GENERATING MOMENT MATCHING SCENARIOS USING OPTIMIZATION TECHNIQUES

SANJAY MEHROTRA AND DÁVID PAPP

Abstract. An optimization based method is proposed to generate moment matching scenarios for numerical integration and its use in stochastic programming. The main advantage of the method is its flexibility: it can generate scenarios matching any prescribed set of moments of the underlying distribution rather than matching all moments up to a certain order, and the distribution can be defined over an arbitrary set. This allows for a reduction in the number of scenarios and allows the scenarios to be better tailored to the problem at hand. The method is based on a semi-infinite linear programming formulation of the problem that is shown to be solvable with polynomial iteration complexity. A practical column generation method is implemented. The column generation subproblems are polynomial optimization problems, however, they need not be solved to optimality. It is found that the columns in the column generation approach can be efficiently generated by random sampling. The number of scenarios generated matches a lower bound of Tchakaloff's. The rate of convergence of the approximation error is established for continuous integrands, and an improved bound is given for smooth integrands. Extensive numerical experiments are presented in which variants of the proposed method are compared to Monte Carlo and quasi-Monte Carlo methods both on numerical integration problems and on stochastic optimization problems. The benefits of being able to match any prescribed set of moments, rather than all moments up to a certain order, is also demonstrated using optimization problems with 100-dimensional random vectors. Empirical results show that the proposed approach outperforms Monte Carlo and quasi-Monte Carlo based approaches on the tested problems.

1. Introduction

We consider the scenario generation problem (or cubature formula generation problem) for numerical integration within the framework of general stochastic programs of the form

$$\min_{x\in\mathcal{X}} \int_{\Xi} f(x,\xi)\mu(d\xi), \tag{1}$$

where f is a convex (hence, continuous) real-valued function, $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex set, and $(\Xi, \mathcal{F}, \mu)$ is a Kolmogorov probability space. We assume that $\Xi \subseteq \mathbb{R}^n$, and that moments of the measure μ can be computed. In numerical solutions (1) is usually approximated by replacing the measure μ by another one, concentrated on finitely many points:

$$\int_{\Xi} f(x,\xi)\mu(d\xi) \approx \sum_{k=1}^{K} w_k f(x,\xi_k). \tag{2}$$

The points $\xi_k \in \Xi$ can be interpreted as *scenarios* representing the original sample space Ξ: scenario ξ_k occurs with probability w_k.

The problem of *scenario generation*, that is, the construction of the right w_k and ξ_k subject to various constraints (derived from the application at hand or imposed by limited computing power) has been a fundamental problem in numerical mathematics and operations research. From an abstract viewpoint, ignoring the dependence of f on x, the approximation of (1) by (2) is simply the problem of constructing *cubature formulas* for the numerical approximation of the

Date: January 29, 2013.

multiple integral $f(x,\cdot)$, which has a long history of study in mathematical analysis going back to Gauss [9]. In this context, ξ_k are the *nodes* and w_k are the *weights* of the cubature formula; the weights need not be probabilities, although they sum to one in every reasonable formula. We will use the terminology of both fields interchangeably in this paper, as they are completely analogous.

In this paper we revisit the classic idea of "polynomial exactness" in the theory of cubature (numerical integration), which corresponds to the approach of "moment matching" scenario generation in stochastic programming. We propose a novel optimization based method to generate formulas of the form (2) with nonnegative weights w_k. This method finds a cubature formula with a K that matches Tchakaloff's lower bound [30]. The algorithm is based on convex optimization, its iteration complexity is polynomial in the number of moments to be matched. The generated formulas can flexibly be adapted to the particular instance of (1) at hand, as the set of moments to be matched can be chosen arbitrarily, and also the domain of integration and the underlying distribution can be arbitrary, as long as moments of the underlying distribution can be computed (approximately, at least), and random samples from the distribution can be drawn efficiently. To our knowledge, this setup is considerably more general than the known approaches in the numerical analysis and stochastic programming literature. We proceed with the precise definitions of our problem.

Definition 1 (cubature formula)**.** *A* cubature formula *is a finitely supported finite signed measure on $\mathbb{R}^n$, represented by a pair $(\xi, w) \in (\mathbb{R}^n)^K \times \mathbb{R}^K$; the components $\xi_1, \dots, \xi_K$ of ξ are the* nodes *or* support points *of the formula, the components $w_1, \dots, w_K$ of w are the corresponding* weights*. The formula is* positive *if $w > 0$ componentwise, and it is* interior *if it is supported on Ξ.*

In the numerical integration setting the most important property cubature formulas must have is that the approximation (2) is exact in some space of "interesting" functions, such as polynomials up to a given total degree, which then allows one to estimate the approximation error in larger classes of functions, in particular in the classes of analytic and continuous functions [29].

As customary, we shall denote by $\mathbb{R}[x]$ the ring of n-variate polynomials whose variables are represented by the components of the vector x; $\mathbb{R}[x]_d$ denotes the space of polynomials of (total) degree up to d, with real coefficients. This is a linear space of dimension $N(n,d) = \binom{n+d}{n}$.

Definition 2 (degree of precision or exactness)**.** *The* degree of exactness *or* degree of precision *of a cubature formula (ξ, w) with respect to the measure μ is the largest integer d with the property that the approximation* (2) *is exact for polynomials up to degree d, that is,*

$$\int_\Xi f(x)\mu(dx) = \sum_{k=1}^K w_k f(\xi_k) \text{ for every polynomial } f \in \mathbb{R}[x]_d.$$

A cubature formula is exact for polynomials of degree up to d if and only if its values for monomials up to total degree d agree with the moments of μ of up to order d. This requirement has a clear interpretation in stochastic programming: a positive cubature formula with degree of exactness d corresponds to a discrete distribution whose moments up to order d agree with the corresponding moments of the probability measure μ. Hence, this approach to finding a finite set of representative scenarios for μ in stochastic programming is referred to as *moment matching scenario generation*, and the corresponding cubature formulas are also referred to as *moment matching formulas*.

The main difficulty of solving (1) is that it requires the computation of a potentially large number of integrals, and that the evaluation of the integrands may also be expensive. Hence, we must rely on formulas (2) with as small K as possible. The focus of the theory of cubature has always been the trade-off between the error of approximation in (2) and the magnitude of K, however, methods successful in the integration of simple integrands (outside the field of

optimization) always assume that the number of function evaluations can be much larger than the typical number of scenarios in stochastic programming. This is highlighted both by the importance given to asymptotic analysis of cubature formulas (as the dimension and the number of function evaluations both tend to infinity) and by the numerical experiments typical in the numerical analysis literature, which often take $K \approx 10^5$ or $K \approx 10^6$; see, e.g., [10, 27].

Another concern is that existing methods of cubature do not provide their user with the flexibility to adapt the cubature formulas to the stochastic optimization problem at hand. In stochastic programming available computing power puts a constraint on the size of the node set of the cubature formula. Also, mixed moments of random variables may have different importance due to their association with different sources of uncertainty. Hence the need for flexible methods for designing cubature formulas that can match any prescribed set of moments, rather than all moments up to a certain order. To our knowledge, this problem has not been addressed in the numerical integration literature. An example of such an approach used in stochastic programming is presented in [16], where a scenario generation method is proposed to match all marginal moments of order 4 and lower, and all correlations. Unfortunately no convergence proof is given in [16] and this approach does not seem to admit a generalization for higher order moments or other combination of moments.

The positivity of the weights of a cubature formula is desirable for multiple reasons: if the weights also sum to one, then the weights can be interpreted as probabilities. It also helps prevent cancellation errors in the formula. Lastly, but most importantly, in convex optimization models, where $f(\cdot, \xi)$ is convex for every $\xi \in \Xi$, the approximation $\int_\Xi f(x, \xi)\mu(d\xi) \approx \sum_k w_k f(x, \xi_k)$ preserves the convexity of f in x if $w_k > 0$. Especially motivated by the last argument, in this paper we concentrate only on positive cubature formulas. Note that the weights sum to one for every formula that is exact for constant polynomials, thus the weights of moment matching positive formulas are always probabilities.

The main contributions of this paper are twofold. First, we provide a systematic approach to moment matching scenario generation, and establish their rate of convergence (as the number of scenarios and the degree of exactness increase) both for general continuous integrands and for smooth integrands with continuous derivatives. Second, we present a novel optimization based method designed to find cubature formulas. The method generates moment matching positive interior formulas with a small number of nodes. It can be used to match any prescribed set of moments. It is applicable to problems with arbitrary probability measures μ as long as the moments of μ are known, and sampling from the distribution represented by μ is possible. A variant using the ellipsoid method is shown to an iteration complexity polynomial in the number of moments to be matched, while a simplex method based column generation variant has even lower empirical iteration complexity. Focusing on the column generation approach, it is shown that the column generation subproblem is a polynomial optimization problem, and various approaches to its solution are discussed. Extensive numerical experiments compare the different column generation approaches (and the resulting moment matching cubature formulas) to other scenario generation methods. The results of these experiments indicate that moment matching scenarios generated by our approach consistently outperform Monte Carlo and quasi-Monte Carlo integration in both integration and stochastic programming problems with small number of scenarios.

We close this section with a brief outline of the paper. The next section is a broad review of the immense literature on cubature and scenario generation; readers familiar with the topic may skip it altogether. Rates of convergence of moment matching formulas, as the number of scenarios increases, are derived in Section 3 under different assumptions on the integrand. Our new scenario generation method is introduced in Section 4, where we also show that its iteration complexity is polynomial. In Section 5 variants of the method are compared to each other and to Monte Carlo and quasi-Monte Carlo scenario generation on a variety of problems. Both their

efficiency in formula generation and their performance in integration and stochastic optimization are considered; see the beginning of Section 5 for a detailed outline of our numerical experiments.

2. Methods of numerical integration and scenario generation

2.1. **Past work on cubature formulas.** A large number of scenario generation methods are available that are based on the discretization of the probability measure μ, [26] is an up-to-date survey. These all have direct interpretation as cubature formulas. Below we only provide a brief summary of these approaches.

2.1.1. *Optimal cubature formulas.* The study of cubature of univariate functions, in which case the term *quadrature* rather than cubature is commonly used, goes back to Gauss [9]. Minimally supported quadrature formulas, which are all positive, are completely characterized for an arbitrary measure μ. The following observations are known.

Proposition 3 (Gaussian quadrature [7])**.** *Suppose $n = 1$, and μ is a given probability measure supported on the closed interval $\Xi = [a, b]$. Then every quadrature formula with K nodes has degree of exactness at most $2K - 1$. Furthermore, there is a unique K-node quadrature formula with exactness $2K - 1$. This quadrature formula is a positive and interior formula.*

The quadrature formulas described in the previous proposition are called *Gaussian* formulas. It is elementary that $2K - 1$ is also an upper bound on the degree of exactness of every (uni- or multivariate) K-node cubature formula. By extension from the univariate case, cubature formulas that achieve this exactness are also called Gaussian. However, the question which multivariate measures μ admit a Gaussian formula is largely unsolved. Whenever exist, Gaussian cubature formulas are also positive, but are not necessarily interior. They are characterized as the common roots of orthogonal polynomials. For more on Gaussian cubature and orthogonal polynomials, see [7, Section 3.7].

The minimum number of nodes is more easily found for positive interior formulas. It can be shown that for every n-dimensional Ξ and μ and every degree d there exists a positive interior formula with $\binom{n+d}{d}$ nodes with degree of exactness d, see Theorem 10 below. On the other hand, a result of Tchakaloff implies that for every n there exist measures for which this bound is sharp [30]. The method proposed in this paper constructs formulas with exactly this many scenarios or fewer.

2.1.2. *Product formulas.* If $\Xi = [0, 1]^n$, simple cubature formulas can be obtained by considering direct products of univariate (quadrature) formulas. Using the same K-node quadrature formula (ξ, w) for each dimension, we obtain a positive formula with K^n points. The nodes of such a formula are vectors of the form $(x_{i_1}, \dots, x_{i_n})$ where $1 \le i_j \le K\}$ for each $j = 1, \dots, n$, the corresponding weight is $\prod_{j=1}^{n} w_{i_j}$. The degree of exactness of this formula is the same as the exactness of the univariate formula (ξ, w). Furthermore, product formulas match several higher-order mixed moments than their degree of exactness.

This approach is also helpful to derive formulas with different degree of exactness for different marginals. However, owing to the large number of points used, product formulas are only useful in low dimensions, or when the required degree of exactness is very low, say, 3.

Product formulas are directly applicable only when the domain of integration is the unit cube $[0, 1]^n$. For other domains, a change of variables might be necessary. This is also true for most other approaches of this section.

2.1.3. *Sparse grid formulas.* Sparse grid integration methods are also based on quadrature formulas and their product formulas, but they avoid the exponential increase in the number of nodes as the dimension increases. The classic sparse grid approximation framework was established by

Smolyak in [28]; see [13] for a transparent introduction to its application to approximate numerical integration. Sparse grid formulas can also be adapted to match moments of different orders for each marginal, but they are not flexible enough to match any prescribed set of moments, and they have negative weights. Assigning nonnegative weights to sparse grid points while still matching a prescribed set of moments is generally also not possible, unless the sparse grid contains the nodes of a product formula with a required degree of exactness. Hence, we do not consider sparse grid formulas any further.

2.1.4. *Optimal quantization of probability measures.* Given a probability measure μ and a probability metric D, the problem of *optimal quantization* is to find a discrete probability measure ν_K that minimizes the distance $D(\mu, \cdot)$ among all probability measures supported on K points. This approach is particularly reasonable when the stochastic program is known to be stable with respect to the chosen metric D. This approach is still a very active research area. Optimal quantization yields positive formulas, and may yield interior formulas as well. It may also be extended to multi-stage problems by incorporating the tree structure in the constraints of the optimization model. The main drawback of this approach is its complexity: for most popular metrics it is very difficult to find the optimal ν_K even for single-stage problems. The exact formulation leads to a non-differentiable non-convex optimization model, whose solution requires global optimization or integer programming techniques; see [26] and the references therein.

2.1.5. *Other non-product formulas.* The construction of cubature rules with a small number of points and a given degree of exactness is a largely open problem. The existing methods usually have a very narrow scope: the formulas are essentially derived one by one for every domain of integration (for instance, hypercube of a given dimension), every measure μ, and every degree of exactness, with little hope of generalization to the level necessary for our purposes. The most general and most popular methods rely on the theory of orthogonal polynomials and require the solution of a large nonlinear system of equations. While these approaches have provided excellent formulas, often with considerably fewer nodes than Tchakaloff's bound (and what is obtained in the present paper), they are numerically difficult, they have not yielded formulas for high-dimensional problems, and they have no guarantee that they yield either positive or interior formulas. Hence, we do not consider these formulas either, but direct the interested reader to two compendiums of such formulas: Stroud's classic book [29], and the electronic encyclopedia of cubature formulas [3].

2.1.6. *Monte Carlo and quasi-Monte Carlo methods.* Monte Carlo integration amounts to using the formula (2) with equal weights: $w_k = 1/K$ for each k, and with K independent random ξ_k sampled from Ξ according to the distribution corresponding to μ. The convergence as $K \to \infty$ is almost certain by the strong law of large numbers, at a rate $O(K^{-1/2})$.

When μ is uniform, quasi-Monte Carlo methods are similar to Monte Carlo integration, but the random numbers used to generate the samples ξ_k are replaced with elements of a *low-discrepancy* sequence, which fill Ξ more evenly than (pseudo-)random samples. For other distributions appropriately transformed low-discrepancy sequences are used. Popular low-discrepancy sequences include those constructed by Niederreiter, Halton, and Sobol. These formulas involve no randomization. Their rate of convergence is $O((\log^n K)/K)$ for sufficiently "smooth" integrands (defined precisely as functions of bounded variation in the sense of Hardy and Krause); the rate of convergence for non-smooth functions is discussed in Section 3.

Both Monte Carlo and quasi-Monte Carlo methods yield positive interior formulas. For more on these methods see for example the classic text [21].

2.2. **Moment matching scenario generation.** Moment matching methods in stochastic programming have been proposed, but primarily in the context of sampling from partially specified distributions, where to goal is to efficiently sample some (unknown) distribution that has given

moments up to a certain order (usually three or four). The motivation behind this line of research is different from ours. Moment matching sampling algorithms do not yield cubature formulas. They are to be used in a Monte Carlo setting when the usual Monte Carlo integration is not applicable, because the measure μ is not given, only its moments up to some order.

There are limitations of this approach in addition to the fact that they do not yield cubature formulas. One is that constraints on the support of the distributions are not considered, hence, the methods may yield scenarios outside the sample space Ξ. Moreover, such sampling methods so far have only been derived on a case-by-case basis. For instance in [19] the marginal distributions are given along with a covariance matrix, and the latter is used to determine a transformation which is then used to generate samples with the given marginals and the desired correlation. A similar approach is used in [16], where marginal moments up to the fourth order are matched along with the mixed second moments. (The method of this paper has an additional drawback that it is not known to be convergent.) The covariance matrix is matched and the first four marginal moments are approximated using semidefinite optimization in [4]. None of these methods can be generalized to match higher order moments, or any other set of moments.

2.3. **Scenario reduction, and other non-moment matching methods.** An entirely different approach, both for single-stage and multi-stage problems is *scenario reduction*, where a large number of sample scenarios are generated first (for example, randomly, or using a regular grid), and then a subset of them is removed to achieve the desired complexity of the model, measured by the number of scenarios K.

In one variant the scenarios to remove are determined by minimizing some probability metric between the original and the reduced scenario set. This problem can be formulated as a very large-scale mixed integer linear programming problem that is very difficult to solve exactly; instead, heuristics are used [8].

Another family of scenario reduction heuristics comes from data mining: traditional clustering methods can be applied to a large number of simulated scenarios to obtain a smaller number of scenarios that are "representative" to the original scenario set. An example of this approach is [12].

It is out of the scope of this paper, but another interesting problem is the generation of scenario trees for multi-stage models, preserving information given by the tree structure in addition to the information provided by the leaves. See [14] for some recent results in this area.

3. The rate of convergence of moment matching cubature formulas

The main motivation behind moment matching is the observation that cubature formulas with high degree of polynomial exactness should provide accurate estimates of the integrals of functions that can be approximated well by polynomials – this includes continuous functions on closed and bounded domains. In this section we formalize this statement and provide an upper bound on the rate of convergence of moment matching cubature formulas as a function of their modulus of smoothness. We also show that this rate of convergence cannot be improved by any other formula, aside from constants in this function, without additional assumptions on the integrand. At the end of the section we give a tighter bound for integrands with continuous derivatives; this rate of convergence improves with the existence of each additional derivative. We need the following definition.

Definition 4 (modulus of smoothness)**.** *The* modulus of smoothness *of a function $f\colon \Xi \mapsto \mathbb{R}$, denoted by ω_f is the function given by*

$$\omega_f(\delta) \stackrel{\text{def}}{=} \sup\left\{|f(x) - f(y)|\colon x \in \Xi, y \in \Xi, \|x - y\| \leq \delta\right\}.$$

It is known that the error of the best polynomial approximation of every continuous f can be bounded by a function of ω_f as follows.

Theorem 5 (see, e.g. [1])**.** *Let $f\colon \Xi \mapsto \mathbb{R}$ be continuous on a compact set $\Xi \subseteq \mathbb{R}^n$. Then for every nonnegative integer d there is a polynomial P_d of degree d satisfying*

$$\sup_{\Xi} |f - P_d| \leq C\omega_f(1/d),$$

where C is a positive constant depending only on Ξ, but not on d or f.

This rate of convergence (as the degree increases) is inherited by the error of cubature formulas.

Theorem 6. *Assume that the support $\Xi \subseteq \mathbb{R}^n$ of the probability measure μ is compact. Then there is a constant $\bar{C}$ (depending only on Ξ) such that for every continuous $f\colon \Xi \mapsto \mathbb{R}$ and for every cubature formula with nodes $\xi_1, \ldots, \xi_K$, nonnegative weights $w_1, \ldots, w_K$, and degree of exactness d,*

$$\left| \int_{\Xi} f(\xi)\mu(d\xi) - \sum_{k=1}^{K} w_k f(\xi_k) \right| \leq \bar{C}\omega_f(1/d).$$

Proof. Let P_d be the best degree-d polynomial approximation of f in the uniform norm. Then we have $\int_{\Xi} P_d(\xi)\mu(d\xi) = \sum_{k=1}^{K} w_k P_d(\xi_k)$, therefore,

$$\begin{aligned}
\left| \int_{\Xi} f(\xi)\mu(d\xi) - \sum_{k=1}^{K} w_k f(\xi_k) \right| &\leq \left| \int_{\Xi} f(\xi)\mu(d\xi) - \sum_{k=1}^{K} w_k P_d(\xi_k) \right| + \left| \sum_{k=1}^{K} w_k f(\xi_k) - \sum_{k=1}^{K} w_k P_d(\xi_k) \right| \\
&= \left| \int_{\Xi} \left(f(\xi) - P_d(\xi)\right) \mu(d\xi) \right| + \left| \sum_{k=1}^{K} w_k (f(\xi_k) - P_d(\xi_k)) \right| \\
&\leq \int_{\Xi} |f(\xi) - P_d(\xi)| \, \mu(d\xi) + \sum_{k=1}^{K} w_k \, |f(\xi_k) - P_d(\xi_k)| \\
&\leq 2 \sup_{\Xi} |f - P_d| \leq 2C\omega_f(1/d),
\end{aligned}$$

using in the last two steps $\mu(\Xi) = \sum_{k=1}^{K} w_k = 1$ and Theorem 5. Hence, our assertion holds with $\bar{C} = 2C$. □

The rate of convergence of sequences of cubature formulas and scenario generation methods is usually expressed as a function of the number of nodes (or scenarios) K.

Theorem 7 (rate of convergence)**.** *Assume that the support $\Xi \subseteq \mathbb{R}^n$ of the probability measure μ is compact. Then for every K_0 there exist a cubature formula with $K \geq K_0$ nodes $\xi_1, \ldots, \xi_K$ and nonnegative weights $w_1, \ldots, w_K$ satisfying*

$$\left| \int_{\Xi} f(\xi)\mu(d\xi) - \sum_{k=1}^{K} w_k f(\xi_k) \right| \leq O(\omega_f(O(K^{-1/n})))$$

for every continuous function $f\colon \Xi \mapsto \mathbb{R}$. In particular, formulas with increasing polynomial exactness have this property.

Proof. By virtue of Theorem 6 it suffices to show that if μ (and hence, n and Ξ) are fixed, we can construct for every degree of exactness d a cubature formula with positive weights and $K = O(d^n)$ nodes, since then we have $\bar{C}\omega_f(1/d)) = \bar{C}\omega_f(O(K^{-1/n}))$. This follows from Tchakaloff's theorem ([30]; see also Theorem 10 in Section 4), which states that we have such a formula with $K \leq \binom{n+d}{n}$ nodes and positive weights for every $n, d \geq 1$. □

The following theorem shows that this rate of convergence is essentially the best possible even when $\Xi = [0,1]^n$, and integration is with respect to the uniform distribution on Ξ.

Theorem 8. *Let $\Xi = [0,1]^n$ and $0 < C_1 < 1$ and $0 < C_2 < 2^{-n}$ be fixed constants. Then there exist no cubature formula on K nodes with positive weights that satisfies*

$$\left| \int_\Xi f(\xi) d\xi - \sum_{k=1}^K w_k f(\xi_k) \right| \le C_1 \omega_f(C_2 K^{-1/n})$$

for every continuous f.

Proof. Let B_x denote the set $[0, x_1) \times \cdots [0, x_n)$ for every $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, and for every $A \subseteq [0,1]^n$, let $\lambda(A)$ and $\chi(A, \cdot)$ denote the Lebesgue measure and the characteristic function of A, respectively. For a formula with nodes $\xi_1, \ldots, \xi_K$ and corresponding weights $w_1, \ldots, w_K$, its *(weighted) star-discrepancy* is the quantity

$$D(\xi, w) \stackrel{\text{def}}{=} \sup_{x \in [0,1]^n} \left| \lambda(B_x) - \sum_{k=1}^K w_k \chi(B_x, \xi_k) \right|.$$

Suppose that ξ and w are fixed, and that (without loss of generality) the nodes are indexed in increasing order of their first coordinate. Let $\hat{k} = \arg\max w_k$, and take $\hat{x} = (\hat{x}_1, 1, \ldots, 1)$ with $\hat{x}_1 = \sum_{k=1}^{\hat{k}-1} w_k + w_{\hat{k}}/2$. From the definitions it is clear that $\lambda(B_{\hat{x}}) = \hat{x}_1$ and that $\sum_{k=1}^K w_k \chi(B_{\hat{x}}, \xi_k) \in \{0, w_1, w_1 + w_2, \ldots, \sum_{k=1}^K w_k = 1\}$, meaning that either $\sum_{k=1}^K w_k \chi(B_{\hat{x}}, \xi_k) \le \sum_{k=1}^{\hat{k}-1} w_k$ or $\sum_{k=1}^K w_k \chi(B_{\hat{x}}, \xi_k) \ge \sum_{k=1}^{\hat{k}} w_k$. Therefore,

$$\left| \lambda(B_{\hat{x}}) - \sum_{k=1}^K w_k \chi(B_{\hat{x}}, \xi_k) \right| \ge \tfrac{1}{2} w_{\hat{k}},$$

yielding the estimate

$$D(\xi, w) \ge \tfrac{1}{2} w_{\hat{k}} \ge \tfrac{1}{2K}. \tag{3}$$

It is a result of Proinov [24] that if $C_1 < 1$, then there exist no cubature formula on K nodes with positive weights that satisfies $\left| \int_\Xi f(\xi) d\xi - \sum_{k=1}^K w_k f(\xi_k) \right| \le C_1 \omega_f(D(\xi, w)^{1/n})$ for every continuous f. Combining this inequality with (3) yields our assertion. □

The estimates in Theorems 6 and 7 can be improved considerably for smooth functions.

Theorem 9. *Suppose that in Theorem 6 all rth order partial derivatives of f are continuous. Then the error bound of moment matching cubature formulas with degree of exactness d can be improved to*

$$\left| \int_\Xi f(\xi) \mu(d\xi) - \sum_{k=1}^K w_k f(\xi_k) \right| \le \hat{C} d^{-r} \omega_f^{(r)}(1/d), \tag{4}$$

where

$$\omega_f^{(r)}(\delta) \stackrel{\text{def}}{=} \sup_{\substack{\rho \in \mathbb{N}^n \\ \sum_i \rho_i = r}} \omega_{D^\rho f}(\delta)$$

is the highest of the moduli of smoothness of the rth partial derivatives of f.

For functions of bounded rth partial derivatives, there exist moment matching formulas on K nodes, including those generated by the algorithms of Section 4, whose error (as a function of K) tends to zero at a rate $O(K^{-r/n})$.

Proof. The proof of the inequality (4) is the same as the proof of Theorem 6, except that in the last step we use the stronger bound

$$\sup_\Xi |f - P_d| \le C d^{-r} \omega_f^{(r)}(1/d),$$

of [1] on the polynomial approximation of r times continuously differentiable functions. The second assertion then follows from the existence of formulas with $K \leq \binom{n+d}{n}$ nodes and degree of exactness d, which we shall prove, constructively, in Section 4. □

It is not easy to compare the results of Theorems 7 and 8 to the known rates of convergence results of quasi-Monte Carlo methods, owing to their different assumptions. Theorem 9 yields increasingly better rates of convergence for functions of increasingly higher differentiability. In particular, for $r \geq n$ the bound provided by Theorem 9 is better than the $O((\log^n K)/K)$ rate of convergence of classic QMC methods. There is also a difference in the hidden constants: the latter rate of convergence depends on the Hardy–Krause variation of the integrand, whereas the rate of convergence of moment matching depends on the modulus of smoothness. (Note that functions of bounded modulus of smoothness can have unbounded Hardy–Krause variation.) On the other hand, some modern quasi-Monte Carlo methods, for example lattice rules and high-order digital nets also exploit the differentiability of the integrand as well as additional regularity conditions, such as fast decay in the Fourier coefficients; see, for example, [21, Chap. 5], [6, Chap. 15], and [17] for the optimal attainable rates of convergence under such assumptions and algorithms.

4. A column generation approach for moment matching

In this section we propose a new approach to constructing cubature formulas that match any prescribed set of moments. This approach is based on a semi-infinite linear optimization formulation of the moment matching problem that allows the construction of formulas for a considerably larger variety of measures than the methods so far described. We shall also underline that the approach we are about to outline is even more general in that it can be used to find cubature formulas that give exact values in *any* given finite dimensional linear space of functions, not just in spaces of polynomials spanned by monomials. However, in this paper we only consider formulas matching polynomial moments. First, we need to introduce some notation.

For a point $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, let $u_x \in \mathbb{R}^N$ denote the N-dimensional vector whose components are the monomials whose corresponding moments we are trying to match, in an arbitrary, but fixed, order (say, the graded lexicographic order). For example, if $n = 3$ and we want to match all moments up to order $d = 2$, then the number of moments to match is $N = \binom{n+d}{d} = 10$, and the moments correspond to the monomials

$$u_x = (1, x_1, x_2, x_3, x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2)^{\mathrm{T}},$$

(all the monomials in an arbitrary, but fixed, order) but if $n = 2$ and we want to match all mixed moments up to order 3 as well as marginal moments up to order 5, then we have $N = 14$, and

$$u_x = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_2^4, x_1^5, x_2^5)^{\mathrm{T}}.$$

Note that we may use bases of polynomials other than the monomial basis; in fact, in our computations we used Legendre polynomials to avoid numerical issues while trying to match high-order moments. In the sequel we will only assume (without loss of generality) that the components of u_x are linearly independent polynomials.

The cubature formula (ξ, w) matches all the required moments if and only if

$$\sum_{k=1}^{K} w_k u_{\xi_k} = m \stackrel{\text{def}}{=} \int_{\Xi} u_\xi \mu(d\xi), \tag{5}$$

with the integral on the right-hand side understood componentwise. The components of the vector $m \in \mathbb{R}^N$ are the required monomial moments of μ, ordered the same way as the components of u_x.

For fixed $\xi_1, \ldots, \xi_K$, finding nonnegative weights to satisfy (5) is a linear programming (feasibility) problem with variables $w_1, \ldots, w_K$. Hence, we can view the scenario generation problem

as a semi-infinite linear programming (feasibility) problem, with a continuum of nonnegative variables indexed by the elements of $\mathbb{R}^n$ (or Ξ, if we are looking for an interior formula), and with N equality constraints. To motivate our approach, we rewrite this feasibility problem as an optimization problem using the notation $\alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix}$:

$$\min_{w:\Xi\to\mathbb{R},\, \alpha\in\mathbb{R}^N} \left\{ \sum_{i=1}^{N} |\alpha_i| \;\middle|\; \int_{\Xi} w(\xi) u_\xi d\xi + \alpha = m;\; w(\xi) \geq 0 \;\forall\, \xi \in \Xi \right\}, \tag{6}$$

whose optimal objective value is 0 if and only if all the required moments of the original distribution can be matched, that is, if m is indeed a vector of moments.

The objective function $\sum_i |\alpha_i|$ may be replaced by any other norm of α. To keep the exposition simple we will continue using the L_1 norm.

To find the right (finite-dimensional) ξ in (5), we start with a candidate set of nodes $\{\xi_1, \ldots, \xi_\ell\}$, possibly empty, and solve the auxiliary LP

$$\min_{w\in\mathbb{R}^\ell,\, \alpha\in\mathbb{R}^N} \left\{ \sum_{i=1}^{N} |\alpha_i| \;\middle|\; \sum_{k=1}^{\ell} w_k u_{\xi_k} + \alpha = m,\; w \geq 0 \right\}. \tag{7}$$

If the optimal objective function value is 0, with optimal solution $(\alpha^* = 0, w^*)$, then the cubature formula (ξ, w^*) is a positive formula matching all the desired moments, and we are done. Otherwise, we can find a point $\xi_{\ell+1}$ with strictly negative reduced cost, and add it to the candidate node set.

Before discussing strategies to find the next node to add to the formula, we make two observations. (Recall the definition of positive and interior cubature formulas from Definition 1.)

Theorem 10. *For every (not necessarily probability) measure μ there exists a positive interior cubature formula with degree of exactness d with respect to μ on $N(n,d) = \binom{n+d}{d}$ points. More generally, if an arbitrary collection of N moments are to be matched, there exists a positive interior cubature formula on N points that matches those moments.*

Proof. The first part of the assertion is known as Tchakaloff's theorem [30].

To show the more general second claim, we need to invoke the following *conic version* of Carathéodory's theorem [25, Corollary 17.1.2]: every vector that belongs to an N-dimensional convex cone C generated by $\{c_i \colon i \in I\}$ can be expressed as a convex combination of N or fewer vectors c_i.

To apply this result, first note that the cone

$$\left\{ v \in \mathbb{R}^N \;\middle|\; \exists\, \mu \colon v = \int_{\Xi} u_\xi \mu(d\xi) \right\}$$

is an N-dimensional non-empty convex cone generated by the vectors u_ξ, $\xi \in \Xi$. The vector m belongs to this cone, hence there exists a collection U of N points, $U = \{u_{\xi_1}, \ldots, u_{\xi_N}\} \subset \Xi^N$ such that m is a convex combination of points in U. □

Also note that if $\{\xi_1, \ldots, \xi_K\}$ is the node set of some positive interior cubature formula, then the convex polytope

$$\left\{ w \geq 0 \;\middle|\; \sum_{k=1}^{K} w_k u_{\xi_k} = m, \right\}$$

is non-empty, and has a vertex (basic feasible solution), which has at most as many non-zero components as the number of equality constraints, which is N. Hence, an N-node formula can be obtained from *every* K-node formula with $K > N$ by solving a single linear programming problem.

Our next observation is that the ellipsoid method can be used to solve our moment matching semi-infinite linear programming formulation using a polynomial number of iterations.

Theorem 11. *Suppose we are given an oracle that finds a node $\xi_{\ell+1}$ with strictly negative reduced cost, given the nodes $\{\xi_1, \dots, \xi_\ell\}$ and the optimal solution to the corresponding auxiliary linear program (7). Using this oracle, a positive formula matching all required moments with absolute precision ε can be found in oracle-polynomial time; here "polynomial" means polynomial jointly in $\log(1/\varepsilon)$ and N.*

Proof. The dual of the semi-infinite LP (6) is the problem

$$\max_{p \in \mathbb{R}^N} \left\{ m^{\mathrm{T}} p \;\middle|\; u_\xi^{\mathrm{T}} p \le 0 \; \forall \, \xi \in \Xi; \; |p_i| \le 1, \; i = 1, \dots, N \right\}, \tag{8}$$

and the oracle described in the assumptions is a (strong) separation oracle for it.

Since (8) is a convex semi-infinite LP with finitely many variables, all of which have an absolute value bounded above by 1, its feasible set is a finite dimensional, circumscribed, convex body. Hence, (8) can be solved in oracle polynomial time with respect to the separation oracle [11, Corollary 4.2.7], in the weak sense (by finding an ε-optimal solution), using the ellipsoid method.

If the cuts generated by the ellipsoid method correspond to $\xi_1, \dots, \xi_K$, then the obtained ε-optimal solution is an optimal solution of

$$\max_{p \in \mathbb{R}^N} \left\{ m^{\mathrm{T}} p \;\middle|\; u_\xi^{\mathrm{T}} p \le 0 \; \forall \, \xi \in \{\xi_1, \dots, \xi_K\}; \; |p_i| \le 1, \; i = 1, \dots, N \right\},$$

which in turn is the dual linear program of (7) with $\ell = K$.

Since $m \in \operatorname{conv}\{u_\xi \,|\, \xi \in \Xi\}$, the constraints of (8) imply $m^{\mathrm{T}} p \le 0$ for every feasible solution, therefore its optimal solution is also non-positive. As this optimal solution is ε-optimal for the LP (7) with columns $\xi_1, \dots, \xi_{\ell=K}$, this latter LP has an optimal solution with objective value at most ε.

Since K is a polynomial in N, a formula with N nodes selected from $\{\xi_1, \dots, \xi_K\}$ can be obtained in polynomial time using the argument preceding the theorem. □

In our implementation the auxiliary LP (7) is solved using the simplex method. The dual multipliers give vector p in (8). If it is not ε-optimal, the oracle provides the new column $\xi_{\ell+1}$ to be added to (7), after which the primal simplex method can be used to resolve the auxiliary LP starting from the previous dual feasible solution. This is the method we implemented in the numerical tests, and we found that in this implementation the number of generated columns is considerably lower than the bounds that can be obtained from the ellipsoid method argument; we discuss variants of this approach in Section 4.1, and present extensive numerical results in Section 5.1.

4.1. **Column generation oracles.** Recall the dual problem (8) of the semi-infinite LP (6). This problem has a straightforward interpretation: $u_\xi^{\mathrm{T}} p = p_1 + p_2 \xi_1 + \dots$ is the value of the polynomial with coefficient vector p (in the basis u_x) at the point ξ, while $m^{\mathrm{T}} p$ is simply the integral of the same polynomial over Ξ. Hence, this is an optimization problem over the set of polynomials in the linear space spanned by u_x that are non-positive over Ξ. With a slight abuse of notation we can identify the coefficient vector p with the polynomial p itself, and (8) can be equivalently written as follows.

$$\max_{p \in \operatorname{span}\{u_x\}} \left\{ \int_\Xi p(\xi) \mu(d\xi) \;\middle|\; p(\xi) \le 0 \; \forall \, \xi \in \Xi, \; \|p\|_\infty \le 1 \right\}, \tag{9}$$

where $\|p\|_\infty$ denotes the L_∞ norm of the coefficient vector of p in the basis $\{u_x\}$ (not of the polynomial p!). For example, if all moments up to order d are matched, the optimization is over the set $\operatorname{span}\{u_x\} = \mathbb{R}[x]_d$.

Similarly, the dual LP of the auxiliary problem (7) can be equivalently written as

$$\max_{p\in\operatorname{span}\{u_x\}} \left\{ \int_\Xi p(\xi)\mu(d\xi) \;\middle|\; p(\xi_k) \le 0 \;\forall\, k = 1,\dots,\ell,\ \|p\|_\infty \le 1 \right\}. \tag{10}$$

Thus, scenario generation with the column generation approach outlined at the end of the previous section can be summarized as Algorithm 1.

Algorithm 1: Moment matching scenario generation with column generation

```
   parameter: ε > 0
 1 initialize ξ = {x_1, ..., ξ_ℓ}                          /* arbitrary */
 2 repeat
 3     solve the primal-dual pair (7)-(10) for w*, α* and p*
 4     if Σ_i |α*_i| < ε then
 5         return (ξ, w*)
 6     else
 7         find a ξ_{ℓ+1} ∈ Ξ satisfying p*(ξ_{ℓ+1}) > 0     /* oracle */
 8         ξ ← ξ ∪ {ξ_{ℓ+1}}
 9     end if
10 until false
```

Ideally $\varepsilon = 0$, in practice we shall use a small positive value; see Section 5.8 for details. Elaborating on the step in line 7: if the optimal objective function of (7) is non-zero, then the dual optimal solution p^* cannot be feasible for (9), and the column generation oracle must find a violated dual inequality, i.e., a $\xi_{\ell+1}$ satisfying $p^*(\xi_{\ell+1}) > 0$.

Thus, the task of the column generation oracle is to *find a point $\xi_{\ell+1} \in \Xi$ at which the dual optimal polynomial p^* assumes a positive value.* There are several ways to implement such an oracle, we discuss a number of them next.

4.1.1. *Global polynomial optimization.* The most violated inequality can be obtained by finding the global maximum of p^*. However, this is an NP-hard problem in general even for multilinear quadratic polynomials p^*, even for the simplest convex polyhedral domains such as the unit cube $\Xi = [0,1]^n$ or the unit simplex $\Xi = \{x \in \mathbb{R}^n \,|\, x \ge 0, \sum x_i = 1\}$. Hence, it is expected that this approach may only work for small problems.

Virtually the only known case when the global optimizer can be found in polynomial time is the well-known case when the degree of p^* is $d = 2$ and Ξ is an ellipsoid.

Theorem 12. *If the integration domain Ξ is an n-dimensional ellipsoid, a cubature formula matching all moments up to order two can be found in time polynomial in the dimension n.*

Proof. It is well known that quadratic optimization over a full-dimensional ellipsoid can be solved in polynomial time; this is equivalent to the trust-region subproblem of trust-region methods [22, Theorem 4.1]. This yields a polynomial time column generation oracle for the column generation algorithm when Ξ is an ellipsoid and only moments up to order two are matched. □

For polynomials of higher degree, relaxations, convex approximations, or non-convex global optimization techniques may be used. We survey a number of possible directions.

Sum-of-squares relaxations. The most popular global polynomial optimization methods are based on sum-of-squares relaxations: they rely on the observation that the maximum of a polynomial p over a domain Ξ is the smallest number c such that $c - p$ is nonnegative over Ξ. Although the membership problem of the cone of nonnegative polynomials over Ξ is NP-hard even for the

simplest domains, such as $\Xi = \mathbb{R}^n$ and $\Xi = [0,1]^n$, these cones can be approximated from within by cones of sum-of-squares polynomials, which are semidefinite representable.

More precisely, suppose $\Xi = \{\xi \in \mathbb{R}^n \,|\, p_j(\xi) \geq 0, j = 1, \ldots, J\}$ for some polynomials $p_1, \ldots, p_J$ (allowing $J = 0$ for $\Xi = \mathbb{R}^n$). Then the cone of polynomials

$$\Sigma_d \stackrel{\text{def}}{=} \left\{ \sum_{j=1}^{J} p_j \sum_{i \in I_j} q_i^2 \;\middle|\; I_j \text{ finite } \forall\, j,\ q_i \in \mathbb{R}[x]_d \;\forall\, i \right\}$$

is a finite dimensional convex cone for every d, and it is a (usually proper) subset of polynomials nonnegative over Ξ. Cones of this form are called (weighted) *sum-of-squares* polynomial cones.

For every $p_1, \ldots, p_J$ and every $d \geq 0$, Σ_d is semidefinite representable [20], therefore the problem $\max\{c \,|\, c - p \in \Sigma_d\}$ can be solved by solving a semidefinite program. This leads to a hierarchy of semidefinite programs that provide better and better approximation of the true value of the maximum of a give polynomial p: if $c_d - p \in \Sigma_d$ for some constant c_d. then $p \leq c$. Under certain conditions on the polynomials p_j, the sequence $\{c_d\}_{d=1,\ldots}$ of best upper bounds c_d converges to the global maximum of p, for certain problems the convergence is also known to be finite [18].

This approach has been successful in a number of applications, including combinatorial optimization. However, the transformation this approach entails eliminates the arguments of the polynomial from the decision variables, hence, it gives no direct access to the maximizer, only to the (approximate) value of the maximum. Under suitable conditions the maximizer can also be recovered from the optimal solution of the semidefinite relaxations, but in our subproblems we found that these conditions virtually never held. In our experiments we used two implementations of this approach, Gloptipoly by Lasserre *et al.* [15], which is a dual approach (using moment relaxations), and SparsePOP by Kojima *et al.* [31], a primal approach using sum-of-squares polynomials. SparsePOP has yielded some optimal solutions for small subproblems with little running time (see also Section 5.1), but Gloptipoly has not performed well. Hence so far this approach has been insufficient for our purposes; we hope that this application will motivate further research in this area; our subproblems may be useful benchmark problems for global polynomial optimization.

Global non-convex optimization. Since for small problems, where exact global polynomial optimization was possible with SparsePOP, global optimization has not yielded better results than some simpler methods to be discussed below, we have not pursued the use of other global optimization techniques. We did experiment, however, with a simple multi-start local optimization approach, which was capable of generating columns fast enough to be practical when formulas with up to 2000 points were generated; see Section 5.1 for details.

4.1.2. *Approximate polynomial optimization.* Rather than finding the most violated inequality, one can aim at finding a substantially violated inequality using approximation algorithms for polynomial optimization. The following theorem summarizes those special cases of polynomial optimization problems that are known to be approximable in polynomial time.

Theorem 13 ([32, 5])**.** *Let p be an n-variate polynomial of total degree d, and consider the problem $\max\{p(\xi) \,|\, \xi \in \Xi\}$. This problem admits an $4/7$-approximation algorithm if $d = 2$, and Ξ is the unit cube. Moreover, the same problem admits a polynomial time approximation scheme (PTAS) for every fixed degree d if Ξ is the unit simplex.*

Several negative (inapproximability) results are also known for higher degree polynomials over the unit cube and the sphere, see, e.g., [5] for details.

4.1.3. *Monte Carlo column generation.* A violated inequality can be found by choosing *random* points $\xi \in \Xi$, and testing whether $p(\xi) > 0$. If resources permit, multiple such points can be found, and the one with the most positive value can be added to the node set. The intuition behind this heuristic is that p is a polynomial whose integral is known to be positive, at the same time its values are *a priori* bounded from above. Hence, the measure of points at which it assumes a positive value cannot be very small, especially in the early iterations, when the integral of p is large. More precisely, we have the following observation.

Lemma 14. *Let p^* be the optimal solution of* (10) *with objective function value $I = \int_\Xi p(\xi)\mu(d\xi) > 0$. Let B be an upper bound on the maximum of every polynomial satisfying the constraints of* (10)*. (Such a bound can be easily obtained from, say, the constraint $\|p\|_\infty \le 1$.) Take any non-negative x for which $x \le I$, and draw random points from Ξ with the distribution determined by μ. Then the expected number of points ξ needed to be drawn until one that satisfies $p^*(\xi) \ge x$ is found is at most $(B - x)/(I - x)$.*

Proof. As $\mu(\Xi) = 1$, we have $I \le B$, and hence, $0 \le x \le I \le B$. Let $L_x \stackrel{\text{def}}{=} \{\xi \in \Xi \,|\, p^*(\xi) \ge x\}$ be the upper level set of p^* corresponding to x. Bounding the value of the integral from above separately on L_x and its complement, we have

$$I = \int_{L_x} p(\xi)\mu(d\xi) + \int_{\bar{L}_x} p(\xi)\mu(d\xi) \le B\mu(L_x) + x(1 - \mu(L_x)),$$

implying that the expected number of trials until we find a point that belongs to L_x is $1/\mu(L_x) \le (B - x)/(I - x)$ if $0 \le x < I$ and $x < B$. □

If the stopping condition of the column generation algorithm is $I < \varepsilon$ for some $\varepsilon > 0$, this lemma provides an explicit upper bound on the number of samples per iteration, if we choose, say, $x = cI$ for a fixed $0 < c < 1$ in every iteration.

Another advantage of this approach is that it is easily adaptable to arbitrary domains Ξ over which global polynomial optimization may be difficult. This method only requires that we are able to generate random samples from the distribution determined by μ.

In the rest of the paper we refer to this variant of our scenario generation method as the *CG-MC* (column generation with Monte Carlo) method.

4.1.4. *Quasi-Monte Carlo column generation.* As an alternative to the previous algorithm, a low-discrepancy sequence (or "quasi-random" points), such as a Sobol sequence or Halton sequence, can be used in place of randomly generated points to find a point ξ satisfying $p(\xi) > 0$.

If the underlying distribution is not uniform, a transformed set of quasi-random points can be used, if an appropriate change of variables, or more precisely, a diffeomorphism $g\colon [0,1]^n \mapsto \Xi$ satisfying $\int_\Xi f(\cdot,\xi)\mu(d\xi) = \int_{[0,1]^n} f(\cdot, g(\omega))d\omega$, is available. If such a transformation is not available, but we can sample from the distribution corresponding to μ, then CG-MC must be used in place of CG-QMC.

We do not have a rigorous bound on the number of points needed to be examined until we find a suitable one. The heuristic justification is that the quasi-random points fill Ξ rather uniformly, and they avoid clustering better than uniform (pseudo-)random points. Hence, quasi-random points avoid visiting points that are close to previously visited points (where p^* is known to take only negative values) better than true random points.

In the rest of the paper we refer to this variant of our scenario generation method as the *CG-QMC* (column generation with quasi-Monte Carlo) method.

5. Numerical examples

We first assess the practical iteration complexity of the column generation methods in Section 5.1. Then we compare the moment matching formulas obtained by the CG-MC and CG-QMC methods to Monte Carlo and quasi-Monte Carlo integration on a standard test set of numerical integration problems, with uniform μ, in Section 5.2. We chose the test functions from a standard battery of test functions.

In Section 5.3 we consider problems with various non-uniform distributions μ. In Section 5.3.1 we repeat the same set of tests as in Section 5.2 after changing the underlying distribution from uniform to multinormal. These two distributions are perhaps the most important. This test allows us to test the hypothesis that the shape of the underlying distribution does not have a dramatic effect on the performance of the moment matching formulas. We carry out a battery of tests for a wide variety of distributions, including unimodal, bimodal, and U-shaped distributions, in Section 5.3.2.

In Section 5.4 we evaluate the efficacy of selective moment matching. We consider high-dimensional integration problems where matching all moments up to any order higher than two would be impossible, but the integrands are functions with a sparse structure. We use moment matching formulas that match only a subset of all moments up to order 7, selected to best suit the integrands at hand. Again, we compare the results to Monte Carlo and quasi-Monte Carlo integration.

We move on to solving optimization problems in Section 5.5. We consider high-dimensional stochastic convex optimization problems where the objective function to be optimized is expressed as an integral. In Section 5.6 we solve a stochastic programs from portfolio optimization models in the literature.

All algorithms were implemented using Matlab 2011a (version 7.12); the linear programs were solved using CPLEX (version 12).

5.1. **Effect of oracles on formula generation.**

The time complexity of the column generation approach is driven by the number of columns that need to be generated until a formula is found. (After adding a new column, the primal simplex method can be used to warmstart the reoptimization of the auxiliary LP, which is relatively fast, see below.)

Algorithm 1 needs to be initialized with a set of nodes (or columns). In CG-QMC the columns to be tested is completely determined, as we consider the columns corresponding to a fixed, deterministic low-discrepancy sequence. Hence, the initial set of columns can be empty. In our implementation of the CG-MC variant we initialized the method with a singleton set containing the midpoint of Ξ. (It can be shown that if all moments up to a given degree are to be matched, and the Legendre basis is used to represent the polynomials, then using the empty set as the initial set the dual optimal polynomial is the constant 1. That is, in this case we have no guidance in choosing the first column.)

Since the auxiliary linear program (7) is a fully dense LP, and it becomes large as the number of columns increases, the bottleneck of the algorithm becomes the iterative resolving of this linear program. With the current linear programming software available it does not seem possible to generate formulas with higher than a few thousand (perhaps 10,000) nodes, as this would involve solving a series of LPs with up to tens of thousands of variables and constraints with a dense constraint matrix. In our experiments we used formulas having not more than 2000 nodes; these could be generated without difficulty. Numerical problems, in particular, ill-conditioning in the LP (7) may arise during the generation of formulas that match high-order moments, especially if we work with polynomials of high degree represented in the monomial basis. In our computations we represented polynomials in the Legendre polynomial basis, which has better numerical properties than the monomial basis.

While generating all cubature formulas up to 2000 nodes with dimension up to 10, and various 100-dimensional formulas for the experiments of Section 5.4 we have not encountered any instances where the CG-MC and CG-QMC methods required more than $1.3N$ iterations (that is, generated columns) to find a formula matching the required N moments; in fact, in almost every case the number of iterations was under $1.2N$.

We also attempted to find formulas by solving the column generation subproblems to optimality using exact global polynomial optimization. In these experiments we used the Gloptipoly and SparsePOP packages [15, 31]. Except for problems of small dimension and low degree, the running times of the exact polynomial optimization was prohibitive to be really useful, and in many instances the solvers failed to return a solution. As a result, we believe that perhaps some of these subproblems could be good benchmark problems for polynomial optimization software.

A comparison of CG-MC, CG-QMC, global optimization, and multi-start local optimization in the generation of 3-dimensional formulas with degree of exactness 5 is given in Table 1. This was the largest formula where global optimization using SparsePOP was possible. The multi-start local optimization heuristic was to find local optima of the polynomial optimization problem from random starting points until a suitable column is found. In our experiments we generated the moment matching formulas obtained with the exact solution of the column generation subproblems for dimensions 2 and 3, matching all moments up to order 5. In all of these cases the formulas obtained with global or local optimization had no fewer points than those obtained using CG-MC and CG-QMC, and the number of iterations for the methods with global and local optimization was also approximately the same as the number of iterations of CG-MC and CG-QMC; Table 1 shows an example. The last two rows in the table show the total time spent on the column generation steps and the time spent on the one column generation step that took the longest. Clearly, some of the global optimization steps take an excessive amount of time, resulting in an overall running time several orders of magnitude larger than the running time of CG-MC and CG-QMC (and even multistart local optimization). These results suggest that the effort necessary to solve the subproblems to optimality may not be worthwhile when both the degree and the dimension are small, while global optimization is impossible with the existing algorithms if either of these parameters is large.

Table 2 shows how CG-MC and CG-QMC compare to column generation with multistart local optimization (CG-LO) as the dimension increases. CG-MC and CG-QMC are nearly indistinguishable, but CG-LO shows two notable differences: on one hand, the total running time of CG-LO is considerably higher, owing to the excessive amount of time spent on the column generation steps. On the other hand, the number of columns generated (and, consequently, the time spent on simplex iterations) is somewhat smaller than for CG-MC and CG-QMC. This suggests that using optimization to find good columns may be worthwhile, but only if the optimization steps could be accelerated substantially. On the other hand, even random sampling does very well in generating good columns, and it does not generate too many columns that are unused at the end. (Recall that the number of columns ultimately used is the same as the number of moments matched.)

Because of the large difference in efficiency of the column generation oracles, in the numerical experiments of the following sections we used only CG-MC and CG-QMC formulas, and compared these to Monte Carlo and quasi-Monte Carlo sampling.

5.2. **Integration problems.** We follow the principles and recommendations laid out in Genz's paper [10], while adapting his popular experimental setup to the stochastic programming applications. We consider the following families of test functions, defined over the domain $[0,1]^n$:

$$\text{Product peak:} \qquad f_1(x) = \prod_{i=1}^{n} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1}$$

	Global opt.	CG-MC	CG-QMC	Local opt.
# points in formula	56	56	56	56
# iterations	62	63	64	67
# simplex iterations	776	642	636	765
total simplex time (sec.)	0.031	0.032	0.078	0.063
total col. gen. time (sec.)	108.77	0.121	0.168	3.923
max. col. gen. time (sec.)	63.05	0.004	0.004	0.079

TABLE 1. Comparison of the iteration and time complexity of various column generation heuristics when finding 3-dimensional formulas matching all moments up to order 5. Global optimization was not possible while seeking larger formulas.

n	K	# iterations			# simplex iterations			simplex time (sec.)			total time (sec.)		
		MC	QMC	LO	MC	QMC	LO	MC	QMC	LO	MC	QMC	LO
10	286	320	318	288	16630	17164	20762	6	6	4	2	4	129
12	455	512	505	465	45201	41994	56048	32	35	28	5	6	399
14	680	773	791	692	103240	98852	139828	189	201	153	14	16	1160
16	969	1104	1117	991	201766	202762	314921	871	896	723	31	40	3148
18	1330	1566	1547	1372	398926	394612	664315	4009	3669	2956	77	68	8386
20	1771	2117	2095	1794	761655	724454	2473023	11052	10307	15579	147	148	49411

TABLE 2. Comparison of the iteration and time complexity of three column generation heuristics when finding n-dimensional formulas ($n \leq 20$) matching all moments up to order 3. Global optimization in the column generation was impossible. Local optimization (LO) yields fewer column generation iterations, but drawing random columns (MC) or quasi-random columns (QMC) is considerably more efficient in terms of the total running time.

Corner peak: $$f_2(x) = \Big(1 + \sum_{i=1}^{n} a_1 z_i\Big)^{-(n+1)}, \text{ with } z_i = \begin{cases} x_i & \text{if } u_i < 1/2 \\ 1 - x_i & \text{if } u_i \geq 1/2 \end{cases}$$

Gaussian: $$f_3(x) = \exp\Big(-\sum_{i=1}^{n} a_i^2 (x_i - u_i)^2\Big)$$

Piecewise linear: $$f_4(x) = \sum_{i=1}^{n} a_i |x_i - u_i|$$

Discontinuous: $$f_5(x) = \begin{cases} 0 & \text{if } x_1 > u_1 \text{ or } x_2 > u_2 \\ \exp\big(-\sum_{i=1}^{n} a_i x_i\big) & \text{otherwise} \end{cases}$$

Each family is named after the particular property shared by its members. The exact values of the integrals are easily found:

$$\int_{[0,1]^n} f_1(x)\, dx = \prod_{i=1}^{n} a_i \left(\tan^{-1}(a_i u_i) + \tan^{-1}(a_i(1 - u_i))\right),$$

i	1	2	3	4	5
B_i	200	100	10	1000	50
d_i	2	3	1	2	2

TABLE 3. Test parameters used in Section 5.2.

$$\int_{[0,1]^n} f_2(x)\,dx = \prod_{i=1}^{n}(1+a_1 i)^{-1},$$
$$\int_{[0,1]^n} f_3(x)\,dx = \prod_{i=1}^{n} \tfrac{\sqrt{\pi}(\operatorname{erfi}(a_i u_i)+\operatorname{erfi}(a_i(1-u_i)))}{2a_i},$$
$$\int_{[0,1]^n} f_4(x)\,dx = \sum_{i=1}^{n} a_i\left(u_i^2 - u_i + 1/2\right),$$
$$\int_{[0,1]^n} f_5(x)\,dx = \prod_{i=1}^{2}(1-\exp(-a_i u_i)/a_i \prod_{i=3}^{n}(1-\exp(-a_i)/a_i.$$

The functions f_1, f_3, and f_5 are from [10], the remaining two are also very similar to two of Genz's benchmark families. Some changes were made to further reduce bias introduced by the chosen families, to include a piecewise linear function, and to avoid having integrals very close to zero, which makes the relative errors overly sensitive to very small changes in the value of the integrand.

Each family has two distinct sets of parameters. The *unaffective* parameters u_i do not affect the qualitative properties of the functions or the difficulty of numerical integration as long as $0 < u_i < 1$. We draw these parameters independently and uniformly from $[0, 1]$ in order to lower the bias introduced in the experiments by the choice of test functions. On the other hand, the difficulty of integration increases as the *affective* parameters $a_i > 0$ increase. To keep the results of different experiments comparable, we draw each a_i randomly, and then scale a so that $\sum_i |a_i| = b$ for some fixed b depending on the function f_i and the dimension n. It is difficult to rigorously justify any particular choice of b. We chose them keeping two observations in mind: (1) the integrals of the functions must be sufficiently non-zero in order to keep the computation of relative errors stable, and (2) with the optimization applications in mind, the cubature formulas to be tested have fewer nodes than what is customary in the numerical analysis literature, meaning that the test functions need to be "easier" than usual—otherwise all formulas will fare very badly, which renders the tests meaningless. Genz recommends computing b using the formula $b = B_i/n^{d_i}$ for the test function family f_i and dimension n, for some the constants B_i and d_i. We adopted this approach, our constants are shown in Table 3. These are similar to those of Genz, except that the B_i are smaller, following observation (2) above.

We compared the two most popular and widely used scenario generation methods, Monte Carlo sampling (MC) and quasi-Monte Carlo integration (QMC) to two variants of the column generation (CG) method: the one using uniform random sample points in the column generation step (CG-MC), and the one using Sobol points (CG-QMC).

The number of nodes in the MC and QMC methods were chosen to be equal to the number of points in the CG methods, which is $N(d, n) = \binom{n+d}{d}$ for n-dimensional problems with degree of exactness d.

Each experiment (with a given f_i and n) was carried out $S = 200$ number of times; the measure of error is the relative error E_{rel}. The statistics used in the tests are the median relative error

and the m-th smallest and largest values (L_m and U_m) of E_{rel}; here $m = m(S)$ is the smallest number satisfying

$$C \geq 1 - 2^{-(S-1)} \sum_{k=0}^{m-1} \binom{S}{k}, \quad C = 0.95;$$

this way the true sample median lies in $[L_m, U_m]$ with confidence C.

The computations were carried out using Matlab 2011a. For the QMC integration we used the Sobol and the Halton low-discrepancy sequences as implemented by Matlab's `sobolset` and `haltonset` commands. There was no qualitative difference between the two (meaning, precisely that whenever one of them performed significantly better or worse than MC or moment matching, then so did the other). The results on Fig. 1 show the Sobol set.

To assess the performance of the different methods as the number of scenarios increases, we fixed $n = 4$ and increased the degree of exactness of the moment matching formulas from 2 to 10. The median relative errors and their 95% confidence intervals are plotted on Fig. 1 for each family. The numerical results in tabular form are shown in the Appendix (Table 9).

The tests are rather conclusive, and they support strongly the efficacy of the CG-MC and CG-QMC formulas for each continuous function family. The MC method fell significantly behind all of the other methods in every example. The QMC method performed considerably better, but there is a clear difference between the rate of convergence of the QMC and the two moment matching methods. Once their degree of exactness is at least 4, the moment matching formulas significantly outperform both Monte Carlo and quasi-Monte Carlo integration with the same number of points for each of the continuous families. The difference is most notable for the Gaussian family, which contains the smoothest functions.

In the experiments with the discontinuous family each method had a poor rate of convergence, with QMC being the most accurate of the five. This is expected, since on one hand QMC does not exploit the smoothness of the integrand, and on the other hand the idea of moment matching is based on the approximability of the integrands by polynomials of low degree. Based on the results we do not recommend using moment matching formulas for discontinuous integrands.

The two moment matching formulas yielded nearly identical results in all the experiments.

5.3. **Alternate distributions.** We now consider examples where the underlying probability distribution is different from uniform. In the optimization examples we will also have problems with normal and lognormal density. In the rest of this section we carry out a more comprehensive set of experiments to test whether the efficacy of the CG integration formulas changes with the change of distributions. In Section 5.3.1 we concentrate on the normal distribution, as it has prime importance in applications. In Section 5.3.2 we experiment with distributions of a wide variety of shapes, with the help of the logit-normal distribution family.

5.3.1. *Multinormal distribution.* We repeated the same experiments as in Section 5.2 after changing the distribution from uniform to multivariate normal. We chose a normal distribution $N(\mu, \Sigma)$ with an arbitrary mean vector μ and covariance matrix Σ, with only one condition in mind: since the test functions are all defined on the unit cube $[0,1]^n$, we chose a distribution whose mass is largely concentrated on the unit cube. This way it is irrelevant how the test functions are extended outside $[0,1]^n$, and the qualitative properties of the test functions are unaffected by such an extension. (We simply defined each function to be zero outside the unit cube.) With this condition in mind we chose the following parameters:

$$\mu = \begin{pmatrix} .4 \\ .4 \\ .4 \\ .4 \end{pmatrix}, \qquad \Sigma = \frac{1}{300} \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

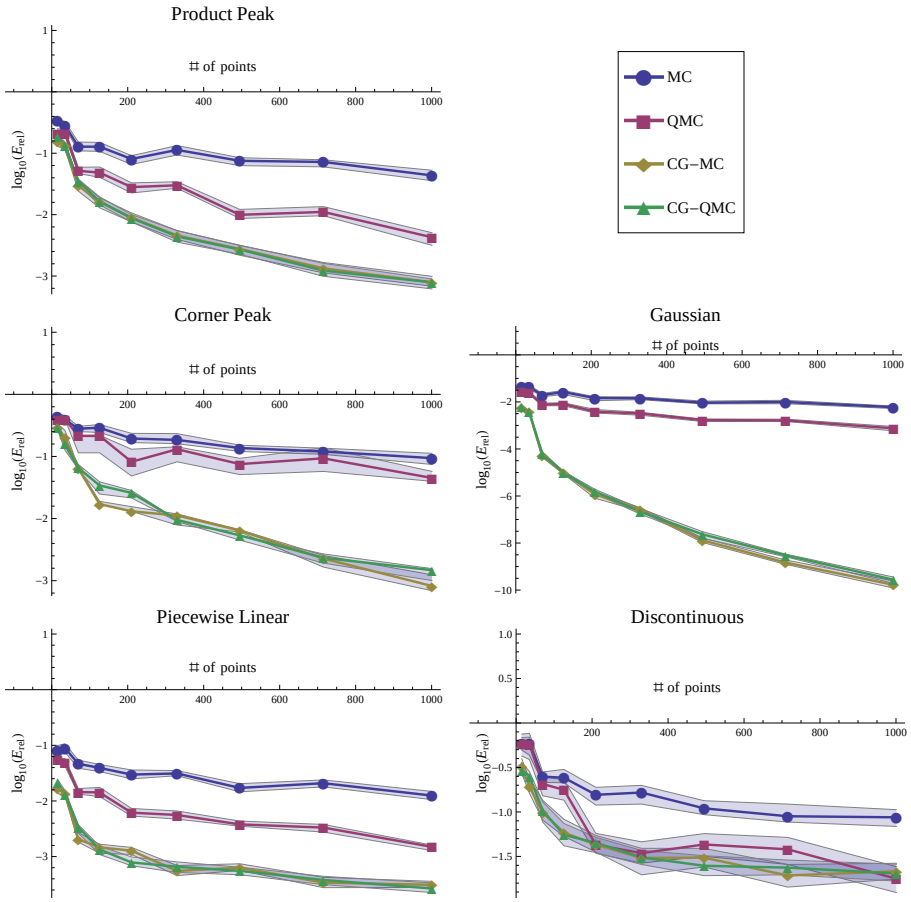

FIGURE 1. Performance of four cubature formulas using the four-dimensional parametric families f_1–f_5 with uniform density. Horizontal axis: number of points used in the formulas; the points correspond to increasing degree of exactness of the CG-MC and CG-QMC formulas from 2 to 10. Vertical axis: base-10 logarithm of the median relative errors from 200 experiments. The gray shaded bands around the median relative errors are 0.95-level confidence intervals around the median. Note the differences on the vertical axes, and that on some of the figures the CG-MC and CG-QMC results are practically indistinguishable.

The experimental setup was identical to that of Section 5.2, we do not repeat the details. The only difference is that with the change in the distribution the true values of the integrals are no longer available in closed form; they need to be numerically approximated using a large-scale formula. The results are summarized in Fig. 2; and in tabular form in the Appendix (Table 10).

The results of the Discontinuous family are inconclusive again; all methods performed badly for these problems. In the other families moment matching outperformed Monte Carlo and quasi-Monte Carlo integration whenever there were enough points to match all moments of order up to 4 or higher. The advantage of CG-MC and CG-QMC appears to be smaller than in the uniform example, but the difference is still significant.

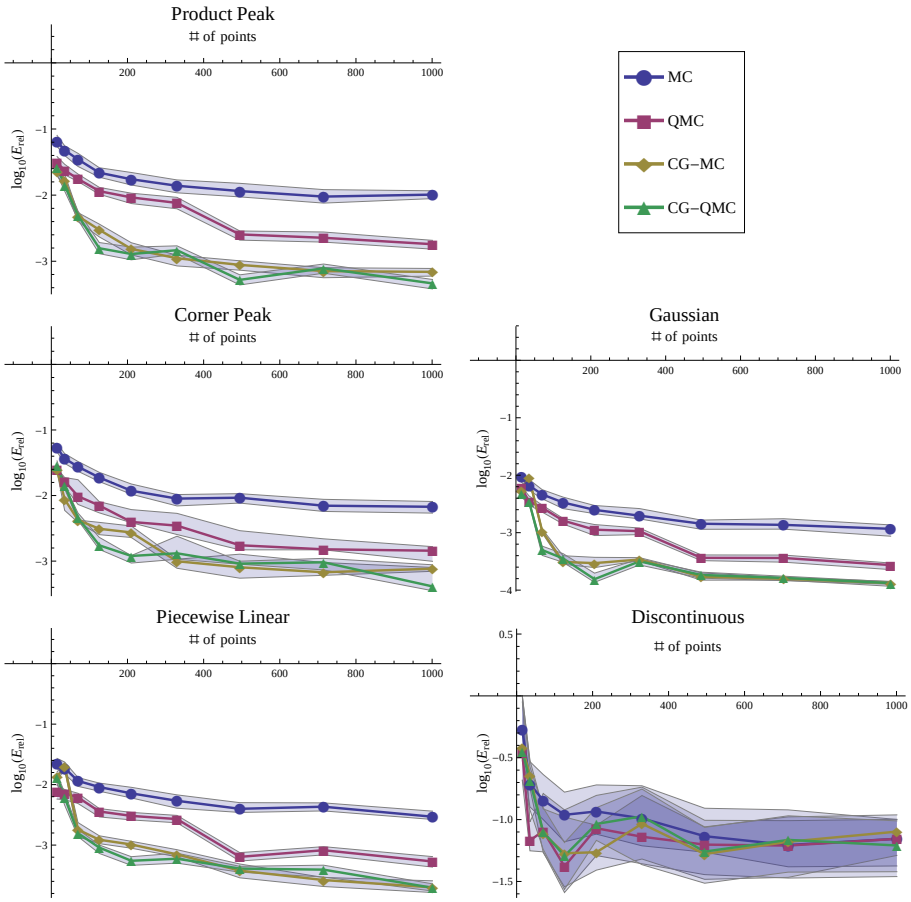

FIGURE 2. Performance of four cubature formulas using the four-dimensional parametric families f_1–f_5 with normal density. Horizontal axis: number of points used in the formulas; the points correspond to increasing degree of exactness of the CG-MC and CG-QMC formulas from 2 to 10. Vertical axis: base-10 logarithm of the median relative errors from 200 experiments. The gray shaded bands around the median relative errors are 0.95-level confidence intervals on the median.

5.3.2. *Distributions with various shapes.* We adapted the experimental setup of Section 5.2 and compared the CG methods to MC and QMC integration in problems where the integration is with respect to probability measures of various shapes.

We used random density functions from the multivariate logit-normal distribution family. Let $\mu \in \mathbb{R}^n$ be arbitrary n-vector, and Σ be an $n \times n$ symmetric positive definite matrix. Then the *logit-normal distribution* $LN(\mu, \Sigma)$ is the one obtained by applying coordinatewise the *logistic function* $t \mapsto (1 + \exp(-t))^{-1}$ on the multivariate normal distribution with mean vector μ and covariance matrix Σ. Logit-normal distributions are supported on the unit hupercube, and take a wide variety of shapes (Fig. 3), including convex and non-convex unimodal, symmetric and asymmetric U-shaped, bimodal, and skewed unimodal shapes, hence are ideal for our experiments.

We created random four-dimensional logit-normal distributions $LN(\mu, \Sigma)$ by drawing random vectors uniformly from $\{0, 1\}^4$ for μ, and drawing random covariance matrices Σ with diagonal

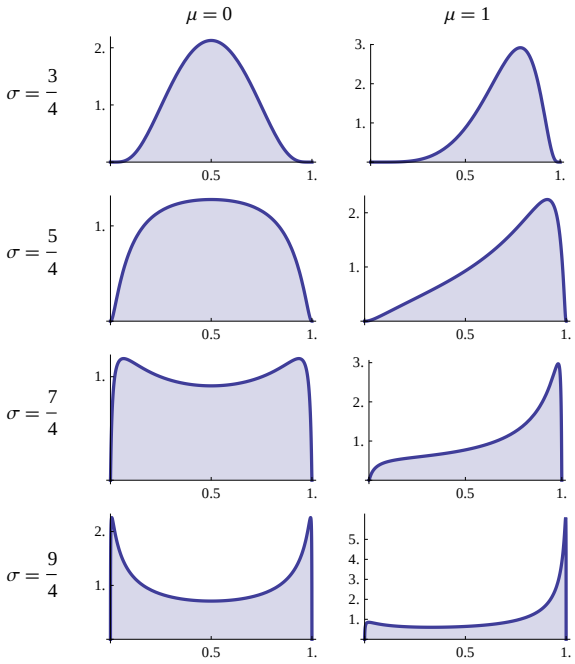

FIGURE 3. Shapes of the univariate logit-normal distribution: the probability density functions of $LN(\mu, \sigma^2)$ are depicted for $\mu \in \{0, 1\}$ and $\sigma \in \{3/4, 5/4, 7/4, 9/4\}$.

entries in $\{3/4^2, 5/4^2, 7/4^2, 9/4^2\}$. These parameters correspond to those shown on Fig. 3. For each generated distribution we repeated a simplified version of the experiments of Section 5.2.

With the change in the distribution the true values of the integrals are no longer available in closed form; they need to be numerically approximated using a large-scale formula. Additionally, the number of four-dimensional shapes is also large. Hence, the number of computed integrals had to be lowered substantially for the experiments to run in reasonable time. Because of this we considered only the first (Product Peak) function family. We experimented with 5000 random distributions, and integrated against each of them 100 random functions from the Product Peak family using both variants of the CG method, and the MC and QMC methods. The degree of exactness of the CG methods were varied between 3 and 10, and for each degree we compared the corresponding CG-MC and CG-QMC formulas to the MC and QMC formulas of the same size, in order to compare the performance of the methods as the number of points increases. The total CPU time used for the experiments exceeded 400 hours, largely due to the time required to satisfactorily estimate the true values of the integrals. For each of the 5000 distributions the median relative errors from the 100 experiments were computed, along with a 0.95-level confidence interval on the median along the same lines as in Section 5.2. Table 4 shows the summary statistics of the experiments; the acronyms in the table are explained next:

(1) We say that in the experiments with a given distribution CG performed *significantly better starting at degree d* (abbreviated "SB@*d*") if the median relative errors of *both* variants of the CG method were significantly lower (at confidence level 0.95) than the median relative errors of the MC and QMC estimates of the integral whenever the degree of exactness of the CG formulas were d or higher. This, with a low degree d, is the best possible result we could have, as it means that once a certain degree of exactness can

be reached by the CG method, it continues to outperform both Monte Carlo and quasi-Monte Carlo formulas of the same size.

(2) A weaker conclusion is reached when the median relative errors of both CG methods are lower than the median relative errors of both MC and QMC for every degree d and higher, but the difference is not always significant at level 0.95. In these experiments we say that CG performed *better, but not significantly* than MC and QMC; abbreviated "BNS@d".

(3) "SW", for *significantly worse*, denotes the cases that do not belong to any of the above categories, and where for at least one degree $d \geq 4$, at least one variant of the CG method had a significantly higher median relative error than either MC or QMC. Note that we require much less from MC and QMC to declare them "better" than what we require from our method.

(4) "WNS", for *worse, but not significantly*, denotes all the yet uncovered cases where for at least one degree $d \geq 4$, at least one variant of the CG method had a higher median relative error than either MC or QMC, although the difference was not significant.

(5) "IC", for *inconclusive* denotes the remaining cases.

outcome type	# occurrences
SB@4	2893
SB@5	916
SB@6	549
BNS@4	508
BNS@5	66
BNS@6	48
SW	0
WNS	3
IC	17

TABLE 4. The distribution of outcomes from the numerical integration experiments with 5000 different logit-normal distributions. The acronyms denoting the various outcomes are explained in the text of Section 5.3.2.

Raw data from the experiments are available from the authors. The numbers in Table 4 clearly show that for nearly all distributions the CG methods outperformed both MC and QMC, in approximately 87% of the distributions the advantage of CG was significant. There were no examples when MC or QMC outperformed the column generation formulas. CG-MC and CG-QMC appears to outperform MC and QMC integration for a wide variety of shapes as long as there are enough points in the formulas to match all moments up to order 4.

5.4. **Selective moment matching.** For high-dimensional problems all moments up to a high order cannot be matched with as few scenarios as in the experiments above, but this may not even be necessary. We considered the following "sparse" variants of the Genz test functions, which do not contain products of univariate functions of non-adjacent variables.

$$\text{Product peak:} \qquad g_1(x) = \sum_{i=1}^{n-1} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1} \left(a_{i+1}^{-2} + (x_{i+1} - u_{i+1})^2\right)^{-1}$$

Corner peak: $g_2(x) = \sum_{i=1}^{n-1} \Big(1 + a_i z_i + a_{i+1} z_{i+1}\Big)^{-3}$, with $z_i = \begin{cases} x_i & \text{if } u_i < 1/2 \\ 1 - x_i & \text{if } u_i \geq 1/2 \end{cases}$

Gaussian: $g_3(x) = \sum_{i=1}^{n-1} \exp\Big(- a_i^2 (x_i - u_i)^2 - a_{i+1}^2 (x_{i+1} - u_{i+1})^2 \Big)$

Piecewise linear: $g_4(x) = \sum_{i=1}^{n} a_i |x_i - u_i|$

Discontinuous: $g_5(x) = \begin{cases} 0 & \text{if } x_1 > u_1 \text{ or } x_2 > u_2 \\ \sum_{i=1}^{n-1} \exp\big(- a_i x_i - a_{i+1} x_{i+1} \big) & \text{otherwise} \end{cases}$

In this subsection we concentrate on the integration of these functions; see the next subsection for actual optimization models involving one of these integrals.

Note that since f_4 is already a sum of univariate functions, g_4 was chosen to be the same as f_4. We ran the same experiments using each of the above g_i as in the previous section, but this time with $n = 100$. For every given degree d, $2 \leq d \leq 6$, we used the CG-MC and CG-QMC methods to generate cubature formulas that match all the moments corresponding to monomials of the form $x_i^a x_{i+1}^b$ with $a + b \leq d$; with a slight abuse of terms in this subsection we call d the degree of exactness of these formulas, although only a subset of the moments of order d and lower are matched. Otherwise the experimental setup was identical to those in the previous experiments. The results are shown on Fig. 4. The qualitative differences between the Monte Carlo, quasi-Monte Carlo, and column generation approaches appears to be same in each function family for dimension 100 as it was for the four-dimensional problems: CG-MC and CG-QMC are comparable across all the experiments, and they both outperform QMC, which in turn outperforms the MC integration in all the continuous families. QMC outperforms CG for the Discontinuous family. The largest difference is measured for the Gaussian family.

5.5. **Convex optimization problems.** The functions g_i from the previous section may also be used as benchmark problems for stochastic optimization, or equivalently, of optimization of integrals. With the affective parameters a_j fixed, the unaffective parameters u_i can be treated as decision variables; the problem then is to find

$$\min_{u \in [0,1]^n} \int_{[0,1]^n} g_i(u, x) dx \quad \text{or} \quad \max_{u \in [0,1]^n} \int_{[0,1]^n} g_i(u, x) dx. \tag{11}$$

Note that whether the resulting problems are convex depends on both the function g_i used as well as on the values of the affective parameters. The problems involving g_2 and g_5 are particularly interesting (and difficult), as they are stochastic integer programming problems. To keep things simple, we considered below the problem involving g_3, whose maximization is a concave maximization problem for every value of the affective parameters a_j, with an easily identifiable global maximum.

The experimental setup was identical to that in the previous section. We considered $n = 100$; the values of the affective parameters and the number of experiments were unchanged. The cubature formulas were used as approximators of the integrand in Matlab's SQP optimization routine. The relative error reported for the Monte Carlo method is the average relative error from 100 repeated experiments, the quasi-Monte Carlo implementation uses the Sobol sequence. As expected, the MC, QMC, and CG-QMC methods all find the approximate global optimum, but the relative errors in the approximation and the empirical rates of convergence are quite different: the MC and QMC formulas gain about one significant digit of precision while the CG-QMC formula gains about two. The relative errors using degrees $d = 2, \ldots, 6$ are shown in Table 5.

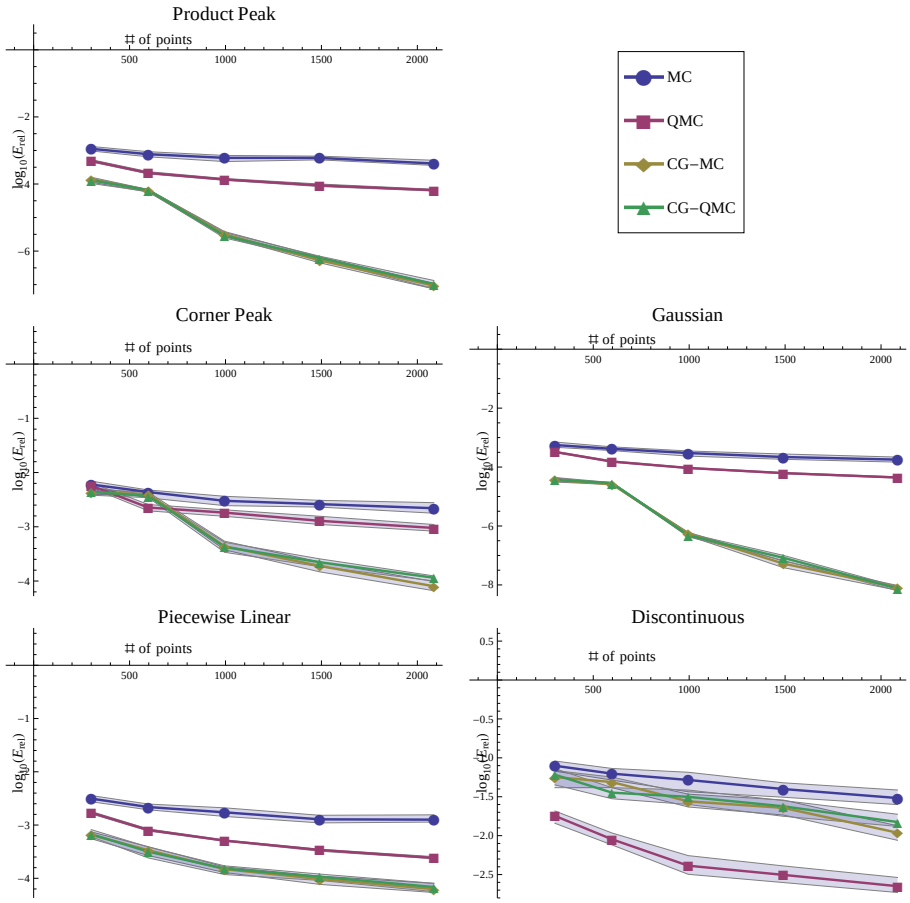

FIGURE 4. Performance of four cubature formulas using the 100-dimensional parametric families g_1–g_5. Horizontal axis: number of points used in the formulas; the points correspond to increasing degree of exactness of the CG-MC and CG-QMC formulas from 2 to 6. Vertical axis: base-10 logarithm of the median relative errors from 200 experiments. The gray shaded bands around the median relative errors are 0.95-level confidence intervals on the median.

5.6. **Stochastic programming problems.** We compared the same methods as in Section 5.5 on the two benchmark problems proposed in [2].

5.6.1. *Markowitz model.* This example was used by Pennanen and Koivu [23] for comparing the MC and QMC methods of scenario generation. With our notation the well-known Markowitz portfolio optimization model can be cast as follows:

$$\min_{x\in\mathcal{X}} \int_{\mathbb{R}^n} (\xi^{\mathrm{T}}x - m^{\mathrm{T}}x)^2 \mu(d\xi), \quad \mathcal{X} = \Big\{ x \in \mathbb{R}^n_+ \;\Big|\; \sum_i x_i \le 1, m^{\mathrm{T}}x \ge R \Big\},$$

The parameters in [23] are $n = 3$, $R = 0.011$, and $\mu \sim N(m, V)$ normally distributed with mean $m = (0.0101110, 0.0043532, 0.0137058)$, and covariance matrix

$$V = \begin{pmatrix} 0.00324625 & 0.00022983 & 0.00420395 \\ 0.00022983 & 0.00049937 & 0.00019247 \\ 0.00420395 & 0.00019247 & 0.00764097 \end{pmatrix}.$$

d	K	MC	QMC	CG-QMC
2	300	$1.243 \cdot 10^{-2}$	$9.663 \cdot 10^{-3}$	$2.402 \cdot 10^{-2}$
3	598	$6.425 \cdot 10^{-3}$	$6.022 \cdot 10^{-3}$	$2.047 \cdot 10^{-2}$
4	995	$4.161 \cdot 10^{-3}$	$3.183 \cdot 10^{-3}$	$5.070 \cdot 10^{-4}$
5	1491	$3.121 \cdot 10^{-3}$	$1.699 \cdot 10^{-3}$	$1.148 \cdot 10^{-3}$
6	2086	$2.627 \cdot 10^{-3}$	$9.005 \cdot 10^{-4}$	$2.846 \cdot 10^{-4}$

TABLE 5. Relative errors of the approximate solutions to the problem $\max_{u\in[0,1]^{100}} \int_{[0,1]^{100}} g_3(u,x)dx$, as a function of the degree of exactness d. The number of scenarios K is also shown.

d	K	MC	QMC	CG-MC	CG-QMC
2	10	0.3727	0.6720	0.1489	0.0826
4	35	0.1839	0.2537	0.0184	0.0970
6	84	0.1431	0.1216	0.0313	0.0170
8	165	0.0822	0.0720	0.0015	0.0057
10	286	0.0623	0.0464	0.0117	0.0135
12	455	0.0519	0.0302	0.0038	0.0094
14	680	0.0420	0.0239	0.0044	0.0026
16	969	0.0317	0.0156	0.0025	0.0023

TABLE 6. Relative errors of the approximate solutions to the Markowitz model, as a function of the degree of exactness d. The number of scenarios $K = \binom{d+4}{4}$ is also shown.

The true optimal objective function value is easily found to be $\approx 3.7853 \cdot 10^{-3}$. After replacing the integral with the approximation given by the various cubature formulas, the resulting optimization problems were solved. The degree of exactness of the CG-MC and CG-QMC cubature formulas was varied between 2 and 16. For comparison, we also generated MC and QMC formulas with the same number of scenarios.

We compared the relative errors in the optimal objective function values, the results are reported in Table 6. The entries in the MC column are averages of 100 independent runs of Monte Carlo sampling. As in the previous sections, Monte Carlo sampling performed very poorly, and quasi-Monte Carlo also fell behind the moment matching methods. The difference between the performances of the CG-MC, CG-QMC methods appears insignificant, both achieve a relative error less than 1% with only 364 scenarios.

5.6.2. *Utility maximization model.* Our next problem, also from [23] and [2], is somewhat more complicated. The goal is to determine a portfolio of n assets that maximizes the expected value of an exponential utility function. The returns of the stocks are random with a joint lognormal distribution. In our notation the optimization model is the following:

$$\min_{x\in\mathcal{X}} \int_{\mathbb{R}^n} \exp(-\xi^{\mathrm{T}} x)\mu(d\xi), \quad \mathcal{X} = \Big\{ x \in \mathbb{R}^n_+ \,\Big|\, \sum_i x_i \le 1 \Big\},$$

with μ corresponding to a lognormal density.

In [2] this model was solved approximately by several methods for $n = 6$, and for a joint lognormal distribution with a randomly drawn mean and covariance matrix. Table 7 shows the

$$m = \begin{pmatrix} 1.186573284 & 1.271987547 & 1.262317169 & 1.165045884 & 1.050464674 & 0.990708953 \end{pmatrix}$$

$$H = \begin{pmatrix} 0.171633 & 0.269857 & 0.108958 & 0.172031 & 0.127180 & 0.234007 \\ 0.158851 & 0.161127 & 0.157614 & 0.189621 & 0.143484 & 0.295287 \\ 0.237658 & 0.245012 & 0.255039 & 0.261027 & 0.227806 & 0.127998 \\ 0.105797 & 0.186469 & 0.254520 & 0.165738 & 0.213629 & 0.193843 \\ 0.278903 & 0.142303 & 0.275562 & 0.217241 & 0.175747 & 0.110844 \\ 0.242071 & 0.165333 & 0.217824 & 0.234243 & 0.104976 & 0.276837 \end{pmatrix}$$

TABLE 7. The mean vector m and the square root H of the covariance matrix corresponding to the lognormal distribution used in the Utility Maximization example of [2].

d	K	MC	QMC	CG-MC	CG-QMC
2	28	0.1994	0.1817	0.0683	0.0102
3	84	0.1139	0.1130	0.0037	0.0726
4	210	0.0661	0.0626	0.0057	0.0015
5	462	0.0457	0.0319	0.0010	0.0019
6	924	0.0299	0.0189	0.0070	0.0028
7	1716	0.0245	0.0078	0.0044	0.0037

TABLE 8. Relative errors of the approximate solutions to the Utility maximization model, as a function of the degree of exactness d. The number of scenarios $K = \binom{d+6}{6}$ is also shown.

values of these parameters. The problem cannot be solved analytically; the optimal objective function value is equal to 0.0553 up to three significant digits.

The experimental setup was identical to that of the previous example, except that dimension of the formulas is increased to six. The relative errors of the optimal objective function values are shown in Table 8 for each method. The degree of exactness of the moment matching methods was increased up to $d = 7$.

As in the previous example, moment matching has a clear advantage over MC and QMC methods using the name number of scenarios. Monte Carlo did especially poorly.

5.7. **The running time of formula generation.** We examined the distribution of the number of simplex iterations between the outer (column generation) iterations, and also the total number of simplex iterations used by the column generation algorithms. The results from CG-MC and CG-QMC were essentially identical, below we show results obtained using CG-QMC only.

First we generated CG-QMC formulas with up to 2000 nodes and plotted the number of simplex iterations against the number of moments matched; the results show an empirical $O(N^2)$ total number of simplex iterations (Figure 5) for formulas matching N moments.

5.8. **Sensitivity of column generation to precision of moment matching.** The stopping condition of the Algorithm 1 is that the optimal objective function value of the auxiliary LP (7) is less than some parameter $\varepsilon > 0$. The question naturally arises how sensitive the method is to the value of this parameter.

Our experiments showed a somewhat surprising insensitivity: we generated CG-QMC formulas for dimensions 4–10 and degree 4–10 with up to 2000 points with two (rather extreme) values of ε: 10^{-1} and 10^{-10}. For each dimension and degree the number of iterations CG-QMC needed to find the corresponding two formulas differed by no more than 10% for these two values of ε. Comparing the results of two less extreme values, $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-10}$ we found that there

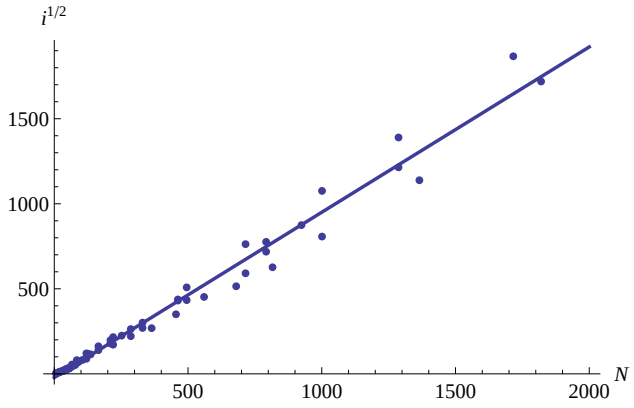

FIGURE 5. The number of simplex iterations i and the number of matched moments, N for formulas with $N \leq 2000$. The vertical axis shows $i^{1/2}$.

was only a single case in which there was a difference in the number of iterations: for dimension 4 and degree 10 two additional columns were needed to reach $\varepsilon = 10^{-10}$ precision from $\varepsilon = 10^{-4}$; in all the other cases there was no difference in the number of iterations.

The formulas used in our numerical experiments were generated using $\varepsilon = 10^{-10}$.

6. Conclusion

Our novel, optimization based, approach to moment matching scenario generation is promising. It is more flexible than the traditional moment matching cubature formulas. It can be generalized to obtain cubature formulas that are exact in any given finite dimensional functional space other than spaces of polynomials. This may be useful for problems where functions from a specific space (or functions that can be approximated well by functions from a specific space) need to be integrated repeatedly. Our approach also allows to match some (say, the low order) moments exactly, while only *bounding* other (say, the higher order) moments. Both of these extensions require only simple changes in the semi-infinite LP formulation.

Furthermore, the method guarantees that the weights of the formulas are positive, and that the number of points in the formula does not exceed the number of moments to be matched. The proposed approach will be effective when the moment matching formulas are generated, saved, and reused. This is the case, for example, when a formulated stochastic optimization model of a fixed dimension is solved multiple times in an application.

The optimization model for the moment matching scenario generation with positive weights is a semi-infinite LP, which in turn can be solved to the desired accuracy using a column generation (dual cutting plane) procedure with polynomial iteration complexity.

The column generation procedure was implemented within the simplex method framework, and the empirical results suggest that the number of columns needed to find an acceptable solution of the semi-infinite LP is in the order of the number of moments required to be matched.

The column generation subproblem can be formulated as a multivariate polynomial optimization problem, which is NP-hard. The popular approach of using sum-of-squares approximations to global polynomial optimization did not prove to be useful for these problems, because the current implementations cannot solve the subproblems when the dimension or the degree is high. Instead of global polynomial optimization we used a sampling based technique to generate acceptable columns in our algorithm. A Monte Carlo and a quasi-Monte Carlo variant of the sampling technique were tested. The numerical results show that the number of columns needed

to solve the semi-infinite LP using the sampling technique is almost the same as those needed if the column generation subproblem is solved to optimality.

We found that the total number of simplex iterations required to generate the moment matching formulas grows quadratically with the number of moments to be matched. However, using a single computer we were able to generate formulas matching about 3000 moments using CPLEX in about 3 days wall clock time. Formulas matching less than 1000 moments could be generated considerably faster, mostly in a few minutes. While this cannot compete with the running time of (quasi-)Monte Carlo methods, these are acceptable running times for a preprocessing step.

The technique of selective moment matching was used to generate formulas for structured integration and stochastic optimization problems with up to 100 variables, matching a subset of the moments up to order six, using 3000 moments. On six-dimensional problems, where all moments are matched, we generated formulas up to degree seven.

Using problems from both the numerical integration literature and the stochastic optimization literature we found that when the order of moments matched is at least four and the integrands are continuous, the moment matching formulas generated by our technique give significantly better numerical accuracy than possible by sample average approximation obtained using either Monte Carlo and quasi-Monte Carlo sampling. For discontinuous integrands neither of the tested methods were adequate.

When comparing different scenario generation methods, one may reasonably ask not only which method yields the best results with a given number of scenarios (the approach taken in the present paper), but also which method yields the best results for a given computational budget. This question can be relatively easily studied in the integration setting, but the optimal trade-off between scenario generation and optimization merits a separate computational study. In numerical integration it is relatively easy to both generate a very large number of (quasi-)Monte Carlo sample points, and evaluate the resulting approximation of the integral fast. (For smooth integrands it is still possible that moment matching scenarios outperform extremely large scale Monte Carlo simulations; recall the Gaussian example from Section 5.2.) We see the primary advantage of moment matching scenarios in settings where scenarios are generated as a preprocessing step, and are reused many times; and also in settings when it is necessary to get the best possible approximation with a small number of scenarios. In particular, in stochastic optimization, the integrand (and its derivatives) may need to be evaluated a large number of times, and the use of a large number of scenarios may be prohibitive. This is especially the case when the number of decision variables is very large, but the number of random variables is small enough so that most important moments can be matched.

A number of questions require further study, we shall outline two of them. It might be useful to study the properties of the polynomial optimization problems generated by this application, and develop improved algorithms for global polynomial optimization, since the existing algorithms could not solve any but the simplest polynomial optimization subproblems. Second, the reason for the large computation times required by CPLEX is that the linear optimization problems arising in the computation are dense. Given the scientific importance of the application, it would be useful to develop more efficient numerical methods for finding solutions to such linear programs.

7. Acknowledgments

The authors are grateful to the associate editor and the referees for their comments. Both the presentation of the material and the numerical experiments were greatly improved using their suggestions.

References

1. T. Bagby, L. Bos, and N. Levenberg, *Multivariate simulatenous approximation*, Constructive Approximation **18** (2002), 569–577.
2. Michael Chen and Sanjay Mehrotra, *Epi-convergent scenario generation method for stochastic problems via sparse grid*, Tech. Report 2007-08, Northwestern University, Evanston, IL, December 2007, `http://www.iems.northwestern.edu/docs/working_papers/MichaelChen_Publication_2_scenarios.pdf`.
3. Ronald Cools, *An encyclopaedia of cubature formulas*, Journal of Complexity **19** (2003), no. 3, 445–453.
4. P. Date, R. Mamon, and L. Jalen, *A new moment matching algorithm for sampling from partially specified symmetric distributions*, Operations Research Letters **36** (2008), no. 6, 669–672.
5. Etienne de Klerk, *The complexity of optimizing over a simplex, hypercube or sphere: a short survey*, Central European Journal of Operations Research **16** (2008), 111–125.
6. Josef Dick and Friedrich Pillichshammer, *Digital nets and sequences*.
7. Charles F. Dunkl and Yuan Xu, *Orthogonal polynomials of several variables*, Encyclopedia of Mathematics and its Applications, vol. 81, Cambridge University Press, Cambridge, UK, 2001.
8. J. Dupačová, N. Gröwe-Kuska, and W. Römisch, *Scenario reduction in stochastic programming*, Mathematical Programming **95** (2003), no. 3, 493–511.
9. C.F. Gauss, *Methodus nova integralium valores per approximationem inviendi*, Werke, vol. 3, Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1876, pp. 163–196.
10. Alan Genz, *Testing multidimensional integration routines*, Tools, Methods and Languages for Scientific and Engineering Computation (B. Ford, J. C. Rault, and F. Thomasset, eds.), Elsevier (North-Holland), 1984, pp. 81–94.
11. Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization*, Algorithms and Combinatorics, vol. 2, Springer Verlag, New York, NY, 1988.
12. Nalan Gülpınar, Berç Rustem, and Reuben Settergren, *Simulation and optimization approaches to scenario tree generation*, Journal of Economic Dynamics and Control **28** (2004), no. 7, 1291–1315.
13. Florian Heiss and Viktor Winschel, *Likelihood approximation by numerical integration on sparse grids*, Journal of Econometrics **144** (2008), no. 1, 62–80.
14. Holger Heitsch and Werner Römisch, *Scenario tree modeling for multistage stochastic programs*, Mathematical Programming **118** (2009), no. 2, 371–406.
15. Didier Henrion and Jean-Bernard Lasserre, *GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi*, ACM Transactions on Mathematical Software **29** (2003), no. 2, 165–194.
16. Kjetil Høyland, Michal Kaut, and Stein W. Wallace, *A heuristic for moment-matching scenario generation*, Computational Optimization and Applications **24** (2003), no. 2, 169–185.
17. F.Y. Kuo, *Component-by-component constructions achieve the optimal rate of convergence for multivariate integration in weighted korobov and sobolev spaces*, Journal of Complexity **19** (2003), no. 3, 301–320.
18. Jean B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM Journal on Optimization **11** (2001), no. 3, 796–817.
19. Philip M. Lurie and Matthew S. Goldberg, *An approximate method for sampling correlated random variables from partially-specified distributions*, Management Science **44** (1998), no. 2, 203–218.
20. Yurii Nesterov, *Squared functional systems and optimization problems*, High Performance Optimization (Hans Frenk, Kees Roos, Tamás Terlaky, and Shuzhong Zhang, eds.), Applied Optimization, vol. 33, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000, pp. 405–440.
21. Harald Niederreiter, *Random number generation and quasi-Monte Carlo methods*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63, SIAM, Philadelphia, PA, 1992.
22. Jorge Nocedal and Stephen J. Wright, *Numerical optimization*, second ed., Springer, New York, NY, 2006.
23. Teemu Pennanen and Matti Koivu, *Epi-convergent discretizations of stochastic programs via integration quadratures*, Numerische Mathematik **100** (2005), no. 1, 141–163.
24. Petko D. Proinov, *Discrepancy and integration of continuous functions*, Journal of Approximation Theory **52** (1988), 121–131.
25. Ralph Tyrrell Rockafellar, *Convex analysis*, Princeton University Press, Princeton, NJ, 1970.
26. Werner Römisch, *Scenario generation in stochastic programming*, Wiley Encyclopedia of Operations Research and Management Science, Wiley, 2010.
27. Rudolf Schürer, *A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems*, Mathematics and Computers in Simulation **62** (2003), no. 3-6, 509–517, 3rd IMACS Seminar on Monte Carlo Methods.
28. S. A. Smoljak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Dokl. Akad. Nauk SSSR (1963), 240–243.
29. A.H. Stroud, *Approximate calculation of multiple integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

30. Vladimir Tchakaloff, *Formules de cubatures mécaniques à coefficients non négatifs*, Bulletin des Sciences Mathématiques. 2e Série **81** (1957), no. 2, 123–134. MR 0094632 (20 #1145)
31. Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu, *Sums of squares and semidefinite programming relaxation for polynomial optimization problems with structured sparsity*, SIAM Journal on Optimization **17** (2006), 218–242.
32. Yinyu Ye, *Approximating quadratic programming with bound and quadratic constraints*, Mathematical Programming **84** (1999), 219–226.

Tables corresponding to Figures 1 and 2

Product peak

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.3463	0.2854	0.3872	0.208	0.1835	0.25	0.1538	0.1295	0.187	0.1859	0.1559	0.2108
35	0.2817	0.2557	0.3216	0.2136	0.188	0.2445	0.1331	0.1164	0.1462	0.1322	0.1118	0.1505
70	0.1281	0.1095	0.1523	0.05177	0.04664	0.05908	0.02946	0.02417	0.03664	0.03514	0.03023	0.03783
126	0.1278	0.1071	0.1501	0.04829	0.04062	0.05965	0.01628	0.01292	0.01885	0.01637	0.01445	0.01951
210	0.0806	0.06564	0.09127	0.02809	0.02252	0.03291	0.008971	0.007606	0.01068	0.008758	0.007598	0.01003
330	0.1133	0.0936	0.1333	0.03007	0.02689	0.03434	0.004578	0.003949	0.005504	0.004427	0.003565	0.005498
495	0.07524	0.06288	0.08461	0.009937	0.008661	0.01217	0.002759	0.002183	0.003149	0.002702	0.002267	0.003176
715	0.07219	0.05999	0.0782	0.01108	0.009606	0.01352	0.001337	0.001125	0.001663	0.001226	0.0009905	0.00161
1001	0.0435	0.03576	0.05277	0.004277	0.003163	0.005077	0.0007913	0.0006827	0.0009893	0.0007903	0.0006169	0.0008943

Corner peak

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.4467	0.3858	0.4883	0.4002	0.4002	0.4802	0.2922	0.2694	0.3342	0.2901	0.2901	0.3558
35	0.3913	0.3524	0.4541	0.4002	0.3934	0.4002	0.201	0.1258	0.2622	0.1624	0.1443	0.1844
70	0.2795	0.2516	0.3046	0.2143	0.1144	0.2186	0.06432	0.06247	0.06697	0.06534	0.0609	0.07433
126	0.2899	0.2433	0.3274	0.2143	0.1144	0.2186	0.01707	0.01703	0.01893	0.03411	0.02486	0.03906
210	0.1943	0.1673	0.2356	0.08203	0.04844	0.1307	0.01308	0.01308	0.01555	0.02591	0.02146	0.02833
330	0.1848	0.1605	0.233	0.1307	0.08203	0.1429	0.01121	0.007872	0.01187	0.009518	0.008969	0.009522
495	0.1366	0.1203	0.152	0.07534	0.05106	0.09329	0.006417	0.006064	0.006623	0.005359	0.004498	0.005367
715	0.1202	0.1082	0.1359	0.09329	0.05739	0.1377	0.002272	0.001655	0.002547	0.002336	0.001916	0.002698
1001	0.09395	0.07431	0.1123	0.04501	0.03978	0.05744	0.0008197	0.0006886	0.001243	0.001475	0.001004	0.001563

Gaussian

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.04667	0.03935	0.05328	0.02649	0.02256	0.03172	0.005678	0.005371	0.005921	0.006126	0.005672	0.006839
35	0.04369	0.0366	0.05336	0.02452	0.02192	0.02815	0.003973	0.003765	0.004097	0.004016	0.003797	0.004278
70	0.01969	0.01664	0.02305	0.00768	0.006875	0.008927	0.00005117	0.00004383	0.00006146	0.00006445	0.00005603	0.00007782
126	0.02622	0.02067	0.03053	0.008161	0.006849	0.009208	0.00001009	$8.794{\cdot}10^{-6}$	0.000012	$9.559{\cdot}10^{-6}$	$8.56\cdot10^{-6}$	0.00001124
210	0.01452	0.01125	0.01657	0.003869	0.00337	0.004772	$1.169\cdot10^{-6}$	$8.009{\cdot}10^{-7}$	$1.527{\cdot}10^{-6}$	$1.548{\cdot}10^{-6}$	$1.254{\cdot}10^{-6}$	$1.925{\cdot}10^{-6}$
330	0.01434	0.01245	0.01633	0.003231	0.002718	0.003645	$2.678{\cdot}10^{-7}$	$2.296{\cdot}10^{-7}$	$3.177{\cdot}10^{-7}$	$2.143{\cdot}10^{-7}$	$1.707{\cdot}10^{-7}$	$2.911{\cdot}10^{-7}$
495	0.009248	0.008249	0.01086	0.00169	0.001462	0.001874	$1.307{\cdot}10^{-8}$	$1.053{\cdot}10^{-8}$	$1.634{\cdot}10^{-8}$	$2.34{\cdot}10^{-8}$	$1.595{\cdot}10^{-8}$	$3.048{\cdot}10^{-8}$
715	0.009906	0.008377	0.01173	0.001639	0.001482	0.001854	$1.478{\cdot}10^{-9}$	$1.249{\cdot}10^{-9}$	$1.908{\cdot}10^{-9}$	$3.006{\cdot}10^{-9}$	$2.598{\cdot}10^{-9}$	$3.39{\cdot}10^{-9}$
1001	0.005992	0.005214	0.006676	0.0007519	0.0005886	0.000903	$1.674{\cdot}10^{-10}$	$1.219{\cdot}10^{-10}$	$2.466{\cdot}10^{-10}$	$2.771{\cdot}10^{-10}$	$1.914{\cdot}10^{-10}$	$3.594{\cdot}10^{-10}$

Piecewise Linear

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.08149	0.06023	0.09901	0.05466	0.04897	0.06105	0.01705	0.01374	0.01912	0.02123	0.01761	0.02459
35	0.0887	0.07537	0.1063	0.05016	0.04177	0.05729	0.01352	0.0121	0.01521	0.01317	0.01226	0.01547
70	0.04652	0.0387	0.05427	0.01446	0.0133	0.01672	0.00204	0.001818	0.002704	0.003394	0.002646	0.003732
126	0.03927	0.03368	0.04524	0.01421	0.01198	0.01751	0.001468	0.001148	0.00167	0.001339	0.001088	0.001549
210	0.02977	0.02493	0.03607	0.006155	0.005353	0.007327	0.001277	0.001095	0.001462	0.0007851	0.0006838	0.000986
330	0.0312	0.02842	0.03508	0.005594	0.004715	0.006597	0.0005521	0.0004543	0.0006913	0.0006559	0.0005175	0.0008025
495	0.01714	0.01459	0.02044	0.003792	0.003512	0.004353	0.0006317	0.0005658	0.0007425	0.0005539	0.0004719	0.0006443
715	0.02086	0.01787	0.02388	0.003321	0.002757	0.003782	0.0003409	0.0002765	0.0004231	0.0003793	0.0003134	0.0004425
1001	0.01247	0.01056	0.01482	0.001496	0.001281	0.00161	0.0003105	0.0002787	0.0003587	0.0002666	0.0002258	0.0003364

Discontinuous

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.5922	0.4962	0.75	0.5827	0.4249	0.6856	0.3211	0.2698	0.3952	0.2965	0.2572	0.358
35	0.5847	0.4346	0.7665	0.5793	0.3886	0.6942	0.1925	0.1622	0.2323	0.2521	0.2141	0.2855
70	0.2494	0.215	0.2821	0.2063	0.1519	0.2473	0.09838	0.08107	0.1353	0.1038	0.077	0.1297
126	0.2424	0.2127	0.3004	0.1772	0.1357	0.2299	0.05864	0.05323	0.08268	0.05518	0.04165	0.07422
210	0.1559	0.1194	0.189	0.04276	0.03419	0.05739	0.04287	0.03543	0.05446	0.04499	0.03486	0.05184
330	0.1653	0.1225	0.199	0.03442	0.02659	0.04617	0.0304	0.01964	0.039	0.03086	0.02568	0.03693
495	0.1098	0.09322	0.1338	0.04309	0.03174	0.05711	0.03061	0.02409	0.03893	0.02483	0.01923	0.03222
715	0.08947	0.07696	0.1225	0.03802	0.02571	0.05175	0.0193	0.0143	0.02569	0.02366	0.01965	0.0288
1001	0.08703	0.06876	0.1061	0.01778	0.01242	0.02426	0.02151	0.01761	0.02649	0.02034	0.01671	0.02586

Table 9. Median relative errors ("median") and 95% level confidence intervals ($[lb, ub]$) around the median for the relative errors from the experiments of Section 5.2. Table corresponds to Figure 1 in the main text.

Product peak

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.06462	0.05457	0.08005	0.03081	0.02546	0.03829	0.02288	0.01944	0.02902	0.02638	0.02055	0.03047
35	0.04787	0.03975	0.05579	0.02376	0.02033	0.02928	0.01695	0.01418	0.02089	0.01416	0.01151	0.01766
70	0.03467	0.02716	0.04284	0.01758	0.01606	0.02083	0.004815	0.003976	0.005428	0.004946	0.004242	0.005751
126	0.02182	0.01865	0.02595	0.01148	0.0103	0.01309	0.002987	0.002333	0.003645	0.001572	0.001301	0.001918
210	0.01738	0.01422	0.02199	0.00923	0.007461	0.01069	0.001537	0.001248	0.001905	0.001287	0.001051	0.001636
330	0.01385	0.01084	0.01697	0.007623	0.006214	0.009252	0.00111	0.0008496	0.001233	0.001469	0.001282	0.00171
495	0.01152	0.009403	0.01508	0.002545	0.002087	0.002868	0.0008808	0.0007329	0.001016	0.000523	0.0004381	0.000623
715	0.009505	0.007576	0.01212	0.002263	0.001948	0.002768	0.0007089	0.0005633	0.0007992	0.000782	0.0006596	0.0009075
1001	0.01018	0.008856	0.01163	0.001811	0.001524	0.002071	0.000691	0.0005963	0.0007779	0.0004623	0.0003816	0.0005299

Corner peak

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.05341	0.04626	0.06248	0.02488	0.02256	0.02608	0.02471	0.02308	0.02888	0.02842	0.02762	0.03371
35	0.03668	0.031	0.04355	0.01657	0.01369	0.01882	0.008766	0.005935	0.01262	0.01443	0.01349	0.01443
70	0.02753	0.02344	0.03305	0.00977	0.007317	0.01744	0.004217	0.0039	0.004218	0.004599	0.004204	0.005412
126	0.01883	0.01633	0.02258	0.007043	0.005444	0.008056	0.003112	0.00253	0.003897	0.00174	0.00151	0.002301
210	0.01184	0.01048	0.015	0.003958	0.003854	0.006104	0.002735	0.002294	0.003436	0.001199	0.0009341	0.001216
330	0.00898	0.006941	0.01035	0.003476	0.002544	0.005349	0.0009952	0.0007781	0.001068	0.001316	0.001067	0.002398
495	0.009268	0.007673	0.01069	0.001745	0.001486	0.002911	0.0008056	0.0005491	0.001276	0.0009148	0.0008961	0.0009967
715	0.007002	0.00572	0.008755	0.001514	0.001486	0.002187	0.0006757	0.0006069	0.0009516	0.0009591	0.0007343	0.001102
1001	0.006737	0.005397	0.008116	0.00144	0.0009931	0.001663	0.0007662	0.0006985	0.0008003	0.0004113	0.0003536	0.0008793

Gaussian

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.009353	0.007571	0.01109	0.005997	0.005807	0.006308	0.005371	0.004413	0.006271	0.004727	0.004081	0.00553
35	0.006629	0.0055	0.008428	0.003508	0.003017	0.004356	0.009063	0.008429	0.009894	0.003472	0.003283	0.003655
70	0.004637	0.003897	0.005612	0.002716	0.002285	0.003019	0.001065	0.0009855	0.001156	0.0004988	0.0004024	0.000583
126	0.003353	0.002625	0.004155	0.001624	0.001414	0.001774	0.0003094	0.0002823	0.0003969	0.0003557	0.0003054	0.0004311
210	0.002494	0.002144	0.002971	0.001127	0.0008906	0.001385	0.0002952	0.0002469	0.0003722	0.0001551	0.0001369	0.0001974
330	0.001977	0.001738	0.002627	0.001063	0.0009251	0.001204	0.0003408	0.0003249	0.0003677	0.0003167	0.0002742	0.0003496
495	0.00143	0.001145	0.001657	0.0003649	0.000326	0.0004096	0.0001708	0.0001492	0.0001997	0.000184	0.0001615	0.0002055
715	0.001367	0.001152	0.001749	0.0003629	0.0003066	0.0004077	0.0001573	0.0001495	0.000166	0.000163	0.000146	0.0001749
1001	0.001167	0.0008677	0.001365	0.000273	0.0002275	0.0002963	0.0001325	0.0001165	0.0001421	0.0001326	0.0001263	0.0001401

Piecewise Linear

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.02251	0.01833	0.02715	0.007676	0.005766	0.009748	0.01358	0.01151	0.0162	0.01316	0.01069	0.0152
35	0.0187	0.015	0.02361	0.007237	0.005621	0.008287	0.01969	0.0181	0.02252	0.006172	0.004572	0.007356
70	0.01151	0.009367	0.01307	0.00614	0.00487	0.007191	0.001804	0.001375	0.002356	0.001571	0.001383	0.00182
126	0.00911	0.007777	0.01059	0.003595	0.002924	0.004017	0.001225	0.001057	0.001425	0.0009088	0.0007276	0.001003
210	0.00717	0.006024	0.009021	0.00304	0.002606	0.003471	0.001024	0.0008881	0.001155	0.0005501	0.0004623	0.0006428
330	0.005397	0.00407	0.006764	0.002691	0.002338	0.003095	0.0007018	0.0006411	0.0008392	0.0005951	0.0005027	0.0007343
495	0.00398	0.003464	0.005029	0.0006388	0.0005443	0.00075	0.0003778	0.0002874	0.0004867	0.0004086	0.0003568	0.0004454
715	0.00431	0.00367	0.004989	0.0008225	0.0006884	0.0009422	0.0002643	0.0002065	0.0002916	0.0003925	0.0002957	0.0004652
1001	0.002946	0.002654	0.003639	0.0005358	0.0004328	0.0006448	0.0001999	0.0001643	0.0002566	0.0001968	0.000174	0.0002287

Discontinuous

	MC			QMC			CG-MC			CG-QMC		
K	median	lb	ub	median	lb	ub	median	lb	ub	median	lb	ub
15	0.5316	0.3162	1.000	0.3507	0.2068	1.000	0.385	0.2979	1.000	0.3576	0.2087	0.8926
35	0.1936	0.1267	0.2941	0.06773	0.05593	0.09592	0.2291	0.181	0.3021	0.2062	0.12	0.268
70	0.1433	0.1001	0.2314	0.08095	0.05489	0.1626	0.07698	0.05409	0.1221	0.07961	0.05749	0.1247
126	0.1093	0.06626	0.1667	0.04186	0.02573	0.1198	0.05347	0.0286	0.1049	0.05086	0.02728	0.06521
210	0.1164	0.07628	0.1907	0.08513	0.05122	0.1569	0.05447	0.03904	0.08998	0.09227	0.06793	0.1249
330	0.1023	0.06139	0.1877	0.07267	0.04396	0.183	0.09262	0.04801	0.1566	0.1063	0.04321	0.1765
495	0.07343	0.05455	0.1235	0.06277	0.03591	0.09868	0.05265	0.0329	0.08735	0.05513	0.03059	0.08651
715	0.0623	0.04153	0.1199	0.06167	0.03368	0.09805	0.06622	0.03413	0.1052	0.06854	0.03762	0.1073
1001	0.06996	0.04219	0.1003	0.06994	0.03463	0.1005	0.07995	0.05134	0.1092	0.06177	0.03791	0.09764

TABLE 10. Median relative errors ("median") and 95% level confidence intervals ($[lb, ub]$) around the median for the relative errors from the experiments of Section 5.3.1. Table corresponds to Figure 2 in the main text.

NORTHWESTERN UNIVERSITY, DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT SCIENCES, EVANSTON, IL, USA. E-MAIL: MEHROTRA@IEMS.NORTHWESTERN.EDU (CORRESPONDING AUTHOR), DPAPP@IEMS.NORTHWESTERN.EDU