Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises*

Ludovic Calès[1], Apostolos Chalkis[2], Ioannis Z. Emiris[3], and Vissarion Fisikopoulos[4]

1 **European Commission, Joint Research Centre, Ispra, Italy**
2 **Department of Informatics & Telecommunications National & Kapodistrian University of Athens, Greece**
3 **Department of Informatics & Telecommunications National & Kapodistrian University of Athens, Greece**
4 **Oracle Corp.**

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. The impact of crises on the economy (labor, income, etc.) makes its detection of prime interest for the public in general and for policy makers in particular. Certain features of dependencies in the markets clearly identify times of turmoil. We describe the relationship between asset characteristics by means of a copula; each characteristic is either a linear or quadratic form of the portfolio components, hence the copula can be constructed by computing volumes of convex bodies.

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; the latter is experimentally extended to non-convex bodies with very encouraging results. Our C++ software, based on CGAL and Eigen and available on `github`, is shown to be very effective in up to 100 dimensions. Our results offer novel, effective means of computing portfolio dependencies and an indicator of financial crises, which is shown to correctly identify past crises.

1998 ACM Subject Classification F.2.2[Analysis of algorithms and problem complexity]: Geometrical problems and computations

Keywords and phrases Polytope volume, convex body, simplex, sampling, financial portfolio

Digital Object Identifier 10.4230/LIPIcs...

* The views expressed are those of the authors and do not necessarily reflect official positions of the European Commission.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Financial markets exhibit 3 types of behavior, see [BGP12]. 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. In accordance with [Mar52], in normal and up-market times, the stocks and portfolios with the lowest volatility present the lowest returns, whereas during crises those with the lowest volatility present the highest returns. These features[1] motivate us to describe the time-varying dependency between portfolios' returns and volatility. The methods introduced here also allow us to describe another market feature, namely the momentum effect [JT93] implied by the dependencies of asset returns with their past returns.

These dependencies are important because

- through the return/volatility dependency, the detection of crisis arises policy makers 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, these descriptions can be given over a single period of time making the information available very early. While this is particularly true for the momentum for which daily data is available, the volatility still has to be estimated over a sufficiently large period of time to be reliable thus delaying the detection of crisis.[2]

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

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

which corresponds to the share of portfolios with a return $\leq R^* = R^T\omega^*$. More generally, this score corresponds to the cumulative distribution function of $f_{ac}(\omega, C)$ where random portfolios are drawn from the simplex. In the following, we consider the cases where f_{ac} is a linear combination or a quadratic form of C. Finally, we describe the relationship between two asset characteristics ac_1 and ac_2 by means of a *copula*, a multivariate probability distribution, whose marginals are ρ_{ac_1}, ρ_{ac_2}. In our applications, the asset characteristics considered are the assets' returns and volatilities. The portfolio quantities will be the same (returns and volatilities), the first being a linear combination of the returns and the last one a quadratic form of the latter.

[1] also called "stylized facts" in the financial literature

[2] More timely estimates could be obtained through the range-based estimation method [BGP12], but requires high-frequency data and is not widely used. The method is beyond the scope of this paper.

These questions are formulated in terms of convex bodies defined by intersecting simplices 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.

1.1 Previous work

The cross-sectional score of portfolio returns has been introduced in [Pou05] and it is estimated by means of a quasi-Monte Carlo method. The applications have been limited in terms of dimensions: the 30 DAX components and the 24 MSCI Netherlands components in [Pou05], the 35 components of the IBEX in [PST04]. This score has also been proposed in [BCG11], considering the set of long/short equally weighted zero-dollar portfolios, whose estimation relied on combinatorics and statistics, and computationally limited to around 20 dimensions, and in [BH11] which was not interested in a precise score.

Given that volume computation of polytopes is #P-hard for both V- and H-representations [DF88] and no poly-time algorithm can achieve better than exponential error [Ele86], 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 [BEF00]. In the approximate setting, following the breakthrough polynomial-time algorithm by random walks [DFK91], several algorithmic improvements ensued. The current best theoretical bounds are in [LV17a] and for polytope sampling in [LV17b]. 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 [EF17], based on the Hit-and-Run paradigm [Lov99], and the Matlab implementation of [CV14], which treats hyperplanes and 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 [CDV09] or, more recently, in [ABGM17].

1.2 Our contribution

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. We thus provide a new description of asset characteristics dependencies. Our methods allow us to propose and to effectively compute a new indicator of financial crises, which is shown to correctly identify all past crises with which we experimented. More importantly, it allows us to establish that periods of momentum nearly never overlap with the crisis events, which is a new result in finance.

The rest of the paper is organized as follows. The next section overviews methods for representing and uniform sampling of simplices. Sect. 3 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. Sect. 4 studies convex bodies defined as the intersection of a simplex and an ellipsoid, for which random walk methods are developed. The implementations are discussed in Sect. 5, along with experiments that show the validity of our approach in answering the open questions in finance. We conclude with current work and open questions. Figures and tables that do not fit are given in the Appendix.

2 Simplex representation and sampling

This section sets the notation, surveys methods for uniform sampling of the simplex, and discusses their efficient implementation.

The d-dimensional 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. It is convenient to use a full-dimensional simplex, by switching to Cartesian coordinates $x = (x_1, \ldots, x_d)$:

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

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

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

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.

A number of algorithms exist for sampling, where some have been rediscovered, while others contain errors; see the survey [ST04]. Let us start with a unit simplex in Cartesian coordinates. A $O(d \log d)$ algorithm is the following [Dav81, Dev86, RK07]: Generate d distinct integers uniformly in $\{1, \ldots, M-1\}$, where M is the largest representable integer. Sort them as follows: $x_0 = 0 < x_1 < \cdots < x_{d+1} = M$. Now $(x_i - x_{i-1})/M$, $i = 1, \ldots, d$, defines a uniform point. Assuming we possess a perfect hash-function, the choice of distinct integers takes $O(d)$. We implement a variant of Bloom filter to guarantee distinctness.

A linear-time algorithm is given in [RM98], which is is generally the algorithm of choice, although it is slower for $d < 20$:

1. Generate $d+1$ independent unit-exponential random variables y_i by uniformly sampling real value $x_i \in (0, 1)$ and setting $y_i = -\log x_i$.
2. Normalize the y_i's by their sum $s = \sum_{i=0}^d y_i$, thus obtaining a uniformly distributed point $(y_0/s, \ldots, y_d/s)$ on the d-dimensional canonical simplex lying in $\mathbb{R}^{d+1}$.
3. Project this point along the x_0-axis to $(y_1/s, \ldots, y_d/s)$, which is a uniform point in the full-dimensional unit simplex.

To sample an arbitrary simplex, we can map sampled points from the unit simplex by transformation (1), which preserves uniformity. Due to applying the transformation, the complexity is $O(d^2)$ to generate a uniform point. The same complexity, though slower in practice, is achieved in [Gri15].

2.1 Rejection sampling

One can sample the simplex and count the percentage of points in the region of interest. The complexity is $O(Nd)$ to generate N points. In the case of a family of k parallel hyperplanes, all sample points are all evaluated at the hyperplane polynomials in time $O(Nd)$. Given the k constant terms characterizing the hyperplanes, for each point we perform a binary search so as to decide in which layer it lies. Hence the total complexity is $O(N \log k)$, which is dominated since $k \leq 100$ typically. Given a family of k ellipsoids with same quadratic form intersecting a simplex, the method requires again $O(Nd)$ to evaluate all sample points and $O(N \log k)$ to assign them to layers.

3 Intersection with hyperplanes

This section considers computing the volume of the intersection of a simplex and one or more linear halfspaces. The most general case is to be given two families of parallel hyperplanes and consider all created polytopes. We assume that the simplex is given in V-representation, i.e. as a set of vertices, and the hyperplanes by their equations.

We can always transform the simplex to be a unit full-dimensional simplex with the origin as one vertex by the transformation of Sect. 2. The same transform applies to the hyperplanes, and volume ratios as preserved.

3.1 Single halfspace formula

Surprisingly, there exist an exact, iterative formula for the volume defined by intersecting a simplex with a hyperplane. A geometric proof is given in [Var73], by subdividing the polytope into pyramids and, recursively, to simplices. We implement a somewhat simpler formula [Ali73], which also requires $O(d^2)$ operations. 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 $h = J$, $A_K = \mathrm{vol}(\Delta^d \cap H)/\mathrm{vol}(\Delta^d)$.

Recall, from Section 2, 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 [Mat07], 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 (see Sect. 6).

3.2 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 [Law91] 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

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

The computational complexity is $O(d^3 n)$. 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 [Law91], 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,

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

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 1. 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 $O(d^3)$.

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 $i \neq k, i \neq m, j \neq kj \neq m$ 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 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 2. 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.

4 Intersection with ellipsoids

This section 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 in `VolEsti` [EF17] so as to handle nonlinear constraints. Our primary motivation here is computing the volume of the intersection of a simplex with an ellipsoid in general dimension.

4.1 Random walks

The method in [EF17] follows the Hit-and-Run algorithm in [Lov99], 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. We have to generalize the method because the input is not a polytope but a general convex body, while `VolEsti`works for d-polytopes. It suffices to solve two subproblems: Compute the maximum inscribed ball of the convex body a.k.a. Chebychev ball, and 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 subsection. 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.

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 `VolEstion` 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. Given these first positive results, various improvements are planned.

4.2 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.

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.

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.

For a convex body that comes from intersecting a polytope with k balls the problem becomes a Second-Order Cone Program (SOCP) 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':

$$\max \ r, \quad \text{subject to} : a_i^T x_c + r||a_i|| \leq b_i, \ ||x'_c - x_c|| \leq r' - r.$$

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 [EI06]. However, for our case it suffices to use the generic SOCP solver from [DCB13] as it is very efficient; for a random simplex and 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 a 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.

4.3 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 (1), 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.

5 Implementation and experiments

Our implementations are in C++, lie in the public domain[3], and are using CGAL and Eigen. All experiments of the paper have performed on a personal computer with Intel Pentium

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

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).

In general, the exact formula (M1) is preferred when available. Rejection sampling (M2, or s/r) 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 Hit-and-Run (M4). The latter is thus the method of choice for volumes $< 1\%$ of the simplex volume, but it is not clear whether it shall be fast beyond $d = 100$. Lawrence decomposition (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.

5.1 Synthetic data

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

Table 1 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 1. On the other hand, M2 is not affected by polytope shape. Up to $d = 30$ and for large volume ratios, 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 2 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 Table 3 we have an arbitrary simplex and two arbitrary hyperplanes that intersect with it. We compare our Lawrence implementation in Sect. 3.2, 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 4 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 5 compares s/r with Hit-and-Run for non-convex bodies, as in Sect. 4. Very small values of Volume means the method failed to approximate the volume.

5.2 Real data

We analyze real data consisting of regular interval (e.g. daily) returns corresponding to indices such as the Dow Jones Europe Stoxx 600™(DJ600). These are points in real space of dimension $d = 600$, respectively: $r_i = (r_{i,1}, \ldots, r_{i,d}) \in \mathbb{R}^d$, $i \geq 1$.

We apply the methodology to assets drawn among the DJ 600 constituents[4]. Since not all assets are tracked for the full period of time, we select the 100 assets with the longest history in the index[5], and juxtapose:

- returns and volatility over the same period to detect crises,
- returns and past returns to observe any momentum effect,

In financial applications, one considers compound returns over periods of k observations, where typically $k = 30$ 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 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.

Volatility is modeled using the shrinkage estimator of the covariance matrix[6] [LW04], 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.

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 formula (M1). For ellipses $E(x) = c_i$, 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.

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. 1 illustrates such copulae, and shows the different relationship between returns and volatility in good (left, dot-com bubbled) and bad (right, bubble burst) times.[7]

5.3 Financial modeling

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 Fig. 1 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,

[4] The data used is daily and ranges from 01/01/1990 to 31/11/2017. It is from Bloomberg™.
[5] This implies a survivor bias, but we use it to assess the effectiveness of the methodology. One would wish to keep 600 constituents, by replacing exiting by incoming stocks along the sample.
[6] Matlab code on `http://www.econ.uzh.ch/en/people/faculty/wolf/publications.html`.
[7] We consider 100 components of DJ 600 with longest history, over 60 days ending at the given date.

discarding the intersection of the two. The construction of the indicator is illustrated in Fig. 2 where the indicator is the ratio of the mass of portfolios in the blue area over the mass of portfolios in the red one.

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 6 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 [**?**]. 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 Fig 4. 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.

6 Conclusion and Future work

Since runtimes are very reasonable, we plan to extend our study to larger subsets of assets of DJ 600 and eventually the whole index in $d = 600$. Another extension is to consider polytopes defined by intersections of both families of parallel hyperplanes and the family of ellipsoids, thus creating 3-D diagrams of dependencies, which have never been studied in finance: one difficulty is to model the outcome since visualization becomes intricate.

Random sampling follows a Monte Carlo (MC) approach by relying on C/C++ functions such as `random` which implement pseudorandom generators. We are experimenting with quasi-MC generators which require fewer points to simulate the uniform distribution. Our preliminary experiments indicate that this may yield a speedup of about 2. 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.

One challenge is to extend the volume formula to the intersection with a ellipsoid. In [MP13], they propose a method to approximate the distribution f of quadratic forms in gamma random variables which is a similar problem to that in [Mat07] (see Sect. 3). It consists in fitting f with a generalized gamma distribution by matching its first 3 moments with those of f and to adjust the distribution with a polynomial in order to fit the higher moments. To get an approximation with a polynomial of degree k, the method requires the first $2k$ moments. In the case of a quadratic form in d random variables, the moment of order m is obtained by a sum over all the partitions of m into d^2 terms. The number of partitions makes the computation of moments challenging even for $d \geq 5$.

References

ABGM17 Yasin Abbasi-Yadkori, Peter L. Bartlett, Victor Gabillon, and Alan Malek. Hit-and-run for sampling and planning in non-convex spaces. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 888–895, 2017.

Ali73 M. Maswood Ali. Content of the frustum oa a simplex. *Pacific J. Math.*, 48(2):313–322, 1973.

BCG11 M. Billio, L. Calès, and D. Guéguan. A cross-sectional score for the relative performance of an allocation. *Intern. Review Appl. Financial Issues & Economics*, 3(4):700–710, 2011.

BEF00 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.

BGP12 M. Billio, M. Getmansky, and L. Pelizzon. Dynamic risk exposures in hedge funds. *Comput. Stat. & Data Analysis*, 56(11):3517–3532, 2012.

BH11 A. Banerjee and C-H. Hung. Informed momentum trading versus uninformed “naive" investors strategies. *J. Banking & Finance*, 35(11):3077–3089, 2011.

CDG95 S. Claessens, S. Dasgupta, and J. Glen. The cross-section of stock returns: Evidence from emerging markets. Technical Report 1505, World Bank Policy Research Working Paper, 1995.

CDV09 Karthekeyan Chandrasekaran, Daniel Dadush, and Santosh Vempala. Thin partitions: Isoperimetric inequalities and sampling algorithms for some nonconvex families. *CoRR*, abs/0904.0583, 2009.

CV14 B. Cousins and S. Vempala. A cubic algorithm for computing Gaussian volume. In *Proc. Symp. on Discrete Algorithms*, pages 1215–1228. SIAM/ACM, 2014.

Dav81 H.A. David. *Order statistics*. Wiley, New York, 2 edition, 1981.

DCB13 A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *Proc. European Control Conference (ECC)*, pages 3071–3076, 2013.

Dev86 L. Devroye. *Non-uniform Random Variate Generation*. Springer-Verlag, Berlin, 1986.

DF88 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.

DFK91 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.

DSP10 I. Dabbene, P.S. Shcherbakov, and B.T. Polyak. A randomized cutting plane method with probabilistic geometric convergence. *SIAM J. Optimization*, 20(6):3185–3207, 2010.

EF17 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.

EI06 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.

Ele86 G. Elekes. A geometric inequality and the complexity of computing volume. *Discrete & Computational Geometry*, 1:289–292, 1986.

FF92 E. Fama and K. French. The cross-section of expected stock returns. *J. Finance*, 47(2):427–465, 1992.

GMW05 Golubitsky, V. Mazalov, and S.M. Watt. An algorithm to compute the distance from a point to a simplex. Technical report, ORCCA, London, Canada, 2015.

Gri15 C. Grimme. Picking a uniformly random point from an arbitrary simplex. Technical report, Information Systems and Statistics, Munster U., Germany, 2015.

JT93 N. Jegadeesh and S. Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *J. Finance*, 48:65–91, 1993.

Law91 J. Lawrence. Polytope volume computation. *Math. of Computation*, 57(195):259–271, 1991.

Lov99 L. Lovász. Hit-and-run mixes fast. *Math. Programming*, 86:443–461, 1999.

LV17a Y.T. Lee and S.S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. *CoRR*, abs/1710.06261, 2017.

LV17b Y.T. Lee and S.S. Vempala. Geodesic walks in polytopes. In *Proc. ACM Symp. on Theory of Computing*, pages 927–940. ACM, 2017.

LW04 O. Ledoit and M. Wolf. Honey, I shrunk the sample covariance matrix. *J. Portfolio Management*, 30(4):110–119, 2004.

Mar52 H. Markowitz. Portfolio selection. *J. Finance*, 7(1):77–91, 1952.

Mat07 A.M. Mathai. On linear combinations of independent exponential variables. *Communications in Statistics: Theory & Methods*, 2007.

MP13 A.A. Mohsenipour and S.B. Provost. On approximating the distribution of quadratic forms in gamma random variables and exponential order statistics. *J. Statistical Theory & Appl.*, 12(2):173–184, 2013.

Pou05 I. Pouchkarev. *Performance evaluation of constrained portfolios*. PhD thesis, Erasmus Research Institute of Management, The Netherlands, 2005.

PST04 I Pouchkarev, J. Spronk, and J Trinidad. Dynamics of the spanish stock market through a broadband view of the IBEX 35 index. *Estudios Econom. Aplicada*, 22(1):7–21, 2004.

RK07 R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method*. Wiley Interscience, New York, 2007.

RM98 R.Y. Rubinstein and B. Melamed. *Modern simulation and modeling*. Wiley, New York, 1998.

ST04 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.

Var73 G. Varsi. The multidimensional content of the frustum of the simplex. *Pacific J. Math.*, 46:303–314, 1973.

A Experiments

Experimental results for arbitrary simplex and two arbitrary hyperplanes													
d	k	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	$2\cdot10^5$	1464	4	39.3877	225859	1638.14	1653.29	0.0092	1551.961	0.0526	0.076	1.189	0.0
5	$2\cdot10^5$	8269	5	240.261	31545.7	1287.54	1304.26	0.0130	1104.214	0.1423	0.072	2.474	0.0
10	$3\cdot10^5$	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	$3\cdot10^5$	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	$3\cdot10^5$	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	$3\cdot10^5$	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	$4\cdot10^5$	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	$4\cdot10^5$	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	$4\cdot10^5$	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	$4\cdot10^5$	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	$4\cdot10^5$	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	$4\cdot10^5$	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	$3\cdot10^5$	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	$4\cdot10^5$	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 1 k is the number of points sampled in the unit simplex, $k = 10^5 \log d$, m the number of points in the intersection, 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; s/r is sampling with rejection; Error denotes relative error $(V - v)/V$ of computed value v over exact volume V.

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 2 k is the number of points sample from the unit simplex, $k = 10^7 \log d$, m is the number of points in the intersection; s/r is sampling with rejection (M2).

B Applications

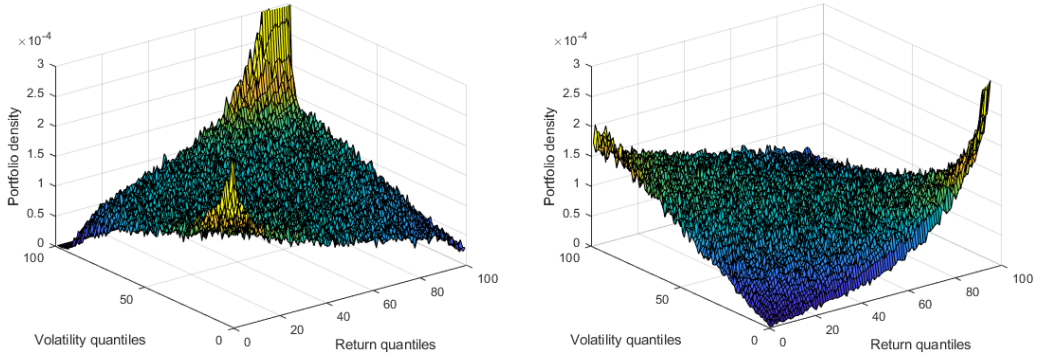

Figure 1 Returns/variance relationship on the 1^{st} September 1999 (left), i.e. during the dot-com bubble, and on the 1^{st} September 2000 (right), at the beginning of the bubble burst. Blue= low density of portfolios, yellow=high density of portfolios.

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%
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	$\cdot 10^7$	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	$\cdot 10^7$	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	$\cdot 10^7$	9979	1.069e-161	45.168	1.072e-161	33.612	—	–	—	–	0.1%

Table 3 k is the number of points sample from the unit simplex, $k = 10^x \log d$, with $x = max\{5, 4 + \lceil -\log_{10}(p) \rceil\}$, where p is the percentage of the unit simplex volume of the defined polytope, m is the number of points in the intersection and last column is p.

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

Table 4 k is the number of points sampled in the simplex, of which m lie in the intersection; N is the number of points generated by `VolEsti` per step, W is the walk length; $N = \frac{1}{\epsilon^2} 400 d \log d$.

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

Table 5 k is the number of points sampled in the simplex, set to constant $k = 5 \cdot 10^5$, of which m lie in the intersection; N is the number of points generated by `VolEsti` per step, W is the walk length. We set $N = \frac{1}{\epsilon^2} 400 d \log d$.

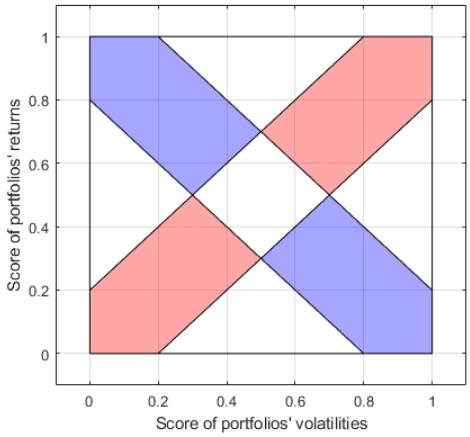

Figure 2 Illustration of the diagonal bands considered to build the indicator.

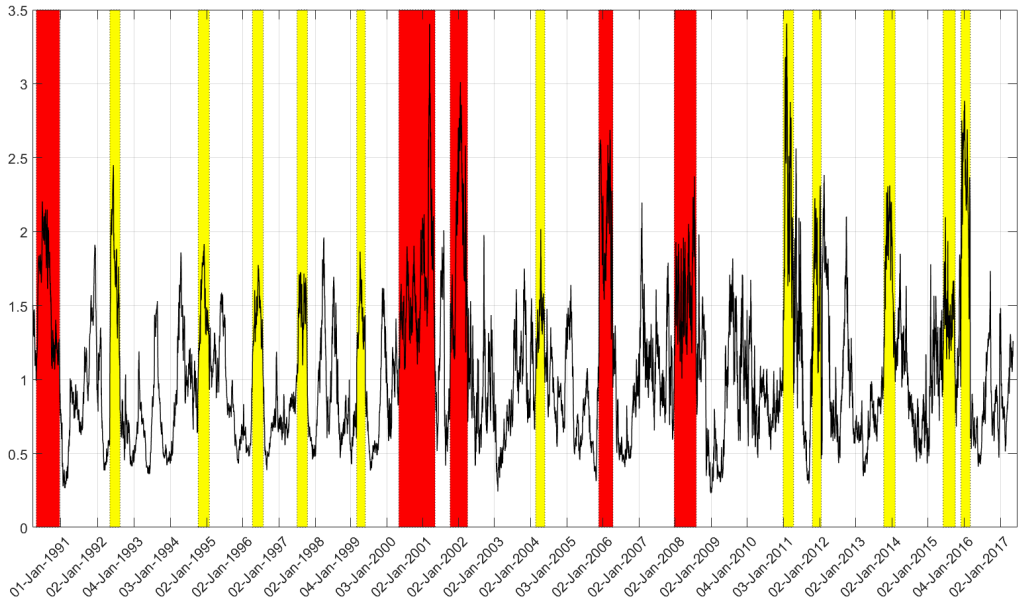

Figure 3 Representation of the periods over which the indicator is greater than one for 61-100 days (yellow) and over 100 days (red)

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

Table 6 All periods over which the return/volatility indicator is greater than one for more than 60 days.

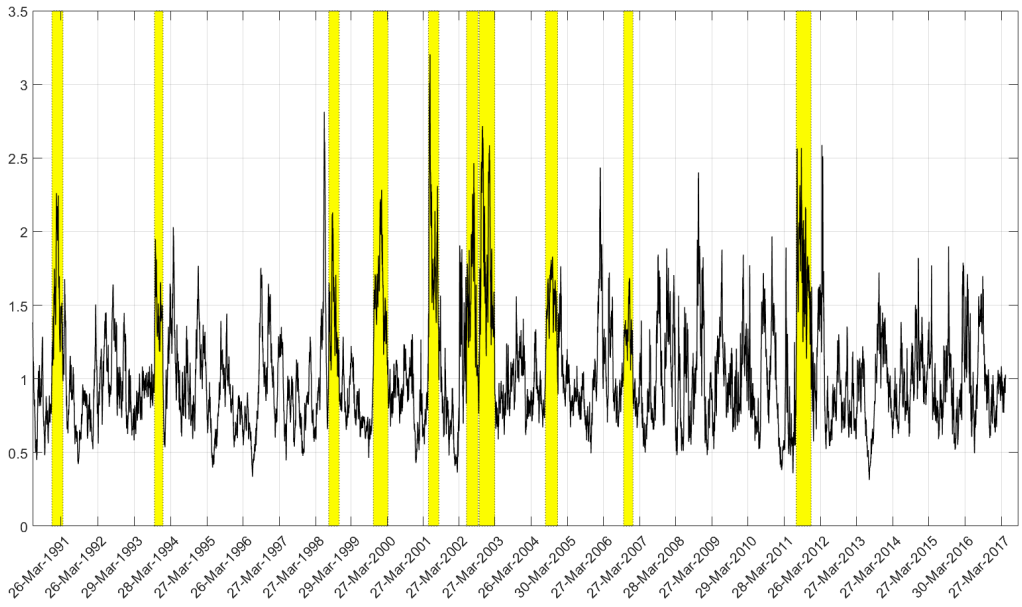

Figure 4 Representation of the periods over which the indicator is greater than one for over 60 days (yellow)

Start date	End date	Duration (days)
26-Dec-1990	16-Apr-1991	79
18-Oct-1993	11-Jan-1994	61
11-Aug-1998	24-Nov-1998	75
08-Nov-1999	04-Apr-2000	105
22-May-2001	04-Sep-2001	75
14-Jun-2002	09-Oct-2002	83
18-Oct-2002	27-Mar-2003	111
20-Aug-2004	21-Dec-2004	87
13-Oct-2006	19-Jan-2007	67
26-Jul-2011	21-Dec-2011	106

Table 7 All periods over which the momentum indicator is greater than one for more than 60 days.