangle$, sample n points as follows:

1. Let $P = \{0, 1, \ldots, N-1\}^2$ and set $\boldsymbol{x} \leftarrow (\boldsymbol{x} + \boldsymbol{U})/N$ for $\boldsymbol{x} \in P$.
2. Map $\boldsymbol{x} \leftarrow 2\boldsymbol{x} - 1 \in [-1, 1]^2$ for all $\boldsymbol{x} \in P$.
3. Set $\boldsymbol{x} \leftarrow \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \boldsymbol{x}$ for all $\boldsymbol{x} \in P$.
4. Set $P_n = P \cap \triangle_R$.
5. If $|P_n| \neq n$, add or remove $|P_n| - n$ points in $\triangle_R$ to P_n.
6. Map the points in P_n to $\triangle$ using (6.1) and return.

Assume that we used a random vector $\boldsymbol{U} \sim \mathrm{U}(0,1)^2$ in Step 1. This randomizes the otherwise deterministic points so that $\boldsymbol{x} \sim \mathrm{U}(\triangle)$ for all $\boldsymbol{x} \in P_n$, which follows readily by observing that the density of each $\boldsymbol{x}$ is constant if a randomization is performed. In Step 5, we can choose arbitrarily which points to add or remove. We remark that the algorithm presented in [7, p. 757] differs slightly from Algorithm 6.2.1 in that their version uses $N = \lceil \sqrt{2n} \rceil + 1$ and the lattice $\{-N, \ldots, N\}^2$.

6.2.2 Extensible triangular lattice constructions

The construction in Algorithm 6.2.1 is non-extensible. In this section, we propose an extensible scheme for which we make use of the one-dimensional van der Corput sequence.

Recall that the ith point of an extensible rank-1 lattice sequence with generating vector $\boldsymbol{z} \in \mathbb{Z}^d$ is defined as

$$\boldsymbol{u}_i = \phi_b(i)\boldsymbol{z} \bmod 1 \in [0,1)^d,$$

where $\phi_b(i)$ is the ith term of the van der Corput sequence in base b, and the mod 1 operation is applied component-wise; see [51].

If we want to use this idea to define a triangular Kronecker lattice sequence, then we need to introduce some generalization of this idea beyond the rank 1 case. In particular, the grid that needs to be generated, given by

$$P = \left\{ -\frac{N}{N}, -\frac{N-1}{N}, \ldots, -\frac{1}{N}, 0, \frac{1}{N}, \ldots, \frac{N}{N} \right\}^2$$

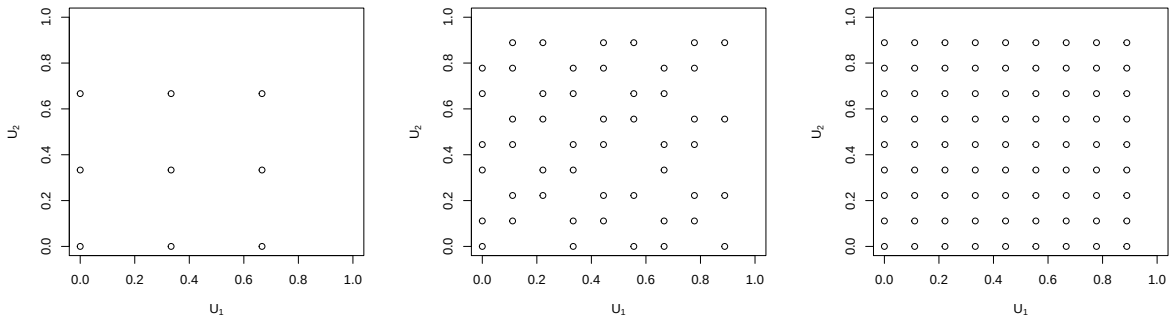

Figure 6.1: First 9 (left), 50 (middle) and 81 points when using $b = 3$.

is a rank-2 lattice rather than a rank-1 lattice. We also note that the only reason why Basu and Owen use a grid over $[-1, 1]^2$ instead of $[0, 1)^2$ seems to be that when they rotate the points from the grid, they can just take the intersection with $\triangle_R$ without having to perform any modulo 1 operation. However with our proposed approach to generate an extensible grid, we will focus on generating points in $[0, 1)^2$; we can then either make use of modulo 1 operations or extend the grid to $[-1, 1]^2$.

First, we must choose the base b in which the grid will be constructed. The choice of base b is such that whenever n is of the form b^{2k} for some $k \geq 1$, the point set P_n obtained with this method will be exactly the same as if we had proceeded with the fixed-size approach based on $N = b^k$; see Proposition 6.2.3 below.

We then make use of the base b decomposition of i as $i = \sum_{j \geq 0} d_j b^j$ to define the point

$$\boldsymbol{u}_i = \left((u_{i,1}, u_{i,2}) = \frac{d_0}{b}(1, 1) + \frac{d_1}{b}(0, 1) + \frac{d_2}{b^2}(1, 1) + \frac{d_3}{b^2}(0, 1) + \dots \right) \bmod 1. \tag{6.2}$$

Note that the modulo 1 operation is only necessary for the second coordinate, as we have the bound

$$\frac{d_0}{b} + \frac{d_2}{b^2} + \frac{d_4}{b^3} + \dots < 1.$$

Figure 6.1 shows the first 9, 50, and 81 points obtained with this method when $b = 3$.

The points $\boldsymbol{u}_i$ lie in $[0, 1)^2$. Next, we rotate the points by α and can then proceed as in Algorithm 6.2.1.

Algorithm 6.2.2 (Extensible Kronecker Lattice)
Given α, n, b, a random vector $\boldsymbol{U} \sim \mathrm{U}(0,1)^2$ (optional) and a skip $s \geq 0$, sample n points in $\triangle$ as follows:

1. Set $P_n = \{\}$ and $k = 0$.
2. While $|P_n| < n$,
   (a) Compute digits d_j such that $k + s = \sum_{j \geq 0} d_j b^j$ and compute $\boldsymbol{u}$ in (6.2).
   (b) Set $k \leftarrow k + 1$, $\boldsymbol{u} \leftarrow 2(\boldsymbol{u} + \boldsymbol{U} \bmod 1) - 1 \in [-1, 1]^2$.
   (c) Set $\boldsymbol{u} \leftarrow \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \boldsymbol{u}$.
   (d) If $\boldsymbol{u} \in \triangle_R$, set $P_n = P_n \cup \{\boldsymbol{u}\}$.
3. Map each $\boldsymbol{u} \in P_n$ to $\triangle$ using (6.1) and return.

Proposition 6.2.3
Let $n = b^{2k}$, $k > 0$ and $s = 0$. The sets P_n of points obtained by Algorithms 6.2.1 with $N = b^k$ and Algorithm 6.2.2 coincide.

Proof. Assume wlog $\boldsymbol{U} = 0$. Consider the set $Q = \{\boldsymbol{u}_i : i = 0, \ldots, n-1\}$ where the $\boldsymbol{u}_i$ are as in (6.2). Then Q is a rank-2 lattice and can be written as $Q = \{(i/b^k, j/b^k) : 0 \leq i, j < b^k\}$. This is exactly the rectangular grid in Step 1 of Algorithm 6.2.1. Since the remaining operations (intersection and rotation) are identical in both algorithms, the result follows. □

Remark 6.2.4
Algorithm 6.2.2 is based on a rank-2 lattice with generating vectors $\boldsymbol{z}_1 = (1, 1), \boldsymbol{z}_2 = (0, 1)$. In our numerical experiments, we also consider a rank-1 lattice with generating vector $\boldsymbol{z} = (1, 182667)$; see [18, p. 26].

6.3 Triangular van der Corput sequence of Basu and Owen

Recall that the ith point of the one-dimensional van der Corput sequence in base b is given by $u_i = \phi_b(i-1)$ where the radical inverse function ϕ_b is defined as

$$\phi_b(i) = \sum_{k \geq 0} d_k b^{-k-1}, \quad i = \sum_{k \geq 0} d_k b^k \in \{0, 1, \ldots\}.$$

Similarly to how this sequence puts points at the left endpoints of intervals $[b^{-m}, b^{-m+1})$, the triangular van der Corput sequence of [6] replaces the intervals by $b^m = 4^m$ congruent sub-triangles and puts the points in the centers. Let $T = \triangle(A, B, C)$ be the triangle we wish to generate points on. Define the sub-triangle of T with index d for $d \in \{0, 1, 2, 3\}$ as

$$T(d) = \begin{cases} \triangle\left(\frac{B+C}{2}, \frac{A+C}{2}, \frac{A+B}{2}\right), & d = 0, \\ \triangle\left(A, \frac{A+B}{2}, \frac{A+C}{2}\right), & d = 1, \\ \triangle\left(\frac{B+A}{2}, B, \frac{B+C}{2}\right), & d = 2, \\ \triangle\left(\frac{C+A}{2}, \frac{C+B}{2}, C\right), & d = 3. \end{cases} \tag{6.3}$$

Let $i \geq 0 = \sum_{k\geq 0} d_k 4^k$, where the sum breaks after at most $K_i = \lceil \log_4(i) + 1 \rceil$ digits. The construction in [6] obtains its ith triangular point by mapping the integer i to the midpoint (mdpt) of the triangle $T_{(d_0,\ldots,d_{K_i})} = T(d_0, \ldots, d_{K_i})$ recursively defined by $T(d_k, d_{k+1}) = T(T(d_k))(d_{k+1})$, where $\text{mdpt}(A, B, C) = (A + B + C)/3$ componentwise; see Figure 6.2 for an example.

Algorithm 6.3.1 (Triangular vdC sequence of K. Basu and A. Owen)
Given input $n \geq 1$ and target triangle $\triangle(A, B, C)$, generate the first n points as follows:

1. For $i = 1, \ldots, n$,
   (a) Compute $(d_0, \ldots, d_{K_i})$ such that $i - 1 = \sum_{k\geq 0} d_k 4^k$;
   (b) Initialize $A' = A, B' = B, C' = C$;
   (c) For $j = 0, \ldots, K_i$,
   Update $\triangle(A', B', C') = T(d_j)$ using (6.3);
   (d) Set $\boldsymbol{x}[i] = \text{mdpt}(A', B', C')$;
2. Return $\boldsymbol{x}$;

The discrepancy of the first $n \geq 1$ points was shown in [6, Theorem 3.3] to be bounded above by $12/\sqrt{n}$, which means there is a gap to the optimal order $\mathcal{O}(\log n/n)$. Nevertheless, Algorithm 6.3.1 has some advantages: i) it gives rise to an extensible sequence; ii) it can be randomized; iii) it is easily implemented. However, the points suffer from poor projection properties, as shown in the following proposition.

Proposition 6.3.2
Let $T = \triangle((0, 0), (0, 1), (1, 0))$ and denote by P_n the point set consisting of the first n points produced by Algorithm 6.3.1 for $n = 4^k$ and $k > 2$. Then the projections of P_n onto the x- and y-axis contain $2\sqrt{n} = 2^{k+1} < n$ points.

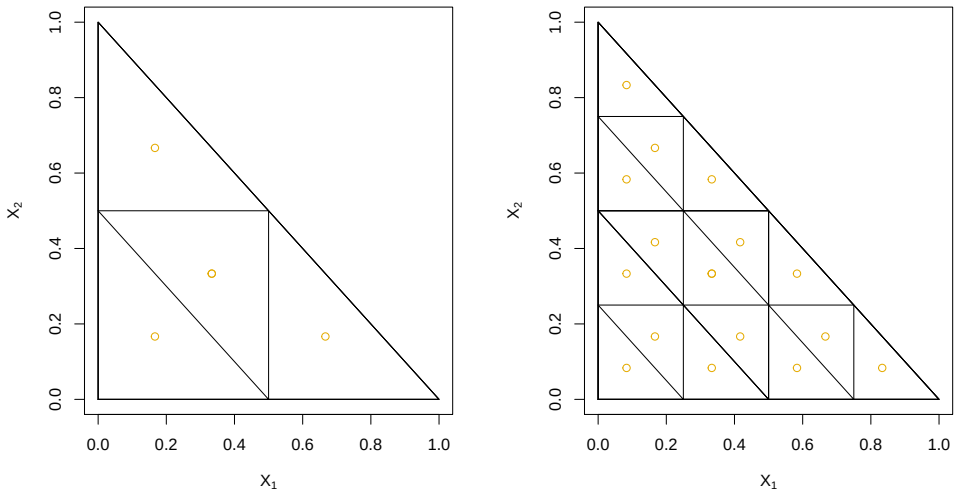

Figure 6.2: Triangular vdC points; $n = 4$ (left) and $n = 16$ (right).

Proof. Since $n = 4^k$, each of the 4^k subtriangles contains one point, and there are 2^k rows of subtriangles. In each row j, the midpoints of all upside triangles have the same y coordinate, say y_{j1}. Similarly, the midpoints of all downside triangles in the same row have the same y coordinate, say y_{j2}. Since there are only these two cases, the point set projects on $\{y_{ji} : j = 1, \ldots, 2^k; i = 1, 2\}$, which has $2 \cdot 2^k = 2^{k+1}$ elements. The x axis case follows similarly. □

[45] generalize this method by replacing the support $\{0, 1, 2, 3\}$ of the transformation in (6.3) by $\{(0, 0), (1, 0), (0, 1), (1, 1)\} = \mathbb{F}_2^2$, where the input strings (from $\mathbb{F}_2^m$ for some m or $\mathbb{F}_2^\infty$) come from a digital net. They prove that their construction gives worst case error in $\mathcal{O}((\log n)^3/n)$ for functions in $C^2(\triangle)$. Furthermore, they show that their construction includes the vdC sequence of Basu and Owen; see [45, p. 369].

As mentioned in [6], one can use scrambling to randomize the deterministic triangular vdC sequence. Not only does this have the advantage that the integration error can be estimated, it also makes sure that, with probability 1, the projections onto the x and y axis contain n points.

6.4 Numerical experiments

Figure 6.3 displays four different triangular point sets, each with $n = 4^5 = 1024$ points. In the case of the transformations "mirror" and "root", the underlying $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n \in [0,1)^2$ come from a deterministic Sobol' sequence. From these plots, it is evident that the "mirror" transformation leaves the largest gaps and is visually outperformed by transformation "root". For the van der Corput point set on the top-right, one can immediately see that this point set suffers from poor projection properties. This is not the case for the triangular Kronecker lattice point set displayed on the bottom left.

Unlike the triangular vdC points, the lattice construction allows for an easy randomization by shifting the underlying grid by a uniform vector. Figure 6.4 displays five independently randomized copies of the lattice points, each having a different color.

To test the performance of the different triangular constructions, we consider the three test functions

$$\begin{aligned} f_1(x,y) &= (|x-y|+y)^d, \\ f_2(x,y) &= \cos(2\pi\beta + \alpha_1 x + \alpha_2 y), \\ f_3(x,y) &= x^{\alpha_3} + y^{\alpha_3}, \end{aligned}$$

where $\beta = 0.4$, $d = -0.9$, $\alpha_1 = e^3$, $\alpha_2 = e^2$, $\alpha_3 = 1.5$ and estimate $\mu_j = \int_{\triangle_E} f_j(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$, whose theoretical values are known for $j = 1, 2, 3$. The first two test functions were also used in [100]. Note that f_1 has a singularity, while f_2 is a smooth oscillatory function. Integrating the function f_3, being a sum of univariate functions, might reveal if the projection properties of the point sets matter. As such, we can already anticipate that f_2 is easier to integrate. Figure 6.5 displays f_k for $k = 1, 2, 3$ and the standard parameter settings.

Figure 6.6 displays absolute errors obtained when integrating f_k for $k = 1, 2, 3$ for various sample sizes n between 2^4 and 2^{15} and various methods. The methods `lattice1` and `lattice2` correspond to the rank-1 and rank-2 constructions.

So far, we have only looked at deterministic point sets. Next, we compare the following randomized methods to estimate μ_j: i) PRNG + root; ii) rSobol' + root, iii) rLattice1, iv) rlattice2 and v) rvdC (randomized vdC). For each method and each sample size, $B = 20$ randomizations were used. The estimates are obtained as the sample average of the realizations, while the variance is estimated as the sample variance of the independent draws. Figure 6.7 displays the results. Note that "rsobol+root" is typically the best performing method, while the randomized lattice or vdC approach still significantly outperform MC.

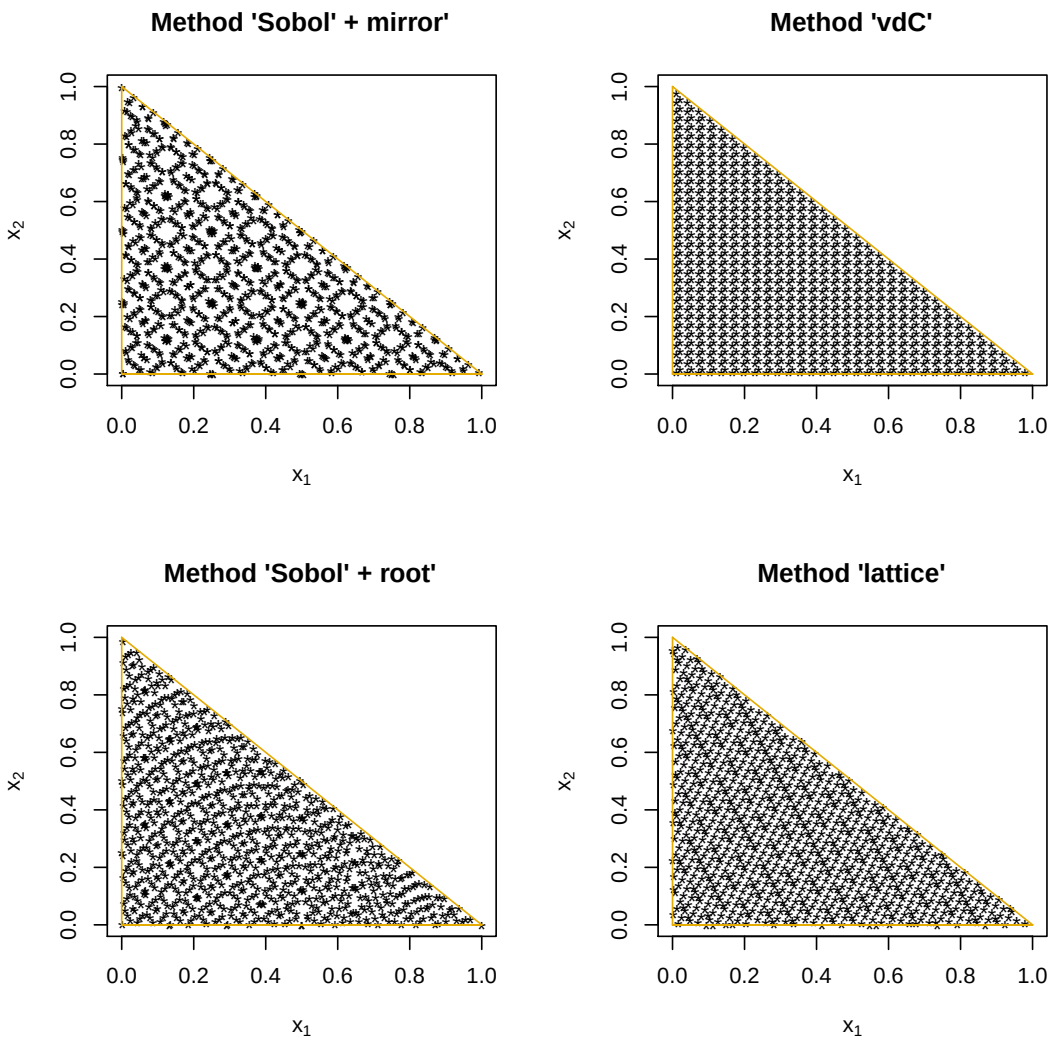

Figure 6.3: $n = 4^5$ points sampled from different methods.

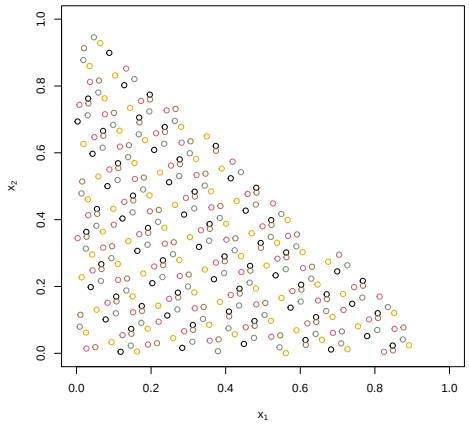

Figure 6.4: 5 independently randomized triangular Kronecker lattice point sets with 2^6 points each.

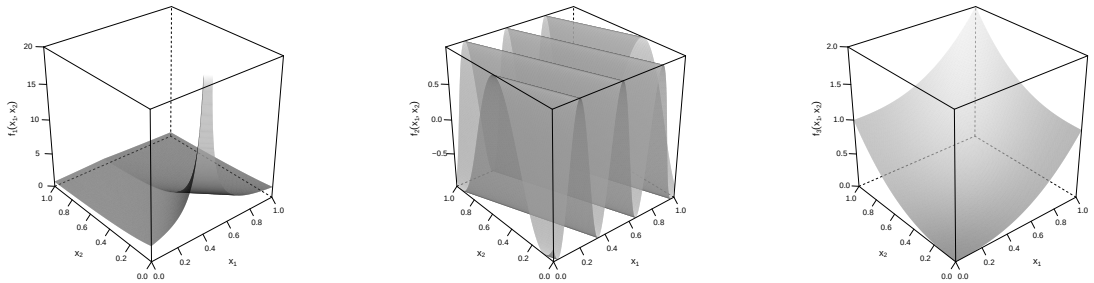

Figure 6.5: Test-functions f_1 (left), f_2 (middle) and f_3 (right) used in the numerical study.

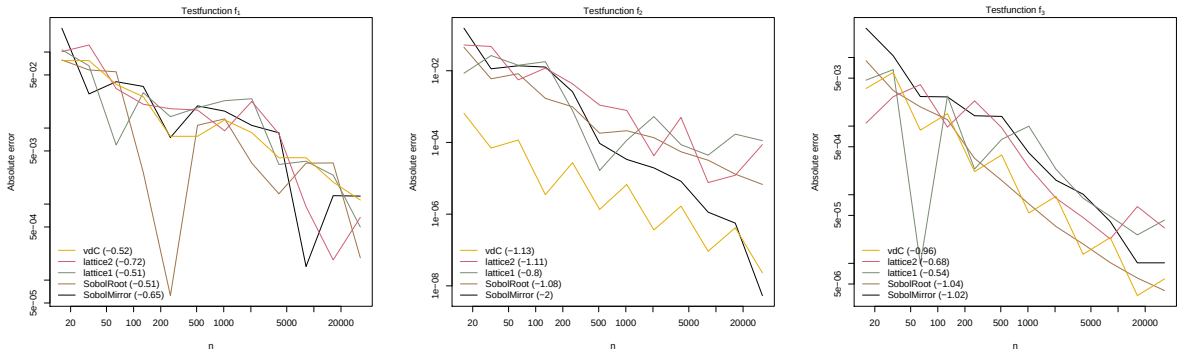

Figure 6.6: Absolute errors when integrating f_1 (left), f_2 (middle) and f_3 (right); regression coefficients in the legend.

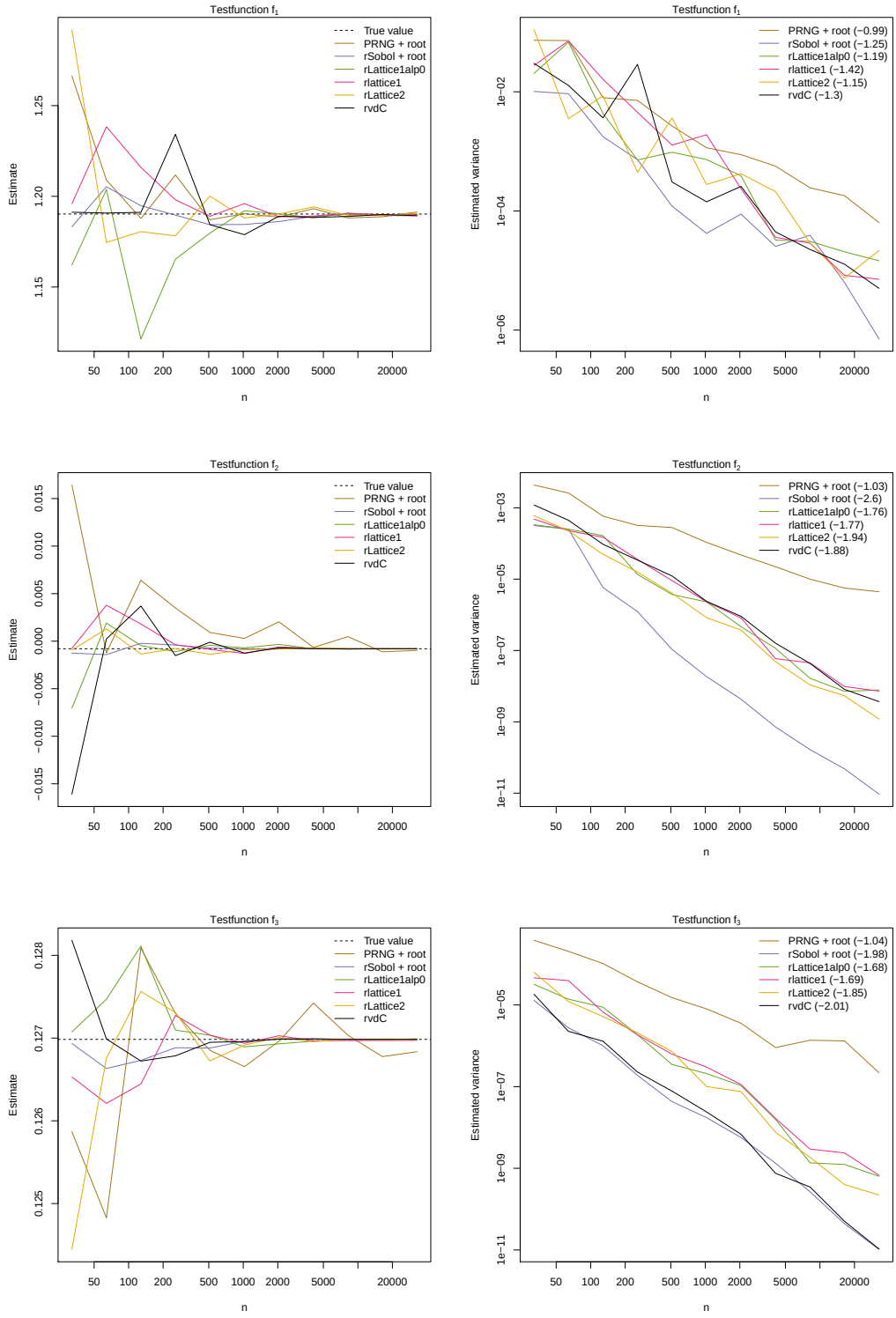

Figure 6.7: Estimates (left) and estimated variances (right) when integrating f_1 (top), f_2 (middle) or f_3 (right). For each n, $B = 15$ randomizations were used.

Chapter 7

Conclusion

In this thesis, we applied RQMC to a wide range of problems and saw that RQMC methods are superior to MC even in high dimensions. We contributed a range of methods to work with grouped and ungrouped normal variance mixtures in Chapter 3 and by doing so filled many gaps in the literature. Due to the importance of these distributions in risk management, our methods and, in particular our R package `nvmix` is widely applicable.

With the EM-like algorithm to fit the t copula at hand, we plan on developing methods to efficiently fit skew-t copulas. The major complication here is the lack of an available quantile function.

In Chapter 4, we explored how NRVGs can be combined with RQMC methods. We also studied AR methods combined with RQMC, in particular, AR-d, which samples along the coordinates rather than moving along the sequence. We highlight that this AR scheme has not gained much attention in the literature. Future research includes both computational questions, such as exploring efficient implementations of AR-d based on point sets that are easily extensible in the number of coordinates, and theoretical questions, such as whether we can prove that our proposed RQMC-based methods outperform MC.

Our (S)SIS framework and our calibration algorithm from Chapter 5 applies to a wide range of problems. As mentioned there, other low-dimensional structures may provide a better fit than single-index models do.

Chapter 6 focused on the construction of low-discrepancy point-sets on triangles. The numerical results indicated that mapping a bivariate Sobol' sequence to a triangle via some transformation can give excellent results, and in particular can outperform the methods that were constructed on the triangle directly. Nevertheless, we saw that all our constructions outperform MC. Future research includes comparing efficient implementations

in terms of their computational time needed and exploring less obvious applications of triangle sampling, such as in the realm of graphical rendering.

References

[1] K. Adragni and R. Cook. Sufficient dimension reduction and prediction in regression. *Phil. Trans.: Math., Phys. and Eng. Sci.*, 367(1906):4385–4405, 2009.

[2] S. Asmussen and P. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer-Verlag, 2007.

[3] S. Au and J. Beck. Important sampling in high dimensions. *Structural Safety*, 25(2):139–163, 2003.

[4] A. Bassamboo, S. Juneja, and A. Zeevi. Portfolio credit risk with extremal dependence: Asymptotic analysis and efficient simulation. *Operations Research*, 56(3):593–606, 2008.

[5] K. Basu. *Quasi-Monte Carlo Methods in Non-Cubical Spaces*. Stanford University, 2016.

[6] K. Basu and A. Owen. Low discrepancy constructions in the triangle. *SIAM Journal on Numerical Analysis*, 53(2):743–761, 2015.

[7] K. Basu and A. Owen. Transformations and hardy–krause variation. *SIAM Journal on Numerical Analysis*, 54(3):1946–1966, 2016.

[8] C. Bellosta. *ADGofTest: Anderson-Darling GoF test*, 2011. R package version 0.3.

[9] Z. Botev and P. L'Écuyer. Efficient probability estimation and simulation of the truncated multivariate student-t distribution. In *Proceedings of the 2015 Winter Simulation Conference*, pages 380–391. IEEE Press, 2015.

[10] L. Brandolini, L. Colzani, G. Gigante, and G. Travaglini. A koksma–hlawka inequality for simplices. In *Trends in harmonic analysis*, pages 33–46. Springer, 2013.

[11] R. Caflisch, W. Morokoff, and A. Owen. *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension.* Department of Mathematics, University of California, Los Angeles, 1997.

[12] J. Chambers, C. Mallows, and B. Stuck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71(354):340–344, 1976.

[13] J. Chan and D. Kroese. Efficient estimation of large portfolio loss probabilities in t-copula models. *European Journal of Oparerational Research*, 205(2):361–367, 2010.

[14] R. Cheng. The generation of gamma variables with non-integral shape parameter. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 26(1):71–75, 1977.

[15] C. Coffey and K. Muller. Properties of doubly-truncated gamma variables. *Communications in Statistics-Theory and Methods*, 29(4):851–857, 2000.

[16] R. Cook. *Regression Graphics.* Wiley, 1998.

[17] R. Cook and L. Forzani. Likelihood-based sufficient dimension reduction. *J. of the American Statistical Association*, 104(485):197–208, 2009.

[18] R. Cools, F. Kuo, and D. Nuyens. Constructing embedded lattice rules for multivariate integration. *SIAM Journal on Scientific Computing*, 28(6):2162–2188, 2006.

[19] R. Cranley and T. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13(6):904–914, 1976.

[20] Wuertz D., Maechler M., and Rmetrics core team members. *stabledist: Stable Distribution Functions*, 2016. R package version 0.7-1.

[21] S. Daul, E. De Giorgi, F. Lindskog, and A. McNeil. The grouped t-copula with an application to credit risk. *Available at SSRN 1358956*, 2003.

[22] P. De Boer, D. Kroese, S. Mannor, and R. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[23] S. Demarta and A. McNeil. The t copula and related copulas. *International statistical review*, 73(1):111–129, 2005.

[24] A. Dempster and D. Laird, N. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.

[25] G. Derflinger, W. Hörmann, and J. Leydold. Random variate generation by numerical inversion when only the density is known. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(4):1–25, 2010.

[26] L. Devroye. *Non-Uniform Random Variate Generation*. Springer New York, 1986.

[27] G. Dong, E. Hintz, and C. Lemieux. RQMC on triangles. Working paper, 2022.

[28] D. Eddelbuettel. Counting CRAN Package Depends, Imports and LinkingTo, 2012.

[29] P. Embrechts, C. Klüppelberg, and T. Mikosch. Modelling extremal events. *British actuarial journal*, 5(2):465–465, 1999.

[30] P. Embrechts, F. Lindskog, and A. McNeil. Modelling dependence with copulas. *Rapport technique, Département de mathématiques, Institut Fédéral de Technologie de Zurich, Zurich*, 2001.

[31] K. Fang and Y. Wang. *Number-theoretic methods in statistics*, volume 51. CRC Press, 1993.

[32] B. Flury. Acceptance–rejection sampling made easy. *SIAM Review*, 32(3):474–476, 1990.

[33] John Fox. *Applied Regression Analysis and Generalized Linear Models*. Sage Publications, 2015.

[34] A. Genz. Numerical computation of multivariate normal probabilities. *Journal of computational and graphical statistics*, 1(2):141–149, 1992.

[35] A. Genz and F. Bretz. Numerical computation of multivariate t-probabilities with application to power calculation of multiple contrasts. *Journal of Statistical Computation and Simulation*, 63(4):103–117, 1999.

[36] A. Genz and F. Bretz. Comparison of methods for the computation of multivariate t probabilities. *Journal of Computational and Graphical Statistics*, 11(4):950–971, 2002.

[37] A. Genz and F. Bretz. *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media, 2009.

[38] A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2019. R package version 1.0-11.

[39] G. Gibson, C. Glasbey, and D. Elston. Monte Carlo evaluation of multivariate normal integrals and sensitivity to variate ordering. *Advances in Numerical Methods and Applications, World Scientific Publishing, River Edge*, pages 120–126, 1994.

[40] P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

[41] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Asymptotically optimal importance sampling and stratification for pricing path-dependent options. *Mathematical finance*, 9(2):117–152, 1999.

[42] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Variance reduction techniques for estimating Value-at-Risk. *Management Science*, 46(10):1349–1364, 2000.

[43] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Portfolio value-at-risk with heavy-tailed risk factors. *Mathematical Finance*, 12(3):239–269, 2002.

[44] P. Glasserman and J. Li. Importance sampling for portfolio credit risk. *Management Science*, 51(11):1643–1656, 2005.

[45] T. Goda, K. Suzuki, and T. Yoshiki. Quasi-monte carlo integration for twice differentiable functions over a triangle. *Journal of Mathematical Analysis and Applications*, 454(1):361–384, 2017.

[46] W. Härdle, P. Hall, and H. Ichimura. Optimal smoothing in single-index models. *The Annals of Statistics*, 21(1):157–178, 1993.

[47] W. Harris and T. Helvig. Marginal and conditional distributions of singular distributions. *Publications of the Research Institute for Mathematical Sciences, Kyoto University. Ser. A*, 1(2):199–204, 1965.

[48] Z. He and X. Wang. Convergence analysis of quasi-monte carlo sampling for quantile and expected shortfall. *Mathematics of Computation*, 90(327):303–319, 2021.

[49] F. Hickernell. Lattice rules: how well do they measure up? In *Random and quasi-random point sets*, pages 109–166. Springer, 1998.

[50] F. Hickernell and H. Hong. Computing multivariate normal probabilities using rank-1 lattice sequences. In *Proceedings of the Workshop on Scientific Computing (Hong Kong)*, pages 209–215, 1997.

[51] F. Hickernell, H. Hong, P. L'Écuyer, and C. Lemieux. Extensible lattice sequences for quasi-monte carlo quadrature. *SIAM Journal on Scientific Computing*, 22(3):1117–1138, 2000.

[52] E. Hintz, M. Hofert, and C. Lemieux. Grouped normal variance mixtures. *Risks*, 8(4):103, 2020.

[53] E. Hintz, M. Hofert, and C. Lemieux. Normal variance mixtures: Distribution, density and parameter estimation. *Computational Statistics and Data Analysis*, 157C:107175, 2021.

[54] E. Hintz, M. Hofert, and C. Lemieux. Computational challenges of t and related copulas. *Journal of Data Science*, 20(1):95–110, 2022.

[55] E. Hintz, M. Hofert, and C. Lemieux. Multivariate Normal Variance Mixtures in R: The R Package nvmix. *Journal of Statistical Software*, To appear, 2022.

[56] E. Hintz, M. Hofert, and C. Lemieux. Quasi-random sampling with black box or acceptance-rejection inputs. To appear., 2022.

[57] E. Hintz, M. Hofert, C. Lemieux, and Y. Taniguchi. Single-index importance sampling with stratification. https://doi.org/10.48550/arXiv.2111.07542, 2021.

[58] E. Hlawka. Funktionen von beschränkter variation in der theorie der gleichverteilung. *Annali di Matematica Pura ed Applicata*, 54(1):325–333, 1961.

[59] M. Hofert, E. Hintz, and C. Lemieux. *nvmix: Multivariate Normal Variance Mixtures*, 2022. R package version 0.0-7.

[60] M. Hofert and K. Hornik. *qrmdata: Data Sets for Quantitative Risk Management Practice*, 2016. R package version 2016-01-03-1.

[61] M. Hofert, I. Kojadinovic, M. Maechler, and J. Yan. *copula: Multivariate Dependence with Copulas*, 2020. R package version 1.0-0.

[62] M. Hofert and C. Lemieux. *qrng: (Randomized) Quasi-Random Number Generators*, 2019. R package version 0.0-7.

[63] M. Hofert and M. Mächler. Nested archimedean copulas meet R: The nacopula package. *Journal of Statistical Software*, 39(9):1–20, 2011.

[64] W. Hörmann and J. Leydold. Continuous random variate generation by fast numerical inversion. *ACM Trans. Model. Comput. Simul.*, 13(4):347–362, 2003.

[65] W. Hörmann and J. Leydold. Generating generalized inverse gaussian random variates. *Statistics and Computing*, 24(4):547–557, 2014.

[66] H. Ichimura. Semiparametric least squares (SLS) and weighted SLS estimation of single-index models. *Journal of Econometrics*, 58(1-2):71–120, 1993.

[67] H. Kahn and A. Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[68] L. Katafygiotis and K. Zuev. Geometric insight into the challenges of solving high-dimensional reliability problems. *Probabilistic Engineering Mechanics*, 23(2–3):208–218, 2008.

[69] P. Keast. Optimal parameters for multidimensional integration. *SIAM Journal on Numerical Analysis*, 10(5):831–838, 1973.

[70] I. Kojadinovic and J. Yan. Modeling multivariate distributions with continuous margins using the copula R package. *Journal of Statistical Software*, 34(9):1 – 20, 2010.

[71] S. Kotz and S. Nadarajah. *Multivariate t Distributions and Their Applications*. Cambridge University Press, 2004.

[72] D. Kundu and R. Gupta. A convenient way of generating gamma random variables using generalized exponential distribution. *Computational Statistics and Data Analysis*, 51(6):2796–2802, 2007.

[73] T. Kvalseth. Cautionary note about R^2. *The American Statistician*, 39(4):279–285, 1985.

[74] P. L'Ecuyer. Quasi-monte carlo methods in finance. In *Proceedings of the 2004 Winter Simulation Conference*, volume 2, pages 1645–1655. IEEE, 2004.

[75] P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235, 2000.

[76] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, 2009.

[77] J. Leydold and W. Hörmann. Generating generalized inverse gaussian random variates by fast inversion. *Computational statistics & data analysis*, 55(1):213–217, 2011.

[78] J. Leydold and W. Hörmann. *Runuran: R Interface to the 'UNU.RAN' Random Variate Generators*, 2020. R package version 0.30.

[79] C. Liu and D. Rubin. The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81(4):633–648, 1994.

[80] C. Liu and D. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5(1):19–39, 1995.

[81] Sh. Liu, H. Wu, and W. Meeker. Understanding and addressing the unbounded "likelihood" problem. *The American Statistician*, 69(3):191–200, 2015.

[82] X. Luo and P. Shevchenko. The t copula with multiple parameters of degrees of freedom: bivariate characteristics and application to risk management. *Quantitative Finance*, 10(9):1039–1054, 2010.

[83] R. Mashal and A. Zeevi. Beyond correlation: Extreme co-movements between financial assets. https://www0.gsb.columbia.edu/faculty/azeevi/PAPERS/BeyondCorrelation.pdf, 2002.

[84] Ma. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[85] A. McNeil, R. Frey, and P. Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, 2015.

[86] O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. R package version 1.4-2.1.

[87] B. Moskowitz and R. Caflisch. Smoothness and dimension reduction in quasi-monte carlo methods. *Mathematical and Computer Modelling*, 23(8-9):37–54, 1996.

[88] S. Nadarajah and S. Kotz. Estimation methods for the multivariate t distribution. *Acta Applicandae Mathematicae*, 102(1):99–118, 2008.

[89] J. Neddermeyer. Non-parametric partial importance sampling for financial derivative pricing. *Quantitative Finance*, 11(8):1193–1206, 2011.

[90] R. Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

[91] N. Nguyen and G. Ökten. The acceptance-rejection method for low-discrepancy sequences. *Monte Carlo Methods and Applications*, 22(2):133–148, 2016.

[92] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63. Siam, 1992.

[93] T. Nitithumbundit and J. Chan. ECM Algorithm for Auto-Regressive Multivariate Skewed Variance Gamma Model with Unbounded Density. *Methodology and Computing in Applied Probability*, 22:1–23, 2019.

[94] C. Oates, M. Girolami, and N. Chopin. Control functionals for monte carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718, 2017.

[95] A. Owen. Necessity of low effective dimension. *Working paper, Stanford University*, 2002.

[96] A. Owen. Better estimation of small Sobol' sensitivity indices. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(2):11, 2013.

[97] S. Paskov and J. Traub. Faster valuation of financial derivatives. *The Journal of Portfolio Management*, 22(1):113–123, 1995.

[98] B. Pfaff and A. McNeil. *QRM: Provides R-Language Code to Examine Quantitative Risk Management Concepts*, 2016. R package version 0.4-13.

[99] R. Piessens, E. de Doncker-Kapenga, C. Überhuber, and D. Kahaner. *Quadpack: a subroutine package for automatic integration*, volume 1. Springer Science & Business Media, 2012.

[100] T. Pillards and R. Cools. Transforming low-discrepancy sequences from a cube to a simplex. *Journal of computational and applied mathematics*, 174(1):29–42, 2005.

[101] J.L. Powell, J.H. Stock, and T.M. Stoker. Semiparametric estimation of index coefficients. *Econometrica*, pages 1403–1430, 1989.

[102] R. Protassov. EM-based maximum likelihood parameter estimation for multivariate generalized hyperbolic distributions with fixed λ. *Statistics and Computing*, 14(1):67–77, 2004.

[103] G. Pujol, B. Iooss, and A. Janon. *sensitivity: Global Sensitivity Analysis of Model Outputs*, 2017. R package version 1.15.0.

[104] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[105] C. Reinsch. Smoothing by spline functions. *Nümerische Mathematik*, 10(3):177–183, 1967.

[106] M. Rosenblatt. Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, 23(3):470–472, 09 1952.

[107] R. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[108] R. Rubinstein and D. Kroesse. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[109] H. Sak, W. Hörmann, and J. Leydold. Efficient risk simulations for linear asset portfolios in the t-copula model. *European Journal of Operational Research*, 202(3):802–809, 2010.

[110] G. Schüeller, H. Pradlwarter, and P. Koutsourelakis. A critical appraisal of reliability estimation procedures for high dimensions. *Probabilistic Engineering Mechanics*, 19(4):463–474, 2004.

[111] I. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[112] I. Sobol'. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[113] T. Stoker. Consistent estimation of scaled coefficients. *Econometrica*, 54(6):1461–1481, 1986.

[114] S. Tezuka. On the necessity of low-effective dimension. *Journal of Complexity*, 21(5):710–721, 2005.

[115] G. Venter, J. Barnett, R. Kreps, and J. Major. Multivariate copulas for financial modeling. *Variance*, 1(1):103–119, 2007.

[116] X. Wang. Improving the rejection sampling method in quasi-monte carlo methods. *Journal of computational and applied Mathematics*, 114(2):231–246, 2000.

[117] X. Wang. On the effects of dimension reduction techniques on some high-dimensional problems in finance. *Operations Research*, 54(6):1063–1078, 2006.

[118] X. Wang and K. Fang. The effective dimension and quasi-Monte Carlo integration. *Journal of Complexity*, 19(2):101–124, 2003.

[119] X. Wang and I. Sloan. Why are high-dimensional finance problems often of low effective dimension? *SIAM Journal on Scientific Computing*, 27(1):159–183, 2005.

[120] J. Yan. Enjoy the joy of copulas: With a package copula. *Journal of Statistical Software*, 21(4):1–21, 2007.

[121] T. Yoshiba. Maximum likelihood estimation of skew-t copulas with its applications to stock returns. *Journal of Statistical Computation and Simulation*, 88(13):2489–2506, 2018.

[122] H. Zhu and J. Dick. Discrepancy bounds for deterministic acceptance-rejection samplers. *Electronic Journal of Statistics*, 8(1):678–707, 2014.