Practical volume approximation of high-dimensional convex bodies, applied to modeling portfolio dependencies and financial crises

Ludovic Calès, Apostolos Chalkis, Ioannis Z. Emiris, Vissarion Fisikopoulos

▶ **To cite this version:**

Ludovic Calès, Apostolos Chalkis, Ioannis Z. Emiris, Vissarion Fisikopoulos. Practical volume approximation of high-dimensional convex bodies, applied to modeling portfolio dependencies and financial crises. Computational Geometry, 2023, 109, pp.101916. 10.1016/j.comgeo.2022.101916 . hal-04294307

HAL Id: hal-04294307
https://hal.science/hal-04294307

Submitted on 28 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Practical volume approximation of high-dimensional convex bodies, applied to modeling portfolio dependencies and financial crises ☆

Ludovic Calès [a], Apostolos Chalkis [b,c,d,*], Ioannis Z. Emiris [c,b], Vissarion Fisikopoulos [b,d]

[a] European Commission, Joint Research Centre, Ispra, Italy
[b] National & Kapodistrian University of Athens, Greece
[c] ATHENA Research Center, Maroussi, Greece
[d] GeomScale org, Greece

A B S T R A C T

We examine volume computation of general-dimensional polytopes and more general convex bodies, defined by 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 estimated by computing volumes of convex bodies. We design and implement practical algorithms in the exact and approximate setting, and experimentally juxtapose them in order to study the trade-off of exactness and accuracy for speed. We also experimentally find an efficient parameter-tuning to achieve a sufficiently good estimation of the probability density of each copula. Our C++ software, based on 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.

1. Introduction

1.1. Financial context and motivation

Modern finance has been pioneered by Markowitz who set a framework to study choice in portfolio allocation under uncertainty, see [1].[1] Within this framework, Markowitz characterized portfolios by their return and their risk which is

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

* Corresponding author.
E-mail addresses: ludovic.cales@ec.europa.eu (L. Calès), achalkis@di.uoa.gr (A. Chalkis), emiris@athenarc.gr (I.Z. Emiris), vfisikop@di.uoa.gr (V. Fisikopoulos).

[1] For which he was awarded the Nobel Prize in economics in 1990.

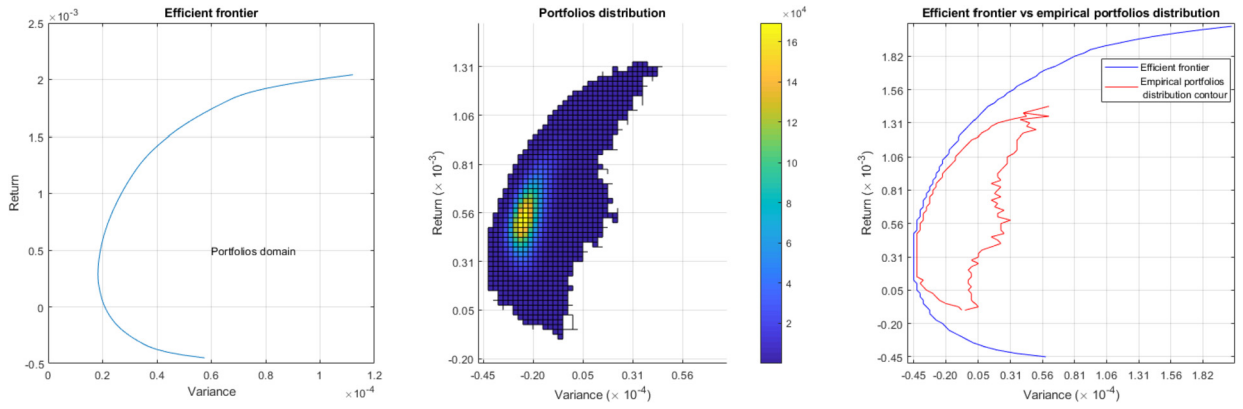

Fig. 1. (left) Efficient frontier, (middle) Empirical portfolio distribution by portfolios' return and variance (10 000 000 portfolios have been sampled uniformly over the set of portfolios as presented later in Section 2.4), (right) Efficient frontier in blue and contour of the empirical portfolio distribution 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. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

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; in the same way, by choosing a level of expected return, an investor can construct a portfolio which minimizes the risk. It has since become 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. Fig. 1 (left panel) presents such an efficient frontier. The region on the right 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. Fig. 1 (middle panel) presents such distribution. When comparing the contour of the empirical portfolios distribution —region over which at least one random portfolio lies— and the portfolio domain bounded by the efficient frontier in Fig. 1 (right panel), we observe that the density of portfolios along the efficient frontier is sparse and that most of the portfolios are located in a small region of the portfolios domain.

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. [2] for details. So, following Markowitz' framework and the capital asset pricing model (CAPM) [3],[2] in normal and up-market times, the stocks and portfolios with the lowest volatility should present the lowest returns, as the less risky shall also be the less profitable. Whereas during crises those with the lowest volatility should present the highest returns, as the most risky assets should be those loosing the most. These features —also called "stylized facts" in the financial literature— 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 Fig. 1 (middle panel), so we will rely on the copula representation of the portfolios distribution. Throughout this paper, we call *copula* a bivariate probability distribution for which the marginal probability distribution of each variable is uniform. More formally,

Definition 1. From [4], a two-dimensional copula is a function $C : [0\ 1]^2 \to [0\ 1]$ with the following properties:

- $C(u, 0) = C(0, v) = 0$ and $C(u, 1) = u, C(1, v) = v$ for every $u, v \in [0\ 1]$;
- C is 2-increasing, i.e. for every $u_1, u_2, v_1, v_2 \in [0\ 1]$ such that $u_1 \leq u_2$, $v_1 \leq v_2$,

$$C(u_1, v_1) + C(u_2, v_2) - C(u_1, v_2) - C(u_2, v_1) \geq 0$$

Consider two random variables (X_1, X_2) whose marginal Cumulative Distribution Functions (CDFs) $F_i(x) = \Pr[X_i \leq x]$, $i = 1, 2$, are continuous functions. Then the random vector $(U_1, U_2) = (F_1(X_1), F_2(X_2))$ has marginals that are uniformly distributed on the interval $[0, 1]$. The copula of (X_1, X_2) is defined as the joint CDF of (U_1, U_2), $C : [0, 1]^2 \to [0, 1]$

$$C(u_1, u_2) = \Pr[U_1 \leq u_1, U_2 \leq u_2]. \tag{1}$$

When there is a, probabilistically, positive relationship between X_1 and X_2 then the largest proportion of the mass induced by the PDF of the copula would concentrate on the up-diagonal (see Fig. 5 left plot). When they have a negative relationship the largest proportion of the mass would concentrate on the down-diagonal (see Fig. 5 right plot). As we follow

[2] The CAPM builds on the model of portfolio choice developed by Markowitz [1].

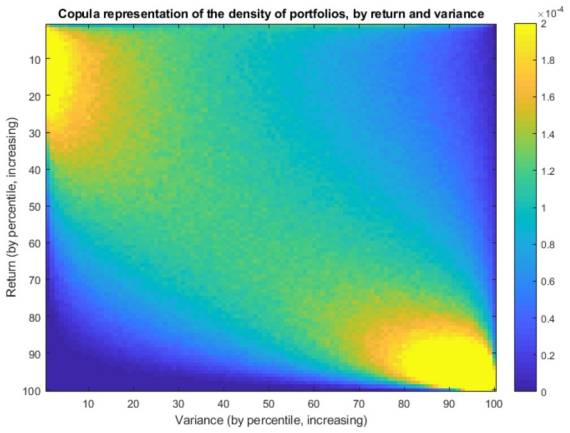

Fig. 2. Copula representation of the portfolios distribution, by return and variance. The market considered is made of the 19 sectoral indices of the DJSTOXX 600 Europe. The data is from October 16, 2017 to January 10, 2018.

Markowitz' framework, the variables considered are the portfolios' return and variance. Fig. 2 illustrates such a copula and shows a positive dependency between portfolios returns and variances. Each line and column sum to 1% of the portfolios. For completeness, we discuss copulae in more detail in Appendix C

The methods introduced here can be used to study other dependencies such as the momentum effect [5] 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 crisis raises policy makers awareness and allows them to act accordingly with potentially large implications in citizens' lives (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 copulae 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 a crisis.[3]

In the general case, the framework to describe the dependencies is as follows. As the set of portfolios, we consider the canonical d-dimensional simplex

$$\Delta^d := \{x \in \mathbb{R}^{d+1} \mid x_i \geq 0, \sum_{i=1}^{d+1} = 1\} \subset \mathbb{R}^{d+1}, \tag{2}$$

where each point represents a portfolio and $d+1$ is the number of assets. The plot in Fig. 3 illustrates the set of portfolios in $\mathbb{R}^3$. 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 up to 1. Notice that Δ^d is a d-dimensional body that lies in $\mathbb{R}^{d+1}$. This is the most common investment set in practice today, as portfolio managers are typically forbidden from short-selling or leveraging.

Next, let us consider the vector of asset returns $R \in \mathbb{R}^{d+1}$. Throughout this paper we use dividend-adjusted stock returns. We define the return of a portfolio as follows.

Definition 2. Given a vector of asset returns $R \in \mathbb{R}^{d+1}$, and a portfolio $x \in \Delta^d$, we say that the return of x is

$$f_{ret}(x, R) = R^T x. \tag{3}$$

Notice that portfolios/points that belong to the same hyperplane with its normal vector being the vector asset returns R, achieve the equal return, while the portfolios above (below) the hyperplane achieve better (worse) return than r. Thus, to model the returns of portfolios we use hyperplanes and halfspaces intersecting Δ^d (see section 2). Given the return of a portfolio, in a certain time period, an important task is to evaluate the performance of that portfolio comparing the alternatives we had. Thus, we define the cross-sectional score of a given portfolio x^* as follows.

[3] Methods exist to estimate the stock returns variance-covariance matrix over short periods, see e.g. the range-based estimation method [2]. However they usually requires high-frequency data and are not widely used. These methods are beyond the scope of this paper.

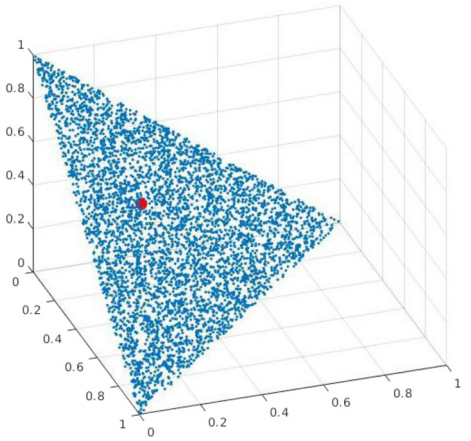

Fig. 3. The set of portfolios, i.e., the canonical simplex Δ^d for $d = 2$. The red point/portfolio corresponds to the asset allocation $x = (0.3, 0.25, 0.45)$.

Definition 3. Given a vector of asset returns $R \in \mathbb{R}^{d+1}$, and a portfolio $x^* \in \Delta^d$ with return $f_{ret}(x, R) = R^T x^*$, we say that the cross-sectional score of the portfolio x^* is

$$\rho_{sc} = \frac{\text{vol}(\Delta^*)}{\text{vol}(\Delta^d)}, \text{ where } \Delta^* = \{x \in \Delta^d : f_{ret}(x, R) \leq f_{ret}(x^*, R)\}. \tag{4}$$

The cross-sectional score of a portfolio x^* corresponds to the share of portfolios with a return lower or equal to $R^* = R^T x^*$. This score corresponds to the cumulative distribution function induced by $\rho_{sc}(x, R)$ where the portfolios are uniformly distributed over the simplex.

It is a standard assumption in literature and in practice to assume that the asset returns follow a multivariate probability distribution. Given the covariance matrix of that distribution we define the portfolio volatility as follows.

Definition 4. Given the covariance matrix $D \in \mathbb{R}^{(d+1)\times(d+1)}$ of the probability distribution that the asset returns follow, and a portfolio $x \in \Delta^d$, we say that the volatility of x is

$$f_{vol}(x, D) = x^T D x. \tag{5}$$

Similarly to portfolio return, notice that portfolios/points the belong to the same ellipsoid centered at the origin and described by matrix D achieve the same volatility, while the portfolios in the interior (complement) of the ellispoid achieve lower (higher) volatility. Thus, to model portfolios' volatility we employ ellipsoids intersecting Δ^d (see section 2).

In a high level, to compute a copula between portfolios' return and volatility (problem of crises detection) we slice Δ^d, i.e. the set of portfolios, along return and volatility. Thus, this question is formulated in terms of convex bodies defined by intersecting Δ^d on one hand by a family of parallel hyperplanes and, on the other hand, by another family of concentric ellipsoids, centered at the origin. When we study the relationship between present returns with past returns (problem of momentum effect detection) we slice Δ^d with two families of parallel hyperplanes. We consider the percentage of the volume that lies inside Δ^d, between two consecutive parallel hyperplanes and between two consecutive concentric ellipsoids (or again two parallel hyperplanes) over the volume of Δ^d. The volume computation of all possible bodies defined the same way, lead directly to the CDF of the corresponding copula.

Finally, to address both financial problems, we study volume computation for the following cases of bodies, that are given as an intersections of Δ^d with: (B1) four halfspaces, where the corresponding hyperplanes are pairwise parallel, (B2) a single halfspace, (B3) the interior of an ellipsoid and two halfspaces, where the corresponding hyperplanes are parallel, and (B4) the interior of an ellipsoid, the complement of a smaller scaled copy of the same ellipsoid and two halfspaces, where the corresponding hyperplanes are parallel. Notice that the body B4 is nonconvex. However, we could obtain the volume of that nonconvex body by the subtraction of the volume of two convex bodies. In particular, we could compute the volume of the convex body defined by the intersection of the simplex with the big ellipsoid and the two halfspaces and subtract the volume of the convex body defined by the same objects and replacing the big ellipsoid with the smaller scaled copy of that.

1.2. Previous work

The cross-sectional score of portfolio returns has been introduced in [6] 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 [6], the 35 components of the IBEX in [7]. This score has also been proposed in [8], for the set of

Table 1
Description of the four bodies for which we study volume computation. The fourth column declares with which financial problem each body is related to: *C* denotes crises detection and *ME* denotes momentum effect.

Type of Body	Description	Convex	Financial Problem
B1	$\{x \in \Delta^d \mid q_1 \leq R^T x \leq q_2,\ q_1' \leq R_1'^T x \leq q_2'\}$	✓	*ME*
B2	$\{x \in \Delta^d \mid R^T x \leq q_2\}$	✓	*C* & *ME*
B3	$\{x \in \Delta^d \mid q_1 \leq R^T x \leq q_2,\ x^T D x \leq c_2\}$	✓	*C*
B4	$\{x \in \Delta^d \mid q_1 \leq R^T x \leq q_2,\ c_1 \leq x^T D x \leq c_2\}$	✗	*C*

long/short equally weighted zero-dollar portfolios and whose estimation relying on combinatorics and statistics is computationally limited to around 20 dimensions, and in [9] where the focus was not on a precise score.

Regarding portfolio optimization, PyPortfolioOpt [10] is a library that contains various methods, including classical mean-variance optimization techniques and Black-Litterman allocation, as well as more recent developments in the field like shrinkage and Hierarchical Risk Parity, along with some novel experimental features like exponentially-weighted covariance matrices.

Given that volume computation of polytopes is #P-hard for both V- and H-representations [11] and no poly-time algorithm can achieve better than exponential error [12], the problem is not expected to admit an efficient deterministic algorithm in general dimensions. Developing algorithms for volume computation has received a lot of attention in the exact setting [13]. In the approximate setting, following the breakthrough polynomial-time algorithm by random walks [14], several algorithmic improvements ensued. The current best theoretical bounds are in [15] and for polytope sampling in [16]. 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 [17], based on the Hit-and-Run paradigm [18], and the Matlab implementation of [19], which supports convex bodies defined by intersecting an ellipsoid with a polytope, and seems competitive to `VolEsti` in very high dimensions. In [20] they provide an FPRAS for the case of a union of convex bodies. For the case of intersection of convex bodies they prove that the FPRAS in [11] is not guaranteed by giving a counterexample, where both no interior point and ball can be computed in polynomial time. However, in our case, since we study special cases of intersections, we compute both in polynomial time. Sampling from nonconvex bodies appears in experimental works, with very few methods offering theoretical guarantees, e.g. in star shaped bodies [21] or, more recently, in [22].

1.3. Our contribution

We design and implement a toolset of methods for volume computation:

M1 Efficient sampling from the simplex then using rejection to approximate the target volume. It can be used to compute the volume of all bodies in Table 1.

M2 Exact formulae of integrals of appropriate probability distribution functions, which are implemented for the case of a single halfspace (body B2).

M3 Optimizing the use of Lawrence's sign decomposition method for body B4 which is a simple polytope.

M4 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 the interior of an ellipsoid (body B3). This is also experimentally generalized for the nonconvex body B4 defined by two ellipsoids with the same quadratic form.

The first method, is fast but inaccurate for small volumes. The second one, is fast and exact for high dimensions. The major issue of the third one is numerical instability for dimension $d \geq 30$. The fourth method achieves accurate approximations are obtained under certain mild conditions. Moreover, this randomized algorithm extends `VolEsti` [17], 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`), rely on Eigen [23] for linear algebra, on `Boost` [24] 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,

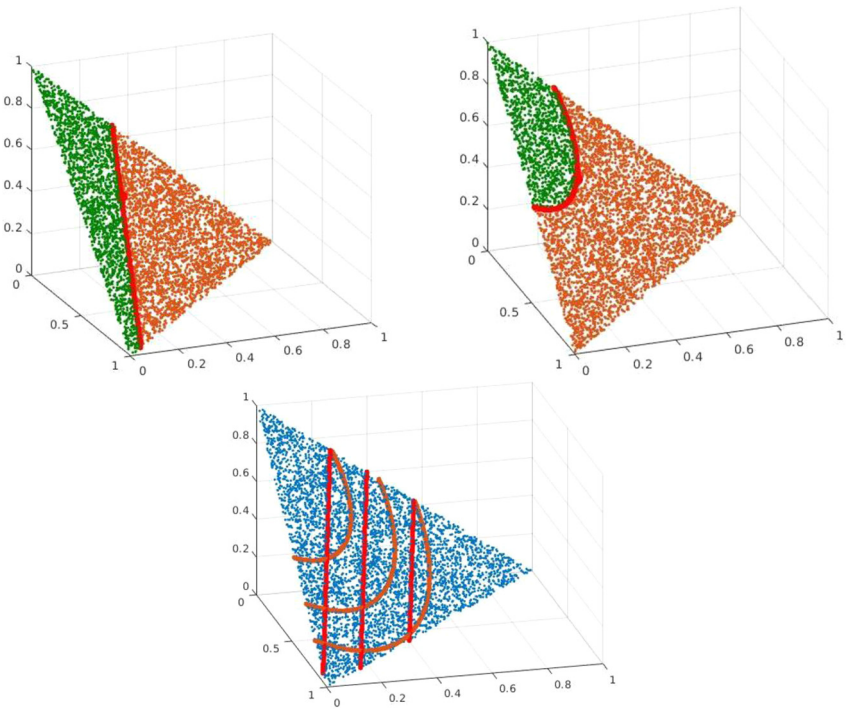

Fig. 4. The top-left plot illustrates the set of portfolios w.r.t. their return. The vector of assets' return is $R = (0.09, 0.13, -0.038)$; the red dot is portfolio $x = (0.3, 0.25, 0.45)$ with return $R^T x = 0.18$; the red segment corresponds to the portfolios with the same return as x; green portfolios have smaller return than x; brown portfolios have higher return than x. The top-right plot illustrates the set of portfolios w.r.t. their volatility. We give the covariance matrix D of the assts' returns in Appendix B; portfolio x has volatility, $x^T Dx = 0.73$; the red curve corresponds to the portfolios with the same volatility as x; green portfolios have smaller volatility than x; brown portfolios have higher volatility than x. The plot below illustrates $\ell = 3$ parallel hyperplanes and concentric ellipsoids intersecting the simplex, defined by vector R and matrix D respectively. Between two consecutive hyperplanes lies the 25% of the volume of the simplex; similarly with the family of ellipsoids.

- it allows us to establish that periods of momentum nearly never overlap with the crisis events,

which are both new result in finance.

The rest of the paper is organized as follows. In section 2 we present geometric modeling of stock markets and connected to copulae. In section 3, we overview methods for sampling uniformly from the simplex and give theoretical guarantees for the sampling - rejection method accuracy. Section 4 considers volume computation of bodies B1 and B2. Section 5 studies convex and nonconvex bodies B3 and B4 respectively, for which random walk methods are developed to extend `VolEsti`. The implementations are discussed in section 6, along with experiments that show the validity of our approach in answering open questions in finance. We conclude with current work and open questions. Tables with experimental results are given in the Appendix.

A preliminary version of this paper appeared in [25]. The current full version contains omitted proofs and detailed experimental results as well as an updated previous work.

2. Convex bodies and financial modeling

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

We apply the methodology to a subset of assets drawn from the DJ 600 constituents. The data used is from Bloomberg™; it is daily and ranges from 01/01/1990 to 31/11/2017. We employ,

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

In the sequel we discuss geometric modeling of portfolios' return and volatility in stock markets.

2.1. Portfolios' returns expressed by hyperplanes

In particular, 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 $R \in \mathbb{R}^{d+1}$ equals:

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

This defines the normal vector to a family of parallel hyperplanes $\{x \in \mathbb{R}^{d+1} \mid R^T x = q,\ q \in \mathbb{R}$. In our framework we consider families with finite number of hyperplanes, whose equations are fully defined by selecting appropriate constants $q_1 < \cdots < q_\ell \in \mathbb{R},\ \ell \in \mathbb{N}$. Each constant q_i correspond to a value of portfolio return. More precisely, the set,

$$\{x \in \Delta^d \mid q_i \leq R^T x \leq q_{i+1}\}, \tag{6}$$

corresponds to the portfolios with return greater or equal to q_i and smaller or equal to q_{i+1}.

For the problem of momentum effect, the second family of parallel hyperplanes is defined similarly by using an adjacent period of k observations. We discuss how we set the values of returns q_i in section 6.

2.2. Market volatility expressed by ellipsoids

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 $D \in \mathbb{R}^{(d+1)\times(d+1)}$ of asset returns. Given the covariance matrix D defines a family of ellipsoids centered at the origin, $\{x \in \mathbb{R}^{d+1} \mid x^T D x = c,\ c \in \mathbb{R}_+\}$. In our framework we consider families with finite number of ellipsoids, whose equations are fully defined by selecting appropriate constants $c_1 < \cdots < c_\ell \in \mathbb{R}_+,\ \ell \in \mathbb{N}$. Each constant c_i correspond to a value of portfolio volatility. More precisely, the set,

$$\{x \in \Delta^d \mid c_i \leq x^T D x \leq c_{i+1}\}, \tag{7}$$

corresponds to the portfolios with volatility greater or equal to c_i and smaller or equal to c_{i+1}. We discuss how we set the values of volatilities c_i in section 6.

2.3. Copulae and volume computation

In this section, we discuss the case of computing the copula between portfolios' return and volatility to address the problem of crises detection. Then, the discussion is directly extended to the case of detecting momentum effect.

Our aim is to estimate the Probability Density Function (PDF) of the copula. Given a vector of asset returns $R \in \mathbb{R}^{d+1}$, let the PDF,

$$g(q) \propto \text{vol}(\{x \in \Delta^d \mid R^T x = q\}),\ q \in \mathbb{R}, \tag{8}$$

which is proportional to the volume of the portfolios which achieve return q. Similarly, for the volatility,

$$h(c) \propto \text{vol}(\{x \in \Delta^d \mid x^T D x = c\}),\ c \in \mathbb{R}_+. \tag{9}$$

Let $G,\ H$ the corresponding CDFs. The copula we would like to estimate is,

$$C(u_1, u_2) = \Pr[G(q) \leq u_1, H(c) \leq u_2] \tag{10}$$

Hence, to estimate the PDF of the copula we first approximate the probability integral transform to each random variable. To achieve this, we compute a sequence of returns $q_1 < \cdots < q_\ell$ and a sequence of volatilities $c_1 < \cdots < c_\ell$, $\ell \in \mathbb{N}$ s.t.,

$$\begin{aligned} &\frac{\text{vol}\Big(\{x \in \Delta^d \mid q_i \leq R^T x \leq q_{i+1}\}\Big)}{\text{vol}(\Delta^d)} = \\ &\frac{\text{vol}\Big(\{x \in \Delta^d \mid c_i \leq x^T D x \leq c_{i+1}\}\Big)}{\text{vol}(\Delta^d)} = \\ &\frac{\text{vol}(\Delta^d)}{(\ell - 1)},\ i = 1, \ldots, \ell - 1 \end{aligned} \tag{11}$$

Finally, by computing the volumes of all possible pairs of intersections we approximate the PDF of the copula by the following probabilities,

Table 2
Overview of the four methods we develop or use for volume computation. The third column gives the type of bodies for which we use each method (see Table 1).

Volume Computation Methods	Description	Bodies
M1	Rejection-sampling method using uniform sampling from the simplex	B1, B2, B3, B4
M2	Exact formula for the volume given by the intersection of a simplex with a halfspace	B2
M3	Optimized Lawrence formula for simple polytopes	B1
M4	Generalization of the `VolEsti` algorithm to non-linear and nonconvex bodies	B3, B4

$$\Pr\left[\frac{i}{\ell-1} \le G(q) \le \frac{i+1}{\ell-1},\ \frac{j}{\ell-1} \le H(c) \le \frac{j+1}{\ell-1}\right] = \frac{\mathrm{vol}\left(\{x \in \Delta^d \mid q_i \le R^T x \le q_{i+1},\ c_j \le x^T D x \le c_{j+1}\}\right)}{\mathrm{vol}(\Delta^d)},\ i, j = 1, \ldots, \ell-1. \tag{12}$$

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, obtain a copula representing the distribution of the portfolios with respect to the portfolios returns and volatilities. Fig. 5 on page 14 illustrates such copulae, and shows the different (probabilistic) relationship between returns and volatility in good (left, dot-com bubble inflating) and bad (right, dot-com bubble burst) times.

Thus, we develop and apply four methods. The first (M1) is to sample from Δ^d and approximate all the volumes directly (Section 3.2). The second (M2) employs an exact formula for the volume defined by the intersection of a simplex with a halfspace (Section 4.1). The third method (M3) is the optimized Lawrence formula for simple polytopes, and is used for the problem of momentum effect detection (Section 4.2). The fourth method (M4) is the generalization of the `VolEsti` algorithm [17] to non-linear and nonconvex bodies (Section 5). We present these methods in the reference Table 2. The Table 2 also shows for which type of body in Table 1 each method is used.

2.4. Simplex representation

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

Since the bodies that we would like to estimate its volumes are defined as intersections of the d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ they are also low-dimensional bodies, i.e., d-dimensional bodies living in the $(d+1)$-dimensions. Thus, we introduce a linear transformation to obtain d-dimensional bodies in $\mathbb{R}^d$. Having full dimensional bodies in $\mathbb{R}^d$ allow us to use our geometric tools (e.g., random walks) more efficiently. 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. $\sum_{i=0}^{d} \lambda_i = 1$, $\lambda_i \ge 0$. 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 $x = (x_1, \ldots, x_d)$:

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

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

$$m_{cb} : \mathbb{R}^d \mapsto \mathbb{R}^{d+1} : x \to \lambda = \begin{bmatrix} -1_d^T \\ I_d \end{bmatrix} M^{-1}(x - v_0) + \begin{bmatrix} 1 \\ 0_d \end{bmatrix}, \tag{13}$$

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.

For the case of ellipsoids, given a full $(d+1)$-dimensional ellipsoid $G : \lambda^T D \lambda - v \le 0$ centered at the origin, where $D \in \mathbb{R}^{(d+1)\times(d+1)}$ is symmetric positive-definite, we compute the equation of the ellipsoid defined by $G \cap \Delta^d \subset \mathbb{R}^d$, by imposing the constraint $\sum_{i=0}^{d} \lambda_i = 1$ by transform m_{cb} in expression (13), thus obtaining:

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

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. The simplex maps to Cartesian coordinates similarly.

3. Volume estimation with rejection-sampling

In this section we discuss Rejection-sampling method (M1) to estimate the volume of a body enclosed to a simplex. First, we present most efficient methods to sample uniformly from a simplex and second, we present method M1.

3.1. Uniform sampling from a simplex

A number of algorithms exist for sampling, where some have been rediscovered, while others contain errors; see the survey [26]. Let us start with the canonical simplex $\Delta^d = \{x \in \mathbb{R}^{d+1} \mid x_i \geq 0,\ \sum_{i=1}^{d+1} x_i = 1,\ i = 1, \ldots, d+1\}$. A $O(d \log d)$ algorithm is the following [27–29]: Generate d distinct integers uniformly in $\{1, \ldots, K-1\}$, where K is the largest representable integer. Sort them as follows: $x_0 = 0 < x_1 < \cdots < x_{d+1} = K$. Now $(x_i - x_{i-1})/K$, $i = 1, \ldots, d$, defines a uniform point. Assuming we possess a perfect hash-function, the choice of distinct integers takes $O(d)$. For $d > 60$ we implement a variant of Bloom filter to guarantee distinctness.

A linear-time algorithm is given in [30], which is generally the algorithm of choice, although it is slower for $d < 80$:

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 canonical simplex Δ^d.

To sample an arbitrary simplex, we can map sampled points from the simplex by transformation in Equation (13), 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 [31].

Sampling could be used in order to approximate all the volumes that arising when a family of parallel hyperplanes and a family of concentric ellipsoids (or another family of parallel hyperplanes) intersect with Δ^d. One can sample the simplex and count the percentage of points that lie between two consecutive hyperplanes from the first family and two concecutive ellipsoids (parallel hyperplanes) from the second family. The complexity is $O(kd)$ to generate k points. Considering a family of ℓ parallel hyperplanes, all sample points are evaluated at the hyperplane, $R^T x = q$, in time $O(kd)$, and at the ellipsoid, $x^T D x = c$, in time $O(kd^2)$. For each point we perform a binary search so as to decide in which layer it lies. Hence the total complexity is $O(kd + kd^2 + k \log \ell)$,

3.2. Sampling - rejection accuracy

In this section we obtain how we could guarantee a bounded error for the sampling - rejection method (M1) with high probability. Let B be a convex or nonconvex full dimensional body in dimension d and 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,

$$\sum_{k=n_1}^{n_2} P(X = k), \quad n_1 = Np(1-e), n_2 = Np(1+e) \tag{14}$$

is the probability that the sampling - rejection method error is at most e. From Poison Limit Theorem [32] we know that if $\lim_{N\to\infty} Np = \lambda$ is a constant independent of N then for any fixed k, $\lim_{N\to\infty} P(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}$. If we set $N = m_1 \cdot 10^z$, where $z = m_2 + \lceil -\log_{10} p \rceil$, we notice that $Np \approx m_1 \cdot 10^{m_2}$. So we can use *Poisson Limit Theorem* to approximate the random variable X as we usually set large enough values for m_1 and m_2. Then from Poison cumulative distribution function we can approximate probability in Equation (14), i.e.,

$$\sum_{k=n_1}^{n_2} P(X = k) = e^{-\lambda} \sum_{i=0}^{\lfloor n_2 \rfloor} \frac{\lambda^i}{i!} - e^{-\lambda} \sum_{i=0}^{\lfloor n_1 \rfloor} \frac{\lambda^i}{i!} \tag{15}$$

4. Intersection with halfspaces

This section considers computing the volume of the intersection of a simplex and one or more linear halfspaces. Clearly, the resulting body is a simple convex polytope. In our framework, the most general case is to be given two sets consisting of parallel hyperplanes each one of them. Then, we consider all created polytopes as follows: we take two consecutive hyperplanes from each set and considering the intersection of the simplex with the spaces in the middle of each pair of hyperplanes; that is a simplex polytope in $\mathbb{R}^d$ with $(d+1)+4$ facets. We assume that the simplex is given in V-representation, i.e. as a set of vertices, and the halfspaces by their equations.

We can always transform the simplex to be a full-dimensional simplex with the origin as one vertex by the transformation of Section 2.4. The same transform applies to the hyperplanes, and volume ratios as preserved.

4.1. Single halfspace formula

Surprisingly, there exist an exact, iterative formula (M2) for the volume defined by intersecting Δ^d with a halfspace in $\mathbb{R}^{d+1}$. A geometric proof is given in [33], by subdividing the polytope into pyramids and, recursively, to simplices. We implement a somewhat simpler formula [34], 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: $A_k \longleftarrow \frac{Y_k A_k - X_h A_{k-1}}{Y_k - X_h}$, for $k = 1, 2, \ldots, K$.

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

Note that an alternative formula exists [35], however, in double precision, the method showed numerical discrepancies above 20 dimensions and was thus abandoned.

4.2. Simple polytopes

This section considers simple polytopes defined by a simplex intersecting four halfspaces, while the corresponding hyperplanes are pairwise parallel. The computation of the volume of such a polytope could be used to approximate the copula in the case of two families of parallel hyperplanes. In particular, we provide an exact algorithm (M3) for the case of a simplex intersecting two arbitrary halfspaces. Then, the discussion is directly expands for the case of four —pairwise parallel— hyperplanes. 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 do not intersect on a simplex edge at the same point.

For a simple polytope P, the decomposition by Lawrence [36] 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 [36], 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 indispensable 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 halfspaces, we exploit the fact that we can map the simplex to the unit simplex 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, we have:

$$A(v) = \begin{bmatrix} -1 & & & 1 & a_1 & b_1 \\ & \ddots & & \vdots & \vdots & \vdots \\ & & -1 & 1 & a_{d-3} & b_{d-3} \\ & & & 1 & a_{d-2} & b_{d-2} \\ & & & 1 & a_{d-1} & b_{d-1} \\ & & & 1 & a_d & b_d \end{bmatrix}, \tag{16}$$

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. *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 the simplex Δ^d. 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 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 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 at most $d\frac{d}{2} = O(d^2)$. □

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 6. *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 a simplex and four, pairwise parallel, hyperplanes. The matrices $A(v)$ remain of the same form because each vertex is incident to at most one hyperplane from each family.

5. 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` [17] so as to handle nonlinear constraints (M4). Our primary motivation here is computing the volume of the intersection of a simplex with an ellipsoid in general dimensions.

5.1. Random walks

The method in [17] follows the Hit-and-Run algorithm in [18], 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. To extend `VolEsti` to the current framework it suffices to solve two subproblems: First, compute the maximum inscribed ball of the convex body a.k.a. Chebychev ball, and second, 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 nonconvex bodies defined by the intersection of two parallel hyperplanes and two concentric ellipsoids. We thus modify `VolEsti` in order to compute the nonconvex 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 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. Given these first positive results, various improvements are planned.

5.2. Chebychev ball

This section offers methods for computing a ball inside the given convex region. Ideally, this is the largest inscribed ball, a.k.a. Chebychev ball, but a smaller ball may suffice. Computing the Chebychev ball reduces to a linear program when P is a polytope (p. 148 in [37]). For general convex regions, more general methods are proposed.

When we do not have the Chebychev ball, an issue is that concentric balls with smaller 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 we find the first inscribed ball. When the body is given as the intersection between the simplex, the interior of an ellipsoid and two halfspaces, 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 constraint.

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 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 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 [38]. However, for our case it suffices to use the generic SOCP solver from [39] 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 the 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.

6. Implementation and experiments

Our implementations are in C++, lie in the public domain as part of `VolEsti` package.[4] They rely on Eigen [23] for linear algebra, on `Boost` [24] for random number generators, and use two SOCP solvers for initializing random walks. Moreover, there is an `R` interface of `VolEsti` in `CRAN` [40]. All experiments of the paper have been performed on a personal computer with Intel Pentium G4400 3.30 GHz CPU and 16 GB RAM. Times are averaged over 100 runs. Some resulting tables and figures are given in the Appendix.

6.1. Compute the families of hyperplanes & ellipsoids

To compute the copulae, we determine the sequence of constants in section 2.3 defining hyperplanes and ellispoids so that the volume between two consecutive such objects is 1% of the simplex volume. Hence, in our implementation, we set the number of constants for both families equal to $\ell = 101$. In the case of hyperplanes, the sequence of constants is determined by bisection using the Varsi's exact formula [33].

In the sequel we set $E(x) = x^T Dx$. For ellipsoids $E(x) = c_i$, the covariance matrix of $d+1$ stock returns is computed using the shrinkage estimator of [41], as it provides a robust estimate even when the sample size is short with respect to the number of assets. Each ellipsoid in this family is fully specified by selecting appropriate constants $c \in \mathbb{R}_+$, we compute the c_i's by sampling the simplex. In particular, we sample N uniform points, and then, we evaluate $E(x)$ at each point. The values are sorted and the c_i selected so as to define intervals containing 1% of the values Thus, between two consecutive ellipsoids —induced say by c_i and c_{i+1}— is contained a percentage of the volume of the simplex close to 1% (see section 3.2).

6.2. Recommendations

We test the methods we developed for all the bodies in Table 1. In general, M2 is preferred when available. Method M1 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 perform exact computing, it becomes too slow.

6.3. Synthetic geometric data

In this section we use synthetic geometric data to test and compare the methods we develop with each where it is possible. We also use packages `Vinci` [42] which provide exact volume computation methods for convex polytopes. In particular, we use the method `rlass` which is a C++ implementation of Lawrence method, and thus, we compare it with method M3.

The formula M2 is used in all Tables where exact computation is needed between two halfspaces —where the corresponding hyperplanes are parallel— and the simplex. Table A.3 considers the intersection of an arbitrary simplex with two random halfspaces. The vertices of each simplex are randomly chosen uniformly from the surface of a ball with radius 100. All hyperplanes' coefficients are randomly chosen in $[-10, 10]$ with `Boost` (mt19937) random generator. For M4 (`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 A.3. On the other hand, M1 is not affected by polytope shape. Up to $d = 30$ and for large volume ratio, namely $> 1\%$, M1 yields very accurate and fastest results. The last two experiments show that M4 achieves the most accurate approximation when the ratio of accepted sampled points is small in M1.

In Table A.4 we use same run-time for M1 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 (M2). 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`. M1 is fast but inaccurate for small volumes; M4 is most accurate but should not scale beyond $d = 100$.

In Table A.5 we have an arbitrary simplex and two arbitrary hyperplanes that intersect with it. We compare our Lawrence implementation (M3) in Section 4.2, using floating-point and rational computation, with `rlass` from `Vinci` package and M1. We have two halfspaces (supported by parallel hyperplanes) intersect the simplex. `Vinci` fails to compute the volume for $d > 31$. Our exact computation works even in $d = 100$ but becomes very slow.

Table A.6 compares M1 with two variants of M4 for ellipsoid intersection. The only difference for the latter is the way we construct an inscribed ball: In s-M4 we implement random sampling until $d+1$ points are found, and in o-M4 we use SOCP. We see M1 is significantly faster than either variant of M4. All methods yield similar output values.

Table A.7 compares M1 with M4 for nonconvex bodies, as in Section 5. Very small values of volume means the method failed to approximate the volume.

[4] https://github.com/GeomScale/volesti.

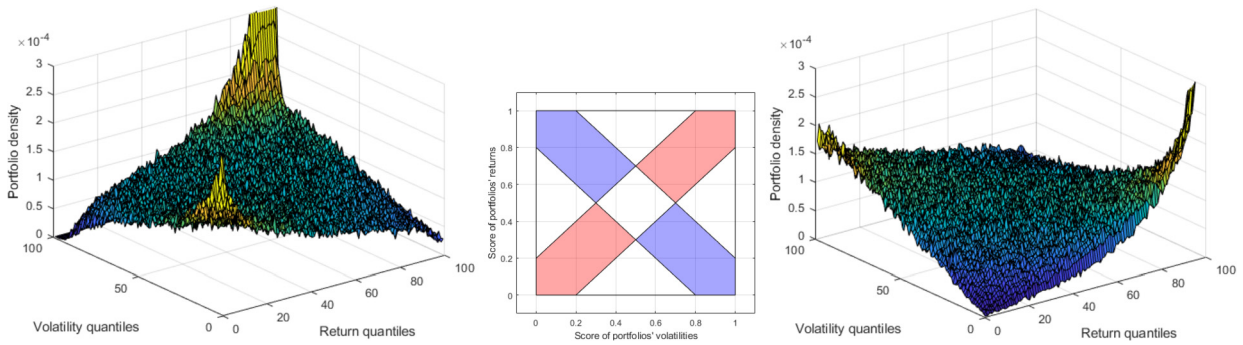

Fig. 5. Returns/variance relationship on the 1st September 1999 (left), i.e. during the dot-com bubble, and on the 1st September 2000 (right), at the beginning of the bubble burst. Blue= low density of portfolios, yellow=high density of portfolios. Illustration of the diagonal bands considered to build the indicator (middle). We consider 100 components of DJ 600 with longest history, over 60 days ending at the given date.

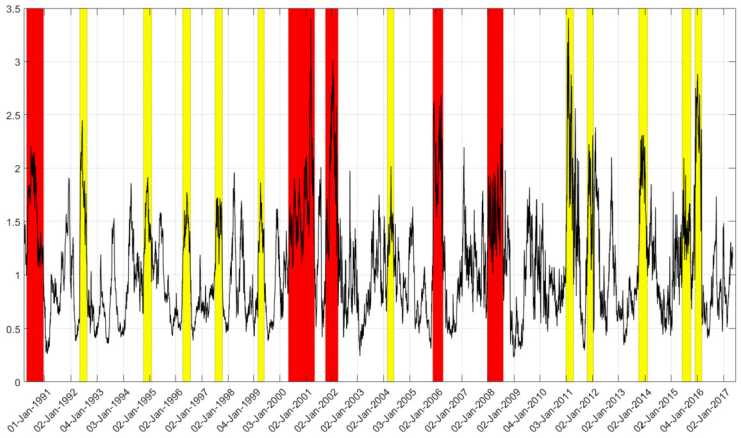

Fig. 6. Representation of the periods over which the indicator is greater than one for 61-100 days (yellow) and over 100 days (red).

6.4. Financial modeling with real data

From the observations of Section 1.1, we expect that, in normal and up-market times, the portfolios with the lowest volatility present the lowest returns. Thus, when considering the copula, the largest proportion of the mass of portfolios should be on the up-diagonal. During crisis the portfolios with the lowest volatility present the highest returns and the largest proportion of the mass of portfolios should be on the down-diagonal of the copula, see Fig. 5 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 Fig. 5 where the indicator is the ratio of the mass of portfolios in the blue area over the mass of portfolios in the red one. We interpret a value of the indicator above 1 as a sign of financial crisis.

In the following, the indicator is computed using copulae estimated using the Rejection-sampling method (M1), drawing 500k points using the sampling method presented in section 3.1. 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 B.8 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 report the crisis indicator in Fig. 6 and the periods we detect as crises periods in Table B.8.

We compare these results with the database for financial crises in European countries proposed in [43]. 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. 7 and the corresponding time periods in Table B.9. We observe that there were only 10 events of lasting momentum effect, mostly around the 1998-2004 period. We remark that they nearly never

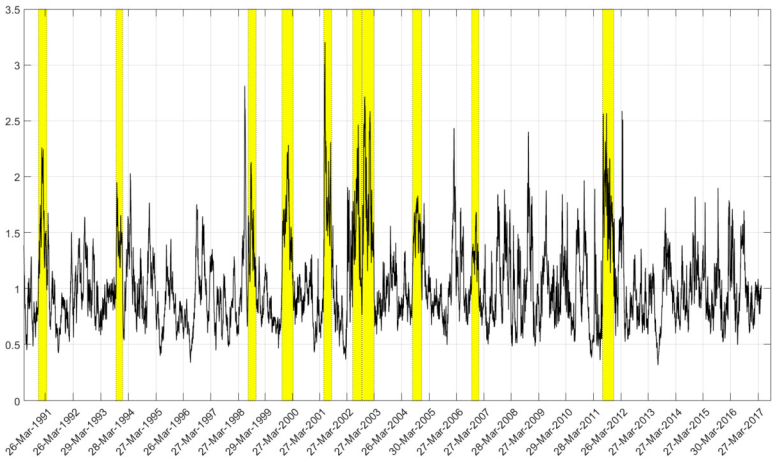

Fig. 7. Momentum effect detection. Representation of the periods over which the indicator is greater than one for over 60 days (yellow).

overlap with the crisis events, with the exception of the end of 2011. To the authors' knowledge, this result is new in finance.

7. Conclusion and future work

In conclusion, we have presented a new approach to the study of financial market dynamics. It consists in representing the distribution of the set of portfolios along two of the portfolio's characteristics as copulae, and in studying the dynamics of those copulae. Based on financial theory, a copula of interest describes the dependence between the portfolios' return and volatility. We build an indicator based on the location of the mass of portfolios within the copula. It provides us with a new way to identify periods of financial turmoil. Another copula of interest describes the dependence between two consecutive vectors of asset returns. With the same indicator we can identify periods of momentum. In an experiment using European stocks, we find that the periods of momentum and financial turmoil, at one exception, did not overlap over the 1990-2017 time range.

The computation of the empirical copulae requires to compute volumes in high-dimension. We develop new methods to handle all possible volume calculation problems that appear in this context. We use a sampling - rejection method based on exact uniform sampling from the simplex. For the volumes that sampling - rejection fails to estimate accurately we develop an optimized Lawrence formula when two arbitrary hyperplanes intersect the simplex and a randomized volume approximation method for the case of a hyperplane and an ellipsoid intersecting the simplex.

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 of a simplex with the interior of an ellipsoid. In [44], they propose a method to approximate the distribution f of quadratic forms in gamma random variables which is a similar problem to that in [35] (see Section 4). 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

[1] H. Markowitz, Portfolio selection, J. Finance 7 (1) (1952) 77–91, https://doi.org/10.1111/j.1540-6261.1952.tb01525.x.
[2] M. Billio, M. Getmansky, L. Pelizzon, Dynamic risk exposures in hedge funds, Comput. Stat. Data Anal. 56 (11) (2012) 3517–3532, https://doi.org/10.1016/j.csda.2010.08.015, http://www.sciencedirect.com/science/article/pii/S0167947310003439.
[3] E.F. Fama, K.R. French, The capital asset pricing model: theory and evidence, J. Econ. Perspect. 18 (3) (2004) 25–46, https://doi.org/10.1257/0895330042162430, https://www.aeaweb.org/articles?id=10.1257/0895330042162430.
[4] R.B. Nelsen, An Introduction to Copulas, Springer Science & Business Media, 2007.
[5] N. Jegadeesh, S. Titman, Returns to buying winners and selling losers: implications for stock market efficiency, J. Finance 48 (1993) 65–91.
[6] I. Pouchkarev, Performance evaluation of constrained portfolios, Ph.D. thesis, Erasmus Research Institute of Management, the Netherlands, 2005.
[7] I. Pouchkarev, J. Spronk, J. Trinidad, Dynamics of the Spanish stock market through a broadband view of the IBEX 35 index, Estud. Econ. Apl. 22 (1) (2004) 7–21.
[8] M. Billio, L. Calès, D. Guéguan, A cross-sectional score for the relative performance of an allocation, Int. Rev. Appl. Financ. Issues Econ. 3 (4) (2011) 700–710.
[9] A. Banerjee, C.-H. Hung, Informed momentum trading versus uninformed “naive” investors strategies, J. Bank. Finance 35 (11) (2011) 3077–3089.
[10] R.A. Martin, “PyPortfolioOpt”: portfolio optimization in Python, J. Open Sour. Softw. 6 (61) (2021) 3066, https://doi.org/10.21105/joss.03066.
[11] M. Dyer, A. Frieze, On the complexity of computing the volume of a polyhedron, SIAM J. Comput. 17 (5) (1988) 967–974, https://doi.org/10.1137/0217060.
[12] G. Elekes, A geometric inequality and the complexity of computing volume, Discrete Comput. Geom. 1 (1986) 289–292.
[13] B. Büeler, A. Enge, K. Fukuda, Exact volume computation for polytopes: a practical study, in: G. Kalai, G. Ziegler (Eds.), Polytopes: Combinatorics and Computation, in: Math. & Statistics, vol. 29, Birkhäuser, Basel, 2000, pp. 131–154.
[14] M. Dyer, A. Frieze, R. Kannan, A random polynomial-time algorithm for approximating the volume of convex bodies, J. ACM 38 (1) (1991) 1–17, https://doi.org/10.1145/102782.102783.
[15] Y. Lee, S. Vempala, Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation, CoRR, arXiv:1710.06261 [abs], 2017, arXiv:1710.06261, http://arxiv.org/abs/1710.06261.
[16] Y. Lee, S. Vempala, Geodesic walks in polytopes, in: Proc. ACM Symp. on Theory of Computing, ACM, 2017, pp. 927–940.
[17] I. Emiris, V. Fisikopoulos, Practical polytope volume approximation, ACM Trans. Math. Softw. 44 (4) (2018) 38, prelim. version: Proc. SoCG 2014.
[18] L. Lovász, Hit-and-run mixes fast, Math. Program. 86 (1999) 443–461, https://doi.org/10.1007/s101070050099.
[19] B. Cousins, S. Vempala, A cubic algorithm for computing Gaussian volume, in: Proc. Symp. on Discrete Algorithms, SIAM/ACM, 2014, pp. 1215–1228.
[20] K. Bringmann, T. Friedrich, Approximating the volume of unions and intersections of high-dimensional geometric objects, Comput. Geom. 43 (6–7) (2010) 601–610, https://doi.org/10.1016/j.comgeo.2010.03.004.
[21] K. Chandrasekaran, D. Dadush, S. Vempala, Thin partitions: isoperimetric inequalities and sampling algorithms for some nonconvex families, CoRR, arXiv:0904.0583 [abs], 2009, arXiv:0904.0583, http://arxiv.org/abs/0904.0583.
[22] Y. Abbasi-Yadkori, P. Bartlett, V. Gabillon, A. Malek, Hit-and-run for sampling and planning in non-convex spaces, in: Proc. 20th Intern. Conf. Artificial Intelligence & Stat., AISTATS, 2017, pp. 888–895, http://proceedings.mlr.press/v54/abbasi-yadkori17a.html.
[23] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org, 2010.
[24] Boost, Boost C++ Libraries, http://www.boost.org/, 2015. (Accessed 30 June 2015).
[25] L. Calès, A. Chalkis, I.Z. Emiris, V. Fisikopoulos, Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises, in: B. Speckmann, C.D. Tóth (Eds.), 34th Intern. Symp. on Computational Geometry (SoCG), in: Leibniz International Proc. in Informatics (LIPIcs), vol. 99, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, 19.
[26] N. Smith, R. Tromble, Sampling uniformly from the unit simplex, Tech. Rep., Center for Language and Speech Processing, Johns Hopkins U., 2004.
[27] H. David, Order Statistics, 2nd edition, Wiley, New York, 1981.
[28] L. Devroye, Non-uniform Random Variate Generation, Springer-Verlag, Berlin, 1986.
[29] R. Rubinstein, D. Kroese, Simulation and the Monte Carlo Method, Wiley Interscience, New York, 2007.
[30] R. Rubinstein, B. Melamed, Modern Simulation and Modeling, Wiley, New York, 1998.
[31] C. Grimme, Picking a uniformly random point from an arbitrary simplex, Tech. Rep., Information Systems and Statistics, Munster U., Germany, 2015, https://doi.org/10.13140/RG.2.1.3807.6968.
[32] G. Last, M. Penrose, Lectures on the Poisson process, https://doi.org/10.1017/9781316104477, https://www.cambridge.org/core/books/lectures-on-the-poisson-process/97F98804D6A3A378C305C8BE1A0D657D, 2017.
[33] G. Varsi, The multidimensional content of the frustum of the simplex, Pac. J. Math. 46 (1973) 303–314.
[34] M.M. Ali, Content of the frustum of a simplex, Pac. J. Math. 48 (2) (1973) 313–322.
[35] A. Mathai, On linear combinations of independent exponential variables, Commun. Stat., Theory Methods (2007).
[36] J. Lawrence, Polytope volume computation, Math. Comput. 57 (195) (1991) 259–271.
[37] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, UK, 2004.

[38] E. Erdougan, G. Iyengar, An active set method for single-cone second-order cone programs, SIAM J. Optim. 17 (2) (2006) 459–484, https://doi.org/10.1137/040612592.
[39] A. Domahidi, E. Chu, S. Boyd, ECOS: an SOCP solver for embedded systems, in: Proc. European Control Conference (ECC), 2013, pp. 3071–3076.
[40] A. Chalkis, V. Fisikopoulos, Volesti: volume approximation and sampling for convex polytopes in R, R J. 13 (2) (2021) 642–660, https://doi.org/10.32614/RJ-2021-077, https://journal.r-project.org/archive/2021/RJ-2021-077/index.html.
[41] O. Ledoit, M. Wolf Honey, I shrunk the sample covariance matrix, J. Portf. Manag. 30 (4) (2004) 110–119, https://doi.org/10.3905/jpm.2004.110, matlab code, https://www.econ.uzh.ch/dam/jcr:ffffffff-935a-b0d6-ffff-ffffde5e2d4e/covCor.m.zip.
[42] B. Büeler, A. Enge, VINCI, http://www.math.u-bordeaux1.fr/~aenge/index.php?category=software&page=vinci, 2000.
[43] M.L. Duca, A. Koban, M. Basten, E. Bengtsson, B. Klaus, P. Kusmierczyk, J. Lang, C. Detken, T. Peltonen, A new database for financial crises in European countries, Tech. Rep. 13, European Central Bank and European Systemic Risk Board, Frankfurt am Main, Germany, 2017.
[44] A. Mohsenipour, S. Provost, On approximating the distribution of quadratic forms in gamma random variables and exponential order statistics, J. Stat. Theory Appl. 12 (2) (2013) 173–184.
[45] M. Sklar, Fonctions de repartition an dimensions et leurs marges, Publ. Inst. Stat. Univ. Paris 8 (1959) 229–231.