NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

MSc THESIS

Practical Volume Computation of Structured Convex Bodies for Modeling Financial Crises

Apostolos G. Chalkis

SUPERVISOR: Ioannis Z. Emiris, Professor N.K.U.A.

ATHENS

MAY 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Υπολογισμού Όγκου Κυρτών Σωμάτων με Εφαρμογή στη Μοντελοποίηση Οικονομικών Κρίσεων

Απόστολος Γ. Χαλκής

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

ΑΘΗΝΑ

ΜΑΪΟΣ 2018

MSc THESIS

Practical Volume Computation of Structured Convex Bodies for Modeling Financial Crises

Apostolos G. Chalkis
Registration Number: M1476

SUPERVISOR: Ioannis Z. Emiris, Professor N.K.U.A.

TWO-MEMBER ADVISORY COMMITTEE:

Ioannis Z. Emiris, Professor N.K.U.A.

Apostolos Giannopoulos, Professor N.K.U.A.

Examination Date: May 8, 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Υπολογισμού Όγκου Κυρτών Σωμάτων με Εφαρμογή στη Μοντελοποίηση Οικονομικών Κρίσεων

Απόστολος Γ. Χαλκής
Α.Μ: Μ1476

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

ΔΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

Απόστολος Γιαννόπουλος, Καθηγητής Ε.Κ.Π.Α.

Ημερομηνία Εξέτασης: 8 Μαΐου 2018

ABSTRACT

We examine volume computation of general-dimensional polytopes and more general convex bodies, defined as the intersection of a simplex by a family of parallel hyperplanes, and another family of parallel hyperplanes or a family of concentric ellipsoids. Such convex bodies appear in modeling and predicting financial crises.

We design and implement practical algorithms in the exact and approximate setting, we experimentally juxtapose them and study the tradeoff of exactness and accuracy for speed. We analyze the following methods in order of increasing generality:

- rejection sampling relying on uniformly sampling the simplex, which is the fastest approach, but inaccurate for small volumes.
- exact formulae based on the computation of integrals of probability distribution functions, which are the method of choice for intersections with a single hyperplane.
- an optimized Lawrence sign decomposition method, since the polytopes at hand are shown to be simple with additional structure.
- Markov chain Monte Carlo algorithms using random walks based on the hit-and-run paradigm generalized to nonlinear convex bodies and relying on new methods for computing a ball enclosed in the given body, such as a second-order cone program. This last is experimentally extended to non-convex bodies with very encouraging results.

We give the Tables of experiments showing our claims. A part of the current Thesis is given in the paper titled ”Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises” [6] which was accepted for presentation at SOCG 2018 (34th International Symposium on Computational Geometry).

SUBJECT AREA: Computational Geometry

KEYWORDS: Polytope volume, convex body, simplex, sampling, financial portfolio

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία αναπτύσουμε και βελτιώνουμε μεθόδους και αλγορίθμους για την αποτελεσματική επίλυση προβλημάτων υπολογισμού όγκου. Τα προβλήματα τα οποία επιλύουμε προκύπτουν από μία εφαρμογή της οικονομικής επιστήμης που αφορά την κατασκευή ενός μαθηματικού μοντέλου πρόβλεψης χρηματιστηριακών κρίσεων.

Τα προβλήματα υπολογισμού όγκου προκύπτουν από την τομή δύο οικογενειών παράλληλων υπερεπιπέδων ή από μια οικογένεια πράλληλων υπερεπιπέδων και μια οικογένεια από ομόκεντρες ελλείψεις με το μοναδιαίο άπλοκο σε αυθαίρετη διάσταση. Επομένως ζητείται ο όγκος απλών πολυτόπων, αλλά και κυρτών ή μη κυρτών μη γραμμικών σωμάτων.

Για τον υπολογισμό αυτών των όγκων αναπτύσουμε ή βελτιώνουμε αλγορίθμους όπως:

- Δημιουργίας τυχαίων σημείων στο εσωτερικό του απλόκου και προσέγγιση του ποσοστού του όγκου όλων των σωμάτων ως προς τον όγκο του απλόκου.
- Χρησιμοποίηση επαναληπτικών τύπων που δίνουν την ακριβή τιμή του όγκου της τομής ενός υπερεπιπέδου και ενός απλόκου.
- Χρησιμοποίηση του τύπου του Lawrence για τον όγκο απλών πολυτόπων και βελτιστοποίησή του στα πλαίσια του δικού μας προβλήματος.
- Χρησιμοποίηση αλγορίθμων Monte Carlo και τυχαίου περιπάτου και πιο συγκεκριμένα τον αλγόριθμο `VolEsti`[14], τον οποίο επεκτείνουμε για μη γραμμικά σώματα, ενώ δείχνουμε πειραματικά ότι λειτουργεί ικανοποιητικά για μικρές σχετικά διαστάσεις και για τα μη κυρτά σώματα που προκύπτουν κάνοντας τις κατάλληλες τροποποιήσεις.

Τέλος δίνονται σε πίνακες όλα τα πειράματα που εκτελέσαμε και χρησιμοποιούνται για να δείξουμε τους ισχυρισμούς που διατυπώνονται στην εργασία. Κομμάτι της εργασίας αναπτύσσεται σε επιστημονικό άρθρο με τίτλο ”Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises” [6] το οποίο θα παρουσιαστεί τον Ιούνιο του 2018 στο συνέδριο με την επωνυμία SOCG 2018 (34th International Symposium on Computational Geometry).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Γεωμετρία

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Όγκος πολυτόπου, κυρτό σώμα, άπλοκο, ανίχνευση κρίσεων στα οικονομικά

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Given that volume computation of polytopes is #P-hard for both V- and H-representations [12] and no poly-time algorithm can achieve better than exponential error [13], the problem is not expected to admit of an efficient deterministic algorithm in general dimension. Developing algorithms for volume computation has received a lot of attention in the exact setting [5]. In the approximate setting, following the breakthrough polynomial-time algorithm by random walks [11], several algorithmic improvements ensued. The current best theoretical bounds are in [25] and for polytope sampling in [26]. Interestingly, only two pieces of software offer practical algorithms in high dimension: `VolEsti`, a public-domain C++ implementation that scales to a few hundred dimensions [14], based on the Hit-and-Run paradigm [27], and the Matlab implementation of [8], which treats hyperplanes as an ellipsoid, and seems competitive to `VolEsti` in very high dimensions. Sampling from non-convex bodies appears in experimental works, with very few methods offering theoretical guarantees, e.g. in star shaped bodies [7] or, more recently, in [1].

In the current thesis we consider some volume computation problems that arise from a financial application which is described in 7.1. For a specific set of assets we want to characterize portfolios by their return and their risk which is the variance of the portfolios' returns. The goal is the detection of crises through return/volatility dependency and to measure the efficiency of financial markets. If we consider the canonical simplex $\Delta^d \subset \mathbb{R}^{d+1}$ where each point represents a portfolio and $d+1$ is the number of assets the first is about a family of parallel hyperplanes and a family of concentric ellipsoids intersecting the canonical simplex and the second consists of two families of parallel hyperplanes intersecting the canonical simplex. The copulas for the two problems arise from all the volume computations defined by the intersections.

We design and implement the following different approaches for volume computation: Efficient sampling of the simplex and using rejection to approximate the target volume, which is fast but inaccurate for small volumes. Exact formulae of integrals of appropriate probability distribution functions, which are implemented for the case of a single hyperplane. Optimizing the use of Lawrence's sign decomposition method, since the polytopes at hand are shown to be simple with extra structure; a major issue here is numerical instability. Extending state-of-the-art random walks based on the hit-and-run paradigm to convex bodies defined as the intersection of linear halfspaces and ellipsoids. The latter is experimentally generalized to non-convex bodies defined by two ellipsoids with same quadratic form, and accurate approximations are obtained under certain mild conditions.

Our randomized algorithms for volume approximation extend VolEsti, where the main problem to address is to compute the maximum inscribed ball of the convex body P a.k.a. Chebychev ball. This reduces to a linear program when P is a polytope. For a convex body defined by intersecting a polytope with k balls, the question becomes a second-order cone program (SOCP) with k cones. When interchanging input balls with ellipsoids, the SOCP yields a sufficiently good approximation of the Chebychev ball.

Our implementations are in C++, lie in the public domain (github), are based on CGAL,

rely on Eigen for linear algebra, on Boost for random number generators, and experiment with two SOCP solvers for initializing random walks. Our software tools are general and of independent interest. They are applied to allow us to extend the computation of a portfolio score to up to 100 dimensions, thus doubling the size of assets studied in financial research.

The rest of the thesis is organized as follows. The Chapter 2 gives the basic definitions and lemmas, the Chapter 3 overviews methods for representing and uniform sampling from simplices. In Chapter 4 we give some theoretical and practical results for the rejection-sampling method. Chapter 5 considers volumes defined as the intersection of a simplex and one hyperplane or more hyperplanes, the latter being organized in at most two families of parallel hyperplanes. Chapter 6 studies convex bodies defined as the intersection of a simplex and an ellipsoid, for which random walk methods are developed. In Chapter 7 we give more details about the financial application and we conclude with experimental results and open questions for future work. Tables of experiments are given in the Appendix.

2. BASIC CONCEPTS

In this chapter we give some basic definitions and theorems which are used in the next chapters.

2.1 Polytopes

We will work with polytopes in *H*-representation and *V*-representation.

Definition 2.1. *A H-representation of a polytope in d dimension is system of inequalities*

$$Ax \leq b$$

*where A is a $m \times d$ matrix and b an m-dimensional vector. The feasible region $S = \{x \in \mathbb{R}^d | Ax \leq b\}$ is called a **polyedron**.*

*A bounded polyhedron is called a **polytope**.*

So every polytope can be expressed as a set of d-dimension halfspaces.

A polytope, P, can be given in *V*-representation where we are given a set of vertices V then

$$P = conv(V)$$

where,

$$conv(V) := \{\sum_{i=1}^{|V|} \alpha_i v_i | v_i \in V, \alpha_i \geq 0, \sum_{i=1}^{|V|} \alpha_i = 1\}$$

So the *V*-representation of the polytope is simply the convex hull $conv(V)$ of the vertices.

Definition 2.2. *We call the $(d-1)$-dimensional faces of a polytope of dimension d a **facet**.*

In some cases we have a *H*-representation of a polytope corresponding to a $m \times d$ matrix A, but some halfspaces, e.g. ℓ, could be **redundant** in the sense that the $m - \ell$ halfspaces define the same polytope.

Definition 2.3. *A row a_i of A corresponding to an inequality $a_i x \leq b_i$ is called redundant if and only if the feasible region S does not change when we delete the row. Otherwise it is called non-redundant.*

Definition 2.4. *The H-redundancy problem of a polytope P is the problem of finding a minimal set of non-redundant constraints representing P, where P is given in H-representation.*

In many problems we will work with simplices which is a specific kind of polytope. A simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. A d-simplex is a d-dimensional polytope which is the convex hull of its $d+1$ vertices.

Definition 2.5. *Let the $d+1$ points $u_0, \ldots, u_d \in \mathbb{R}^d$ are affinely independent, which means $u_1-u_0, \ldots, u_d-u_0$ are linearly independent. Then, the set of points,*

$$R = \left\{ \lambda_0 u_0 + \cdots + \lambda_d u_d | \lambda_i \geq 0, \sum_{i=0}^{d} \lambda_i = 1 \right\}$$

is called a d-simplex.

A special case of a d-simplex is the unit d-simplex, Δ^d.

Definition 2.6. *The convex hull of the $d+1$ vertices,*

$$V_j = (v_{1j}, \ldots, v_{ij})$$

where

$$v_{ij} = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \quad \text{for } j = 1, \ldots, d+1 \quad \text{and } i = 1, \ldots, d$$

is called the unit d-simplex Δ^d in $\mathbb{R}^d$ (or the unit simplex in d dimension).

Another representation of the unit simplex is the canonical simplex which we use in the next chapters.

Definition 2.7. *The convex hull of the d vertices,*

$$V_j = (v_{1j}, \ldots, v_{ij})$$

where

$$v_{ij} = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \quad \text{for } j = 1, \ldots, d \quad \text{and } i = 1, \ldots, d$$

is called the canonical $d-1$-simplex Δ^{d-1} in $\mathbb{R}^d$ (or the canonical simplex in $d-1$ dimension).

2.2 Convex sets and bodies

Definition 2.8. *A set $C \subset \mathbb{R}^d$ is called convex, if for any two points $x, y \in C$ the line segment $[x, y] := \{z \in \mathbb{R}^d | z = tx + (1-t)y$, for $0 \leq t \leq 1\}$ is entirely in C.*

Definition 2.9. *A set $C \subset \mathbb{R}^d$ is called convex body, if it is closed, bounded and convex with non-empty interior.*

Theorem 2.1. *If X is a convex set and $x_1, \ldots, x_k$ are any points in it, then*

$$x = \sum_{i}^{k} \lambda_i x_i$$

where all $\lambda_i \geq 0$ and $\sum_i^k \lambda_i = 1$ is also in X.

*A set that is not convex is called a **non-convex set**.*

3. SAMPLING UNIFORMLY FROM SIMPLEX

In this chapter we give all the known algorithms to sample uniformly from an arbitrary simplex or from the unit simplex. We experimentally compare them and make some improvements. In order to improve the implementations we use a Bloom filter variation. First we give all the known algorithms for sampling from the unit simplex and then from an arbitrary one. Finally we end up with the optimal implementations in practice.

3.1 Smith and Tromble algorithm

Smith and Tromble in [37] give an $O(d \log d)$ algorithm for sampling from the unit Simplex in $\mathbb{R}^d$. They suggest the following:

Algotrithm 1:

- Generate $d+2$ distinct integers $x_0, x_1, \cdots, x_d$ randomly in $\{0, 1, 2, \cdots, M\}$ and set $x_0 = 0$, $x_{d+1} = 1$.
- Sort sequence x_i: $x_0 < x_1 < \cdots < x_{d+1}$
- Define point y in $\mathbb{R}^{d+1}$: $y_i = \dfrac{x_i - x_{i-1}}{M}$, $i = 1, \cdots, d+1$, so $\sum_{i=1}^{d+1} y_i = 1$.
- Then point y belong to the canonical simplex in $\mathbb{R}^{d+1}$. So if we ignore the last coordinate we project them to the Δ^d.

The algorithm above is given by Rubinstein and Krose in [34] too.

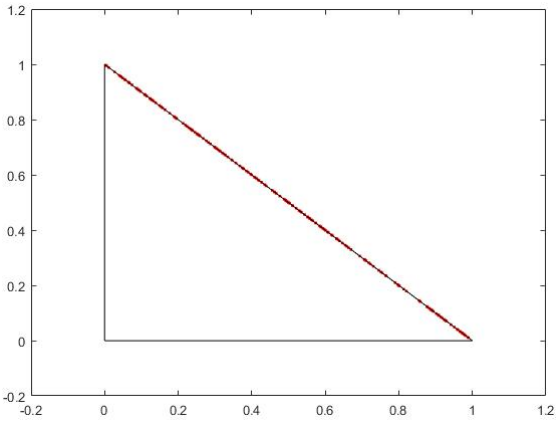

Figure 1: Smith and Tromble algorithm before projection, $d = 1$

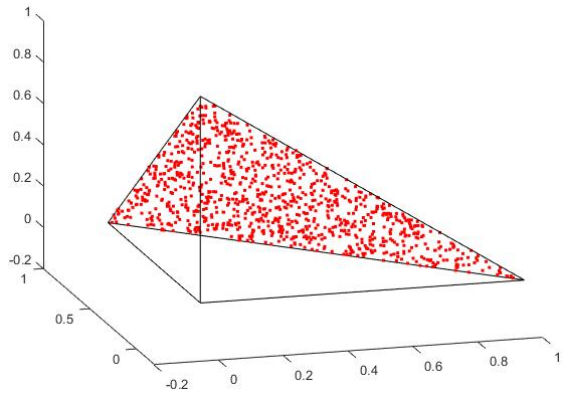

Figure 2: Smith and Tromble algorithm before projection, $d = 2$

Sorting takes $O(d \log d)$ and if we assume that we have a perfect hash function the distinct

choice of integers can be done in $O(d)$. So total complexity is $O(d\ logd)$. Nevertheless in practice distinct choice of d integers causes some problems as the assumption of a perfect hash function is not so simple. If we use $C++$ structures as $std::unordered\_map$ or $std::set$, the implementation becomes worse, for $d \leq 600$, than using $std::vector$ with a linear search in every step (when complexity becomes $O(d^2)$). Figures 3, 4 below shows that $std::vector$ is better for a large number of dimensions.

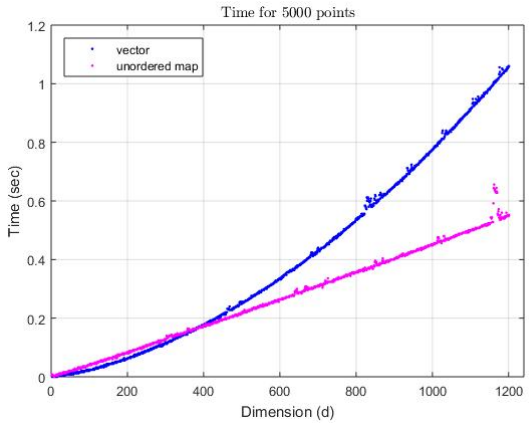

Figure 3: Comparing data structures for Smith's & Tromble's algorithm.

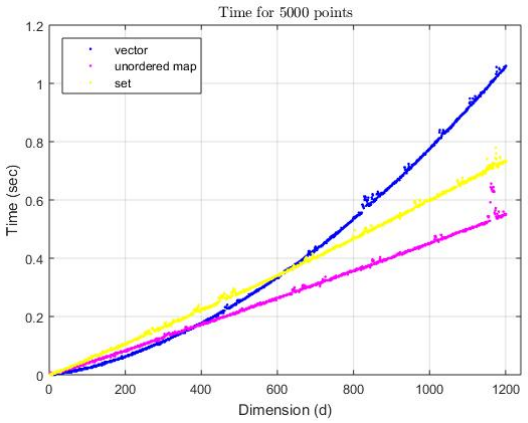

Figure 4: Comparing data structures for Smith's & Tromble's algorithm.

3.1.1 Hash function (division method)

In order to overcome these problems we implemented a hash function with the division method. Our keys are integers, so we can define the following hash function:

$$h(x_i) = x_i mod p$$

And p is the closest prime number to $\left\lceil \frac{d}{3} \right\rceil$ and $p > \left\lceil \frac{d}{3} \right\rceil$. We can assume that the chosen integers are distributed uniformly in the hash table. So in every step we check in $O(1)$ for distinct choice and we use $O(d)$ memory. In figure 5 we see the improvement but $std::vector$ is still better for $d \leq 150$.

3.1.2 Filter

A more efficient implementation is to use a variation of Bloom filter. The difference here is that we take a larger filter than the stream and only one hash function. So in order to guarantee distinct choice we do the following:

- We set p to be the smallest prime number such that $p > 3d$.

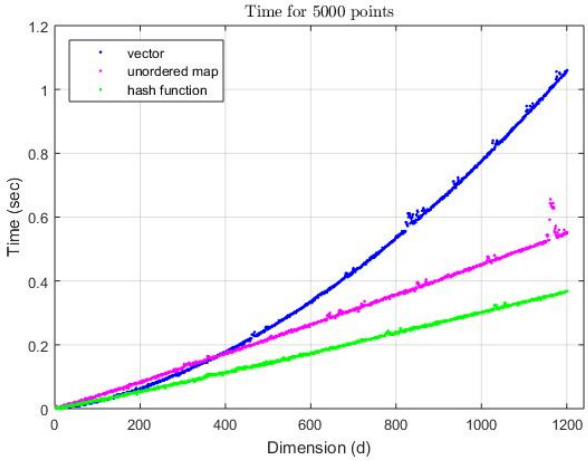

Figure 5: Comparing hash function with default C++ data structures.

- We construct the vector-filter $\in \mathbb{R}^p$, $filter = [0, 0, \cdots, 0]$
- In step i we uniformly pick integer x_i. If $filter(x_i mod p) = 1$ the we reject x_i and generate another integer. Otherwise we accept x_i and add it to the list.

It's easy to notice that in some step i there is a probability to reject an integer that we shouldn't because $filter(x_i mod p) = 1$ from a previous step j with $x_j \neq x_i$ (false positive). But if we assume that the integers are distributed uniformly in the filter then the probability to generate two different integers mapped to the same filter's position is $\frac{1}{3d} \cdot \frac{1}{3d} = \frac{1}{9d^2}$. In figures 6, 7 we can see that filter is better than hash function in order to guarantee distinct choice. But as we can notice in 7 the implementation using $std :: vector$ is still better for $d \leq 60$. So for $d > 60$ we can use our filter.

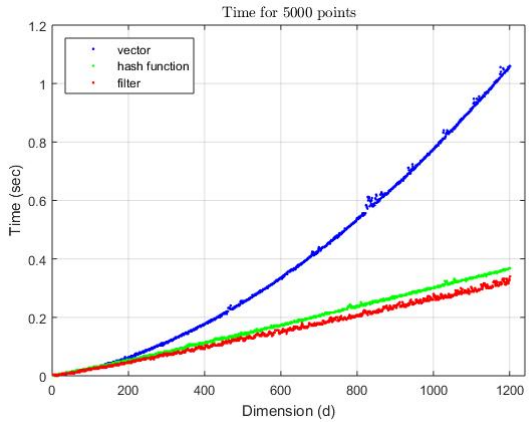

Figure 6: Comparing Bloom filter variation with hash function.

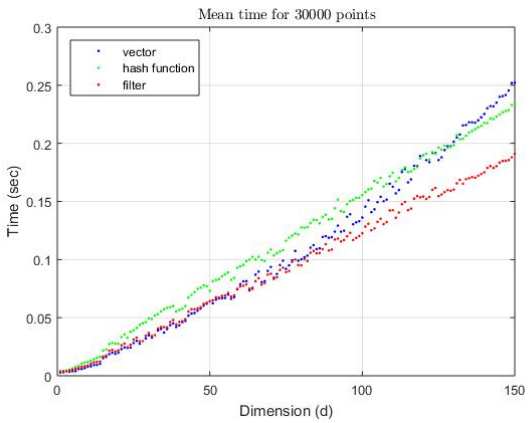

Figure 7: Comparing Bloom filter variation with hash function.

3.2 Rubinstein's and Melamed's algorithm

A more efficient algorithm is given by Rubinstein and Melamed in [35]. They suggest the following:

Algorithm 2:

- Generate $d+1$ independent unit-exponential random variables $Y_1, \cdots, Y_{d+1}$ and normalize, computing $T = \sum_{i=1}^{d+1} Y_i$ and then return $Y_i' = Y_i/T$.
- Y' will belong to the canonical simplex in $\mathbb{R}^{d+1}$ and we can project it to the unit simplex Δ^d as we did in Algorithm 1.

To generate one point in unit simplex takes $O(d)$ time. But in figures 8, 9 we can notice that Algorithm 2 is faster for $d > 80$ than our implementation for Algorithm 1 but for $d \leq 80$ is slower due to constants.

So we can experimentally conclude that the most efficient implementation for sampling from the unit simplex is Algorithm's 1 implementation for $d \leq 80$ and Algorithm's 2 implementation for $d > 80$. Furthermore we can assume with safety that the total complexity for uniformly sampling k points in Δ^d is $O(kd)$ in practice.

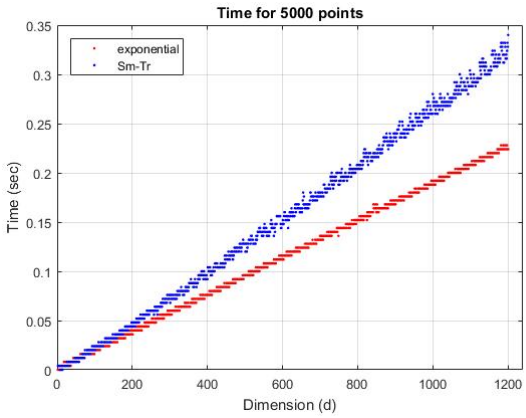

Figure 8: Comparing Rubinstein's & Melamed's algorithm with Smith's & Tromble's algorithm.

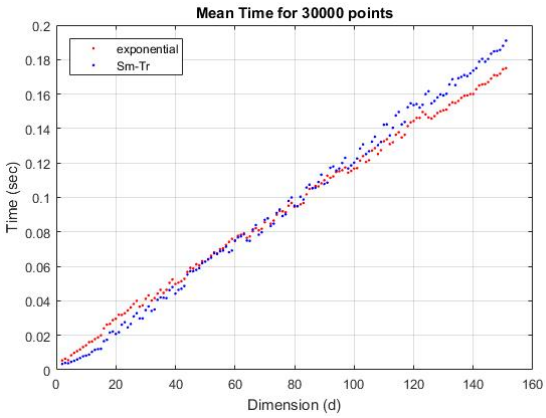

Figure 9: Comparing Rubinstein's & Melamed's algorithm with Smith's & Tromble's algorithm.

3.3 Sampling from an arbitrary simplex

If we uniformly sample a point from the unit simplex we can easily map it to a point in the arbitrary simplex through a linear transformation. The idea outcomes from figure 10 (an

example in $\mathbb{R}^2$). Let $V = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$ be the set of vertices of the arbitrary simplex. First we shift our simplex to the origin. Then if $\mathbf{x} = (x_1, \ldots, x_d)$ is a point in the unit simplex we can map it to the shifted simplex. We define matrix,

$$T = \begin{bmatrix} v_{11} - v_{01} & v_{21} - v_{01} & v_{31} - v_{01} & \ldots & v_{d1} - v_{01} \\ v_{12} - v_{02} & v_{22} - v_{02} & v_{32} - v_{02} & \ldots & v_{d2} - v_{02} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{1d} - v_{0d} & v_{2d} - v_{0d} & v_{3d} - v_{0d} & \ldots & v_{dd} - v_{0d} \end{bmatrix} \tag{3.1}$$

Through matrix in 3.1 we define the linear transformation which maps points from the unit simplex to the arbitrary simplex below,

$$W(\mathbf{x}) = T\mathbf{x} + \mathbf{v}_0 \tag{3.2}$$

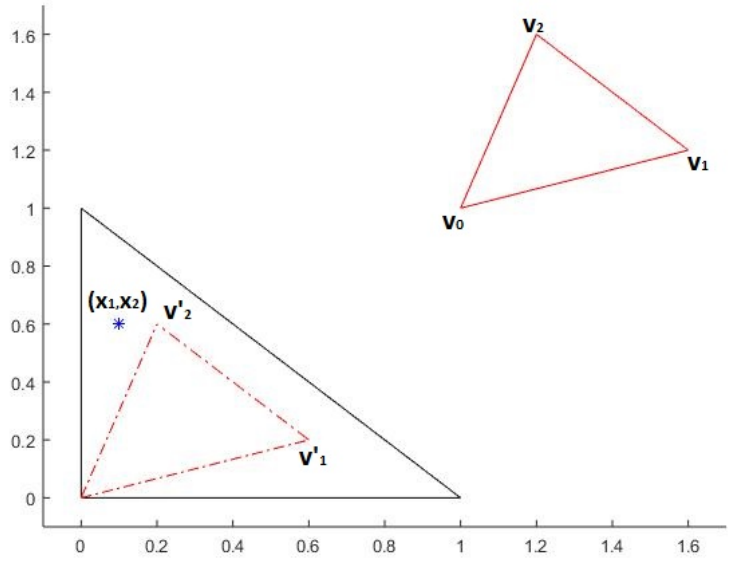

Figure 10: Transform arbitrary simplex.

We can easily prove that the mapping through linear transformation 3.2 is uniform preserving. So in order to sample from an arbitrary simplex we could first sample from the unit and then apply the linear transformation. Expanding Smith's and Tromble's proof about uniform sampling from unit simplex it is enough to show that 3.2 is 1-1 in order to prove uniform preserving. Indeed it's easy to notice that kerT=$\{\mathbf{0}\}$ so 3.2 is bijective.

Applying 3.2 results to a total complexity $O(d^2)$ for uniformly sampling one point from an arbitrary simplex.

In figure 11 we give time execution for sampling 5000 points from an arbitrary simplex using linear transformation 3.2. In figure 12 we give time execution related with the quantity d^2. We notice that we have an almost perfect positive linear relation and Pearson's coefficient is 0.9966. So we can conclude that time complexity in practice is $O(d^2)$.

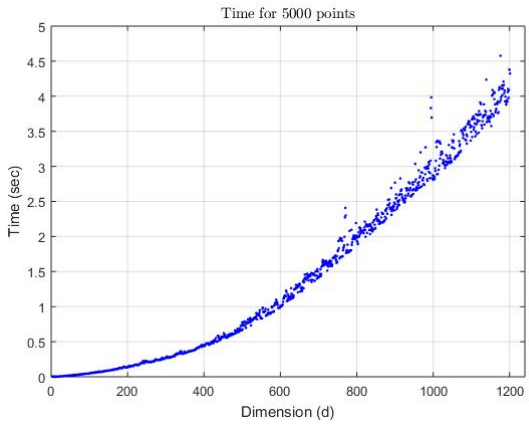

Figure 11: Owen's algorithm time efficiency.

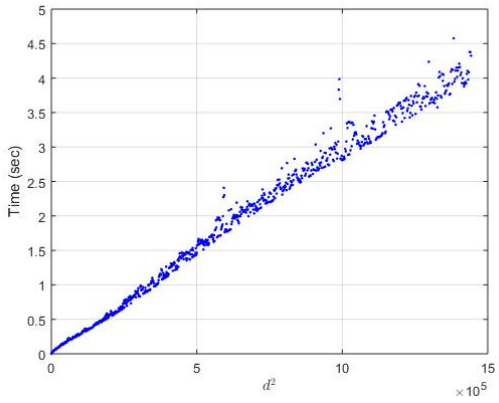

Figure 12: Owen's algorithm time complexity.

3.3.1 Owen's algorithm

Art Owen in [33] gives a seemingly different algorithm for sample from an arbitrary simplex, but it all ends to linear transformation 3.2. His idea begins with Dirichlet Distribution which s.p.p. is:

$$D(\alpha)^{-1}\prod_{j=1}^{d} x_j^{\alpha_j-1}, \quad \mathbf{x}\in\Delta^{d-1}$$

While $D(\alpha)=\prod_{j=1}^{d}(\alpha)/(\sum_{j=1}^{d}\alpha)$ and Δ^{d-1} is the unit simplex in $\mathbb{R}^{d-1}$. If $\alpha=1$, for every j we have a random variable $X\sim U(\Delta^{d-1})$. So in order to sample from X we have to do the followings:

- Let $U_1,\cdots,U_d$ be independent $U(0,1)$ random variables with $U_0\leq U_1\leq\cdots\leq U_d$ and assume $U_0=0, U_{d+1}=1$.
- We define the random variable $X_j=U_j-U_{j-1}, \quad j=1,\cdots,d+1$ and then $X\sim U(\Delta^d)$, where $\Delta^d\subset\mathbb{R}^{d+1}$ is the canonical simplex in $\mathbb{R}^{d+1}$.
- Let v_j, with $j=1,\cdots,d+1$ be the vertices of the arbitrary simplex. Then

$$Y=\sum_{j=1}^{d+1}X_jv_j, \text{ where } X\sim U(\Delta^d) \tag{3.3}$$

  would generate uniformly random points in that arbitrary simplex.

So Art Owen suggest us to sample points from the unit simplex as Smith and Tromble do and then apply mapping 3.3 to the arbitrary simplex. But it is easy to notice that 3.3 is equivalent to 3.2 as 3.3 maps a point that is sampled from the canonical simplex $\Delta^d\subset\mathbb{R}^{d+1}$ before we project it to the unit simplex Δ^d. Moreover there is a bijective relation

between points in the canonical simplex and the unit simplex. So that means that we could equivalently sample points from the unit simplex as we described in 3.1 and then apply 3.3.

Furthermore it is easy to notice that 3.3 takes $d^2 + d$ multiplications and d^2 additions and 3.2 takes d^2 multiplications and $2d^2$ additions. So 3.2 needs more operations than 3.3. In figures 13, 14 we notice that advantage. Figure 15 shows the time ratio between 3.2 over 3.3.

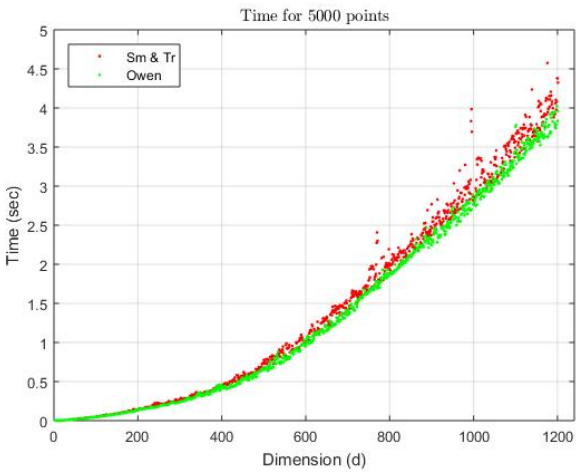

Figure 13: Comparing Owen's algorithm with Smith's & Tromble's for arbitrary simplex.

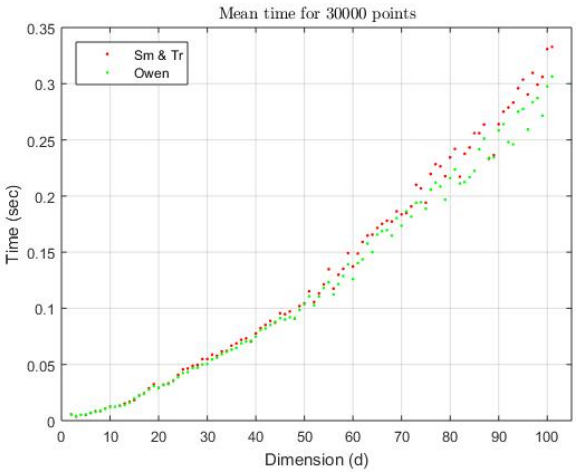

Figure 14: Comparing Owen's algorithm with Smith's & Tromble's for arbitrary simplex.

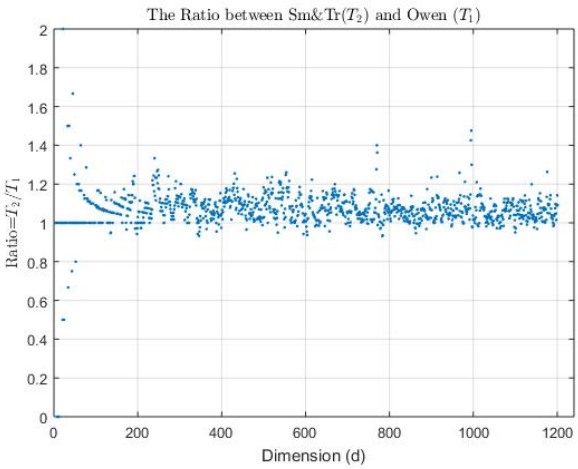

Figure 15: Ratio between time execution of Owen's algorithm over Smith's & Tromble's for arbitrary simplex.

3.3.2 Grimme's algorithm

Cristian Grimme in [16] gives a different but less time efficient algorithm for uniformly sampling points from an arbitrary simplex. His main idea comes from Turk's method. We will not deal with details, so we just give the formula:

$$P_{dS} = \sum_{i=1}^{d} \left((1-\lambda_i) \prod_{j=1}^{i-1} \lambda_j \right) A_i + \left(\prod_{k=1}^{d} \lambda_k \right) A_{d+1} =$$

$$P_{dS} = \sum_{i=1}^{d+1} \left((1-\lambda_i) \prod_{j=0}^{i-1} \lambda_j \right) A_i \quad \text{with } \lambda_0 = 1 \text{ and } \lambda_{d+1} = 0$$

Vectors (vertices) A_i, $i = 1, \cdots, d+1$, spanning the simplex. In order to choose a random point for each $\lambda_j \in (\lambda_1, \cdots, \lambda_d)$ we pick a random number $z_j \in U(0,1)$ and set $\lambda_j = \sqrt[k]{z_j}$ with $k = d+1-j$.

The time complexity for a random point to be generated is $O(d^2)$. In figures 16, 17 we can notice that Grimme's algorithm is slower for every dimension due to constants.

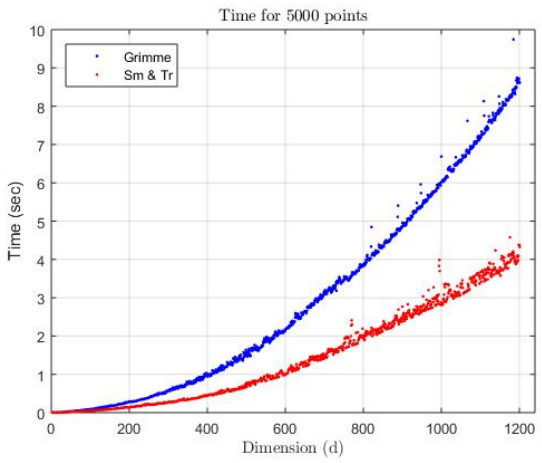

Figure 16: Comparing Grimme's algorithm with Smith's & Tromble's algorithm.

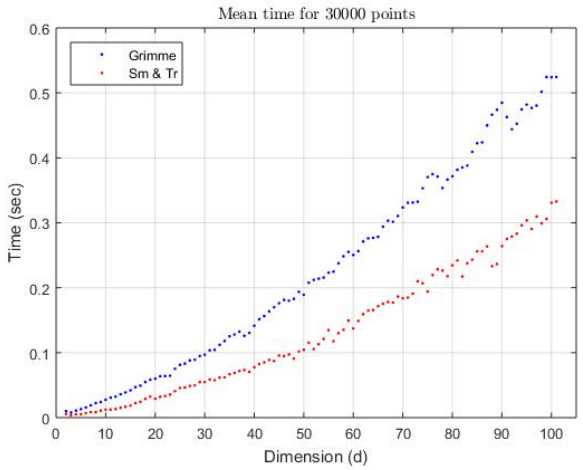

Figure 17: Comparing Grimme's algorithm with Smith's & Tromble's algorithm.

Finally we can conclude that the most efficient method to uniformly sample points from an arbitrary simplex is to sample points as we described in 3.1 and then map them to the arbitrary simplex applying mapping 3.3

4. REJECTION-SAMPLING METHOD

In this chapter we discuss rejection-sampling methods which are used for the volumes' computations in the financial application we describe in 7.1. At first we give some results for the accuracy of the rejection-sampling algorithmic family. We show that we can bound the error with high probability by setting an appropriate number of points we sample. This result can be applied in rejection-sampling method from a simplex or a sphere. Next we use these results in order to solve the main problem efficiently and finally we give formulas for the ellipsoid transformation to a full dimensional ellipsoid intersecting the unit simplex Δ^d. The latter is necessary for the random walk methods we use in chapter 6.

4.1 Sampling-Rejection accuracy

Let B be a convex or non convex full dimensional body in dimension d, let S be an enclosing simplex such that $B \subseteq S$ and let $p = \frac{Vol(B)}{Vol(S)}$. Then if we uniformly sample a point from S it lies in B with probability p. So if we sample N points from S the random variable X which gives the number, k, of points that lie in B follows the binomial distribution. So,

$$P(X = k) = \binom{N}{k} p^k (1-p)^{N-k}$$

Moreover if we sample N points and reject in order to compute an approximation of $Vol(B)$ the sum,

$$\sum_{k=n_1}^{n_2} P(X = k) \tag{4.1}$$

where $n_1 = Np(1-e)$ and $n_2 = Np(1+e)$, is the probability that the rejection method error is at most e.

Theorem 4.1. *Let X be a binomial random variable with parameters N, p and $lim_{N\to\infty} Np = \lambda$ is a constant that is independent of N. Then for any fixed k,*

$$lim_{N\to\infty} P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

Theorem 4.1 is known as *Poisson limit theorem*. If we set $N = m_1 \cdot 10^x$, where $x = m_2 + \lceil -\log_{10} p \rceil$, we notice that $Np \approx m_1 \cdot 10^{m_2}$. So we can use *Poisson limit theorem* to approximate the random variable X which gives the number, k, of points that lie in B through Poisson distribution,

$$P(X = k) \approx e^{-\lambda}\frac{\lambda^k}{k!}, \text{ where } \lambda = m_1 \cdot 10^{m_2}$$

Then from Poison cumulative distribution function,

$$F_{X_n}(x) = P(X_n \leq k) = \frac{\Gamma(\lfloor k+1 \rfloor, \lambda)}{\lfloor k \rfloor !} \tag{4.2}$$

we can approximate probability 4.1. For example for $e = 0.01$ we set $N = 4 \cdot 10^{4+\lceil -\log_{10} p \rceil}$ and we get,

$$\sum_{k=n_1}^{n_2} P(X = k) \approx F_X(n_2) - F_X(n_1) = 0.955$$

So it seems that we can guarantee with high probability that the error of rejection-sampling method is at most 1%, when we sample $N = 4 \cdot 10^{4+\lceil -\log_{10} p \rceil}$ points from S. Notice that when the order of p drops by one we have to increase the order of N by one. This result also applies to rejection-sampling method using enclosing sphere. If we compute probability 4.1 from binomial distribution for one thousand values of p we get figure 18 which seems to agree with previous results.

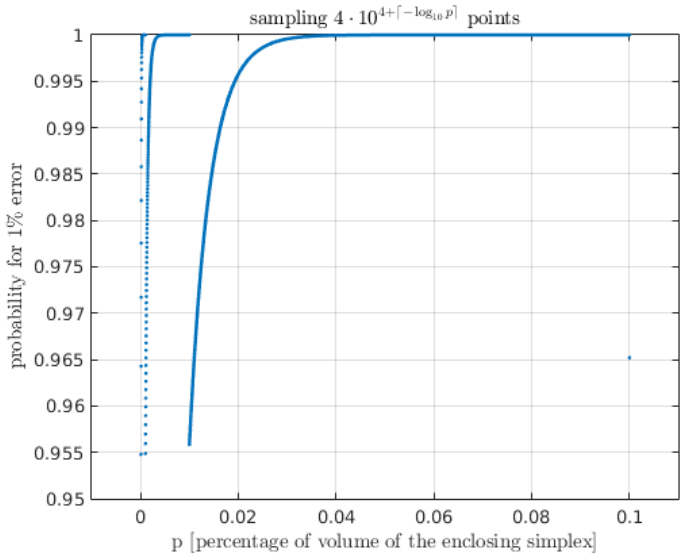

Figure 18: Probabilities related to the proportion of the enclosed body's volume over the enclosing body's volume for 1% error using binomial distribution.

Moreover c.d.f. 4.2 could be used to approximate same probabilities as in 4.1 for a different given error. Table 1 gives the probabilities of maximum errors of rejection-sampling method related to the number of points N we have to sample.

An alternative approach is to use Chebyshev inequality and obtain the following:

$$P\Big(Np(1-e) \leq X \leq Np(1+e)\Big) \geq 1 - \frac{\frac{1}{p} - 1}{4Ne^2} \tag{4.3}$$

Using 4.3 and direct computations from 4.1 we can get almost the same values as in table 1.

Table 1: Maximum rejection-sampling method errors with high probability.

Sampling - Rejection accuracy		
$error$	N	Pr
1%	$4 \cdot 10^{4+\lceil -\log_{10} p \rceil}$	0.955
2%	$9 \cdot 10^{3+\lceil -\log_{10} p \rceil}$	0.942
3%	$4 \cdot 10^{3+\lceil -\log_{10} p \rceil}$	0.942
4%	$4 \cdot 10^{3+\lceil -\log_{10} p \rceil}$	0.972
5%	$2 \cdot 10^{3+\lceil -\log_{10} p \rceil}$	0.975
6%	$1 \cdot 10^{3+\lceil -\log_{10} p \rceil}$	0.942
7%	$8 \cdot 10^{2+\lceil -\log_{10} p \rceil}$	0.952
8%	$6 \cdot 10^{2+\lceil -\log_{10} p \rceil}$	0.951
9%	$5 \cdot 10^{2+\lceil -\log_{10} p \rceil}$	0.956
10%	$4 \cdot 10^{2+\lceil -\log_{10} p \rceil}$	0.955

4.2 Mapping convex bodies to the unit base

In practice when we use sampling for volume computation it is not common to uniformly sample points from an arbitrary simplex because we use 3.2 in order to transform the arbitrary simplex and any convex body that intersects with simplex to the unit base. Then we can equivalently sample from the unit simplex. The transformation of the simplex is trivial because linear transformation 3.2 defines a bijective relation between the points in the unit and in the arbitrary simplex.

Proposition 4.1. *Let C be a convex set and W be a linear transformation. Then $W(C)$ is convex set.*

Proof. Let $x = Wu$ and $y = Wv$, for some $u, v \in C$. Then for any $t \in [0, 1]$ we have,

$$tx + (1-t)y = tWu + (1-t)Wv = W(tu + (1-t)v)$$

but $tu + (1-t)v \in C$ so $tx + (1-t)y \in W(C)$. □

So for every convex body that intersects an arbitrary simplex we can use linear transformation 3.2 in order to get a new convex body that intersects with the unit simplex. Moreover the proposition below is very useful.

Proposition 4.2. *Let S be a set and A be a linear transformation matrix. Then $Vol(TS) = Vol(T)Vol(S)$.*

For more details about proposition 4.2 see [40].

So if S is the set defined by the intersection of the unit simplex and a convex body C and T is a matrix of a linear transformation then $\frac{Vol(S)}{Vol(\Delta^d)} = \frac{Vol(TS)}{Vol(T\Delta^d)}$. So if we transform a convex body C and an arbitrary simplex R to the unit base using 3.2 we can have an approximation of the ratio $\frac{Vol(R \cap C)}{Vol(C)}$ by uniformly sampling the unit simplex.

For example, if a halfspace $H := c^T x \leq z$, where $c \in \mathbb{R}^d, z \in \mathbb{R}$ intersects with a simplex R then $\frac{Vol(H \cap R)}{Vol(R)} = \frac{Vol(H' \cap \Delta^d)}{Vol(\Delta^d)}$, where $H' := c^T Ax \leq z$.

So the main idea is to pay some time for the transformation of the convex body and then sample in $O(kd)$ rather than sampling in $O(kd^2)$, where k is the number of points we sample. In practice this is always beneficial.

4.3 Two parallel families of hyperplanes or ellipsoids

In the financial application described in chapter 2 the main problem was to compute the volumes defined by the intersection of two full-dimensional parallel families of hyperplanes or one family of hyperplanes and one family of concentric parallel ellipsoids with the canonical simplex in $\mathbb{R}^{d+1}$. So, for both problems, we have to make some modifications to the sampling algorithms in chapter 3, because we have to sample from the canonical simplex and not from the unit.

In section 3.1 we first sample from the canonical simplex $\Delta^d \subset \mathbb{R}^{d+1}$ in both variants of the sampling algorithm and then project to the unit simplex. So it is easy to apply only the sampling and not the projection. After sampling we set the distance between two successive hyperplanes of the same family such as to define volume equal to 1% of the canonical simplex Δ^d. We compute the inner products between the direction of the family and all the N points, then we sort them and finally we choose the constants that leave $N/100$ points between successive hyperplanes. An exact and better in time efficiency method could be applied using Varsi's or Lasserre's formulas described in section 5.1.

Furthermore we use the results from section 4.1 in order to achieve an efficient approximation. In practice the smallest volume in dimension $d = 100$ is $Vol(\Delta^d) \cdot 10^{-5}$. So achieving 1% error is unrealistic for these volumes but if we set $N = 4 \cdot 10^7$ we achieve 3% error at most with probability 0.942 for the 99% of the volumes. If better accuracy is required we can follow table 1 in order to set the number of points we sample accordingly.

4.3.1 Transform the ellipsoids

The d-dimensional canonical simplex $\Delta^d \subset \mathbb{R}^{d+1}$ may be represented by barycentric coordinates $\lambda = (\lambda_0, \ldots, \lambda_d)$ s.t. $\sum_{i=0}^{d} \lambda_i = 1$, $\lambda_i \geq 0$. The points are $\sum_{i=0}^{d} \lambda_i v_i$, where $v_0, \ldots, v_d \in \mathbb{R}^d$ are affinely independent. In order to run random walk methods we have to

use a full-dimensional simplex, by switching to Cartesian coordinates $x = (x_1, \ldots, x_d)$:

$$m_{bc} : \mathbb{R}^{d+1} \mapsto \mathbb{R}^d : \lambda \to x = M(\lambda_1, \ldots, \lambda_n) + v_0, \text{ where } M = [v_1 - v_0 \cdots v_d - v_0],$$

is a $d \times d$ invertible matrix. The inverse transform is:

$$m_{cb} : \mathbb{R}^d \mapsto \mathbb{R}^{d+1} : x \to \lambda = \begin{bmatrix} -1_d^T \\ I_d \end{bmatrix} M^{-1}(x - v_0) + \begin{bmatrix} 1 \\ 0_d \end{bmatrix}, \tag{4.4}$$

where $0_d, 1_d$ are d-dimensional column vectors of 1's and 0's, respectively, and I_d is the d-dimensional identity matrix.

In our financial application, portfolios are points in the canonical simplex. The simplex lies in hyperplane $\sum_{i=0}^{d} \lambda_i = 1$. To model levels of volatility, a family of full-dimensional ellipsoids in $\mathbb{R}^{d+1}$, centered at the origin, is defined by the covariance matrix C of asset returns. We wish to compute the volume of intersections of this family with the simplex and, moreover, with a family of hyperplanes on the simplex. Except Sampling-Rejection which would work in this context, however methods employing random walks require a full-dimensional convex body. Given a full $(d+1)$-dimensional ellipsoid $G : \lambda^T C \lambda - c = 0$ centered at the origin, where $C \in \mathbb{R}^{(d+1)\times(d+1)}$ is symmetric positive-definite, we compute the equation of the ellipsoid defined $G \cap \Delta^d \subset \mathbb{R}^d$, by imposing the constraint $\sum_{i=0}^{d} \lambda_i = 1$ by transform m_{cb} in expression (4.4), thus obtaining:

$$(x - v_0)^T \left(M^{-T} [-1\, I_d]\, C \begin{bmatrix} 1 \\ 0_d \end{bmatrix} M^{-1} \right) (x - v_0) + A(x - v_0) = c',$$

where the expression in parenthesis is the matrix defining the new d-dimensional ellipsoid in Cartesian coordinates, and $A \in \mathbb{R}^{d\times d}, c' \in \mathbb{R}$ are obtained by direct calculation. Similarly the simplex maps to Cartesian coordinates.

5. EXACT COMPUTATIONS

In this chapter we are dealing with the polytopes defined by the two families of parallel hyperplanes applying exact computation algorithms. First we give Varsi's and Lassere's formulas which are exact formulas for the volume defined by the intersection of a simplex with a hyperplane. This formula is useful in order to define the distances between successive hyperplanes that belong to the same family. Next we give an optimized Lawrence sign decomposition method since the polytopes at hand are shown to be simple.

5.1 Varsi's formula

There exist an exact, iterative formula for the volume defined by intersecting a simplex with a hyperplane. A geometric proof is given in [39], by subdividing the polytope into pyramids and, recursively, to simplices. Here we give a somewhat simpler formula [2], which also requires $O(d^2)$ operations. This formula was expressed by Ali using Fourier-Stiltjes transform. Let $H = \{(x_1, \ldots, x_d) | \sum_{i=1}^{d} a_i x_i \leq z\}$ be the linear halfspace.

1. Compute $u_j = a_j - z$, $j = 1, \ldots, d$. Label the nonnegative u_j as $Y_1, \ldots, Y_K$ and the negatives as $X_1, \ldots, X_J$. Initialize $A_0 = 1, A_1 = A_2 = \cdots = A_K = 0$.
2. For $h = 1, 2, \ldots, J$ repeat: $A_k \longleftarrow \dfrac{Y_k A_k - X_h A_{k-1}}{Y_k - X_h}$, for $k = 1, 2, \ldots, K$.

If $\Delta^d \subset \mathbb{R}^d$ is the unit simplex then, for $h = J$, $A_K = \text{vol}(\Delta^d \cap H)/\text{vol}(\Delta^d)$.

Recall, from Section 3.1, that sampling uniformly over the simplex can be obtained by drawing exponential random variables. Thus, an alternative formula follows from computing the cumulative distribution of a linear combination of exponential random variables. In [30], they propose an exact method to compute the distribution f of such linear combination. It consists in representing f as its moment generating function, analogous to a Laplace transform, simplifying it with a generalized partial-fraction technique of integration, before inverting its terms. However, in double precision, the method showed numerical discrepancies above 20 dimensions and was thus abandoned. However, it has the advantage of being generalizable to nonlinear combinations.

5.2 Lasserre's formula

Lasserre, in [22], gives another formula for the volume defined by intersecting a simplex with a hyperplane 22 years after Varsi, using Laplace transform and the following Lemma.

Lemma 5.1. *Let* $\mathbf{c} := (c_1, \ldots, c_n) \in \mathbb{R}^d$, $c_0 := 0$, *with* $c_i \neq c_j$ *for every distinct pair (i, j).*

Then

$$\int_{\Delta^d} exp(-\mathbf{c}^T\mathbf{x}) = \sum_{i=0}^{d} \frac{exp(-c_i)}{\prod_{j\neq i}(c_i - c_j)}$$

Let $H = \{(x_1, \ldots, x_d) | \sum_{i=1}^{d} a_i x_i \leq t\}$ be the linear halfspace and $a_0 = 0$. Then

$$Vol(\Delta^d \cap H) = \frac{1}{d!} \sum_{i=0}^{d} \frac{(t - a_i)_+^d}{\prod_{j\neq i}(a_j - a_i)} \tag{5.1}$$

Given a scalar $x \in \mathbb{R}$, the notation $(x)_+$ stands for $max[0, x]$. The formula 5.1 takes $O(d^2)$ and it seems to be simpler than Varsi's formula. In [22] Lasserre gives an exact formula for the case of identical weights of the halfspace H and for the case of an arbitrary simplex.

5.3 Simple polytopes

This section considers simple polytopes defined by a constant number of families of parallel hyperplanes; in our application there are two such families. The defined polytopes are simple, i.e., all vertices are defined at the intersection of d hyperplanes, assuming that no hyperplane contains any of the simplex vertices and, moreover, two hyperplanes does not intersect on a simplex edge at the same point.

For a simple polytope P, the decomposition by Lawrence [23] picks $c \in \mathbb{R}^d$, $q \in \mathbb{R}$ such that $c^T x + q$ is not constant along any edge, i.e. $c, -c$ do not lie on the normal fan of any edge. For each vertex v, let $A(v)$ be the $d \times d$ matrix whose columns correspond to the equations of hyperplanes through v. Then $A(v)$ is invertible and vector $\gamma(v)$ such that $A(v)\gamma(v) = c$ is well defined up to a permutation. The assumption on c assures no entry vanishes, then

$$\text{vol}(P) = \frac{1}{d!} \sum_{v} \frac{(c^T v + q)^d}{|\det A(v)| \prod_{i=1}^{d} \gamma(v)_i}.$$

The computational complexity is $O(d^3 n)$, where n is the number of vertices. We set $q = 0$ for simplicity in the implementation. An issue is to choose c so as to avoid that $c^T x + q$ be nearly constant on some edge, because this would result in very small entries in the denominator and numerical issues. A theoretical choice is given in [23], but its practical importance is very small. The main drawback of Lawrence's decomposition remains numerical instability when executed with floating point numbers, and high bit complexity, when executed over rational arithmetic. The latter is indispensible for $d > 30$ in our applications, because then numerical results become very unstable.

To compute the volume defined by the intersection of a simplex and two arbitrary hyperplanes, we exploit the fact that the simplex is unit in order to compute more effectively the determinants and the solutions of the linear system. The hardest case is when vertex v is defined by the two arbitrary hyperplanes H_a, H_b, the supporting hyperplane

$H_0 : \sum_{i=1}^{d} x_i = 1$, and $d-3$ hyperplanes of the form $H_i : x_i = 0$. Then, up to row permutations,

$$A(v) = \begin{bmatrix} -1 & & & 1 & a_1 & b_1 \\ & \ddots & & \vdots & \vdots & \vdots \\ & & -1 & 1 & a_{d-3} & b_{d-3} \\ & & & 1 & a_{d-2} & b_{d-2} \\ & & & 1 & a_{d-1} & b_{d-1} \\ & & & 1 & a_d & b_d \end{bmatrix}, \tag{5.2}$$

where the i_j, $i = a, b$ are the coefficients of the equation of H_i up to permutation. Then we solve the lowest right 3×3 linear system in $O(1)$ and then the computation of each remaining unknown $\gamma(v)_i$, $i = 1, \ldots, d-3$ requires $O(1)$ operations for a total of $O(d)$. The corresponding determinant is computed in $O(1)$.

Lemma 5.2. *Polytopes in H-representation, defined by intersecting the simplex with two arbitrary hyperplanes in $\mathbb{R}^d$, have $O(d^2)$ vertices, which are computed in $O(1)$ each.*

Proof. A vertex in the new polytope is of one of 3 types: (i) It may be a vertex of unit simplex Δ. It suffices to check all simplex vertices against hyperplanes H_a, H_b in total time $O(d)$. (ii) It may be the intersection of a simplex edge with H_a, which is easy to identify and compute by intersecting simplex edges whose vertices lie on different sides of H_a, with H_a. Each such edge is defined by at least one coordinate hyperplane, so computing the edge intersection with H_a is in $O(1)$. These vertices are checked against H_b in $O(1)$ each, since they contain at most two nonzero coordinates. There are $O(d^2)$ such edges, hence the total complexity is $O(d^2)$.

(iii) It may be defined as $H_a \cap H_b \cap \Delta$, i.e. the intersection of H_a with the edges of $H_b \cap \Delta$. Let B_1, B_2 be vertices on $H_b \cap \Delta$. Then B_1 is defined by the intersection of H_b and an edge (v_i, v_j) of the unit simplex, when v_i and v_j lie on different sides of H_b and B_2 by the intersection of H_b and an edge (v_k, v_m). That means that every vertex in $H_b \cap \Delta$ corresponds to a unit simplex edge. Then we have 3 cases:

1. B_1, B_2 lie on the same side of H_a: no vertex is defined.
2. If $i \neq k, i \neq m, j \neq k, j \neq m$ there is not an edge between B_1 and B_2.
3. If B_1, B_2 correspond to simplex egdes that have a common vertex and lie on different sides of H_a, then a polytope's vertex is defined, which has at most 3 nonzero coordinates.

In the worst case $d/2$ simplex vertices lie on the same side of H_b and $d/2$ on the other. Then the polytope's vertices that are defined by $H_a \cap H_b \cap \Delta$ are $d\frac{d}{2} = O(d^2)$. □

Lawrence's formula requires both H- and V-representation. In our setting, the H-representation is known, but the previous lemma allows us to obtain vertices as well.

Proposition 5.1. *Let us consider polytopes defined by intersecting the simplex with two arbitrary hyperplanes. The total complexity of the Lawrence sign decomposition method, assuming that the H-representation is given, is* $O(d^3)$.

The entire discussion extends to polytopes defined by two families of parallel hyperplanes. The matrices $A(v)$ remain of the same form because each vertex is incident to at most one hyperplane from each family.

6. RANDOM WALKS

This chapter considers more general convex bodies, defined as a finite, bounded intersection of linear and nonlinear halfspaces. For this, we extend the polynomial-time approximation algorithm `VolEsti` in [14] so as to handle nonlinear constraints. Our primary motivation here is computing the volume of the intersection of a simplex with an ellipsoid and two parallel hyperplanes in general dimension.

6.1 Previous work

The method in [14] follows the Hit-and-Run algorithm in [27], and is based on an approximation algorithm in $O^*(d^5)$. It scales in a few hundred dimensions by integrating certain algorithmic improvements to the original method.

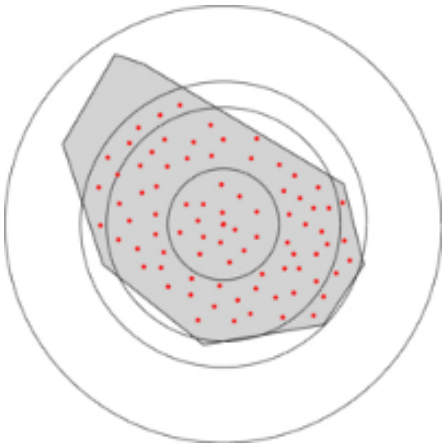

Figure 19: A $2D$ representation of VolEsti.

Given a polytope $P = \{x \in \mathbb{R}^d : a_i^T x \leq b, \quad i = 1, \ldots, m\}$ in H-representation, the `VolEsti` algorithm could be divided into four steps:

1. Rounding of the polytope (optional). An efficient method to get a well rounded polytope (or a convex body) is to sample uniformly from the polytope a large number of points denoted as the set S and then compute an ϵ-approximate minimum volume ellipsoid E covering S:
$$E/(1+\epsilon)d \subset CH(S) \subset E$$
Then we could apply a linear transformation to the ellipsoid and get the unit sphere and transform the polytope similarly, see Figure 20. We can repeat this process until the ratio of the maximum over minimum ellipsoid axis reaches threshold.

2. Computation of the chebychev ball $B(c, r_{min})$ and approximate the minimum enclosing ball of the polytope P. The computation of the polytope's Chebychev ball reduces

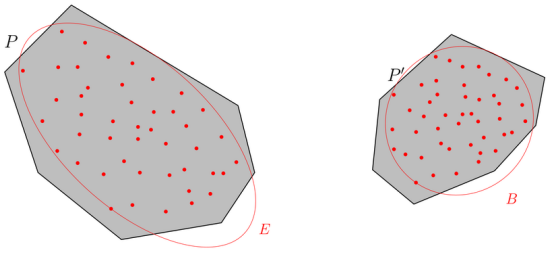

Figure 20: Rounding a polytope.

to a Linear program. We want to compute the largest ball $B = \{x_c + u : ||u||_2 \leq r\}$ in the interior of P, so we have to solve the following LP:

$$\begin{aligned} &\text{maximize} && r \\ &\text{subject to} && a_i^T x_c + r||a_i||_2 \leq b_i, \quad i = 1, \ldots, m \end{aligned}$$

In order to find an approximate enclosing ball of the polytope P we sample uniformly from P, obtaining a set of random points S and then define the ball $B(x_c, r_{max})$, where $r_{max} = \max\{||x_c - p||_2 : p \in S\}$.

3. Define the sequence $B(c, 2^{i/d})$ of concentric balls $B(c, 2^{i/d})$, $i = \lfloor dlog(r_{min}) \rfloor, \ldots, \lceil dlog(r_{max}) \rceil$ and $B(c, r_{min}) \subset P \subset B(c, r_{max})$. Then $P_i := P \cap B(c, 2^{i/d})$

4. At each iteration i sample uniformly from P_i in order to estimate all the ratios $\frac{P_{i+1}}{P_i}$. We sample uniformly using Coordinate Directions Hit-and-Run from the P_{i+i} and reject points in the P_i.

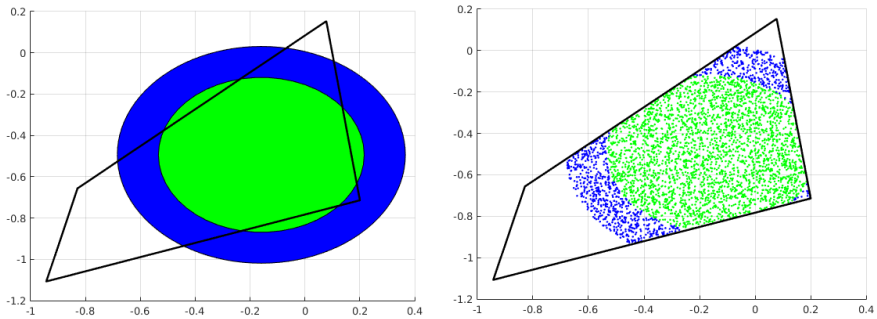

Figure 21: Approximating ratios of volume using uniform sampling.

Then we have that,

$$Vol(P) = Vol(P_{dlog(r_{min})}) \prod_{i=\lfloor dlog(r_{min}) \rfloor + 1}^{\lceil dlog(r_{max}) \rceil} \frac{P_{i+1}}{P_i}$$

For the uniform sampling from P_i there are many algorithms based on random walks as you can see in Table 2. We use Coordinate Direction Hit-and-Run because is more efficient in practice.

Table 2: Random walk algorithms

Random walk algorithms			
Year	Algorithm	Steps	Cost/step
1997 [19]	Ball Walk	n^3	mn
2003 [28]	Hit-and-Run	d^3	md
2009 [20]	Dikin Walk	md	$md^{\omega-1}$
2016 [26]	Geodesic Walk	$md^{3/4}$	$md^{\omega-1}$
2017 [25]	RHMC	$md^{2/3}$	$md^{\omega-1}$

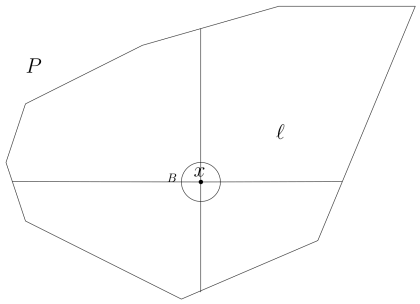

Figure 22: Coordinate Directions HnR.

Coordinate Directions Hit-and-Run

Input: point $x \in P$ and polytope $P \subset \mathbb{R}^d$
Output: a uniform point in P

- line ℓ through x, uniform on $\{e_1, \ldots, e_d\}, e_i = (\ldots, 0, 1, 0, \ldots)$
- x is uniformly distributed on $P \cap \ell$

Perform W steps, return x.

In Figure 22 you can see a $2D$ representation of CDHR algorithm. In `VolEsti` which is a practical method for volume approximation we set $W = O(d)$.

6.2 Extend VolEsti

We have to generalize `VolEsti` because the input is not a polytope but a general convex body, while the method works for d-polytopes. It suffices to solve two subproblems:

1. Compute the maximum inscribed ball of the convex body a.k.a. Chebychev ball
2. Compute the intersection points of a line that crosses the interior of the convex body P with the boundary of P.

The first problem is treated in the next section. For the second one, when the body is the intersection of linear and quadratic halfspaces, it suffices to solve systems of linear or quadratic equations. In our case where P has few input hyperplanes we can optimize that procedure by transforming a base of our polytope to an orthonormal base thus obtaining very simple linear systems. One heuristic is to first compute the intersection of the line with all hyperplanes and test whether the intersection points lie inside the ellipsoid so as to avoid intersecting the line with the ellipsoid. Formally, every ray ℓ in Coordinate Direction Hit-and-Run is of the form $p + \lambda e_k$ and parallel to $d-1$ simplex facets. The roots of,

$$\lambda^2 + 2\lambda p_k + |p|^2 - R^2$$

define the intersection of a sphere with radius R, centered at the origin, and a coordinate direction ray ℓ. If C is the matrix of an ellipsoid centered at the origin its intersections with ℓ are roots of:

$$C_{kk}x^2 + bx + c = 0, \quad b = 2C_{kk}p_k + 2\sum_{j=k+1}^{d} C_{kj}p_j + 2\sum_{i=0}^{k-1} C_{ik}p_i,$$

$$c = \sum_{i=0}^{d} C_{ii}p_i^2 + 2\sum_{j=i+1}^{d} C_{ij}p_i p_j, \quad i = 0, \ldots, d.$$

Computing the roots, and keeping the largest negative and smallest positive λ is quite fast, as it takes $O(d)$.

In our application, there are non-convex bodies defined by the intersection of two parallel hyperplanes and two concentric ellipsoids. We thus modify `VolEsti` in order to compute the non convex volume. We make two major changes:

- First, in ray shooting, we have to check whether one quadratic equation has only complex solutions, which implies the ray does not intersect the ellipsoid. For λ, we take the largest negative and the smallest positive root in every step as well.
- Second, for the initial interior point, we sample from the unit simplex and when we find a point inside the intersection we stop and use it for initialization. We define an inscribed ball with this center and radius equal to some small $\epsilon > 0$. We stop the algorithm when we find the first inscribed ball as described in the next subsection. So we can set ϵ sufficiently small so it always defines an inscribed ball in practice, but the enclosing ball is enough to run the algorithm and do not stop until we find an inscribed ball.

The method works fine for $d < 35$ using the same walk length and number of points as for the convex case, and has time complexity and accuracy competitive to running `VolEsti` on the convex set defined by one ellipsoid. For $d > 35$, the method fails to approximate volume for most of the cases. This should be due to inaccurate rounding bodies and the inscribed ball we define.

6.2.1 Chebychev ball

This section offers methods for computing a ball inside the given convex region. Ideally, this is the largest inscribed ball, aka Chebychev ball, but a smaller ball may suffice. Computing the Chebychev ball reduces to a linear program when P is a polytope. For general convex regions, more general methods are proposed.

We start with some simple approaches. Let us consider the case of intersecting a simplex with an ellipsoid.

- If there are z_1 simplex vertices inside the ellipsoid and z_2 outside, then we have $(z_2+1)z_1$ vertices on the boundary of the convex intersection. Since $z_1+z_2=d+1$, then $(z_2+1)z_1 \geq d+1$ and a new inscribed simplex is defined. In this case we take its largest inscribed ball and start hit and run.
- More generally, we sample from the unit simplex until we have $d+1$ points inside our section and then take the largest inscribed ball of this new simplex that is defined by the $d+1$ points.
- Another approach is to consider the transformation mapping the ellipsoid to a sphere and apply it both to the simplex and to the ellipsoid. We compute the distance from the sphere's center to the new simplex and compare it with the sphere's radius.

At the very least, one point must be obtained inside the convex region. When we do not have the Chebychev ball, an issue is that concentric balls with largest radii will again be entirely contained in the convex region, thus wasting time in the computation. In practice we use the one interior point as center of an enclosing ball, then reduce the radius until the first inscribed ball. To decide whether a given ball is inscribed, with high probability, we check whether all boundary points in Hit-and-Run belong to the sphere instead of any other constrain.

For a convex body that comes from intersecting a polytope with k balls the problem becomes a Second-Order Cone Program (SOCP) [4] with k cones. However in our case we need to consider input ellipsoids. Assume that we transformed the ellipsoid to a ball $B'=\{x'_c+u' \ : \ \|u'\| \leq r'\}$, and applied the same transformation to the simplex to have $a_i x \leq b_i$ for $i \in [d+1], a_i \in \mathbb{R}^d, b_i \in \mathbb{R}$. The following SOCP computes the maximum ball $B=\{x_c+u \ : \ \|u\| \leq r\}$ in the intersection of the simplex and B':

$$\begin{aligned} & \max \ r \\ \text{subject to}: \quad & a_i^T x_c + r||a_i|| \leq b_i \\ & ||x'_c - x_c|| \leq r' - r \end{aligned}$$

There are several ways to solve SOCP's such as to reformulate it to as a semidefinite program or perform a quadratic program relaxation. Moreover, since in our case we only have a single cone we could utilize special methods as in [15]. However, for our case it suffices to use the generic SOCP solver from [9] as it is very efficient; for a random simplex

and a ball, it takes 0.06 sec in $d = 100$ and < 20 sec in $d = 1000$, on Matlab using `ecos` and `yalmip` packages.

It is possible to apply the inverse transformation and get an inscribed ellipsoid, which is not necessarily largest possible. However we can use the maximum inscribed ball in that ellipsoid as an approximation of the Chebychev ball, by taking the center of that ellipsoid and the minimum eigenvalue of its matrix as the radius.

7. APPLICATION AND EXPERIMENTS

Our implementations are in C++, lie in the public domain[1], and are using CGAL and Eigen. All experiments of the paper have been performed on a personal computer with Intel Pentium G4400 3.30GHz CPU and 16GB RAM. Times are averaged over 100 runs. Some resulting tables and figures are given in the Appendix.

We test the following convex bodies: a d-simplex intersected with: (1) two arbitrary halfspaces, (2) two parallel halfspaces, (3) an ellipsoid, (4) two parallel halfspaces and two cocentric ellipsoids (non convex body).

Here we use the following notation for the methods we develop: The (M1) method is the Varsi's exact formula for the volume defined by the intersection of simplex with a hyperplane. The second (M2 or s/r) is to sample the unit simplex and approximate all the volumes directly. The third method (M3) is the optimized Lawrence formula for simple polytopes and is used for the first problem. The fourth method (M4) is the generalization of the `VolEsti` algorithm to non-linear and non-convex bodies.

In general, M1 is preferred when available. Method M2 is the fastest and scales easily to 100 dimensions, so it is expected to be useful for larger dimensions. However, for small volumes its accuracy degrades; sampling more points makes it slower than M4. The latter is thus the method of choice for volumes $< 1\%$ of the simplex volume, but it is not clear whether it would be fast beyond $d = 100$. Method M3 is useful, even for small volumes, but it cannot scale to $d = 100$ due to numerical instability; if we opt for exact computing, it becomes too slow.

7.1 Financial context and application

Modern finance has been pioneered by Markowitz who set a framework to study choice in portfolio allocation under uncertainty, see [29].[2] Within this framework, Markowitz characterized portfolios by their return and their risk which is defined as the variance of the portfolios' returns. And an investor would build a portfolio that will maximize its expected return for a chosen level of risk. It has since be common for asset managers to optimize their portfolio within this framework. And it has led a large part of the empirical finance research to focus on the so-called efficient frontier which is defined as the set of portfolios presenting the lowest risk for a given expected return. Figure 23 (left panel) presents such an efficient frontier. The region on the left of the efficient frontier represent the portfolios domain.

Interestingly, despite the fact that this framework considers the whole set of portfolios, no attention has been given to the distribution of portfolios. Figure 23 (middle panel) presents

[1] https://github.com/TolisChal/volume_approximation

[2] for which he was awarded the Nobel Prize in economics in 1990.

such distribution. When comparing the contour of the empirical portfolios distribution[3] and the portfolio domain bounded by the efficient frontier in Figure 23 (right panel), we observe that the density of portfolios along the efficient frontier is dim and that most of the portfolios are located in a small region of the portfolios domain.

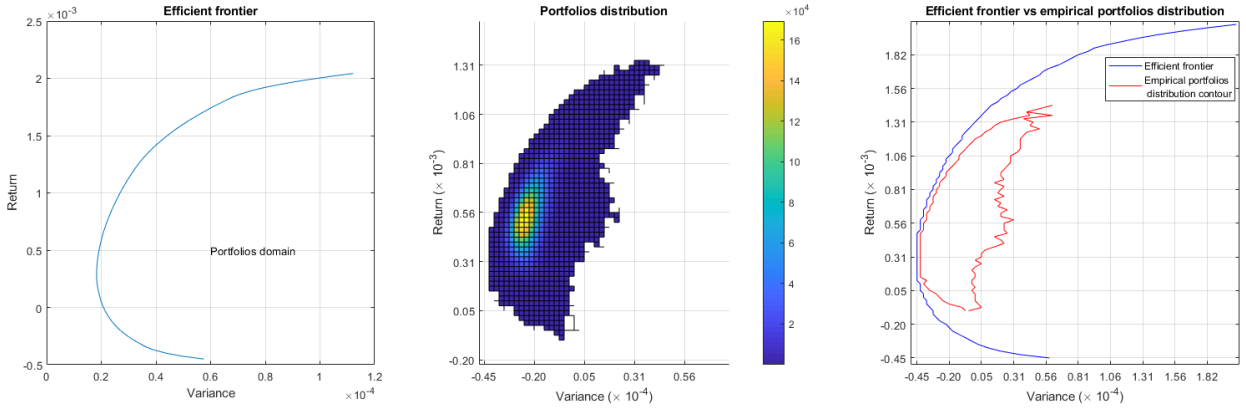

Figure 23: (left) Efficient frontier, (middle) Empirical portfolio distribution by portfolios' return and variance, (right) Efficient frontier in blue and contour of the empirical portfolio distribution in red. The market considered is made of the 19 sectoral indices of the DJSTOXX 600 Europe. The data is from October 16, 2017 to January 10, 2018.

We also know from the financial literature that financial markets exhibit 3 types of behavior. In normal times, stocks are characterized by slightly positive returns and a moderate volatility, in up-market times (typically bubbles) by high returns and low volatility, and during financial crises by strongly negative returns and high volatility, see e.g. [3] for details. So, following Markowitz' framework, in normal and up-market times, the stocks and portfolios with the lowest volatility should present the lowest returns, whereas during crises those with the lowest volatility should present the highest returns. These features motivate us to describe the time-varying dependency between portfolios' returns and volatility.

However this dependency is difficult to capture from the usual mean-variance representation, as in Figure 23 (middle panel), so we will rely on the copula representation of the portfolios distribution. A copula is a bivariate probability distribution for which the marginal probability distribution of each variable is uniform. As we following Markowitz' framework, the variables considered are the portfolios' return and variance. Figure 24 illustrates such a copula and shows a positive dependency between portfolios returns and variances. Each line and column sum to 1% of the portfolios.

The methods introduced here can be used to study other dependencies such as the momentum effect [18] which is implied by the dependencies of asset returns with their past returns.

The dependencies mentioned here are important because

- through the return/volatility dependency, the detection of crises raises policy makers

[3] Region over which at least 1 random portfolio lies.

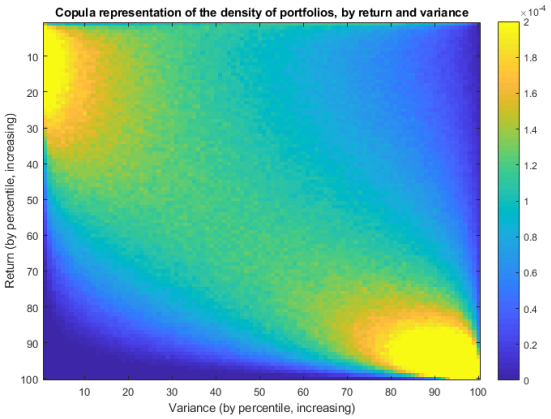

Figure 24: Copula representation of the portfolios distribution, by return and variance. The market considered is made of the 19 sectoral indices of the DJSTOXX 600 Europe. The data is from October 16, 2017 to January 10, 2018.

awareness and allows them to act accordingly with potentially large implications in citizens' life (employment, wages, pensions, etc).

- the momentum, if persistent, questions the efficiency of financial markets, a strong assumption which still cannot be proven wrong.

Interestingly, the copulas can be computed over a single period of time making the information available as early as the sample allows. The copula for the momentum dependency can be computed over very short periods (even intra-daily). The copula for the return/volatility dependency requires the estimation of the stock returns variance-covariance matrix which has to be estimated over a sufficiently large period of time to be reliable thus delaying the detection of crises.

In the general case, the framework to describe the dependencies is as follows. First, as the set of portfolios, we consider the canonical d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ where each point represents a portfolio and $d+1$ is the number of assets. The vertices represent portfolios composed entirely of a single asset. The portfolio weights, i.e. fraction of investment to a specific asset, are non-negative and sum to 1. This is the most common investment set in practice today, as portfolio managers are typically forbidden from short-selling or leveraging. Second, considering some asset characteristic ac quantified by $C \in \mathbb{R}^{d+1}$, we define a corresponding quantity $f_{ac}(\omega, C)$ for any portfolio $\omega \in \Delta^d$. For instance, considering the vector of asset returns $R \in \mathbb{R}^{d+1}$, ω has the return $f_{ret}(\omega, R) = R^T\omega$. Then, we define the cross-sectional score of a given portfolio ω^* as

$$\rho_{ac} = \frac{\text{vol}(\Delta^*)}{\text{vol}(\Delta^d)}, \quad \text{where } \Delta^* = \{\omega \in \Delta^d \,:\, f(\omega, C) \leq f(\omega^*, C)\},$$

which corresponds to the share of portfolios with a return lower or equal to $R^* = R^T\omega^*$. This score corresponds to the cumulative distribution function of $f_{ac}(\omega, C)$ where the portfolios are uniformly distributed over the simplex. In the following, we consider the cases where

f_{ac} is a linear combination or a quadratic form of C. Finally, the relationship between two asset characteristics ac_1 and ac_2 is presented in the form of a *copula* whose marginals are ρ_{ac_1}, ρ_{ac_2}. In our applications, the asset characteristics considered are the assets' returns and variances, and their values correspond to a linear combination of the returns and a quadratic form of the returns, respectively.

A copula is computed by slicing a simplex, i.e. the set of portfolios, along the asset characteristics. Thus, these questions are formulated in terms of convex bodies defined by intersecting simplices on one hand by a family of parallel hyperplanes and, on the other hand, by another family of parallel hyperplanes in the linear case or a family of concentric ellipsoids in the quadratic case. Furthermore, the latter case yields non-convex bodies between two ellipsoids.

In financial applications, one considers compound returns over periods of k observations, where typically $k = 20$ or $k = 60$; the latter corresponds to roughly 3 months when observations are daily. Compound returns are obtained using k observations starting at the i-th one where the j-th coordinate corresponds to asset j and the component j of the new vector equals:

$$(1 + r_{i,j})(1 + r_{i+1,j}) \cdots (1 + r_{i+k-1,j}) - 1, \quad j = 1, \ldots, d.$$

This defines the normal vector to a family of parallel hyperplanes, whose equations are fully defined by selecting appropriate constants. The second family of parallel hyperplanes is defined similarly by using an adjacent period of k observations.

The covariance matrix of the stock returns is computed using the shrinkage estimator of [24],[4] as it provides a robust estimate even when the sample size is short with respect to the number of assets. A covariance matrix C defines a family of ellipsoids centered at the origin $0 \in \mathbb{R}^d$ whose equations $x^T C x = c$ are fully specified by selecting appropriate constants c.

To compute the copulas, we determine constants defining hyperplanes and ellispoids so that the volume between two consecutive such objects is 1% of the simplex volume. The former are determined by bisection using the Varsi's exact formula. For ellipsoids $E(x) = c_i$, we look for the c_i's by sampling the simplex, then evaluating $E(x)$ at each point. The values are sorted and the c_i selected so as to define intervals containing 1% of the values. Two consecutive ellipsoids intersecting the simplex and the family of parallel hyperplanes define a non-convex body for which we practically extend `VolEsti` algorithm.

The volume between two consecutive hyperplanes and two consecutive ellipsoids defines the density of portfolios whose returns and volatilities lie between the specified constants. We thus get a copula representing the distribution of the portfolios with respect to the portfolios returns and volatilities. Fig. 25 illustrates such copulae, and shows the different relationship between returns and volatility in good (left, dot-com bubble) and bad (right, bubble burst) times.[5]

[4] Matlab code on `http://www.econ.uzh.ch/en/people/faculty/wolf/publications.html`.

[5] We consider 100 components of DJ 600 with longest history, over 60 days ending at the given date.

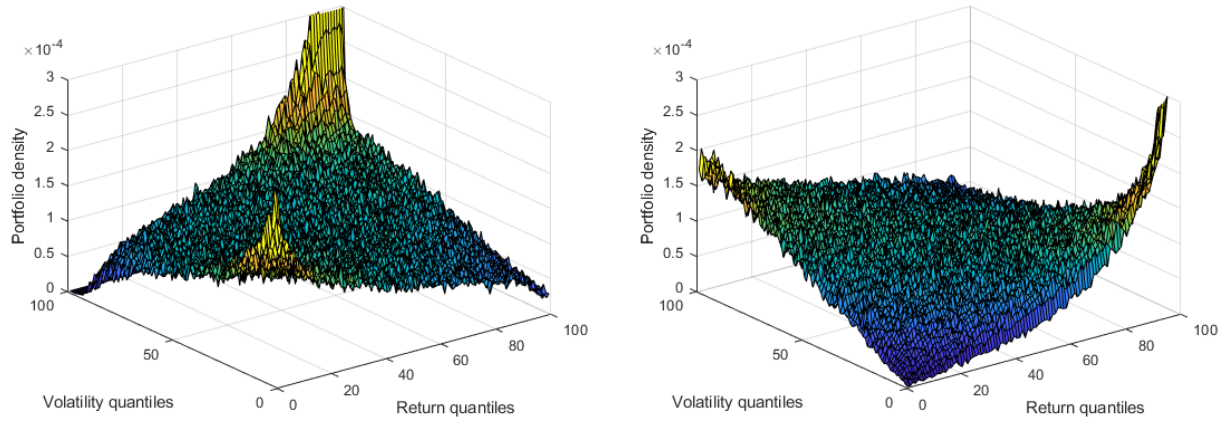

Figure 25: Returns/variance relationship on the 1^{st} September 1999 (left), i.e. during the dot-com bubble, and on the 1^{st} September 2000 (right), at the beginning of the bubble burst. Blue= low density of portfolios, yellow=high density of portfolios.

The main problem is to compute all the volumes that arise from the intersection of the two families with the unit simplex. We totally have to handle three types of full dimensional bodies and thus we develop or use existing methods for three different problems. The first is to compute the volume of the polytope defined by the intersection of the unit simplex with four hyperplanes which are pairwise parallel. The second arises when an ellispoid intersects with the unit simplex and a family of parallel hyperplanes. The third is to compute the volume of a non-convex body defined by the intersection of two concentric ellipsoids with a simplex and a family of parallel hyperplanes.

7.1.1 Market volatility expressed by ellipsoids

In our financial application, portfolios are points in the unit d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ defined as the convex hull of $v_0, \ldots, v_d \in \mathbb{R}^d$, where v_i lies on the i-th axis. The simplex lies in hyperplane $\sum_{i=0}^{d} \lambda_i = 1$. To model levels of volatility, a family of full-dimensional ellipsoids in $\mathbb{R}^{d+1}$, centered at the origin, is defined by the covariance matrix C of asset returns. We wish to compute the volume of intersections of this family with the simplex and, moreover, with a family of hyperplanes on the simplex. Rejection sampling would work in this context, however methods employing random walks require a full-dimensional convex body. Given a full $(d+1)$-dimensional ellipsoid $G : \lambda^T C \lambda - c = 0$ centered at the origin, where $C \in \mathbb{R}^{(d+1)\times(d+1)}$ is symmetric positive-definite, we compute the equation of the ellipsoid defined $G \cap \Delta^d \subset \mathbb{R}^d$, by imposing the constraint $\sum_{i=0}^{d} \lambda_i = 1$ by transform m_{cb} in expression (4.4), thus obtaining:

$$(x - v_0)^T \left(M^{-T} \left[-1 \; I_d \right] C \begin{bmatrix} 1 \\ 0_d \end{bmatrix} M^{-1} \right) (x - v_0) + A(x - v_0) = c',$$

where the expression in parenthesis is the matrix defining the new d-dimensional ellipsoid in Cartesian coordinates, and $A \in \mathbb{R}^{d\times d}, c' \in \mathbb{R}$ are obtained by direct calculation. Similarly

the simplex maps to Cartesian coordinates.

7.2 Moments' computation

Equivalently to the framework we described in 7.1 we can define a simplex representation using barycentric coordinates. Let consider a portfolio x investing in n assets and whose weights are $x = (x_1, \ldots, x_n)$. The portfolios in which a long-only asset manager can invest are subject to $\sum_{i=1}^{n} x_i = 1$ and $x_i \geq 0, \forall i$. Thus the set of portfolios available to this asset manager is the unit $(n-1)-$simplex, denoted Δ^{n-1} and defined as

$$\Delta^{n-1} = \left\{ \sum_{i=1}^{n} x_i v_i \,\middle|\, (x_1, \ldots, x_n) \in \mathbb{R}^n, \sum_{i=1}^{n} x_i = 1 \text{ and } x_i \geq 0, \forall i \in (1, \ldots, n) \right\}.$$

where $v_1, \ldots, v_n \in \mathbb{R}^{n-1}$ are a set of n affinely independent points in some Euclidean space of dimension $n-1$. The vertices $(v_i)_{i=1,\ldots,n}$ represent the n portfolios made of a single asset and the simplex is the convex hull of these vertices.

We define $v_1, \ldots, v_n$ such that:

1. the center of the simplex is set to the origin,
2. the distances of the simplex vertices to its center are equal,
3. the angle subtended by any two vertices through its center is $arccos(\frac{-1}{n-1})$.

The weights $(x_i)_{i=1,\ldots,n}$ of portfolio x are called its barycentric coordinates, whereas in $\mathbb{R}^{n-1}$ they are called its Cartesian coordinates and are denoted $\breve{x} = (\breve{x}_1, \ldots, \breve{x}_{n-1})$.

There are affine maps to pass

- from barycentric to Cartesian coordinates:

$$\begin{array}{rccc} m_{bc}: & \mathbb{R}^n & \mapsto & \mathbb{R}^{n-1} \\ & x & \to & \breve{x} = Tx + v_n \end{array}$$

  where $T = \left[\begin{array}{ccc} v_1 - v_n & \ldots & v_{n-1} - v_n \end{array} \right]$

- from Cartesian to barycentric coordinates:

$$\begin{array}{rccc} m_{cb}: & \mathbb{R}^{n-1} & \mapsto & \mathbb{R}^n \\ & \breve{x} & \to & x = \left[\begin{array}{c} I_{n-1} \\ -1'_{n-1} \end{array} \right] T^{-1}(\breve{x} - v_n) + \left[\begin{array}{c} 0_{n-1} \\ 1 \end{array} \right] \end{array}$$

  where 0_{n-1} and 1_{n-1} are the $n-1$ column vectors of 1's and 0's, respectively, and I_{n-1} is the $n-1$ identity matrix.

7.2.1 Moments of the portfolio returns distribution

We are interested in the distribution of the portfolio returns in the case of observed asset returns $R = (R_1, \ldots, R_n)$.

The return of portfolio x is given by $R'x = A\breve{x} - Av_n + R_n$, where $A = R' \begin{bmatrix} I_{n-1} \\ -1'_{n-1} \end{bmatrix} T^{-1}$.
The moments of the portfolio returns distribution is given by

$$\begin{aligned} M_1 &= \frac{1}{Vol(\Delta^{n-1})} \int\limits_{\Delta^{n-1}} A\breve{x} - Av_n + R_n \, d\breve{x} \\ M_2 &= \frac{1}{Vol(\Delta^{n-1})} \int\limits_{\Delta^{n-1}} (A\breve{x} - Av_n + R_n - M_1)^2 \, d\breve{x} \\ M_k &= \frac{1}{Vol(\Delta^{n-1})(\sqrt{M_2})^k} \int\limits_{\Delta^{n-1}} (A\breve{x} - Av_n + R_n - M_1)^k d\breve{x} \quad , n \geq 3 \end{aligned} \tag{7.1}$$

where the term $\frac{1}{Vol(\Delta^{n-1})}$ is normalizing the equations. Indeed, the distance between the vertices v_i is arbitrary, and so is the volume of Δ^{n-1}. An alternative is to choose the distance between the vertices v_i such that $Vol\,(\Delta^{n-1}) = 1$.

From Lasserre and Avranchenkov in [21], and slightly adapted to our notations, we have

Theorem 7.1. *Let $v_1, \ldots, v_n$ be the vertices of an $(n-1)-$dimensional simplex Δ^{n-1}. Then, for a symmetric q-linear form $H : (\mathbb{R}^{n-1})^q \to \mathbb{R}$, we have*

$$\int\limits_{\Delta^{n-1}} H(X, \ldots, X) \, d\breve{x} = \frac{Vol\,(\Delta^{n-1})}{\begin{pmatrix} n-1+q \\ q \end{pmatrix}} \sum_{1 \leq i_1 \leq i_2 \leq \cdots \leq i_q \leq n} H(v_{i_1}, v_{i_2}, \ldots, v_{i_q})$$

where $Vol\,(\Delta^{n-1}) = \int\limits_{\Delta^{n-1}} 1 \, d\breve{x}$ stands for the volume of the simplex Δ^{n-1}.

7.2.2 Complete homogeneous symmetric polynomials in terms of power sums

In our real data application we are not able to compute moments in 7.1 for $k > 4$ if we just use the Theorem 7.1. The bottleneck is the computation of the complete homogeneous symmetric polynomial (CHSP) in the right side of the equation of Theorem 7.1. In real data applications the direct computation of CHSPs is very complex and we are not able to compute moments for $k > 4$. For the efficient computations of 7.1, for $k \geq 3$, we will need the following identity:

Lemma 7.1. *For $n, k \in$ **N**, it holds*

$$(k!)\sum_{i_1=1}^{n}\sum_{i_2=I_1}^{n}\cdots\sum_{i_k=i_{k-1}}^{n} x_{i_1}x_{i_2}\cdots x_{i_k} = \sum_{m_1+2m_2+\cdots+km_k=k} k!\prod_{j=1}^{k}\frac{1}{m_j!j^{m_j}}\left(\sum_{i=1}^{n}x_i^j\right)^{m_j}$$

where $m_1 \geq 0, m_2 \geq 0, \cdots, m_k \geq 0$.

The right sum of the identity is the sum over all the partitions of k. The $\frac{k!}{\prod_{j=1}^{k} m_j!j^{m_j}}$ is the number of permutations of k elements with the corresponding cyclic type.

- 1^{st} **moment:** M_1

Lemma 7.2. *In a market of n assets, $n \in$ **N**, whose returns are $R = (R_i)_{i=1}^n$, the 1^{st} moment of the portfolios' returns is*

$$M_1 = \frac{1}{n}\sum_{i=1}^{n} R_i$$

Proof. By definition, we have

$$Vol\left(\Delta^{n-1}\right) M_1 = \int\limits_{\Delta^{n-1}} A\breve{x} - Av_n + R_n \, d\breve{x} = \int\limits_{\Delta^{n-1}} A\breve{x} \, d\breve{x} + (-Av_n + R_n)Vol\left(\Delta^{n-1}\right)$$

From Lemma 7.1, and simplifying by $Vol\left(\Delta^{n-1}\right)$, we get

$$M_1 = \left(\frac{1}{n}\sum_{i=1}^{n} Av_i\right) + (-Av_n + R_n) = \frac{1}{n}\sum_{i=1}^{n}(Av_i - Av_n + R_n) = \frac{1}{n}\sum_{i=1}^{n} R_i$$

which concludes the proof.

- 2^{nd} **moment:** M_2

Lemma 7.3. *In a market of n assets, $n \in$ **N**, whose returns are $R = (R_i)_{i=1}^n$, the 2^{nd} moment of the portfolios' returns is*

$$M_2 = \frac{1}{n(n+1)}\sum_{i=1}^{n}(R_i - M_1)^2 = \frac{1}{n+1}Var(R)$$

where Var is the biased sample variance.

Proof. By definition, we have

$$\begin{aligned} Vol\left(\Delta^{n-1}\right) M_2 &= \int_{\Delta^{n-1}} (A\breve{x} - Av_n + R_n - M_1)^2 \, d\breve{x} \\ &= \int_{\Delta^{n-1}} (A\breve{x})^2 \, d\breve{x} + 2(-Av_n + R_n - M_1) \int_{\Delta^{n-1}} A\breve{x} \, d\breve{x} + (-Av_n + R_n - M_1)^2 \end{aligned}$$

From Lemmas 7.1, and simplifying by $Vol\left(\Delta^{n-1}\right)$, we get

$$M_2 = \frac{1}{n(n+1)} \left(\left(\sum_{i=1}^{n} Av_i \right)^2 + \sum_{i=1}^{n} (Av_i)^2 \right) + 2(-Av_n + R_n - M_1) \frac{1}{n} \sum_{i=1}^{n} Av_i + (-Av_n + R_n - M_1)^2$$

And, as $\frac{1}{n} \sum_{i=1}^{n} Av_i = M_1 + Av_n - R_n$, we get

$$\begin{aligned} M_2 &= \frac{1}{n(n+1)} \left(\left(\sum_{i=1}^{n} Av_i \right)^2 + \sum_{i=1}^{n} (Av_i)^2 \right) - \frac{1}{n^2} \left(\sum_{i=1}^{n} Av_i \right)^2 \\ &= -\frac{1}{n^2(n+1)} \left(\left(\sum_{i=1}^{n} Av_i \right)^2 \right) + \frac{1}{n(n+1)} \sum_{i=1}^{n} (Av_i)^2 \\ &= \frac{1}{n+1} \left(\left(\frac{1}{n} \sum_{i=1}^{n} (Av_i)^2 \right) - \left(\frac{1}{n} \sum_{i=1}^{n} Av_i \right)^2 \right) \\ &= \frac{1}{n+1} \left(\frac{1}{n} \sum_{i=1}^{n} \left(Av_i - \frac{1}{n} \sum_{j=1}^{n} Av_j \right)^2 \right) \end{aligned}$$

Moreover,

$$\begin{aligned} M_2 &= \frac{1}{n+1} \left(\frac{1}{n} \sum_{i=1}^{n} \left(Av_i - \frac{1}{n} \sum_{j=1}^{n} Av_j \right)^2 \right) \\ &= \frac{1}{n+1} \left(\frac{1}{n} \sum_{i=1}^{n} \left((Av_i - Av_n + R_n) - \frac{1}{n} \sum_{j=1}^{n} (Av_j - Av_n + R_n) \right)^2 \right) \end{aligned}$$

Thus,

$$M_2 = \frac{1}{n+1}\left(\frac{1}{n}\sum_{i=1}^{n}\left(R_i - \frac{1}{n}\sum_{j=1}^{n} R_j\right)^2\right)$$
$$= \frac{1}{n(n+1)}\sum_{i=1}^{n}(R_i - M_1)^2$$

which concludes the proof.

- k^{tk} **moment:** M_k

For $k \geq 3$ we have,

$$M_k Vol\left(\Delta^{n-1}\right)\left(\sqrt{M_2}\right)^k = \int_{\Delta^{n-1}} (A\breve{x} - \frac{1}{n}\sum_{i=1}^{n} Av_i)^k d\breve{x}$$
$$= \binom{k}{0}\int_{\Delta^{n-1}} (A\breve{x})^k d\breve{x} + \binom{k}{1}\sum_{i=1}^{n} Av_i \int_{\Delta^{n-1}} (A\breve{x})^{k-1} d\breve{x} + \cdots + \binom{k}{k}\left(\sum_{i=1}^{n} Av_i\right)^k \int_{\Delta^{n-1}} 1 d\breve{x}$$

As it holds $\frac{1}{n}\sum_{i=1}^{n} Av_i = M_1 + Av_n - R_n$, and

$$\int_{\Delta^{n-1}} (A\breve{x})^k = \frac{Vol\left(\Delta^{n-1}\right)}{n(n+1)\cdots(n+k-1)}(k!)\sum_{i_1=1}^{n}\sum_{i_2=I_1}^{n}\cdots\sum_{i_k=i_{k-1}}^{n} Av_{i_1}Av_{i_2}\cdots Av_{i_k}$$

For the last computation we could use directly the Lemma 7.1.

Now we can compute ~ 35 moments in real data applications with $d = 600$, e.g. DJSTOXX 600 Europe's data set. The bottleneck of the identity in lemma 7.1 and for the moments' computation as well is the computation of the partitions of k. For the latter we use the algorithm given in [38].

7.3 Experiments with synthetic data

The formula M1 is used in all Tables where exact computation is needed between two parallel hyperplanes intersecting the simplex.

In Tables 4, 5, 6, 7, 8, 9, k is the number of points sampled in the unit simplex, m the number of points in the intersection, p is the percentage of the unit simplex volume of the defined polytope, time is in seconds. In Tables including experiments with `VolEsti` W is the walk length and $N = \frac{1}{\epsilon^2}400d\log d$ is the number of points we sample in each step of

the algorithm. In Table 4, n the number of vertices in the intersection. R/r is the ratio of radii of the smallest enclosing over the largest inscribed ball of the simplex

Table 4 considers the intersection of an arbitrary simplex with two hyperplanes. The vertices of each simplex are randomly chosen uniformly from the surface of a ball with radius 100, using `CGAL` random point generator. All hyperplanes' coefficients are randomly chosen in $[-10, 10]$ with `Boost` (mt19937) random generator. For `VolEsti` we do not use the rounding option for the input polytope. This means that skinny polytopes have low accuracy since the random walk mixes slow, cf. row 10 of Table 4. On the other hand, M2 is not affected by polytope shape. Up to $d = 30$ and for large volume ratio, namely $> 1\%$, M2 yields very accurate and fastest results. The last two experiments show that `VolEsti` achieves the most accurate approximation when the ration of accepted sampled points is small.

In Table 5 we use same runtime for M2 and M4 (analogous numbers of sampled points) and compare their accuracy. We perform two experiments per dimension. For the first, for each dimension we compute the volume between two parallel hyperplanes which is 1% of the simplex volume. For exact volume computation we use (M1). For the second experiment, for each dimension we compute volumes defined by the intersection of 4 hyperplanes which are pairwise parallel with the simplex, which is close to 0.01% of the simplex volume. For exact computation we used `vinci` default method, `rlass`. M2 is fast but inaccurate for small volumes; M4 is most accurate but should not scale beyond $d = 100$.

In Tables 6 and 7 we have an arbitrary simplex and two arbitrary hyperplanes that intersect with it. We compare our Lawrence implementation in Sect. 5.3, using floating-point and rational computation, with `rlass` and M2. We have two parallel hyperplanes intersect the unit simplex. `vinci` fails to compute the volume for $d > 31$. Our exact computation works even in $d = 100$ but becomes very slow.

Table 8 compares M2 (s/r) with two variants of M4 for ellipsoid intersection. The only difference for the latter is the way we construct an inscribed ball: In s/V we implement random sampling until $d + 1$ points are found, and in o/V we use SOCP. We see M2 is significantly faster than either variant of M4. All methods yield similar output values.

Table 9 compares s/r with Hit-and-Run for non-convex bodies. Very small values of Volume means the method failed to approximate the volume.

7.4 Experiments with real data

When we work with real data in order to build the indicator, we wish to compare the densities of portfolios along the two diagonals. In normal and up-market times, the portfolios with the lowest volatility present the lowest returns and the mass of portfolios should be on the up-diagonal. During crisis the portfolios with the lowest volatility present the highest returns and the mass of portfolios should be on the down-diagonal, see Figure 25 as illustration. Thus, setting up- and down-diagonal bands, we define the indicator as the ratio of

the down-diagonal band over the up-diagonal band, discarding the intersection of the two. The construction of the indicator is illustrated in Figure 26 where the indicator is the ratio of the mass of portfolios in the blue area over the mass of portfolios in the red one.

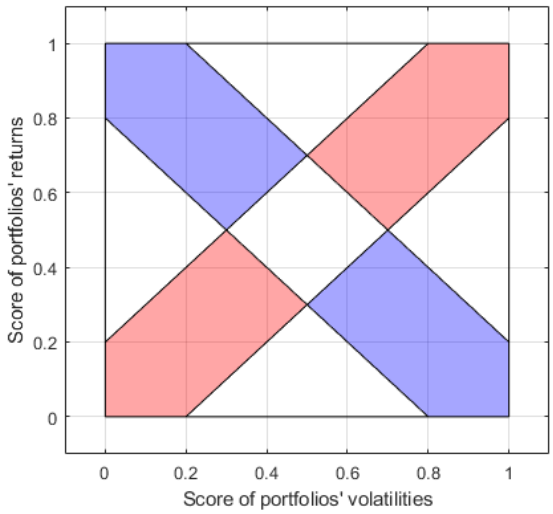

Figure 26: Illustration of the diagonal bands considered to build the indicator.

In the following, the indicator is computed using copulae estimated using the sampling method, drawing 500000 points. Computing the indicator over a rolling window of $k = 60$ days and with a band of $\pm 10\%$ with respect to the diagonal, we report in Table 3 all the periods over which the indicator is greater than 1 for more than 60 days. The periods should be more than 60 days to avoid the detection of isolated events whose persistence is only due to the auto-correlation implied by the rolling window. All these periods offer warnings, but only the longest ones correspond to crises.

We compare these results with the database for financial crises in European countries proposed in [10]. The first crisis (from May 1990 to Dec. 1990) corresponds to the early 90's recession, the second one (from May 2000 to May 2001) to the dot-com bubble burst, the third one (from Oct. 2001 to Apr. 2002) to the stock market downturn of 2002, the fourth one (from Nov. 2005 to Apr. 2006) is not listed and it is either a false signal or it might be due to a bias in the companies selected in the sample, and the fifth one (from Dec. 2007 to Aug. 2008) to the sub-prime crisis.

Regarding the momentum effect, i.e. the effect of the compound returns of the last 60 days on the following 60-day compound returns, we report the indicator in Figure 27. We observe that there were only 10 events of lasting momentum effect, mostly around the 1998-2004 period. We remark that they nearly never overlap with the crisis events, with the exception of the end of 2011. To the authors' knowledge, this result is new in finance. In Figure 28 we see the warning indicator which detects past crises correctly.

Table 3: All periods over which the return/volatility indicator is greater than one for more than 60 days.

Start date	End date	Duration (days)
02-May-1990	20-Dec-1990	166
06-May-1992	14-Aug-1992	72
06-Oct-1994	27-Jan-1995	80
08-Apr-1996	24-Jul-1996	77
01-Jul-1997	13-Oct-1997	74
03-Mar-1999	01-Jun-1999	61
04-May-2000	09-May-2001	258
05-Oct-2001	05-Apr-2002	124
25-Feb-2004	28-May-2004	65
18-Nov-2005	11-Apr-2006	101
20-Dec-2007	04-Aug-2008	157
28-Dec-2010	12-Apr-2011	75
18-Oct-2011	16-Jan-2012	63
08-Oct-2013	04-Feb-2014	82
04-Jun-2015	05-Oct-2015	87
30-Nov-2015	03-Mar-2016	66

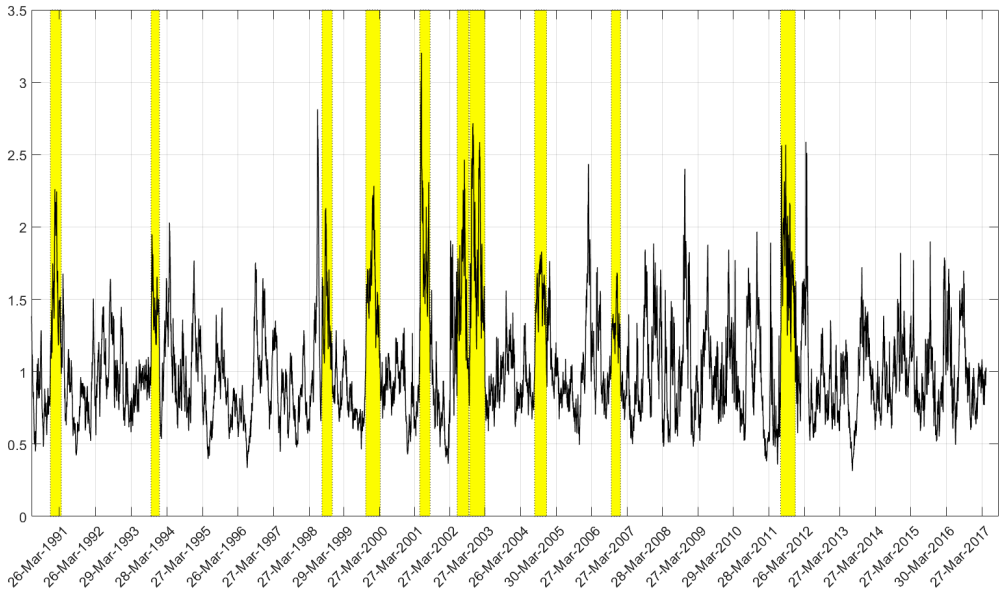

Figure 27: Representation of the periods over which the indicator is greater than one for over 60 days (yellow)

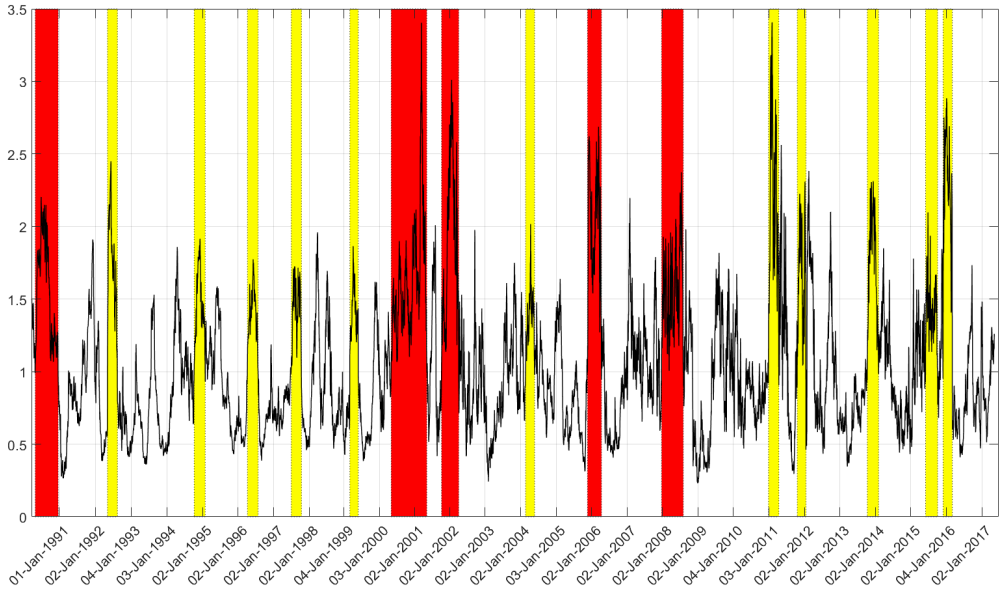

Figure 28: Representation of the periods over which the indicator is greater than one for 61-100 days (yellow) and over 100 days (red)

8. CONCLUSION AND FUTURE WORK

In the current thesis we give some competitive methods for volume computation problems that arise from financial applications. By applying these methods to real data we show that past crises could be detected successfully as we are able to work on higher dimensions that before. The next challenge is to use real data in even higher dimensions and to obtain better results.

Moreover there are some other extensions and improvements we could do. Random sampling follows a Monte Carlo (MC) approach by relying on C/C++ functions such as `random` which implement pseudorandom generators. A potential extension would be to experiment with quasi-MC generators which require fewer points to simulate the uniform distribution. An obvious enhancement is to parallelize our algorithms, which seems straightforward. Then results can be obtained for larger classes of assets such as the entire DJ 600.

Another challenge is to speedup `VolEsti` algorithm for the convex case and to improve it or to work on some different algorithms for the non-convex case. In the next two subsections we provide some more details about future work we are planning on two topics.

8.1 Computing the volume of an ellipsoid intersecting the canonical simplex

Let $A \in \mathbb{R}^{d\times d}$ be a symmetric and positive definite matrix, e.g. a matrix of an ellipsoid centered at the origin. The goal is to approximate the volume defined by the intersection of the ellipsoid $E(A,0) \subset \mathbb{R}^{d+1}$ and the canonical simplex $\Delta^d \subset \mathbb{R}^{d+1}$. Recall that a point in the canonical simplex can be seen as a vector of idepedent exponential random variables. In [32], [31] they give an algorithm for the approximation of the distribution of quadratic forms of exponential and gamma random variables which is a similar problem to that in [30].

When the components of the random vector $\mathbf{X} = (X_1, \ldots, X_n)$ are exponentially distributed with parameter β and density function,

$$f(x) = \frac{1}{\beta} e^{-x/\beta} H(x), \quad \beta > 0,$$

the m^{th} moment of $Q(\mathbf{X}) = Q(X_1, \ldots, X_n) = \mathbf{X}^T A \mathbf{X} = \sum_{i=1}^n \sum_{j=1}^n a_{ij} X_i X_j$, is

$$E(Q(\mathbf{X})^m) = m! \sum_{(m)} \left[\prod_{i,j}^{n} \frac{a_{ij}^{m_{ij}}}{m_{ij}!} \right] \beta^{\sum_{\ell=1}^n \delta_\ell} \prod_{\ell=1}^{n} \Gamma(1+\delta_\ell) \tag{8.1}$$

where $\sum_{(m)}$ denotes the sum over all the partitions of m into n^2 terms such that $m_{11} + m_{12} + \cdots + m_{nn} = m$ with $0 \le m_{ij} \le m$, the m_{ij}'s being nonnegative integers, and $\delta_\ell = \sum_{j=0}^n (m_{\ell j} + m_{j\ell})$. Then we can consider the following generalized gamma density function as a base density for the approximation of $Q(\mathbf{X})$,

$$\psi(x) = \frac{\gamma}{\beta^{a\gamma}\Gamma(a)} z^{a\gamma-1} e^{-(z/\beta)^\gamma} H(x), \quad a > 0, \beta > 0, \gamma > 0 \tag{8.2}$$

The parameters a, β and γ are determined by solving the following nonlinear system,

$$\mu_j = m_j \quad \text{for } j = 1, 2, 3$$

where

$$m_j = \frac{\beta^j \Gamma(a + j/\gamma)}{\Gamma(a)}, \quad j = 0, 1, \ldots$$

are the moments of 8.2 and μ_j can be computed from 8.1. Then a moment-based density approximation of the following form is assumed for Q_1:

$$f_n(x) = \psi(x) \sum_{j=0}^{n} \xi_j x^j \tag{8.3}$$

In order to determine the polynomial coefficients, ξ_j, we have to equating the h^{th} moment of $Q(\mathbf{X})$ denoted by μ_h to the h^{th} moment of the approximate distribution $f_n(x)$ for $h = 1, \ldots, n$,

$$\mu_h = \int_p^q x^h \psi(x) \sum_{j=0}^{n} \xi_j \int_p^q x^{h+j} \psi(x) dx = \sum_{j=0}^{d} \xi_j m_{h+j}, \quad h = 0, 1, \ldots, n$$

where m_{h+j} is the $(h+j)^{th}$ moment determined from $\psi(x)$. This leads to a linear system of $(n+1)$ equations,

$$\begin{bmatrix} m_0 & m_1 & \ldots & m_n \\ m_1 & m_2 & \ldots & m_{n+1} \\ \ldots & \ldots & \ldots & \ldots \\ m_n & m_{n+1} & \ldots & m_{2n} \end{bmatrix} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}$$

The $f_n(x)$ is referred as a n^{th} degree polynomially adjusted density approximant. More details in [32] and [31]. Then approximating the volume is left as a future extension of this work.

8.2 Modeling financial crises using clustering

An alternative approach for modeling financial crises is to compute the copulas and try to divide them into clusters rather than computing the indicator described in 7.1. Then each cluster will correspond to a specific behavior of assets' market, e.g. normal period, crisis period etc. So in order to apply any clustering algorithm we have to define a distance function between copulas. A choice would be to consider the Earth Mover's Distance [36] (EMD) which is a metric between two distributions and is based on the minimal cost that must be paid to transform one distribution into the other.

We constructed the distance matrix using EMD and applied k-medoids algorithm for the clustering. In Figure 29 you can see the representatives of each cluster for $k = 12$ and $k = 15$. Notice that the silhouette score is too small, while the same occurs for $k = 12, \ldots, 20$, which means that we have to use alternative approaches. For example in [17] they give a clustering algorithm which might perform for the copulas'-clustering problem.

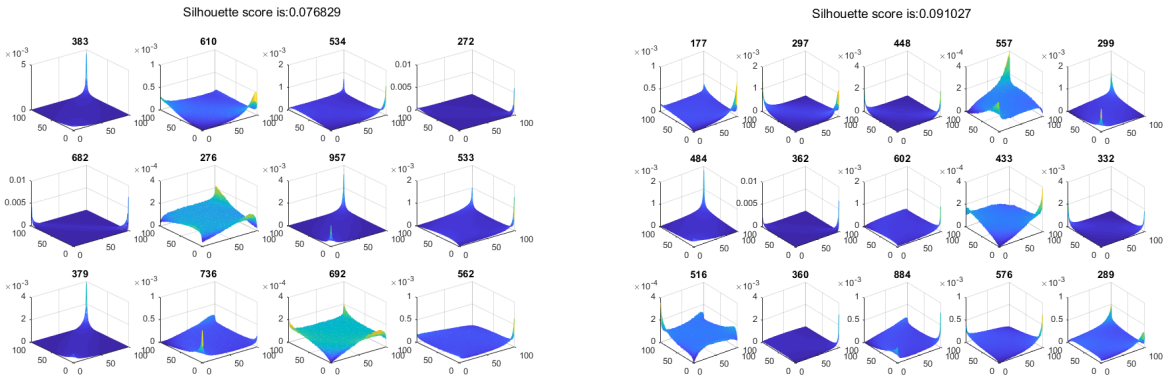

Figure 29: Here we used Earth Mover's Distance and K-medoids for clustering 6616 copulas from DJSTOXX 600 Europe's data set.

APPENDIX A. TABLES OF EXPERIMENTS

Table 4: Experimental results for arbitrary simplex and two arbitrary hyperplanes. We set $k = 10^5 \log d$; Error denotes relative error $(V - v)/V$ of computed value v over exact volume V.

Experimental results for arbitrary simplex and two arbitrary hyperplanes													
d	k (10^5)	m	n	R/r	Simplex Vol.	`Vinci` Vol.	s/r Vol.	s/r Error	`VolEsti` Vol.	`VolEsti` Error	s/r Time	`VolEsti` Time	`Vinci` Time
5	2	1464	4	39.3877	225859	1638.14	1653.29	0.0092	1551.961	0.0526	0.076	1.189	0.0
5	2	8269	5	240.261	31545.7	1287.54	1304.26	0.0130	1104.214	0.1423	0.072	2.474	0.0
10	3	111018	7	31.1786	1.14352e+09	4.22648e+08	4.2317e+08	0.0012	4.399476e+08	0.0409	0.156	8.290	0.0
10	3	16279	9	752.594	2.21485e+07	1.20556e+06	1.20185e+06	0.0031	0.023537e+06	0.9805	0.164	19.980	0.0
15	3	1695	11	112.756	2.87936e+10	1.62617e+08	1.62684e+08	0.0004	1.284843e+08	0.2099	0.204	43.547	0.0
15	3	168639	10	51.9497	1.8289e+11	1.02984e+11	1.02808e+11	0.0017	1.018419e+11	0.0111	0.224	31.848	0.0
20	4	52657	17	50.351	2.47765e+17	3.24630e+16	3.26163e+16	0.0047	3.201464e+16	0.0138	0.416	135.685	0.0
20	4	13952	17	140.094	6.76692e+15	2.38561e+14	2.3603e+14	0.0106	2.334992e+14	0.0212	0.42	181.058	0.0
25	4	4982	23	135.804	1.37457e+18	1.70146e+16	1.71202e+16	0.0062	1.119995e+16	0.3417	0.52	333.052	0.0
25	4	3809	25	89.8112	4.17323e+18	4.03833e+16	3.97396e+16	0.0159	5.017313e+18	123.2	0.508	384.346	0.0
30	4	118304	22	4164.1	1.28638e+17	4.12910e+16	4.10773e+16	0.0052	5.02297e+16	0.2165	0.64	863.056	11.4
30	4	27523	24	177.613	4.08094e+18	2.80038e+17	2.80799e+17	0.0027	1.891857e+17	0.3244	0.616	622.995	7.3
10	3	1151	10	61.3936	2.99231e+08	1.17756e+06	1.14805e+06	0.0251	1.185146e+06	0.0064	0.152	10.367	0.0
18	4	1318	16	57.0641	8.58015e+11	2.96758e+09	2.82716e+09	0.0473	2.908083e+09	0.0200	0.376	93.7450	0.0

Table 5: Experimental results for rejection and `VolEsti`. We set $k = 10^7 \log d$.

Experimental results for rejection and `VolEsti`.											
d	k	m	s/r Vol	s/r time	N	W	ϵ	`VolEsti`	Time `VolEsti`	Exact Vol	Exact Time
15	$3{\cdot}10^7$	300345	7.66e-15	14.716	101551	11	0.4	7.52e-15	20.86	7.65e-15	0.0
15	$3{\cdot}10^7$	744	1.90e-17	14.796	101551	11	0.4	2.15e-17	21.49	2.01e-15	0.0
20	$3{\cdot}10^7$	299842	4.11e-21	23.532	66571	12	0.6	4.44e-21	36.17	4.11e-21	0.0
20	$3{\cdot}10^7$	2040	2.80e-23	23.688	66571	12	0.6	2.72e-23	34.88	2.74e-23	0.1
25	$3{\cdot}10^7$	299976	6.44e-28	30.74	50294	12	0.8	5.81e-28	34.03	6.45e-28	0.0
25	$3{\cdot}10^7$	980	2.10e-30	30.664	65691	12	0.7	2.00e-30	46.03	1.985e-30	0.1
30	$4{\cdot}10^7$	400395	3.77e-35	51.104	50388	13	0.9	3.42e-35	48.71	3.77e-35	0.0
30	$4{\cdot}10^7$	4769	4.49e-37	60.32	63772	13	0.8	4.52e-37	63.91	4.56e-37	3.2

Table 6: Experimental results for Lawrence and rejection methods. We set $\mathbf{k} = \mathbf{10^x} \log \mathbf{d}$, with $\mathbf{x} = \mathbf{max}\{\mathbf{5}, \mathbf{4} + \lceil -\log_{\mathbf{10}}(\mathbf{p}) \rceil\}$.

Experimental results for Lawrence and rejection methods.											
d	k	m	s/r Vol	s/r Time	ex/Law Vol	ex/Law time	fl/Law Vol	fl/Law Time	Vinci Vol	Vinci Time	per. Vol
2	10^5	969	0.0049	0.036	0.005	0.0	0.0005	0.0	0.005	0.0	1%
5	$2{\cdot}10^5$	2034	8.475e-05	0.056	8.33e-05	0.0	8.33e-05	0.0	8.33e-05	0.0	1%
5	$2{\cdot}10^6$	19967	8.320e-05	0.492	8.33e-05	0.0	8.33e-05	0.0	8.33e-05	0.0	1%
10	$3{\cdot}10^5$	2952	2.711e-09	0.136	2.76e-09	0.0	2.76e-09	0.0	2.76e-09	0.0	1%
10	$3{\cdot}10^6$	2986	2.743e-09	1.132	2.76e-10	0.0	2.76e-10	0.0	2.76e-10	0.0	0.1%
15	$3{\cdot}10^5$	2991	7.624e-15	0.156	7.64e-15	0.02	7.64e-15	0.0	7.64e-15	0.0	1%
20	$4{\cdot}10^5$	4096	4.209e-21	0.332	4.11e-21	0.052	4.11e-21	0.0	4.11e-21	0.0	1%
20	$4{\cdot}10^6$	39800	4.09e-21	3.204	4.11e-21	0.052	4.11e-21	0.0	4.11e-21	0.0	1%
20	$4{\cdot}10^6$	3894	4.001e-22	3.14	4.11e-22	0.02	4.11e-22	0.0	4.11e-22	0.0	0.1%
25	$4{\cdot}10^5$	4049	6.526e-28	0.416	6.45e-28	0.076	6.45e-28	0.0	6.45e-28	0.0	1%
25	$4{\cdot}10^6$	39858	6.424e-28	4.108	6.45e-28	0.076	6.45e-28	0.0	6.45e-28	0.0	1%
30	$4{\cdot}10^5$	3986	3.757e-35	0.52	3.77e-35	0.12	2.37e-35	0.0	3.77e-35	0.0	1%
30	$4{\cdot}10^6$	40155	3.785e-35	4.808	3.77e-35	0.12	3.77e-35	0.0	3.77e-35	0.0	1%
30	$4{\cdot}10^6$	3979	3.750e-36	4.96	3.77e-36	0.08	3.77e-35	0.0	3.77e-36	0.0	0.1%
35	$4{\cdot}10^5$	4077	9.864e-43	0.588	9.67e-43	0.184	9.68e-43	0.004	—	–	1%
35	$4{\cdot}10^6$	40155	9.696e-43	5.852	9.67e-43	0.184	6.22e-42	0.0	—	–	1%
40	$5{\cdot}10^5$	4977	1.220e-50	0.864	1.226e-50	0.34	1.06e-50	0.0	—	–	1%
40	$5{\cdot}10^6$	50074	1.227e-50	8.56	1.226e-50	0.34	1.23e-50	0.0	—	–	1%
40	$5{\cdot}10^6$	4923	1.207e-51	8.464	1.226e-51	0.344	-1.38e-49	0.0	—	–	0.1%
50	$5{\cdot}10^5$	5003	3.290e-67	1.088	3.28e-67	1.276	3.29e-67	0.0	—	–	1%
50	$5{\cdot}10^6$	49923	3.283e-67	11.0	3.28e-67	1.276	2.99e-67	0.0	—	–	1%
50	$5{\cdot}10^6$	5011	3.295e-68	11.068	3.28e-68	0.924	3.16e-68	0.0	—	–	0.1%
60	$5{\cdot}10^5$	5093	1.224e-84	1.356	1.20e-84	2.6	3.59e-84	0.0	—	–	1%
60	$5{\cdot}10^6$	50122	1.204e-84	13.5	1.20e-84	2.6	-4.20e-83	0.0	—	–	1%
60	$5{\cdot}10^6$	4897	1.177e-85	13.512	1.20e-84	2172	-2.27e-80	0.0	—	–	0.1%

Table 7: Experimental results for Lawrence and rejection methods. We set k $= 10^{x} \log d$, with x $= \max\{5, 4 + \lceil -\log_{10}(p) \rceil\}$.

Experimental results for Lawrence and rejection methods.										
d	k	m	s/r Vol	s/r Time	ex/Law Vol	ex/Law time	fl/Law Vol	fl/Law Time	Vinci Vol	Vinci Time	per. Vol
70	$6{\cdot}10^5$	6069	8.444e-103	1.988	8.348e-103	5.776	-1.85e-95	0.0	—	–	1%
70	$6{\cdot}10^6$	59911	8.336e-103	19.436	8.348e-103	5.776	-8.78e-97	0.0	—	–	1%
70	$6{\cdot}10^6$	6105	8.494e-104	19.512	8.348e-104	5.048	9.37e-99	0.0	—	–	0.1%
70	10^7	10125	8.453e-104	32.208	8.348e-104	5.776	-1.28e-95	0.0	—	–	0.1%
80	$6{\cdot}10^5$	6059	1.410e-121	2,24	1.397e-121	11.564	2.33e-91	0.0	—	–	1%
80	$6{\cdot}10^6$	59991	1.397e-121	22.576	1.397e-121	11.564	—	–	—	–	1%
80	$6{\cdot}10^6$	5965	1.389e-122	22.424	1.397e-121	11.272	—	–	—	–	0.1%
90	$6{\cdot}10^5$	6045	6.781e-141	2.492	6.73e-141	25.036	—	–	—	–	1%
90	$6{\cdot}10^6$	59873	6.717e-141	24.384	6.73e-141	25.036	—	–	—	–	1%
90	$6{\cdot}10^6$	6083	6.823e-142	24.416	6.73e-142	20.764	—	–	—	–	0.1%
90	10^7	10036	6.755e-142	41.232	6.73e-142	25.3	—	–	—	–	0.1%
100	$6{\cdot}10^5$	6020	1.075e-160	2.696	1.072e-160	41.56	—	–	—	–	1%
100	$6{\cdot}10^6$	60190	1.075e-160	27.096	1.072e-160	41.56	—	–	—	–	1%
100	$6{\cdot}10^6$	6034	1.077e-161	27.472	1.072e-161	37.352	—	–	—	–	0.1%
100	10^7	9979	1.069e-161	45.168	1.072e-161	33.612	—	–	—	–	0.1%

Table 8: Experimental results for the unit simplex and ellipsoid intersection.

Experimental results for the unit simplex and ellipsoid intersection.											
d	k (10^5)	m	s/r Vol.	s/r Time	N	W	ϵ	s/V Vol.	s/V Time	o/V Vol.	o/V Time
3	1	1318	0.0804667	0.004	14648	10	0.3	0.0792319	0.592	0.0798146	0.564
6	1	7668	0.001065	0.004	47780	10	0.3	0.00107003	14.172	0.00105103	13.412
8	1	8798	2.18204e-05	0.012	73935	10	0.3	2.18847e-05	48.672	2.22077e-05	55.324
15	2	19827	7.58102e-13	0.02	64993	11	0.5	7.68531e-13	96.888	7.46954e-13	105.06
20	3	29951	4.1036e-19	0.036	66571	12	0.5	3.93709e-19	178.54	3.97954e-19	170.476
25	3	39987	6.44486e-26	0.056	89413	12	0.6	6.4879e-26	457.196	6.51637e-26	442.68
30	3	39987	3.76754e-33	0.056	63772	14	0.8	3.92896e-33	311.772	3.56866e-33	311.872
40	4	39974	1.22559e-48	0.096	40987	14	1.2	1.21068e-48	253.38	1.30804e-48	242.172
40	4	39999	1.22559e-48	0.096	59022	14	1.0	1.28713e-48	436.976	1.26529e-48	448.472

Table 9: Experimental results for non convex bodies. We set k $= 5 \cdot 10^5$.

Experimental results for non convex bodies.									
d	k	m	s/r Vol.	s/r Time	N	W	ϵ	s/V Vol.	s/V Time
3	$5{\cdot}10^5$	6384	0.00213	0.172	14648	10	0.3	0.00174	2.028
6	$5{\cdot}10^5$	43210	0.000120	0.22	47780	10	0.3	0.000120	22.036
8	$5{\cdot}10^5$	72915	3.616e-06	0.26	73935	10	0.3	3.633e-06	114.596
15	$5{\cdot}10^5$	38012	5.814e-14	0.448	64993	11	0.5	5.834e-14	139.908
15	$5{\cdot}10^5$	41824	6.044e-14	0.476	64993	12	0.5	8.109e-14	240.74
20	$5{\cdot}10^5$	31824	2.616e-20	0.644	95863	12	0.5	2.642e-20	1016.15
20	$5{\cdot}10^5$	36273	2.981e-20	0.620	66571	12	0.6	2.895e-20	323.536
25	$5{\cdot}10^5$	27650	3.565e-27	0.86	89413	12	0.6	3.787e-27	999.352
25	$5{\cdot}10^5$	27055	3.488e-27	0.82	65691	12	0.7	3.301e-27	586.496
30	$5{\cdot}10^5$	26451	1.994e-34	1.032	83294	13	0.7	2.171e-34	1051.19
30	$5{\cdot}10^5$	26265	1.980e-34	1.072	83294	13	0.7	2.179e-34	1005.43
35	$5{\cdot}10^5$	2158	4.176e-43	1.196	49774	14	1.0	2.904e-44	630.908
35	$5{\cdot}10^5$	1115	2.158e-43	1.348	61450	13	0.9	1.198e-166	1417.01
35	$5{\cdot}10^5$	10160	1.966e-42	1.292	61450	13	0.9	1.061e-42	810.248
40	$5{\cdot}10^5$	8753	1.22559e-48	1.36	72866	13	0.9	2.087e-192	1873.56

REFERENCES

[1] Y. Abbasi-Yadkori, P.L. Bartlett, V. Gabillon, and A. Malek. Hit-and-run for sampling and planning in non-convex spaces. In *Proc. 20th Intern. Conf. Artificial Intelligence & Stat. (AISTATS)*, pages 888–895, April 2017.

[2] M. Maswood Ali. Content of the frustum oa a simplex. *Pacific J. Math.*, 48(2):313–322, 1973.

[3] M. Billio, M. Getmansky, and L. Pelizzon. Dynamic risk exposures in hedge funds. *Comput. Stat. & Data Analysis*, 56(11):3517–3532, 2012.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: A practical study. In G. Kalai and G.M. Ziegler, editors, *Polytopes: Combinatorics and Computation*, volume 29 of *Math. & Statistics*, pages 131–154. Birkhäuser, Basel, 2000.

[6] L. Calès, A. Chalkis, I.Z. Emiris, and V. Fisikopoulos. Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises. *CoRR*, arXiv:1803.05861, March 2018. To appear in Proc. 34th Int. Sympos. on Comput. Geometry, Budapest June 2018.

[7] K. Chandrasekaran, D. Dadush, and S. Vempala. Thin partitions: Isoperimetric inequalities and sampling algorithms for some nonconvex families. *CoRR*, abs/0904.0583, 2009.

[8] B. Cousins and S. Vempala. A cubic algorithm for computing Gaussian volume. In *Proc. Symp. on Discrete Algorithms*, pages 1215–1228. SIAM/ACM, 2014.

[9] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *Proc. European Control Conference (ECC)*, pages 3071–3076, 2013.

[10] M. Lo Duca, A. Koban, M. Basten, E. Bengtsson, B. Klaus, P. Kusmierczyk, J.H. Lang, C. Detken, and T. Peltonen. A new database for financial crises in european countries. Technical Report 13, European Central Bank and European Systemic Risk Board, Frankfurt am Main, Germany, 2017.

[11] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[12] M.E. Dyer and A.M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.

[13] G. Elekes. A geometric inequality and the complexity of computing volume. *Discrete & Computational Geometry*, 1:289–292, 1986.

[14] I.Z. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. Math. Soft.*, 2017. To appear. Prelim. version: Proc. Sympos. on Comput. Geom., 2014.

[15] E. Erdougan and G. Iyengar. An active set method for single-cone second-order cone programs. *SIAM J. Optimization*, 17(2):459–484, February 2006.

[16] C. Grimme. Picking a uniformly random point from an arbitrary simplex. Technical report, Information Systems and Statistics, Munster U., Germany, 2015.

[17] A. Iscen, Y. Avrithis, G. Tolias, T. Furon, and O. Chum. Fast spectral ranking for similarity search. *CoRR*, arXiv:1703.06935v3, March 2017.

[18] N. Jegadeesh and S. Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *J. Finance*, 48:65–91, 1993.

[19] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $o*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.

[20] R. Kannan and H. Narayanan. Random walks on polytopes and an affine interior point method for linear programming. *In STOC*, pages 561–570, 2009.

[21] J.B. Lasserre and K.E. Avranchenkov. The multi-dimensional version of $\int_a^b x^p dx$. *The American Mathematical Monthly*, 108, No. 2, 2001.

[22] Jean-Bernard Lasserre. Volume of slices and sections of the simplex in closed form. *Optimization Letters, Springer Verlag, 2015, 9 (7)*, pages 1263–1269.

[23] J. Lawrence. Polytope volume computation. *Math. of Computation*, 57(195):259–271, 1991.

[24] O. Ledoit and M. Wolf. Honey, I shrunk the sample covariance matrix. *J. Portfolio Management*, 30(4):110–119, 2004.

[25] Y.T. Lee and S.S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. *CoRR*, abs/1710.06261, 2017.

[26] Y.T. Lee and S.S. Vempala. Geodesic walks in polytopes. In *Proc. ACM Symp. on Theory of Computing*, pages 927–940. ACM, 2017.

[27] L. Lovász. Hit-and-run mixes fast. *Math. Programming*, 86:443–461, 1999.

[28] L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM J. Computing*, 35:985–1005, 2006.

[29] H. Markowitz. Portfolio selection. *J. Finance*, 7(1):77–91, 1952.

[30] A.M. Mathai. On linear combinations of independent exponential variables. *Communications in Statistics: Theory & Methods*, 2007.

[31] A. Mohsenipour and S. Provost. On approximating the distributions of ratios and differences of non-central quadratic forms in normal vectors. 2010.

[32] A. Mohsenipour and S. Provost. On approximating the distribution of quadratic forms in gamma random variables and exponential order statistics. *Journal of Statistical Theory and Applications*, 12(2):173–184, 2013.

[33] A.B. Owen. *Monte Carlo theory, methods and examples*. Copyright A. Owen, 2009.

[34] R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method*. Wiley Interscience, New York, 2007.

[35] R.Y. Rubinstein and B. Melamed. *Modern simulation and modeling*. Wiley, New York, 1998.

[36] Y. Rubner, C. Tomasi, and J. Guibas. The earth mover's distance as a metric for image retrieval.

[37] N.A. Smith and R.W. Tromble. Sampling uniformly from the unit simplex. Technical report, Center for Language and Speech Processing, Johns Hopkins U., 2004.

[38] I. Stojmenović and A. Zoghbi. Fast algorithms for generating integer partitions. *International Journal of Computer Mathematics*, 70(2):319-332, 1998.

[39] G. Varsi. The multidimensional content of the frustum of the simplex. *Pacific J. Math.*, 46:303–314, 1973.

[40] Santosh Vempala. Geometric random walks: A survey, p. 31. 2005.