Computational Geometry 46 (2013) 232–252

Contents lists available at SciVerse ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Software for exact integration of polynomials over polyhedra

J.A. De Loera, B. Dutra *, M. Köppe, S. Moreinis [1], G. Pinto, J. Wu [2]

Department of Mathematics, UC Davis, One Shields Avenue, Davis, CA 95616, USA

ARTICLE INFO

Article history:
Received 30 July 2011
Received in revised form 2 April 2012
Accepted 5 September 2012
Available online 11 September 2012
Communicated by Komei Fukuda

Keywords:
Exact integration
Volume computation
Integration of polynomials
Valuations
Polyhedral computation

ABSTRACT

We are interested in the fast computation of the exact value of integrals of polynomial functions over convex polyhedra. We present speed-ups and extensions of the algorithms presented in previous work by some of the authors. We provide a new software implementation and benchmark computations. The computation of integrals of polynomials over polyhedral regions has many applications; here we demonstrate our algorithmic tools solving a challenge from combinatorial voting theory.

1. Introduction

Integration is a fundamental problem arising in many contexts, from engineering to algebraic geometry. For example, center of mass, moments and products of inertia about various axes are integrals used in any type of dynamic simulation or modeling such as computer games (see [1–3]), similarly, normalized volumes indicate the degrees of toric varieties and are closely related to moment maps of symplectic manifolds (see [4,5]). Integration over polyhedra is particularly useful because many domains can be approximated by polyhedra and then decomposed into convex polyhedra (e.g., a tetrahedral mesh decomposition etc.).

In this work we are interested in the exact *evaluation of integrals* over convex polyhedral regions. More precisely, let P be a d-dimensional rational convex polyhedron in $\mathbb{R}^n$ and let $f \in \mathbb{Q}[x_1, \ldots, x_n]$ be a polynomial with rational coefficients. We consider the problem of efficiently computing the *exact* value of the integral of the polynomial f over P, which we denote by $\int_P f \, \mathrm{d}m$. Here we use the *integral Lebesgue measure* $\mathrm{d}m$ on the affine hull $\langle P \rangle$ of the polytope P. This general setting is quite useful because, when the polytope is full-dimensional, the integral Lebesgue measure coincides with the standard Riemann integral but generalizes it naturally to cases when the polytope is not full-dimensional. Another reason to compute in this setting is that the volume of P and every integral of a polynomial function with rational coefficients yields *rational numbers*. Finally this normalization of the measure occurs naturally in Euler–Maclaurin formulas for a polytope P, which relate sums over the lattice points of P with integrals over the various faces of P. We remark that the computer algebra community has dedicated a great deal of effort to develop a different kind of exact integration, sometimes called symbolic

* Corresponding author.
E-mail addresses: deloera@math.ucdavis.edu (J.A. De Loera), bedutra@ucdavis.edu (B. Dutra), mkoeppe@math.ucdavis.edu (M. Köppe), moreinis@stanford.edu (S. Moreinis), gpinto@ucdavis.edu (G. Pinto), jwu2@caltech.edu (J. Wu).
[1] Present address: Stanford University, Stanford, CA 94305, USA.
[2] Present address: California Institute of Technology, Pasadena, CA 91125, USA.

0925-7721/$ – see front matter
http://dx.doi.org/10.1016/j.comgeo.2012.09.001

integration (see [6]), and understood to be the automatic computation of the antiderivatives of functions, as predicted by the fundamental theorem of calculus, but we do not discuss this kind of exact integration here.

Regarding the theoretical computational complexity of our problem, it is very educational to look first at the case when f is the constant polynomial 1, and the answer is simply a volume. It has been proved that already computing the volume of polytopes of varying dimension is #P-hard [7–11], and that even approximating the volume is hard [12]. More recently in [13] it was proved that computing the center of mass of a polytope is #P-hard.

We report on a new C++ implementation, `LattE integrale` [14], which extends the work done in [15] and [16]. This paper is mostly an experimental and practical study, but it also slightly develops the theory of [15]. This article presents useful formulas for integration of powers of linear forms over simplicial cones that complement those presented in [15].

Our method of computation relies on powerful mathematical ideas. It was proved in [15] that, unlike general polynomials, integrals over simplices of arbitrary powers of linear forms, or of polynomials with a fixed number of variables, can be computed in polynomial time. In this case our algorithms use known properties of integrals of exponentials of linear forms (see [17,18]). This allows for fast calculation over general polytopes using two methods that depend on two different decompositions of polyhedra. General polyhedra can be decomposed as either a disjoint union of simplices, i.e., triangulations, or as signed cone decompositions of the kind proposed by Brion, Lasserre, Lawrence, and Varchenko [19,20,10,21]. The polynomial-time complexity for integration over simplices shown in [15] can be extended to more polyhedra as long as their decompositions are of "small" size (note that this is always the case in fixed dimension).

This paper is organized as follows. We begin in Section 2 recalling the mathematical ideas at the heart of our algorithms (although we omit details of proofs, they can be found in the references). We begin with a short review of polyhedral geometry, specially valuations. In Section 3 we discuss details about the implementation including main subroutines and data structures. In Section 4 we first discuss speed improvements for integrating over simplices from earlier work in [15], and then we report on several benchmarks of integration over arbitrary polytopes. More experimental tables are available online [14]. We conclude our paper with an application: we solve a computational challenge from combinatorial voting theory.

2. Mathematical preliminaries

In this section we recall the necessary mathematical background used in our algorithms. We state results without proof but excellent background sources for what is going to be discussed here include [15,22–24] and the references mentioned there.

2.1. Polyhedra and polynomials

A *convex rational polyhedron* P in $\mathbb{R}^d$ (we will simply say *polyhedron*) is defined as the intersection of a finite number m of closed halfspaces bounded by rational affine hyperplanes. We say that P is *full-dimensional* (in $\mathbb{R}^d$) if the affine span of P is $\mathbb{R}^d$. When $P = \{x\colon\ Ax \leqslant b\}$ for a $m \times d$ matrix A and m-vector b, P is said to be given by a *halfspace* or *h-representation*. For our purposes we focus on two special kinds of polyhedra. First, for integration, we consider only compact polyhedra, which are called *polytopes*. When P is a polytope it is know to be the convex hull $\mathrm{conv}(V)$ of finitely many points in $\mathbb{R}^d$, $V = \{v_1, \ldots, v_n\}$. In that case P is said to be given by a *vertex* or *v-representation*. We can switch between the h- and v-representations of a d-dimensional polytope using well-known algorithms (see [25,26]).

Second, in many computations we use cones: a *polyhedral cone* is a polyhedron C containing the origin and if $y \in C$, then any dilation λy is also in C. We call a *cone* C any polyhedral cone (with vertex 0) and an *affine cone* is a translation $s + C$ of a cone C. A cone C is called *simplicial* if it is generated by linearly independent vectors of $\mathbb{R}^d$. A simplicial cone C is called *unimodular* if it is generated by linearly independent integral vectors $v_1, \ldots, v_k$ such that $\{v_1, \ldots, v_k\}$ can be completed to an integral basis of $\mathbb{Z}^d$. An affine cone C is called simplicial (respectively, simplicial unimodular) if the associated cone is. The set of vertices of P is denoted by $V(P)$. For each vertex $s \in V(P)$, the *cone of feasible directions* $C_s(P)$ at the vertex s is the cone of all vectors y such that $s + \varepsilon y \in P$ for some $\varepsilon > 0$. The *tangent cone* of a polytope P at a vertex s is the affine cone $s + C_s(P)$ (this is a translation of $C_s(P)$). For details on all these notions see, e.g., [22].

For the integration of a full-dimensional polytope we consider the standard Lebesgue measure, which gives volume 1 to the fundamental domain of the lattice $\mathbb{Z}^n$. But if P lies inside an affine subspace $L + a$, with L a rational linear subspace of dimension $n \leqslant d$, we will normalize the Lebesgue measure on L, so that the volume of the fundamental domain of the lattice $L \cap \mathbb{Z}^d$ is 1. Thus for any affine subspace $L + a$ parallel to L, we define the *integral Lebesgue measure* $\mathrm{d}m$ by translation. For example, the diagonal of the unit square has length 1 instead of $\sqrt{2}$. This has the great advantage that for rational input our output will always be an (exact) rational number $\int_P f\,\mathrm{d}m$ in the usual binary encoding.

One important point of our method is that all computations are done in the representation of polynomials given by *powers of linear forms*. It is well known that any homogeneous polynomial of degree M can be decomposed as a sum of M-th powers of linear forms. For example, one can decompose the polynomial f as a sum $f = \sum_\ell c_\ell \langle \ell, x\rangle^M$ with at most 2^M terms. This decomposition is given by the following well-known identity for monomials: If $x^{\mathbf{M}} = x_1^{M_1} x_2^{M_2} \cdots x_n^{M_n}$, then

$$x^{\mathbf{M}} = \frac{1}{|\mathbf{M}|!} \sum_{0 \leqslant p_i \leqslant M_i} (-1)^{|\mathbf{M}| - (p_1 + \cdots + p_n)} \binom{M_1}{p_1} \cdots \binom{M_n}{p_n} (p_1 x_1 + \cdots + p_n x_n)^{|\mathbf{M}|}, \tag{1}$$

Table 1
Average number of powers of linear forms plus or minus one standard deviation necessary to express one monomial in d variables, averaged over 50 monomials of the same degree.

d	Monomial degree					
	5	10	20	30	40	50
3	14 ± 3	$(6.6 \pm 1.2) \cdot 10^1$	$(4.0 \pm 0.5) \cdot 10^2$	$(1.2 \pm 0.1) \cdot 10^3$	$(2.7 \pm 0.2) \cdot 10^3$	$(5.2 \pm 0.2) \cdot 10^3$
4	16 ± 5	$(1.1 \pm 0.2) \cdot 10^2$	$(1.1 \pm 0.2) \cdot 10^3$	$(4.5 \pm 0.6) \cdot 10^3$	$(1.3 \pm 0.2) \cdot 10^4$	$(3.0 \pm 0.2) \cdot 10^4$
5	19 ± 4	$(1.5 \pm 0.4) \cdot 10^2$	$(2.2 \pm 0.6) \cdot 10^3$	$(1.2 \pm 0.3) \cdot 10^4$	$(4.7 \pm 0.7) \cdot 10^4$	$(1.4 \pm 0.2) \cdot 10^5$
6	20 ± 5	$(2.0 \pm 0.6) \cdot 10^2$	$(4.1 \pm 1.2) \cdot 10^3$	$(3.2 \pm 0.8) \cdot 10^4$	$(1.5 \pm 0.3) \cdot 10^5$	$(5.2 \pm 0.6) \cdot 10^5$
7	21 ± 5	$(2.4 \pm 0.9) \cdot 10^2$	$(6.7 \pm 2.4) \cdot 10^3$	$(7.1 \pm 2.1) \cdot 10^4$	$(4.0 \pm 1.0) \cdot 10^5$	$(1.7 \pm 0.3) \cdot 10^6$
8	21 ± 5	$(2.9 \pm 0.9) \cdot 10^2$	$(1.1 \pm 0.5) \cdot 10^4$	$(1.4 \pm 0.5) \cdot 10^5$	$(9.8 \pm 2.7) \cdot 10^5$	$(4.8 \pm 1.1) \cdot 10^6$
10	24 ± 5	$(3.5 \pm 1.1) \cdot 10^2$	$(2.1 \pm 0.9) \cdot 10^4$	$(4.1 \pm 1.6) \cdot 10^5$	$(4.5 \pm 1.7) \cdot 10^6$	$(3.1 \pm 1.0) \cdot 10^7$

where $|\mathbf{M}| = M_1 + \cdots + M_n \leqslant M$. Of course, when dealing with general polynomials, this same formula can be applied for as many monomials as is necessary. For example, the polynomial $7x^2 + y^2 + 5z^2 + 2xy + 9yz$ can be written as $\frac{1}{8}(12(2x)^2 - 9(2y)^2 + (2z)^2 + 8(x+y)^2 + 36(y+z)^2)$.

It is worth noting that the above formula does not yield an optimal decomposition, but it suffices to generate a polynomial-time algorithm on fixed degree $|\mathbf{M}|$ or fixed number of variables [15]. The problem of finding a decomposition with the smallest possible number of summands is known as the *polynomial Waring problem*. What is the smallest integer $r(M,n)$ such that a generic homogeneous polynomial $f(x_1,\ldots,x_n)$ of degree M in n variables is expressible as the sum of $r(M,n)$ M-th powers of linear forms? This problem was solved for general polynomials by Alexander and Hirschowitz [27] (see [28] for an extensive survey), but there is no computational or constructive version of this result that would yield the optimal decomposition for a specific input polynomial and the bounds may be much too pessimistic on the average situation. Only very recently Carlini et al. [29] gave efficient decompositions of a monomial. However, their decomposition involves roots of unity, and here we are interested in an arithmetic version of the problem where everything is expressed using rational forms and rational coefficients. But we can see that the explicit rational construction we use in our code is not too far away from the optimum.

Table 1 lists the average number of powers of linear forms necessary to decompose monomials of given degree generated uniformly at random. To create the monomials, we keep adding 1 to the power of a randomly chosen variable until the monomial has the desired degree. The table shows mild exponential growth as degree or dimension grow. This was predicted in the theory.

In conclusion, to integrate a multivariate polynomial, we first algebraically decompose each monomial into a sum of powers of linear forms which, as we will see next, can be integrated very fast in practice over simplices or over simplicial cones using a few useful formulas. Thus we will need a geometric decomposition of our polytopes into those pieces.

2.2. Valuations and formulas of integration of exponentials over cones and simplices

We now recall several formulas for the integrals of a power of a linear form over a simplex or over a simplicial cone. The idea is that if we can do fast integration for those two structures, then we can always rely on two polyhedral decompositions of the input polyhedron to obtain the integral. See Section 2.5 for details.

One of the most important properties of integrals over polyhedra is that they can be seen as valuations. A *valuation* F is a linear map from the rational vector space of the indicator functions of rational polyhedra $P \subset \mathbb{R}^d$ into a rational vector space M. Whenever the indicator functions $[P_i]$ of a family of polyhedra P_i satisfy a linear relation $\sum_i r_i[P_i] = 0$, then the elements $F(P_i)$ satisfy the same relation $\sum_i r_i F(P_i) = 0$ (for a formal definition within the polytope algebra, see Chapter 2 of [22]).

Let $C = \sum_{i=1}^d \mathbb{R}_+ u_i$ be the simplicial cone spanned by linearly independent integral vectors $u_1, u_2, \ldots, u_d$. The *fundamental parallelepiped* Π_C of the cone C (with respect to the generators $u_i, i = 1,\ldots,d$) is the set of points $\Pi_C = \sum_{i=1}^d [0,1[\, u_i$. Let us denote by $\mathrm{vol}(\Pi_C)$ its volume.

Proposition 1. *(See Theorem 8.4 in [22].) There exists a unique valuation $I(P)(\ell)$ which associates to every polyhedron $P \subset V$ a meromorphic function so that the following properties hold*

(i) *If ℓ is a linear form such that $\mathrm{e}^{\langle \ell, x\rangle}$ is integrable over P, then*

$$I(P)(\ell) = \int_P \mathrm{e}^{\langle \ell, x\rangle}\, \mathrm{d}m.$$

(ii) *For every point $s \in \mathbb{R}^n$, one has*

$$I(s+P)(\ell) = \mathrm{e}^{\langle \ell, s\rangle} I(P)(\ell).$$

(iii) *If P contains a straight line, then $I(P) = 0$.*

A consequence of the valuation property is the following fundamental theorem. It follows from the Brion–Lasserre–Lawrence–Varchenko decomposition theory of a polyhedron into the supporting cones at its vertices [19,22,21,20].

Lemma 2. *Let P be a polyhedron with set of vertices $V(P)$. For each vertex s, let $C_s(P)$ be the cone of feasible directions at vertex s. Then*

$$I(P)(\ell) = \sum_{s\in V(P)} I\big(s + C_s(P)\big)(\ell). \tag{2}$$

Note that the cone $C_s(P)$ in Lemma 2 may not be simplicial, but for simplicial cones their integrals have explicit rational function formulas. As we see in Proposition 4, one can derive an explicit formula for the rational function $I(s + C_s(P))$ in terms of the geometry of the cones.

Lemma 3. *Using the valuation property for the valuation $I(P)(\ell)$ and the linearity over the integrands we have that*:

(i) *For any triangulation $\mathcal{T}$ of the polytope P, we have $I(P)(\ell) = \sum_{\Delta\in\mathcal{T}} I(\Delta)(\ell)$.*
(ii) *For any triangulation $\mathcal{D}_s$ of the feasible cone $C_s(P)$ at each of the vertices s of the polytope P we have $I(P)(\ell) = \sum_{s\in V(P)} \sum_{C\in\mathcal{D}_s} I(s + C)(\ell)$.*

Lemma 3 says that *if* we know how to integrate over simplices or simplicial cones, we can integrate over a polytope. We are close to knowing how to do this. By elementary integration, and Proposition 1, we have the following.

Proposition 4. *For a simplicial cone C generated by rays $u_1, u_2, \ldots, u_d$ (with vertex 0) and for any point s*

$$I(s + C)(\ell) = \operatorname{vol}(\Pi_C) e^{\langle \ell, s\rangle} \prod_{i=1}^{d} \frac{1}{\langle -\ell, u_i\rangle}. \tag{3}$$

This identity holds as a meromorphic function of ℓ and pointwise for every ℓ such that $\langle \ell, u_i\rangle \neq 0$ for all u_i.

2.3. From exponentials to powers of linear forms

We now consider powers of linear forms instead of exponentials. Similar to $I(P)$, we now let $L^M(P)$ be the meromorphic extension of the function defined by

$$L^M(P)(\ell) = \int_P \langle \ell, x\rangle^M \, dm$$

for those ℓ such that the integral exists. To transfer what we know about integrals of exponentials to those of powers of linear forms, we can consider the formula of Proposition 4 as a function of the auxiliary parameter t:

$$\int_{s+C} e^{\langle t\ell, x\rangle} \, dm = \operatorname{vol}(\Pi_C) e^{\langle t\ell, s\rangle} \prod_{i=1}^{d} \frac{1}{\langle -t\ell, u_i\rangle}. \tag{4}$$

Using the series expansion of the left in the variable t, we wish to recover the value of the integral of $\langle \ell, x\rangle^M$ over the cone. This is the coefficient of t^M in the expansion; to compute it, we equate it to the Laurent series expansion around $t = 0$ of the right-hand side expression, which is a meromorphic function of t. Clearly

$$\operatorname{vol}(\Pi_C) e^{\langle t\ell, s\rangle} \prod_{i=1}^{d} \frac{1}{\langle -t\ell, u_i\rangle} = \sum_{n=0}^{\infty} t^{n-d} \frac{\langle \ell, s\rangle^n}{n!} \cdot \operatorname{vol}(\Pi_C) \prod_{i=1}^{d} \frac{1}{\langle -\ell, u_i\rangle}.$$

We say that ℓ is *regular* if $\langle \ell, u_i\rangle \neq 0$ for every ray u_i of the cone. With this, we can conclude the following.

Corollary 5. *For a regular linear form ℓ and a simplicial cone C generated by rays $u_1, u_2, \ldots, u_d$ with vertex s*

$$L^M(s + C)(\ell) = \frac{M!}{(M + d)!} \operatorname{vol}(\Pi_C) \frac{(\langle \ell, s\rangle)^{M+d}}{\prod_{i=1}^{d} \langle -\ell, u_i\rangle}. \tag{5}$$

Otherwise when ℓ is not regular, there is a nearby perturbation which is regular. To obtain it, we use $\ell + \hat{\varepsilon}$ where $\hat{\varepsilon} = \varepsilon a$ is any linear form with $a \in \mathbb{R}^n$ such that $\langle -\ell - \hat{\varepsilon}, u_i \rangle \neq 0$ for all u_i, to define a new linear form (depending of a) on the space of meromorphic functions in the variable ε. Then, applying (5) on the limit as ε goes to zero we obtain:

$$L^M(s+C)(\ell) = \frac{M!}{(M+d)!}\operatorname{vol}(\Pi_C)\operatorname{Res}_{\varepsilon=0}\frac{(\langle \ell+\hat{\varepsilon}, s\rangle)^{M+d}}{\varepsilon\prod_{i=1}^{d}\langle -\ell-\hat{\varepsilon}, u_i\rangle}. \tag{6}$$

We recall some useful facts on complex analysis (see, e.g., [30] for details). As we observed, there is a pole at $\varepsilon = 0$ for our univariate rational function given in formula (6) of Corollary 5. Recall that if a univariate rational function $f(\varepsilon) = p(\varepsilon)/q(\varepsilon)$ has a Laurent series expansion $f(\varepsilon) = \sum_{k=-m}^{\infty} a_k \varepsilon^k$, the residue is defined as a_{-1}. Given a rational function $f(\varepsilon)$ with a pole at $\varepsilon = 0$ there are a variety of well-known techniques to extract the value of the residue. For example, if $\varepsilon = 0$ is a simple pole ($m = 1$), then $\operatorname{Res}_{\varepsilon=0}(f) = \frac{p(0)}{q'(0)}$. Otherwise, when $\varepsilon = 0$ is a pole of order $m > 1$, we can write $f(\varepsilon) = \frac{p(\varepsilon)}{\varepsilon^m q_1(\varepsilon)}$. Then expand p, q_1 in powers of ε with $p(\varepsilon) = a_0 + a_1\varepsilon + a_2\varepsilon^2 + \cdots$ and $q_1(\varepsilon) = b_0 + b_1\varepsilon + b_2\varepsilon^2 + \cdots$. This way the Taylor expansion of $p(\varepsilon)/q_1(\varepsilon)$ at ε_0 is $c_0 + c_1\varepsilon + c_2\varepsilon^2 + c_3\varepsilon^3 + \cdots$, where $c_0 = \frac{a_0}{b_0}$, and $c_k = \frac{1}{b_0}(a_k - b_1c_{k-1} - b_2c_{k-2} - \cdots - b_kc_0)$. Thus we recover the residue $\operatorname{Res}_{\varepsilon=0}(f) = c_{m-1}$. We must stress that the special structure of the rational functions in Corollary 5 can be exploited to speed up computation further rather than using this general methodology. For more on this see [31,22,15] and the following discussion.

Finally, we have all the tools necessary to write down our formula for integration using cone decompositions.

Corollary 6. *For any triangulation $\mathcal{D}_s$ of the tangent cone $C_s(P)$ at each of the vertices s of the polytope P we have*

$$L^M(P)(\ell) = \sum_{s\in V(P)}\sum_{C\in\mathcal{D}_s} L^M(s+C)(\ell). \tag{7}$$

2.4. The formula for the simplex

Suppose now that $\Delta \subset \mathbb{R}^n$ is a d-dimensional simplex (as it may appear in a triangulation of the polytope P), and ℓ is a linear form on $\mathbb{R}^n$. We say that the linear form ℓ is *regular* for the simplex Δ if it is not orthogonal to any of the edges of the simplex. If ℓ is regular for Δ, then it is regular for all tangent cones at each of the vertices. We then find the following result as a special case of Corollary 6.

Corollary 7. *(Brion, see [19].) Let Δ be a d-simplex with vertices $s_1, \ldots, s_{d+1} \in \mathbb{R}^d$. Let ℓ be a linear form which is regular w.r.t. Δ, i.e., $\langle \ell, s_i\rangle \neq \langle \ell, s_j\rangle$ for any pair $i \neq j$. Then we have the following relation.*

$$L^M(\Delta)(\ell) = \int_{\Delta}\langle \ell, x\rangle^M \,\mathrm{d}m = d!\operatorname{vol}(\Delta, \mathrm{d}m)\frac{M!}{(M+d)!}\left(\sum_{i=1}^{d+1}\frac{\langle \ell, s_i\rangle^{M+d}}{\prod_{j\neq i}\langle \ell, s_i - s_j\rangle}\right). \tag{8}$$

When ℓ is regular, Brion's formula is very short; it is a sum of $d+1$ terms. When ℓ is not regular, we can again use a perturbation $\ell + \hat{\varepsilon}$ where $\hat{\varepsilon} = \varepsilon a$ as in Corollary 5, so that the expression of the integral over the simplex reduces to a sum of residues as in (6). However, in the special case of a simplex, there exists a computationally more efficient method that avoids the calculation of a perturbation a; see [15].

Here is how it works. From [15, Theorem 10] we find that $L^M(\Delta)(\ell)$ is the coefficient of the term t^M in the Laurent series of the rational function

$$d!\operatorname{vol}(\Delta, \mathrm{d}m)\frac{M!}{(M+d)!}\frac{1}{\prod_{j=1}^{d+1}(1 - t\langle \ell, s_j\rangle)} \tag{9}$$

in the variable $t \in \mathbb{C}$. This rational function can be expanded into partial fractions. To this end, let $K \subseteq \{1, \ldots, d+1\}$ be an index set of the different poles $t = t_k := 1/\langle \ell, s_k\rangle$, and for $k \in K$ let m_k denote the order of the pole, i.e.,

$$m_k = \#\big\{i \in \{1, \ldots, d+1\}\colon \langle \ell, s_i\rangle = \langle \ell, s_k\rangle\big\}.$$

Then the rational function can be written as

$$\sum_{k\in K}\left(\frac{a_{k,1}}{1 - t\langle \ell, s_k\rangle} + \frac{a_{k,2}}{(1 - t\langle \ell, s_k\rangle)^2} + \cdots + \frac{a_{k,m_k}}{(1 - t\langle \ell, s_k\rangle)^{m_k}}\right),$$

where the coefficients $a_{k,r}$ are given by certain residues about the pole $t = t_k$. After a change of variables, $t = t_k + \varepsilon$, one obtains the following formula.

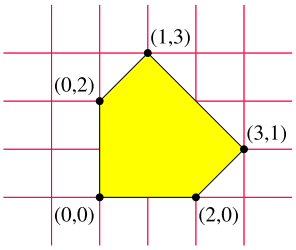


Fig. 1. A pentagon.

Corollary 8. *(See Corollary 13 in [15].) Let Δ be a d-dimensional simplex. Then for an arbitrary power $\langle \ell, x\rangle^M$ of a linear form, we have*:

$$\int_{\Delta} \langle \ell, x\rangle^M \,\mathrm{d}m = d!\,\mathrm{vol}(\Delta,\mathrm{d}m)\frac{M!}{(M+d)!}\sum_{k\in K}\mathrm{Res}_{\varepsilon=0}\frac{(\varepsilon+\langle \ell, s_k\rangle)^{M+d}}{\varepsilon^{m_k}\prod_{\substack{i\in K\\ i\neq k}}(\varepsilon+\langle \ell, s_k-s_i\rangle)^{m_i}}. \tag{10}$$

To conclude we note that one can even extend the formula above on integrating a power of a linear form to the case of a *product of powers of several linear forms* (see [15]).

2.5. Should one triangulate or cone decompose?

One could triangulate the whole polytope, or integrate over each tangent cone. However, each cone must be decomposed into simplicial cones. This is the trade-off: we can get away with not doing one large polytope triangulation, but we might have to do many smaller cone triangulations.

The number of simplices in a triangulation and the number of simplicial cones in a polytope decomposition can significantly differ. Depending on the polytope, choosing the right method can determine its practicality. Our experimental results agree with [16] in showing that triangulating the polytope is better for polytopes that are "almost simplicial" while cone decomposition is faster for simple polytopes. The details will be discussed in Section 4.

Lemma 3 together with the formulas we stated for integration over simplices and cones give a general process for computing integrals:

- Decompose a polynomial as a sum of powers of linear forms.
- Select a decomposition of the polyhedron in question, either a triangulation or a cone decomposition.
- Apply the formulas to each piece and add up the results via the above results.

2.6. Examples

2.6.1. Integral values encoded by rational function identities

Before working out a simple integration example, let us highlight the fact that for regular linear forms the integration formulas are given by sums of rational functions which we read from the geometry at vertices and possibly a cone decomposition method: Consider a pentagon P with vertices $(0,0)$, $(2,0)$, $(0,2)$, $(3,1)$, and $(1,3)$ as in Fig. 1.

Then the rational function giving the value of $\int_P (c_1x+c_2y)^M\,\mathrm{d}x\,\mathrm{d}y$ is

$$\frac{M!}{(M+2)!}\left(\frac{(2c_1)^{M+2}}{c_1(-c_1-c_2)}+4\frac{(3c_1+c_2)^{M+2}}{(c_1+c_2)(2c_1-2c_2)}+4\frac{(c_1+3c_2)^{M+2}}{(c_1+c_2)(-2c_1+2c_2)}+\frac{(2c_2)^{M+2}}{(-c_1-c_2)c_2}\right).$$

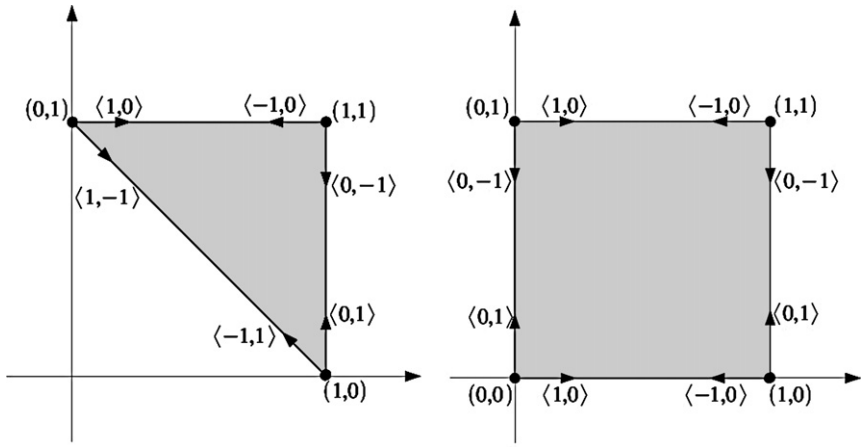
This rational function expression encodes *every integral* of the form $\int_P (c_1x+c_2y)^M\,\mathrm{d}x\,\mathrm{d}y$. For example, if we let $M=0$, then the integral is equal to the area of the pentagon, and the rational function simplifies to a number by simple high-school algebra:

$$\frac{1}{2}\left(4\frac{c_1}{-c_1-c_2}+4\frac{(3c_1+c_2)^2}{(c_1+c_2)(2c_1-2c_2)}+4\frac{(c_1+3c_2)^2}{(c_1+c_2)(-2c_1+2c_2)}+4\frac{c_2}{-c_1-c_2}\right)=6.$$

Hence the area is 6. When M and (c_1,c_2) are given and (c_1,c_2) is not perpendicular to any of the edge directions we can simply plug in numbers to the rational function. For instance, when $M=100$ and $(c_1=3,\ c_2=5)$ the answer is a fraction with numerator equal to

227276369386899663893588867403220233833167842959382265474194585

311501951704481580782855497399198118376955797967280316412539699 2



(a) Triangulation method (b) Cone decomposition method

Fig. 2. Example polytopes.

and denominator equal to 1717. When (c_1, c_2) is perpendicular to an edge direction, we encounter (removable) singularities in the rational functions, thus using complex residues we can do the evaluation. Note that those linear forms that are perpendicular to some edge direction form a measure zero set inside a hyperplane arrangement.

2.6.2. Using the triangulation method

Take the problem of integrating the polynomial $x + y$ over the triangle Δ with vertices $s_1 = (1, 1)$, $s_2 = (0, 1)$, and $s_3 = (1, 0)$ in Fig. 2a.

The polynomial is already a power of a linear form, and the polytope is a simplex. Because $\ell = (1, 1)$ is not regular (it is perpendicular to the edge spanned by s_2 and s_3), we have to build the index set K. Note $\langle \ell, s_1 \rangle = 2$, $\langle \ell, s_2 \rangle = 1$, and $\langle \ell, s_3 \rangle = 1$; pick $K = \{1, 2\}$ with $m_1 = 1$, $m_2 = 2$. We proceed below with this choice, but note that we have a choice in picking the indices and we could have instead $K = \{1, 3\}$. This would yield a different decomposition of the generating function. Note also that the decomposition of the power of a linear form is not necessarily unique either. We now need to compute two values:

Vertex s_1: We are not dividing by zero, we can simply plug vectors into Corollary 7, $\frac{\langle \ell, s_1 \rangle^3}{\langle \ell, s_1 - s_2 \rangle^2} = 8$.

Vertex s_2: Here, we need to compute a residue.

$$\mathrm{Res}_{\varepsilon=0} \frac{(\varepsilon + \langle \ell, s_2 \rangle)^{1+2}}{\varepsilon^2(\varepsilon + \langle \ell, s_2 - s_1 \rangle)} = \mathrm{Res}_{\varepsilon=0} \frac{(\varepsilon + 1)^{1+2}}{\varepsilon^2(\varepsilon - 1)} = -4.$$

Finally, $\int_\Delta (x + y)\,\mathrm{d}x\,\mathrm{d}y = 2! \times \frac{1}{2} \times \frac{1!}{3!}(8 - 4) = 2/3$.

2.6.3. Using the cone decomposition method

Next, integrate the polynomial x over the unit square in Fig. 2b using the cone decomposition algorithm. The polynomial is already a power of a linear form so $\ell = (1, 0)$. The polytope has four vertices that we need to consider, and each tangent cone is already simplicial. The linear form ℓ is not regular at vertices s_1, s_2. We let the reader verify in these cases the residue-based calculation gives the value of the integrals on the corresponding cones to be zero. We only do in detail the same calculation for vertex $s_3 = (1, 0)$. At this vertex, the rays are $u_1 = (0, 1)$, $u_2 = (-1, 0)$. Because $\langle \ell, u_1 \rangle = 0$, we need a perturbation vector $\hat{\varepsilon}$ so that when $\ell := \ell + \hat{\varepsilon}$, we do not divide by zero on any cone (we have to check this cone and the next one). Pick $\hat{\varepsilon} = (\varepsilon, \varepsilon)$. Then the integral on this cone is

$$\frac{M!}{(M + d)!} \mathrm{vol}(\Pi_C)\, \mathrm{Res}_{\varepsilon=0} \frac{(1 + \varepsilon)^{1+2}}{\varepsilon(-\varepsilon)(1 + \varepsilon)} = \frac{1!}{(1 + 2)!} \times 1 \times -2 = -2/6.$$

Vertex $s_4 = (1, 1)$: The rays are $u_1 = (-1, 0)$, $u_2 = (0, -1)$. Again, we divide by zero, so we perturbate ℓ by the same $\hat{\varepsilon}$. The integral on this cone is

$$\frac{M!}{(M + d)!} \mathrm{vol}(\Pi_C)\, \mathrm{Res}_{\varepsilon=0} \frac{(1 + 2\varepsilon)^{1+2}}{\varepsilon(\varepsilon)(1 + \varepsilon)} = \frac{1!}{(1 + 2)!} \times 1 \times 5 = 5/6.$$

The integral $\int_P x\,\mathrm{d}x\,\mathrm{d}y = 0 + 0 - 2/6 + 5/6 = 1/2$ as it should be.

3. How the software works

LattE was originally developed in 2001 as software to study lattice points of convex polytopes [32]. The algorithms used combinations of geometric and symbolic computation. Two key data structures are rational generating functions and cone decompositions, and it was the first ever implementation of Barvinok's algorithm. LattE was improved in 2007 with various software and theoretical modifications, which increased speed dramatically. This version was released under the name LattE macchiato; see [33]. Now in 2011, our new release LattE integrale has extended its capabilities to include the computation of exact integrals of polynomial functions over convex polyhedra. The new integration functions are C++ implementations of the algorithms provided in [15] with additional technical improvements (including an important new set of data structures for the manipulation of truncated series). A key distinction between LattE integrale and other software tools is that our algorithms give the exact evaluation of the integral since our implementation uses exact rational arithmetic. The code of this software is freely available at [14] under the GNU license.

The new implementation of LattE integrale allows us to calculate the integral of a sum of powers of linear forms over an arbitrary polytope. Alternatively, we can calculate the integral of a sum of monomials by decomposing each monomial into a sum of powers of linear forms using formula (1), then integrating these powers of linear forms.

This section starts with a discussion of our new data structure for manipulating polynomials and linear forms, then we describe the format LattE integrale expects for the input polytopes, and we end with a detailed explanation of the two main algorithms.

3.1. Input format and data structures

The input format for the polynomials is identical to that of the Maple programs of [15]:

- A polynomial is represented as a list of its monomials in the form

  [*monomial*$_1$, *monomial*$_2$, ...],

  where *monomial*$_i$ is represented by

  [*coefficient*,[*exponent-vector*]].

  For example, $3x_0^2x_1^4x_2^6 + 7x_1^3x_2^5$ is input as [[3,[2,4,6]], [7,[0,3,5]]].
- To deal directly with powers of linear forms, the input format is

  [*linear-term*$_1$, *linear-term*$_2$, ...],

  where *linear-term*$_i$ is represented by

  [*coefficient*,[*power*,[*coefficient-vector*]]].

  For example, $3(2x_0 + 4x_1 + 6x_2)^{10} + 7(3x_1 + 5x_2)^{12}$ is input as [[3,[10,[2,4,6]]], [7,[12,[0,3,5]]]].

In [15], the integration over simplices was first implemented in Maple, and so there was no control over the data structures used to store the data. We have implemented the simplex integration algorithm in C++ with a sophisticated data structure and have developed a new algorithm that integrates over the tangent cones of a polytope. Currently, we are using *burst tries*, a data structure designed to have cache-efficient storage and search, due to the fact that they are prefix trees with sorted arrays of stored elements as leaves [34]. Such a data structure is performance-critical when computing residues, as a comparison with a linked-list implementation showed. In our implementation, each node corresponds to a particular dimension or variable and contains the maximal and minimal values of the exponent on this dimension. The node either points to another node a level deeper in the tree or a list of sorted elements.

The input rational polytope P could be given to LattE integrale by an h-representation or v-representation. The input format is the same as in the previous versions and it is explained in [14]. Although the theory we presented earlier works for both full-dimensional and non-full-dimensional rational polytopes, the current release of LattE integrale is only guaranteed to do integration and volume computation in full-dimensional polyhedra. It is worth stressing the old capabilities for counting lattice points still work for polytopes of all dimensions and we impose no arbitrary limit on the size or dimension of the input. LattE integrale relies on Cddlib [35] or 4ti2 [36] for all basic polyhedral calculations such as computation of dimension.

Our data structures are specialized for polytopes with vertices of integer coordinates. In order to integrate over rational polytopes, we first dilate them and perform a change of variables. If P is a d-dimensional rational polytope and αP is a

dilation by $\alpha > 0$ that makes P integer, then our software operates on the vertices of αP and rescales the final integral by the following well-known change of variables:

$$\int_P x_i^{m_i} \dots x_d^{m_d}\, \mathrm{d}m = \frac{1}{\alpha^d} \int_{\alpha P} \frac{1}{\alpha^{m_1+\dots+m_d}} x_1^{m_1} \dots x_d^{m_d}\, \mathrm{d}m.$$

After this transformation, we apply formula (1) to transform the polynomial into powers of linear forms.

When integrating polytopes other than simplices, there are two options based on the formulas presented in Section 2: (i) Triangulate the polytope and apply the algorithm for each simplices individually, or (ii) triangulate each tangent cone and integrate each one using the cone decomposition algorithm. Therefore, a key step in all our computations is to find either a triangulation of the polytope or a triangulation of each of its tangent cones. Once more, this step relies on `Cddlib` or `4ti2`, because when we triangulate we compute a regular triangulation via a convex hull [32,33]. We now explore the two integration algorithms in greater detail.

3.2. Integrating powers of linear forms by polytope triangulation

After we decompose the polynomial to a sum of powers of linear forms and after finding a triangulation of the polytope, Algorithm 1 loops over these two sets and integrates each linear form/simplex pair individually using Corollaries 7 and 8.

Algorithm 1 Integrate using polytope triangulation.

Input: $F = \sum c_j \langle \ell_j, x\rangle^{M_j}$, sum of powers of linear forms.
Input: P, a full-dimensional polytope.
Output: integral of the linear forms F over the polytope P.

```
integral ← 0 { integral is a rational data type}
let T be a list of simplices that form a triangulation of P
for all simplices Δ in T do
  for all linear forms c⟨ℓ, x⟩^M in F do
    if ℓ is regular on Δ then
      integral ← integral + c × integrateSimplexRegular(ℓ, M, Δ)
    else
      integral ← integral + c × integrateSimplexResidue(ℓ, M, Δ)
    end if
  end for
end for
return integral
```

In Algorithm 1, the linear forms are represented as a burst trie, the triangulations are stored in a linked list, and each simplex is a simple two-dimensional array containing the vertices $s_1, \dots, s_{d+1}$.

When ℓ is regular on Δ, the *integrateSimplexRegular* function plugs in numbers and vectors into Corollary 7. Also, the terms in the numerator and denominator are evaluated in a rational data type, and so no floating-point divisions are performed.

When ℓ is not regular, the *integrateSimplexResidue* function (Algorithm 2) applies Corollary 8 and must find an index set $K \subset \{1, \dots, d+1\}$ of different poles $t = 1/\langle \ell, s_k\rangle$, and compute $|K|$ residues. Let $k \in K$ and let m_k denote the order of the pole, i.e.,

$$m_k = \#\big\{i \in \{1, \dots, d+1\}\colon \langle \ell, s_i\rangle = \langle \ell, s_k\rangle\big\}.$$

The problem has now been reduced to evaluating formula (11).

$$\mathrm{Res}_{\varepsilon=0} \frac{(\varepsilon + \langle \ell, s_k\rangle)^{M+d}}{\varepsilon^{m_k} \prod_{\substack{i\in K\\ i\neq k}} (\varepsilon + \langle \ell, s_k - s_i\rangle)^{m_i}} = \big[\varepsilon^{m_k-1}\big] \frac{(\varepsilon + \langle \ell, s_k\rangle)^{M+d}}{\prod_{\substack{i\in K\\ i\neq k}} (\varepsilon + \langle \ell, s_k - s_i\rangle)^{m_i}}, \tag{11}$$

where $[\varepsilon^a]p$ means the coefficient of ε^a in the Laurent series of expression p.

To compute formula (11), we expand the polynomial in terms of ε in the numerator truncated to degree $m_k - 1$. We then find the first m_k terms in the polynomial expansion of each $1/(\varepsilon + \langle \ell, s_k - s_i\rangle)^{m_i}, i \neq k$ term using the general binomial theorem. To make the notation easy, let $b = \langle \ell, s_k - s_i\rangle \in \mathbb{Z}$, then the degree $m_k - 1$ polynomial of $(\varepsilon + b)^{-m_k}$ is

$$\hat{p}(\varepsilon) = \alpha_0 \varepsilon^0 b^{-m_i} + \alpha_1 \varepsilon^1 b^{-m_i-1} + \dots + \alpha_{m_k-1} \varepsilon^{m_k-1} b^{-m_i-m_k+1}, \quad \alpha_j = \binom{m_i + j - 1}{m_i - 1}(-1)^j.$$

This is a polynomial in ε with rational coefficients. For efficiency reasons, we factor $\hat{p}(\varepsilon) = \frac{1}{b^{m_i+m_k-1}} p(\varepsilon)$, $p \in \mathbb{Z}[\varepsilon]$. The *integrateSimplexResidue* and *truncatedMultiply* functions both implement these ideas.

Finally, Algorithm *truncatedMultiply(p, q, k)* takes two polynomials $p, q \in \mathbb{Z}[\varepsilon]$ and returns their product up to and including terms of degree k. Our implementation is very simple (e.g., not using any special multiplication algorithms) but the

Algorithm 2 integrateSimplexResidue.

Input: ℓ, coefficients of a linear form and M, integer power.
Input: Δ, simplex with vertices $s_1, \ldots, s_{d+1}$.
Output: The integral of $\langle \ell, x\rangle^M$ over Δ.
  Let $p_1 \leftarrow 1$, $p_2 \leftarrow 1$ be polynomials in ε.
  Let $rf \leftarrow 1$ be a rational data type.
  Let *sumResidue* $\leftarrow 0$ be a rational data type.
  Make the index set K of unique poles.
  for all k in K **do**
    $rf \leftarrow 1$
    $p_1 \leftarrow$ the expansion of $(\varepsilon + \langle \ell, s_k\rangle)^{M+d}$ up to degree $m_k - 1$ $\{p_1 \in \mathbb{Z}[\varepsilon]\}$
    for all i in K and $i \neq k$ **do**
      $rf \leftarrow rf \times \langle \ell, s_k\rangle^{-(m_i+m_k-1)}$
      $p_2 \leftarrow$ the expansion of $(\varepsilon + \langle \ell, s_k - s_i\rangle)^{-m_i}$ up to degree $m_k - 1$ with $\langle \ell, s_k\rangle^{-(m_i+m_k-1)}$ factored out. $\{p_2 \in \mathbb{Z}[\varepsilon]\}$
      $p_1 \leftarrow$ *truncatedMultiply*$(p_1, p_2, m_k - 1)$
    end for
    Let c be the coefficient of the degree $m_k - 1$ term in $p_1(\varepsilon)$.
    sumResidue $\leftarrow$ *sumResidue* $+ rf \times c$
  end for
  return $abs(det(s_1 - s_{d+1}, \ldots, s_d - s_{d+1})) \times \frac{M!}{(M+d)!} \times$ *sumResidue*

cache-efficient use of the burst tries leads to speed-ups when compared to a naive implementation with arrays. We note that asymptotically faster multiplication algorithms exists (see, e.g., [37]), which might lead to further improvements.

3.3. Integrating powers of linear forms by cone decomposition

After triangulating each tangent cone into simplicial cones, the computation is very similar to the polytope-triangulation case: if ℓ is regular on the rays of the cone, we plug in values into Corollary 5, else we perturb ℓ and perform a residue calculation. Algorithm 3 implements this idea.

Algorithm 3 Integrate using the cone decomposition method.

Input: $F = \sum c_j \langle \ell_j, x\rangle^{M_j}$, powers of linear forms.
Input: P, a full-dimensional polytope.
Output: integral of the linear forms over P.
  integral $\leftarrow 0$ { *integral* is a rational data type}
  Let C be a list of tangent cones P.
  Make T be a list of triangulated cones in C. {A cone in T is in the form $(s; u_1, \ldots, u_d)$, where s is a vertex and u_i are rays}
  for all linear forms $c\langle \ell, x\rangle^M$ in F **do**
    Let $R \subseteq T$ be cones that ℓ is regular on.
    for all $(s; u_1, \ldots, u_d)$ in R **do**
      integral $\leftarrow$ *integral* $+ c \times$ *integrateConeRegular*$(\ell, M, s, u_1, \ldots, u_d)$
    end for
    Pick $\hat{\varepsilon} = \varepsilon(a_1, \ldots, a_d)$ where $a_i \in \mathbb{Z}$ so that we do not divide by zero.
    for all $(s; u_1, \ldots, u_d)$ in $T \setminus R$ **do**
      integral $\leftarrow$ *integral* $+ c \times$ *integrateConeResidue*$(\ell, M, \hat{\varepsilon}, s, u_1, \ldots, u_d)$
    end for
  end for
  return *integral*

In *integrateConeResidue*, ℓ is perturbed by setting $\ell := \ell + \hat{\varepsilon}$, where $\hat{\varepsilon}$ is a vector in terms of ε with coefficients picked on the moment curve with alternating signs. We repeatedly pick a random $t \in \mathbb{Z}_+$ and set $\hat{\varepsilon}_i = t^{i-1}(-1)^{i-1}\varepsilon$ for $i = 1, 2, \ldots, d$ until $\langle -(\ell + \hat{\varepsilon}), u_i\rangle$ is non-zero for every simple cone at every vertex in Corollary 5. Then the residues are computed using the general binomial theorem and truncated series multiplication like in *integrateSimplexResidue*.

3.4. A special case: computing volumes

Computing the volume of a polytope is equivalent to integrating the monomial 1 over the polytope. We again have the two same options when computing volumes as we did when computing integrals. Instead of using the algorithms above, we can simplify the computation. In the triangulation based approach, we find a triangulation of the polytope and sum the volume of each simplex. The volume of a specific simplex is obtained by taking a determinant. In the cone decomposition approach, we triangulate each tangent cone and apply Corollary 5 with $M = 0$ and any random vector ℓ. If we do divide by zero, instead of finding residues, we simply pick a new random ℓ and start the computation over.

Table 2
Average integration time plus or minus one standard deviation when integrating a power of a linear form over a random d-simplex (in seconds over 50 random forms).

Dimension d	Exponent M						
	2	10	20	50	100	300	1000
10	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00
20	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00
50	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.01	0.01 ± 0.00	0.03 ± 0.01
100	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.09 ± 0.00
300	0.35 ± 0.01	0.36 ± 0.01	0.36 ± 0.01	0.36 ± 0.01	0.38 ± 0.01	0.42 ± 0.01	0.66 ± 0.02
400	0.78 ± 0.02	0.79 ± 0.03	0.79 ± 0.02	0.80 ± 0.03	0.82 ± 0.03	0.90 ± 0.03	1.25 ± 0.04

4. Experiments

We did thorough testing of the implementation and performed new computational benchmarks. We report on four different test classes:

1. We expand the computational limits for integrating over simplices described in [15].
2. Next, we integrate random monomials over three families of polytopes: (1) simple, (2) simplicial, and (3) neither simple nor simplicial.
3. Because our volume methods are optimized versions of the integration methods, we also compute the volumes of the same polytopes in the last case above.
4. Finally, we compare `LattE integrale` to other software tools and computational studies [16,38,39,3].

Our integration and volume experiments input data, along with running times and the results of integration and volumes are available on the `LattE` website [14].

4.1. Integration over simplices

In [15], the theory of integration over simplices was developed and a fair amount of `Maple` experiments were carried out to show the potential of the methods. In this section, the experiments we performed clearly indicate that this C++ implementation is at least two orders of magnitude faster than the preceding `Maple` code; compare Tables 5 and 6 in [15] with Tables 2 and 3 in this paper. In Table 2, we used `Maple` to generate powers of random linear forms and randomly generated simplices. The coefficients of each linear form were picked uniformly over $[0, 100] \cap \mathbb{Z}$. We did the integration using `LattE integrale`.

Next, in Table 3, we used `Maple` to generate monomials and simplices. We again integrate using `LattE integrale`. We measure time from the start of program execution to termination, which includes file I/O, system calls, child process time, the time to find tangent-cones, and triangulation time. All triangulations were computed with the software package `cddlib` version 0.94f [35]. All computations were performed on a 64-bit Ubuntu machine with 64 GB of RAM and eight Dual Core AMD Opteron 880 processors. We applied a 600-second maximum running time to this program; tasks taking longer are not benchmarked.

4.2. Integration over general polytopes

We tested the triangulation and cone decomposition integration methods on polytopes and their duals across dimension, vertex counts, and over monomials with different degrees. For each polytope dimension and vertex count we constructed 50 random polytopes by taking the convex hull of random points using `Polymake` [40]. For primal polytopes of dimension d, the number of vertices considered goes from $d+2$ to $d+25$. When zero is not in the interior of the polytope, we translated the centroid to the origin before constructing the dual polytope. Then we integrated each polytope and its corresponding dual polytope over a new random monomial of a set degree. Because of the construction method, most primal polytopes are simplicial and the duals are mostly simple polytopes. We also integrated over G.M. Ziegler's database of polytopes [41], which contains polytopes that are not simplicial nor simple.

Simple and simplicial polytopes We present the results for dimensions 5, 6, and 7. We tested both algorithms on the primal polytopes starting with their v-representation. For their duals we tested the triangulation and cone decomposition methods starting from their h-representations. We also did experiments in dimension 3 and 4 but the numbers are too close to each other to show a clear trend of which is the fastest method (see [42]). We only report those test classes for which every polytope in the test class finished under 600 seconds for both the triangulation and the cone decomposition method. We define the *relative time difference* as the time taken by the triangulation method *minus* the time taken by the cone decomposition method, all divided by the time of the

Table 3
Average and standard deviation of integration time in seconds of a random monomial of prescribed degree by decomposition into linear forms over a d-simplex (average over 50 random forms).

Dimension d	Degree										
	1	2	5	10	20	30	40	50	100	200	300
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.0	3.8
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.4	1.7
3	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	2.3	38.7	162.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.4	24.2	130.7
4	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.7	22.1	–	–
	0.0	0.0	0.0	0.0	0.0	0.1	0.3	0.7	16.7	–	–
5	0.0	0.0	0.0	0.0	0.1	0.3	1.6	4.4	–	–	–
	0.0	0.0	0.0	0.0	0.0	0.2	1.3	3.5	–	–	–
6	0.0	0.0	0.0	0.0	0.1	1.1	4.7	15.6	–	–	–
	0.0	0.0	0.0	0.0	0.1	1.0	4.3	14.2	–	–	–
7	0.0	0.0	0.0	0.0	0.2	2.2	12.3	63.2	–	–	–
	0.0	0.0	0.0	0.0	0.2	1.7	12.6	66.9	–	–	–
8	0.0	0.0	0.0	0.0	0.4	4.2	30.6	141.4	–	–	–
	0.0	0.0	0.0	0.0	0.3	3.0	31.8	127.6	–	–	–
10	0.0	0.0	0.0	0.0	1.3	19.6	–	–	–	–	–
	0.0	0.0	0.0	0.0	1.4	19.4	–	–	–	–	–
15	0.0	0.0	0.0	0.1	5.7	–	–	–	–	–	–
	0.0	0.0	0.0	0.0	3.6	–	–	–	–	–	–
20	0.0	0.0	0.0	0.2	23.3	–	–	–	–	–	–
	0.0	0.0	0.0	1.3	164.8	–	–	–	–	–	–
30	0.0	0.0	0.0	0.6	110.2	–	–	–	–	–	–
	0.0	0.0	0.1	4.0	779.1	–	–	–	–	–	–
40	0.0	0.0	0.0	1.0	–	–	–	–	–	–	–
	0.0	0.0	0.3	7.0	–	–	–	–	–	–	–
50	0.0	0.0	0.1	1.8	–	–	–	–	–	–	–
	0.0	0.1	0.5	12.9	–	–	–	–	–	–	–

triangulation method. Note that when the triangulation method is faster we obtain a negative number. We will use this quantity throughout.

In Figs. 3, 4, and 5, we display histograms on three axes. The first horizontal axis is the relative time difference between the two integration methods. The second horizontal axis shows the degrees of monomials and finally the vertical axis presents the number of random polytopes (in the respective dimensions 5, 6, 7). The height of a particular solid bar in position (a_k, b^*) tallies the number of random polytopes for which the relative time difference between the two algorithms, when integrating a monomial of degree b^*, was between a_{k-1} and a_k with a_k included in that bar. Thus, the bars with relative time difference zero should be counted as experiments where triangulation is faster. Note that the left-most bar a_0 on the graph always contains values from $-\infty$ to a_0. Also note that when the right-most bar is at position 1 it then accounts for relative time differences between the second-to-last relative time difference and 1.

Each figure has one histogram for the primal polytopes and one for the dual polytopes of respective dimensions. For example, Fig. 3 is for polytopes of dimension 5. On the top part of the figure (primal), there are eight colors on the bars, one for each degree (corresponds to a row). Note that large majority of the bars for monomials of degrees 20 or less have a negative relative time difference. This indicates the triangulation method was faster when integrating low degree monomials. In contrast, for higher degrees, say the bars in the degree 50 row, we see the relative time difference in some cases was positive, this shows that the cone method was faster for those simplicial problems, but it is a minority of cases. Similar trends were observed in higher dimensions (see the other figures). On the bottom of Fig. 3 (dual), all the bars have positive relative time difference, which indicates the cone method wins over the triangulation method for the integration of higher degree monomials. More tables are available online [42].

In conclusion, our experiments for dimension higher than four on integrating monomials have the same qualitative behavior as those of [16] for volume computation (polynomial of degree zero): the triangulation method is faster for simplicial polytopes (mass on histograms is highly concentrated on negative relative time differences) while the cone decomposition is faster for simple polytopes (mass on histograms is concentrated on positive relative time differences). The trends are very clear while in dimension less than four, the timings are too close to each other to give a clear-cut trend.

Zero-one polytopes In Fig. 6, we present another histogram comparing the cone decomposition and triangulation methods on Ziegler's database of polytopes [41], which contains many zero-one polytopes and a few other polytopes. We translate each polytope so that its centroid is the origin, thus its dual is well-defined. Then for each polytope and its dual, we integrate 50 random monomials of a set degree. We skipped non-full-dimensional polytopes and a few others that did not finish within 30 minutes. The figure displays the histogram of the relative time differences

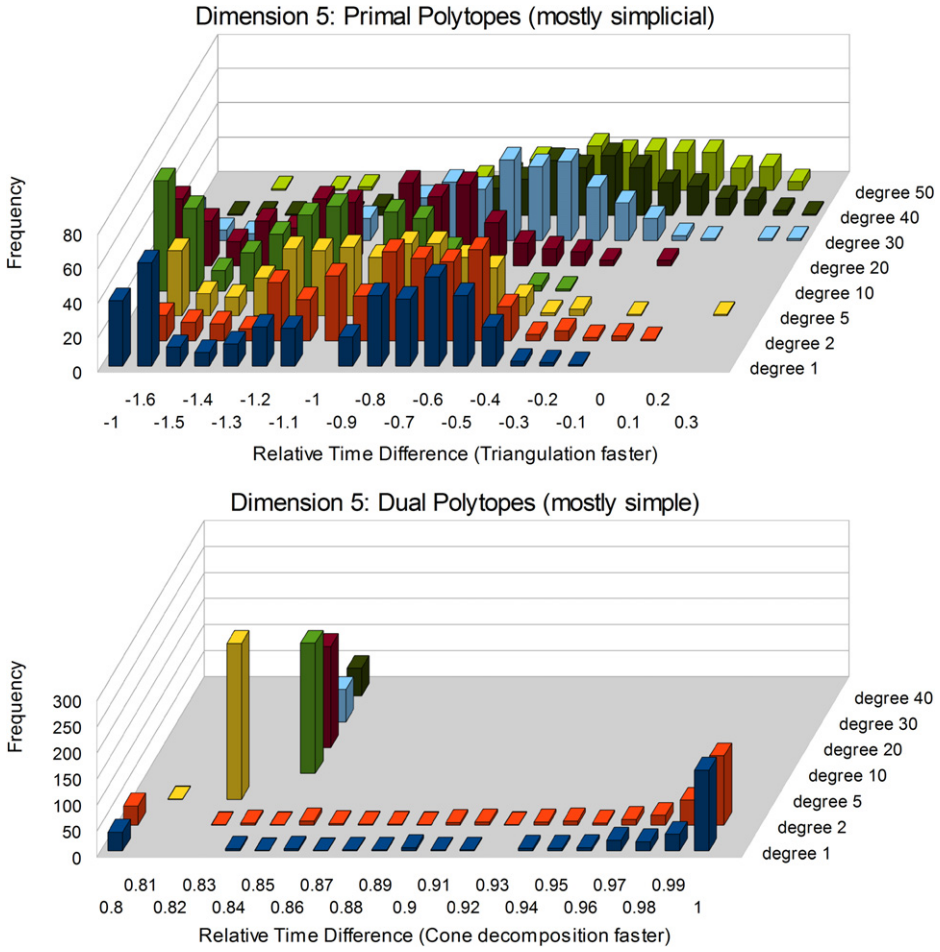


Fig. 3. Histogram of the relative time difference between the triangulation and cone decomposition methods for integrating over random polytopes in dimension 5.

between the two integration methods for monomials of eight different degrees (1, 2, 5, 10, 20, 30, 40, and 50). The description of all the histograms is the same as in the histograms for Figs. 3, 4, and 5.

The behavior we observed before for simple vs. simplicial polytopes still mostly holds for these tests, except we see that the two methods are closer to each other as the degree grows (the positive mass and the negative mass are comparable). The variation is then not as strong as before, but the polytopes are also not separated by simpliciality.

4.3. Volume experiments

Volume computation is an important special case of integration that has received attention by several researchers, thus we also tested the triangulation and cone decomposition methods on the same database of random polytopes and their duals, and on Ziegler's database to see the performance of volume evaluation.

Simple and simplicial polytopes Each test class contains 50 polytopes for each dimension and we only considered tests where both methods finished within 600 seconds for the same polytope. The histograms in Fig. 7 indicate the triangulation method is still faster for simplicial polytopes and the cone decomposition method is faster for simple polytopes (the mass of the histogram is concentrated in the positive part).

Zero-one polytopes Note that Tables 4 and 5 show *running times* for each method. Each row of the tables contains running times for a polytope and its dual, their numbers of vertices, and their dimension. The columns of the tables are roughly separated in half by the primal and dual information. We apply the triangulation and cone decomposition volume methods to Ziegler's database [41] and their duals. If a polytope did not contain the origin, we centered it so that its dual is defined. Again, we skipped non-full-dimensional polytopes and a few others that did not finish within 30 minutes. For each instance the faster timings are shown in bold. When computing volumes of primal polytopes in Ziegler's database, triangulation is faster more often. For finding the volume of the dual polytopes

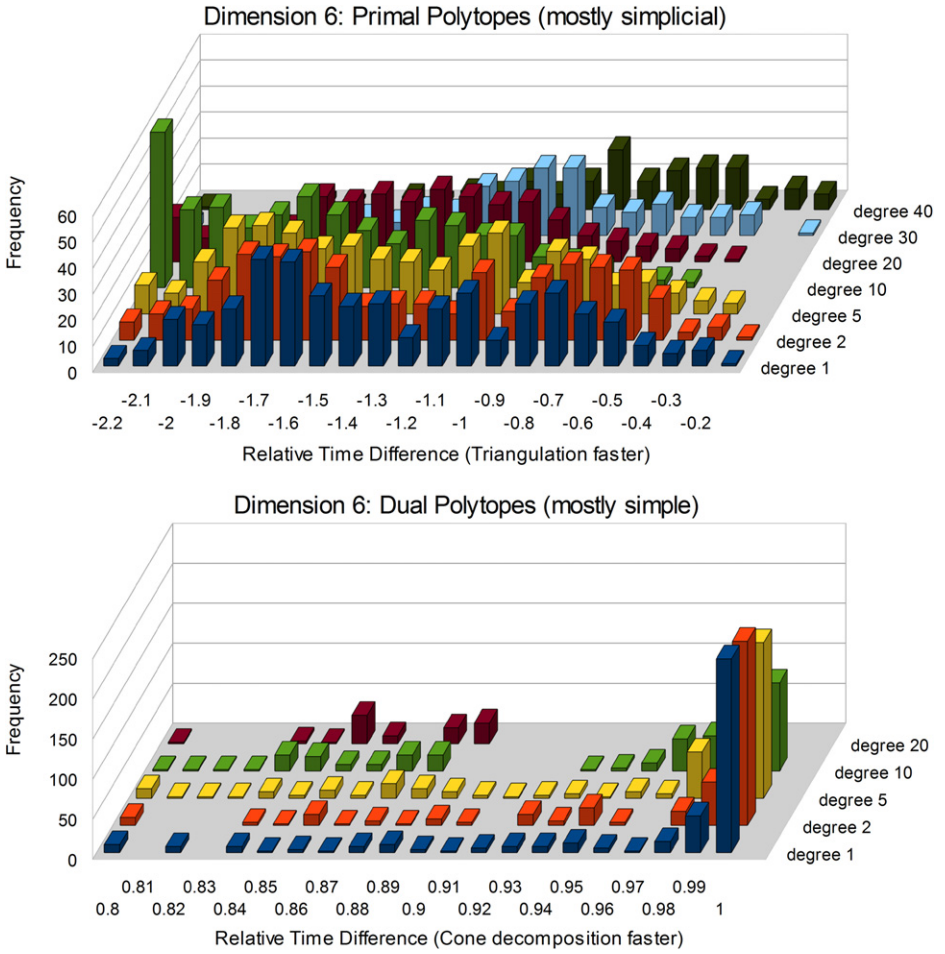


Fig. 4. Histogram of the relative time difference between the triangulation and cone decomposition methods for integrating over random polytopes in dimension 6.

there is no clear faster method if we just look at the number of times cone decomposition won, but even when it lost, it did not loose by much.

4.4. Comparison to other software

There are two general classes of algorithms for finding volumes and integrals over polytopes: numerical and exact. Numerical algorithms approximate the valuation on the polytope and involve error bounds, whereas exact algorithms do not contain a theoretical error term. However, exact algorithms may contain errors when they use finite digit integers or use floating-point arithmetic. In order to sidestep this problem, `LattE integrale` uses NTL's arbitrary length integer and rational arithmetic [43] compiled with the GNU Multiple Precision Arithmetic Library [44]. The obvious downside to exact arithmetic is speed, but this cost is necessary to obtain exact answers. In this section, we compare our exact algorithms with other software tools and algorithms that use numerical algorithms or non-exact arithmetic.

`Vinci` contains different algorithms for finding polytope volumes and in fact implemented the same decompositions we used in our software (see [16]).

We tested against `Vinci` 1.0.5, and Table 6 shows comparison of `LattE`'s cone decomposition method with `Vinci`'s HOT method (Hybrid Orthonormalisation Technique). We did not compare with Lasserre's method in `Vinci` because we found that `Vinci`'s Lasserre's method contained a bug: `Vinci` found the correct volumes for the cubes and random-hyperplane polytopes, but reported incorrect or negative volumes for most other polytopes in the Vinci database.

We ran `LattE`'s cone decomposition method starting from the h-representation. Because the HOT method requires both an h- and v-representation of the polytope, we also report the time used by CDD [26] to convert an h-representation to a v-representation. We also break down time spent in `LattE` for finding the vertices, finding the rays at each vertex, triangulation, and the time spent in the main cone decomposition integration method. The timings are, for `Vinci`, the sum of the entries in the first two columns and for `LattE`, the sum of the entries in the last four columns. We should note

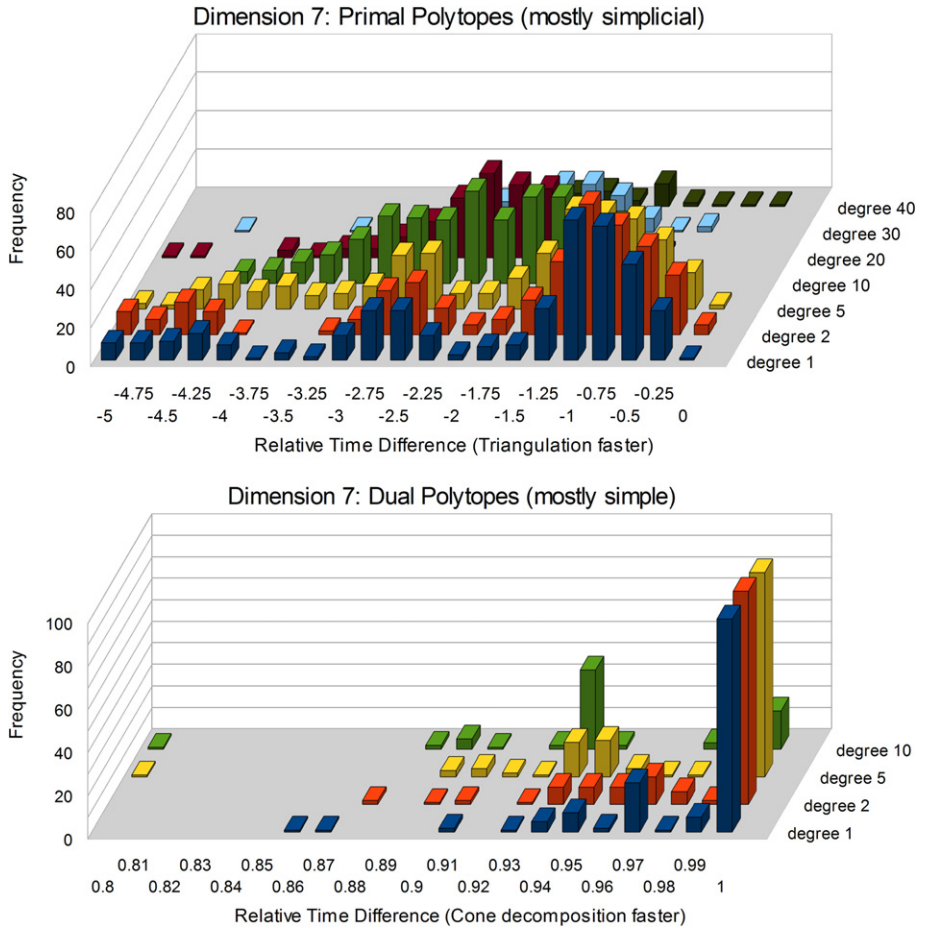


Fig. 5. Histogram of the relative time difference between the triangulation and cone decomposition methods for integrating over random polytopes in dimension 7.

that `LattE` and `Vinci` use different algorithms for finding the vertices, except in three cases `ccp5`, `cross 8`, `cross 9` where both use CDDlib, then numbers are essentially the same.

It is clear that the HOT method is faster and usually accurate when applied on the `Vinci` database (these polytopes are available from [16]), but because of non-exact arithmetic, it can give incorrect results. To explore the difference between exact and non-exact arithmetic we also tested how well `Vinci` can compute volumes of polytopes where each vertex contains small and large positive numbers. In Table 7, we tested the accuracy of `Vinci`'s HOT method on cyclic polytopes. We constructed these d-dimensional polytopes by taking the convex hull of $k+d$ points $(t, t^2, t^3, \ldots, t^d) \in \mathbb{Z}^d$ for $t = 5, 6, \ldots, 5+k+d-1$. For very small dimensions, the HOT method does well, but gives incorrect or zero volumes already in dimension six.

Another comparison we made was to the paper [3], where it is claimed that exact volumes are computed by integration. The authors report seven volumes for different polytopes. `LattE integrale`'s triangulation and cone decomposition method agree with their calculations except in the last case. For P_7 the correct volume is $1/622080 \approx 1.607510 \times 10^{-6}$ but they calculate 1.56439×10^{-6}. Presumably, because of non-exact arithmetic, their answer has only one digit of accuracy.

4.5. Numerical methods

M. Korenblit and E. Shmerling present a numerical integration algorithm in [39] which is based on a special decomposition of the integral into regions that have well-defined upper and lower limits of integration that, on an ordering of the variables, $x_1, x_2, \ldots, x_d$, x_i is expressed only in terms of $x_1, \ldots, x_{i-1}$. It is known that achieving such a decomposition is equivalent to the so-called Fourier–Motzkin elimination procedure [45] and as such it is of exponential complexity. The paper [39] gives an application to finding the probability a random-coefficient polynomial has one or two real roots in the interval $[-1, 1]$. To do this, they use their software to find the volume of a polytope. They calculate 2.79167; however, we verified that the correct volume is $31/12 = 2.58\bar{3}$ which gives their method one digit of accuracy.

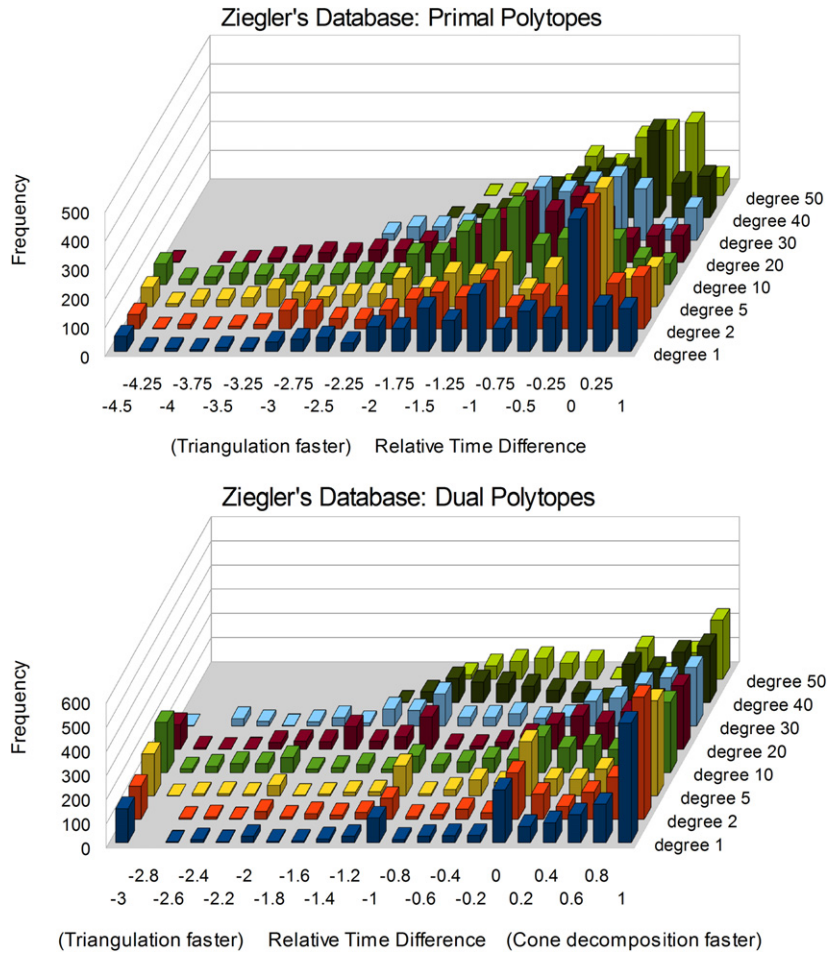


Fig. 6. Histogram of the relative time difference between the triangulation and cone decomposition methods for integrating over the polytopes in Ziegler's database.

A more interesting comparison is to CUBPACK, a Fortran 90 library which estimates the integral of a function (or vector of functions) over a collection of d-dimensional hyper-rectangles and simplices [38]. This comparison is very interesting because CUBPACK uses an adaptive grid to seek better performance and accuracy. All integration tests with CUBPACK in dimension d were done with a product of linear forms with a constant term over a random d-dimensional simplex where the absolute value of any coordinate in any vertex does not exceed 10. For example, we integrated a product of inhomogeneous linear forms such as $(\frac{1}{5}+2x-\frac{37}{100}y)(2-5x)$ over the simplex with vertices $(10,0)$, $(9,9)$, $(1,1)$. In Table 8, LattE was run 100 times to get the average running time, while CUBPACK was run 1000 times due to variance. Both the dimension and number of linear forms multiplied to construct the integrand were varied.

As shown in Table 8, LattE integrale tends to take less time, especially when the number of forms and dimension increases. The table does not show the high variance that CUBPACK has in its run times. For example, the 5-dimensional test case with 6 linear forms had a maximum running time of 2874.48 seconds, while the minimum running time was 0.05 seconds on a different random simplex. This contrasted starkly with LattE integrale, which had every test be within 0.01 (the minimum time discrepancy recognized by its timer) of every other test case.

CUBPACK differs from LattE integrale in that since it is based on numerical approximations, one can ask for different levels of precision. Table 9 illustrates how CUBPACK scales with requested precision on a single, 4-dimensional, 10 linear form test case. It seems that CUBPACK scales linearly with the inverse of the requested precision—10 times the precision requires about 3 times the work. All reported tests were done by expanding the multiplication of linear forms, and coding a Fortran 90 function to read in the resulting polynomial and evaluate it for specific points.

5. One application: Voting theory

Computation of integrals of polynomials over polyhedral regions is fundamental for many applications, including combinatorics, probability and statistics. In this last section we wish to demonstrate the power of LattE integrale by attacking problems arising in the social sciences. In the mathematical theory of voting it was observed that the probability

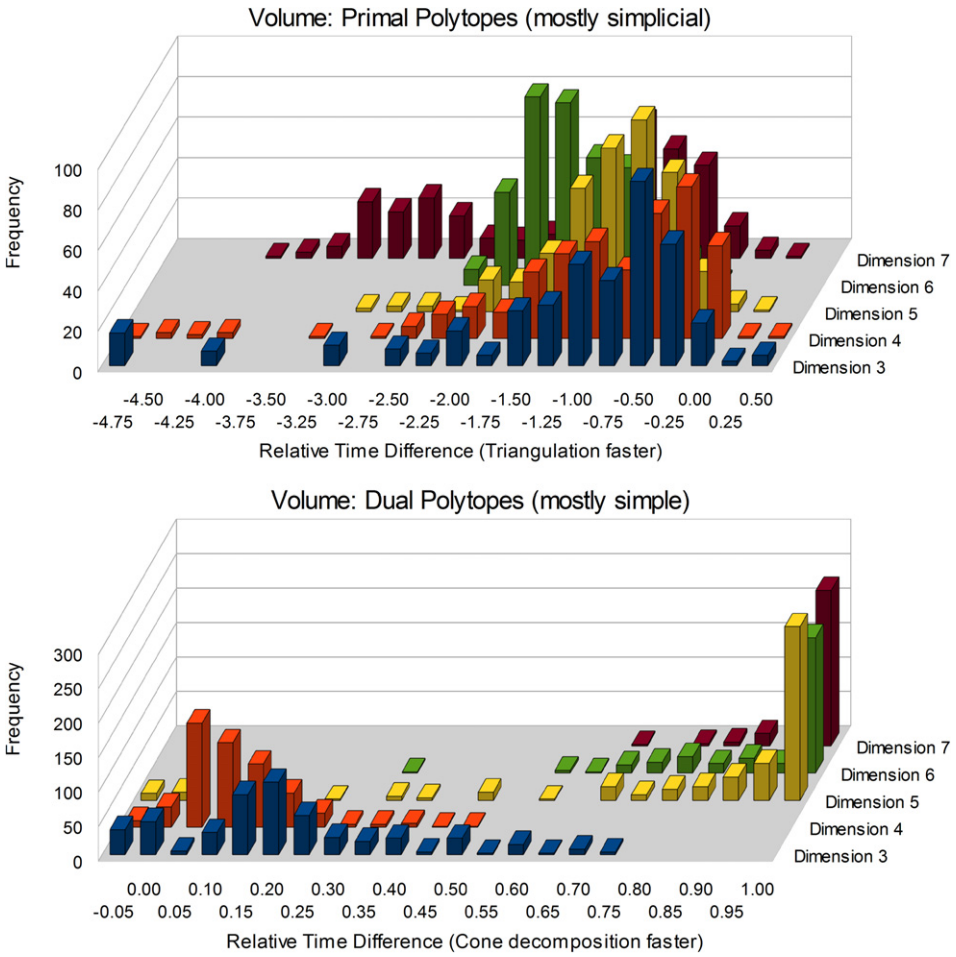


Fig. 7. Histogram of the relative time difference between the triangulation and cone decomposition methods for finding the volume of random polytopes.

Table 4
The triangulation vs. cone decomposition method for finding volumes in Ziegler's database: Part I.

Polytope	Dimension	Primal			Dual		
		Vertices	Time (s)		Vertices	Time (s)	
			Cone	Triang.		Cone	Triang.
3simp3simp.vrep.latte	6	44	**5.61**	6.10	32	**1.11**	1.15
cyclic_4_8.vrep.latte	4	8	0.09	**0.06**	20	**0.02**	0.10
neighborly_4_8.vrep.latte	4	8	0.12	**0.03**	20	**0.03**	0.06
SharirCube.vrep.latte	3	8	0.03	0.03	6	0.11	**0.02**
HC_6-32.vrep.latte	6	32	2.29	**2.06**	44	3.25	**3.22**
HC_7-64.vrep.latte	7	64	**13.42**	75.85	78	**61.68**	762.12
HC_8-128.vrep.latte	8	128	**85.85**	–	144	**15 007.50**	–
MJ_16-17.vrep.latte	16	17	2.60	**2.48**	17	0.07	**0.04**
OA_5-10.vrep.latte	5	10	0.22	**0.08**	22	0.18	**0.11**
OA_5-18.vrep.latte	5	18	0.46	**0.32**	19	0.34	**0.10**
OA_5-24.vrep.latte	5	24	0.81	**0.58**	18	0.22	**0.13**
OA_6-13.vrep.latte	6	13	0.53	**0.20**	56	**0.52**	5.37
OA_7-18.vrep.latte	7	18	3.36	**0.82**	146	**13.96**	1827.83
OA_8-25.vrep.latte	8	25	38.55	**10.44**	524	**4116.93**	–
OA_9-33.vrep.latte	9	33	–	**648.77**	1870	–	–
AS_6-18.vrep.latte	6	18	1.26	**0.52**	121	**1.40**	65.63
BIR3_4-6.vrep.latte	4	6	0.12	**0.02**	9	0.01	0.01
BIR4_9-24.vrep.latte	9	24	6.26	**2.22**	16	1.42	**0.17**
BIR5_16-120.vrep.latte	16	120	–	–	25	–	**488.78**

Table 5
The triangulation vs. cone decomposition method for finding volumes in Ziegler's database: Part II.

Polytope	Dimension	Primal			Dual		
		Vertices	Time (s)		Vertices	Time (s)	
			Cone	Triang.		Cone	Triang.
CF_10-11.vrep.latte	10	11	0.37	**0.33**	11	0.03	**0.00**
CF_4-5.vrep.latte	4	5	0.01	**0.02**	5	0.02	0.02
CF_5-6.vrep.latte	5	6	0.04	**0.03**	6	0.00	0.00
CF_6-7.vrep.latte	6	7	**0.04**	0.05	7	0.01	**0.00**
CF_7-8.vrep.latte	7	8	**0.08**	0.09	8	0.02	0.02
CF_8-9.vrep.latte	8	9	0.14	**0.13**	9	0.02	**0.01**
CF_9-10.vrep.latte	9	10	0.22	**0.20**	10	**0.01**	0.02
CRO_3-6.vrep.latte	3	6	0.04	**0.01**	8	**0.00**	0.02
CRO_4-8.vrep.latte	4	8	0.12	**0.05**	16	**0.00**	0.03
CRO_5-10.vrep.latte	5	10	0.17	**0.10**	32	**0.01**	0.33
CUT3_3-4.vrep.latte	3	4	0.00	0.01	4	0.01	0.01
CUT4_6-8.vrep.latte	6	8	0.16	**0.06**	16	**0.00**	0.05
CUT5_10-16.vrep.latte	10	16	2.72	**0.94**	56	**38.38**	2046.74
CYC_5-8.vrep.latte	5	8	0.11	**0.05**	20	**0.00**	0.10
EG_5-12.vrep.latte	5	12	0.32	**0.13**	40	**0.15**	0.79
EQU_5-7a.vrep.latte	5	7	0.06	**0.05**	10	**0.01**	0.04
EQU_5-7b.vrep.latte	5	7	0.09	**0.05**	10	0.01	**0.00**
HAM_8-16.vrep.latte	8	16	1.57	**0.60**	256	**0.15**	–
HC_3-4.vrep.latte	3	4	0.03	**0.00**	4	0	0
HC_4-8.vrep.latte	4	8	0.15	**0.05**	16	**0.02**	0.04
HC_5-16.vrep.latte	5	16	0.48	**0.24**	26	0.32	**0.25**
CNG_5-6a.vrep.latte	5	6	0.04	**0.03**	6	0.02	**0.01**
MJ_32-33.vrep.latte	32	33	**82.90**	83.86	33	1.44	**0.14**
CNG_5-6b.vrep.latte	5	6	**0.02**	0.03	6	**0.01**	0.02

Table 6
Time breakdown between `LattE integrale`'s cone decomposition and `Vinci`'s HOT method with CDD.

Polytope	`Vinci`		`LattE`			
	HOT	`Cddlib`	Vertices	Rays	Triang.	Cone
cube-9	0.03	0.08	0.02[1]	0.06	0.02	0.02
cube-10	0.11	0.18	0.04[1]	0.15	0.02	0.06
cube-14	141.65	7.99	1.24[1]	4.67	0.69	1.26
rh-8-20	0.13	0.89	0.11[1]	0.49	0.04	7.00
rh-8-25	0.43	2.63	0.32[1]	1.14	0.14	80.25
rh-10-20	0.96	2.21	0.25[1]	1.80	0.14	98.26
rh-10-25	5.71	12.49	1.07[1]	8.80	0.44	3989.25
$CC_8(9)$	0.04	0.22	0.07[1]	0.12	0.39	0.40
$CC_8(10)$	0.08	0.52	0.16[1]	0.22	0.97	0.88
$CC_8(11)$	0.18	1.18	0.03[1]	0.42	1.84	1.76
ccp 5	0.00	0.07	0.09[2]	0.00	0.10	0.09
cross 8	0.00	0.39	0.50[2]	0.00	0.06	0.04
cross 9	0.00	1.57	2.15[2]	0.00	0.12	0.11
rv-8-10	0.00	0.08	0.00[1]	0.03	0.02	0.00
rv-8-11	0.00	1.99	0.08[1]	0.23	0.03	0.01
rv-10-12	0.00	0.12	0.02[1]	0.09	0.04	0.01
rv-10-14	0.00	1061.49	29.50[1]	64.96	0.10	0.07

[1] Computed with `4ti2`.
[2] Computed with `Cddlib`.

of events that can lead to singular election outcomes can be modeled as the number of lattice points inside a polytope divided by the number of lattice points of a simplex (see [46] and the references therein). Note that both the polytope and the simplex dilate proportional to the number n of voters. It is very well-known from the theory of Ehrhart functions that the counting functions are quasipolynomials (polynomials with periodic coefficients) that depend on n [22]. Thus when the quotient is evaluated the answer is asymptotically equal to the quotient of the leading coefficients of the two Ehrhart quasipolynomials involved.

Table 7
Comparison between `LattE integrale` and `Vinci` on finding the volume of cyclic polytopes.

Dimension	Tool	k				
		1	2	3	4	5
2	LattE	1	4	10	20	35
	Vinci	1	4	10	20	35
3	LattE	2	16	70	224	588
	Vinci	1.999999999988	15.99999999999	69.99999999991	224.0000000006	587.9999999986
4	LattE	12	192	1512	8064	33 264
	Vinci	11.99999993201	191.9999999913	1511.99999999	8063.999999892	33 263.99999989
5	LattE	288	9216	133 056	1 216 512	8 154 432
	Vinci	287.9996545868	9216.000252236	133 055.9883262	1 216 511.998301	8 154 431.872519
6	LattE	34 560	2 211 840	59 304 960	948 879 360	10 600 761 600
	Vinci	34 561.951223	1 935 359.822684	58 060 819.63341	885 910 920.3761	10 336 274 212.34
7	LattE	24 883 200	3 185 049 600	160 123 392 000	4 554 620 928 000	86 502 214 656 000
	Vinci	25 744 201.0524	0	0	0	0

Table 8
Average time for `LattE integrale` and `CUBPACK` for integrating products of inhomogeneous linear forms over simplices.

Dimension	Tool	Number of linear factors									
		1	2	3	4	5	6	7	8	9	10
2	LattE	**0.0001**	**0.0002**	**0.0005**	**0.0008**	**0.0009**	**0.0019**	0.0038	0.0048	0.0058	**0.0089**
	CUBPACK	0.0027	0.0014	0.0016	0.0022	0.0064	0.0052	**0.0014**	**0.0002**	**0.0026**	0.0213
3	LattE	**0.0002**	**0.0005**	**0.0009**	**0.0016**	**0.0043**	**0.0073**	**0.0144**	**0.0266**	**0.0453**	**0.0748**
	CUBPACK	0.0134	0.0145	0.0018	0.0054	0.0234	0.0219	0.0445	0.0699	0.1170	0.2420
4	LattE	**0.0003**	**0.0012**	**0.0018**	0.0044	0.0121	**0.0274**	**0.0569**	**0.1094**	**0.2247**	**0.4171**
	CUBPACK	0.0042	0.0134	0.0028	**0.0019**	**0.0076**	0.5788	4.7837	4.3778	22.3530	54.3878
5	LattE	**0.0005**	**0.0008**	0.0048	**0.0108**	0.0305	**0.0780**	**0.0800**	–	–	–
	CUBPACK	0.0013	0.0145	0.0048	0.0217	**0.0027**	37.0252	128.2242	–	–	–

Table 9
`CUBPACK` scaling with increased relative accuracy. "Relative Error" is a user-specified parameter of `CUBPACK`; "Expected Error" is an estimate of the absolute error, produced by `CUBPACK`'s error estimators. Finally, the "Actual Error" is the difference of `CUBPACK`'s result to the exact integral computed with `LattE integrale`.

Relative Error	Result	Expected Error	Actual Error	# Evaluations	Time (s)
10^{-2}	1 260 422 511.762	9 185 366.414	94 536.015	4467	0.00
10^{-3}	1 260 507 955.807	1 173 478.333	9091.974	9820	0.01
10^{-4}	1 260 516 650.281	123 541.490	397.496	34 411	0.04
10^{-5}	1 260 517 042.311	12 588.455	5.466	104 330	0.10
10^{-6}	1 260 517 047.653	1257.553	0.124	357 917	0.31
10^{-7}	1 260 517 047.691	126.042	0.086	1 344 826	1.16
10^{-8}	1 260 517 047.775	12.601	0.002	4 707 078	4.15
10^{-9}	1 260 517 047.777	1.260	$< 10^{-3}$	16 224 509	14.09
10^{-10}	1 260 517 047.777	0.126	$< 10^{-3}$	55 598 639	48.73

To illustrate this, consider the following example from [46]: There are three candidates a, b and c, and let the preference orders of the $n = \sum_{i=1}^{6} n_i$ voters be

$$abc(n_1),\ acb(n_2),\ bac(n_3),\ bca(n_4),\ cab(n_5),\ cba(n_6).$$

Here, there are n_1 voters who rank candidate a as first, b second, and c third, n_2 voters who rank b first, a second, c third, etc. Under simple plurality voting, the candidate with the most votes wins. But in a plurality runoff system, if no candidate wins more than 50% of the vote, the two candidates with the highest vote count advance to a second voting round. In [46], the authors compute the probability that the simple plurality and plurality runoff systems give different winners. This requires setting up a system of equations that describes the situation that a wins by plurality but, using plurality runoff b obtains higher score than c and a majority of voters then prefer b to a.

$$0 < n_1 + n_2 - n_3 - n_4,$$
$$0 < n_3 + n_4 - n_5 - n_6,$$

$$-\frac{1}{2} < -n_1 - n_2 - n_5,$$
$$1 = n_1 + n_2 + n_3 + n_4 + n_5 + n_6,$$
$$0 \leqslant n_i, \quad i = 1, \ldots, 6.$$

This is done by computing the Ehrhart quasipolynomial of the above polyhedron and dividing by the Ehrhart quasipolynomial of the simplex $\{(n_1, n_2, \ldots, n_6)\colon\ n_1 + n_2 + \cdots + n_6 = 1, n_i \geqslant 0\}$ (which is the space of all possible voting possibilities assuming that all 6 rankings of three candidates are equally likely). All must be multiplied by 6 because the plurality winner may be a, b or c and the second position could be c not just b. As the authors observed, asymptotically, the leading coefficients of these two quasipolynomials is all that matter. In the concluding remarks the authors then posed the challenge of pushing the limit of such calculations for four-candidate elections which they observed is too big for their calculations.

However, we have observed their calculation can be further simplified and accelerated because it is very well known (see [22]) that the leading coefficient of the quasipolynomial is always equal to the volume of the polytope with $n = 1$, thus one can directly perform the calculation of the volume (the volume of the simplex is well known) and do a quotient of two numbers. The key step in finding the probabilities requires only finding the volume directly. Our algorithm corroborates that for the previous example the volume is $\frac{71}{414\,720}$, and when multiplied by 6×120 gives the probability these two voting systems give different winners for a large population: 12.33%.

Using our code for exact integration we tackled the same problem for four candidates. In this case we have 24 variables associated to the orderings

$$abcd(n_1), abdc(n_2), acbd(n_3), acdb(n_4), adbc(n_5), adcb(n_6),$$
$$bacd(n_7), badc(n_8), bcad(n_9), bcda(n_{10}), bdac(n_{11}), bdca(n_{12}),$$
$$cabd(n_{13}), cadb(n_{14}), cbad(n_{15}), cbda(n_{16}), cdab(n_{17}), cdba(n_{18}),$$
$$dabc(n_{19}), dacb(n_{20}), dbac(n_{21}), dbca(n_{22}), dcab(n_{23}), dcba(n_{24}).$$

The equations and inequalities associated to the problem codify the following facts: The sum of all variables n_i must be equal to the total number of voters. We have four inequalities expressing that when a is the plurality winner, b obtained a score higher than c, and c obtained a score higher than d, thus

$$n_1 + n_2 + n_3 + n_4 + n_5 + n_6 > n_7 + n_8 + n_9 + n_{10} + n_{11} + n_{12},$$
$$n_7 + n_8 + n_9 + n_{10} + n_{11} + n_{12} > n_{13} + n_{14} + n_{15} + n_{16} + n_{17} + n_{18},$$
$$n_{13} + n_{14} + n_{15} + n_{16} + n_{17} + n_{18} > n_{19} + n_{20} + n_{21} + n_{22} + n_{23} + n_{24}.$$

These inequalities assume that the order was $a > b > c > d$ but the answer we get should be multiplied by $4! = 24$ to take into account other possible orders. Finally, we have to express the fact that but a majority of voters prefer b over a and that a did not achieve more than 50 percent of the vote ($n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_{13} + n_{14} + n_{17} + n_{19} + n_{20} + n_{23} < n/2$). The volume of this polytope when $n = 1$ is

$$\frac{2\,988\,379\,676\,768\,359}{7\,552\,997\,065\,814\,637\,134\,660\,504\,411\,827\,077\,120\,000}.$$

The probability is then the volume times 4! divided by the volume of the simplex

$$\left\{(n_1, n_2, \ldots, n_{24})\colon\ \sum_i n_i = 1,\ n_i \geqslant 0\right\},$$

which equals $\frac{1}{23!}$. After a minute of computation using the cone decomposition method, we obtain the probability is 12.27%.

We can continue the example by considering the same problem for five candidates. The five-candidate polytope has $5! = 120$ variables. However, after LRS [25] enumerated over 12.5 million vertices, we terminated the program and decided the polytope is beyond our limits. We close by mentioning that after the first version of this paper was made available other authors proposed new ideas to compute these values using symmetries of the problem. See [47].

Acknowledgements

We thank the anonymous referees for the excellent suggestions that we received from them and greatly improved the presentation of the paper. They read the paper in great detail and we are grateful for the attention received. We are truly grateful to our collaborators V. Baldoni, N. Berline, and M. Vergne for important discussions that led to this software. The senior authors, J.A. De Loera and M. Köppe, were partially supported by NSF grants DMS-0914107 and DMS-0914873. Most of the students were supported by those grants and by summer fellowships provided through the UC Davis VIGRE grant DMS-0636297.

References

[1] B. Li, The moment calculation of polyhedra, Pattern Recogn. 26 (8) (1993) 1229–1233.
[2] B. Mirtich, Fast and accurate computation of polyhedral mass properties, J. Graphics Tools 1 (2) (1996) 31–50.
[3] H. Ong, H. Huang, W. Huin, Finding the exact volume of a polyhedron, Adv. Eng. Softw. 34 (6) (2003) 351–356, http://dx.doi.org/10.1016/S0965-9978(03)00030-9.
[4] B. Sturmfels, Gröbner Bases and Convex Polytopes, Univ. Lecture Ser., vol. 8, American Mathematical Society, 1996.
[5] N. Berline, M. Vergne, Local Euler–Maclaurin formula for polytopes, Moscow Math. J. 7 (2007) 355–386.
[6] M. Bronstein, Symbolic Integration I – Transcendental Functions, vol. 1, Springer, Heidelberg, 2005.
[7] M.E. Dyer, A.M. Frieze, On the complexity of computing the volume of a polyhedron, SIAM J. Comput. 17 (5) (1988) 967–974.
[8] G. Brightwell, P. Winkler, Counting linear extensions, Order 8 (3) (1991) 225–242.
[9] L. Khachiyan, Complexity of polytope volume computation, in: New Trends in Discrete and Computational Geometry, in: Algorithms Combin., vol. 10, Springer, Berlin, 1993, pp. 91–101.
[10] J. Lawrence, Polytope volume computation, Math. Comp. 57 (195) (1991) 259–271.
[11] P. Gritzmann, V. Klee, On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes, Universität Trier, Mathematik/Informatik, Forschungsbericht 94-07.
[12] G. Elekes, A geometric inequality and the complexity of computing volume, Discrete Comput. Geom. 1 (4) (1986) 289–292.
[13] L. Rademacher, Approximating the centroid is hard, in: Proceedings of 23rd Annual ACM Symposium of Computational Geometry, Gyeongju, South Korea, 2007, pp. 302–305.
[14] J. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, J. Wu, A users guide for latte integrale v1.5, available from URL http://www.math.ucdavis.edu/~latte/, 2011.
[15] V. Baldoni, N. Berline, J.A. De Loera, M. Köppe, M. Vergne, How to integrate a polynomial over a simplex, Math. Comp. 80 (273) (2011) 297–325, http://dx.doi.org/10.1090/S0025-5718-2010-02378-6.
[16] B. Büeler, A. Enge, K. Fukuda, Exact volume computation for polytopes: A practical study, in: G. Kalai, G.M. Ziegler (Eds.), Polytopes – Combinatorics and Computation, in: DMV-Semin., vol. 29, Birkhäuser Verlag, Basel, 2000.
[17] A.I. Barvinok, Computation of exponential integrals, Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) Teor. Slozhn. Vychisl. 5 (1991) 149–162, 175–176, translation in J. Math. Sci. 70 (4) (1994) 1934–1943.
[18] A.I. Barvinok, Partition functions in optimization and computational problems, Algebra i Analiz 4 (1992) 3–53, translation in St. Petersburg Math. J. 4 (1) (1993) 1–49.
[19] M. Brion, Points entiers dans les polyèdres convexes, Ann. Sci. Éc. Norm. Super. 21 (4) (1988) 653–663.
[20] J.B. Lasserre, An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbf{R}^n$, J. Optim. Theory Appl. 39 (1983) 363–377, http://dx.doi.org/10.1007/BF00934543.
[21] M. Beck, C. Haase, F. Sottile, Formulas of Brion, Lawrence, and Varchenko on rational generating functions for cones, Math. Intelligencer 31 (2009) 9–17, http://dx.doi.org/10.1007/s00283-008-9013-y.
[22] A.I. Barvinok, Integer Points in Polyhedra, Zürich Lect. Adv. Math., European Mathematical Society (EMS), Zürich, Switzerland, 2008.
[23] A.I. Barvinok, J.E. Pommersheim, An algorithmic theory of lattice points in polyhedra, in: L.J. Billera, A. Björner, C. Greene, R.E. Simion, R.P. Stanley (Eds.), New Perspectives in Algebraic Combinatorics, in: Math. Sci. Res. Inst. Publ., vol. 38, Cambridge Univ. Press, Cambridge, 1999, pp. 91–147.
[24] M. Beck, S. Robins, Computing the Continuous Discretely: Integer-Point Enumeration in Polyhedra, Undergrad. Texts Math., Springer, 2007.
[25] D. Avis, lrs: A revised implementation of the reverse search vertex enumeration algorithm, http://cgm.cs.mcgill.ca/~avis/C/lrs.html, 1999.
[26] K. Fukuda, A. Prodon, Double description method revisited, in: M. Deza, R. Euler, I. Manoussakis (Eds.), Combinatorics and Computer Science, in: Lecture Notes in Comput. Sci., vol. 1120, Springer, Berlin/Heidelberg, 1996, pp. 91–111, http://dx.doi.org/10.1007/3-540-61576-8_77.
[27] J. Alexander, A. Hirschowitz, Polynomial interpolation in several variables, J. Algebraic Geom. 4 (1995) 201–222.
[28] M.C. Brambilla, G. Ottaviani, On the Alexander–Hirschowitz theorem, J. Pure Appl. Algebra 212 (5) (2008) 1229–1251, available at arXiv: math.AG/0701409.
[29] E. Carlini, M. Catalisano, A. Geramita, The solution to Waring's problem for monomials, eprint arXiv:1110.0745v1 [math.AC], 2011.
[30] P. Henrici, Applied and Computational Complex Analysis, Vol. 1, Power Series—Integration—Conformal Mapping—Location of Zeros, Wiley Classics Lib., John Wiley & Sons Inc., New York, 1988, Reprint of the 1974 original, a Wiley–Interscience Publication.
[31] A.I. Barvinok, K. Woods, Short rational generating functions for lattice point problems, J. Am. Math. Soc. 16 (4) (2003) 957–979.
[32] J.A. De Loera, R. Hemmecke, J. Tauzer, R. Yoshida, Effective lattice point counting in rational convex polytopes, J. Symbolic Comput. 38 (4) (2004) 1273–1302.
[33] M. Köppe, A primal Barvinok algorithm based on irrational decompositions, SIAM J. Discrete Math. 21 (1) (2007) 220–236, http://dx.doi.org/10.1137/060664768.
[34] M. Gastineau, J. Laskar, Development of TRIP: Fast sparse multivariate polynomial multiplication using burst tries, in: V. Alexandrov, G. van Albada, P. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, in: Lecture Notes in Comput. Sci., vol. 3992, Springer, Berlin/Heidelberg, 2006, pp. 446–453, http://dx.doi.org/10.1007/11758525_60.
[35] K. Fukuda, `cddlib`, version 094f, available from URL http://www.ifor.math.ethz.ch/~fukuda/cdd_home/, 2008.
[36] 4ti2 team, 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces, available at www.4ti2.de.
[37] G. Hanrot, P. Zimmermann, A long note on Mulders' short product, Rapport de recherche RR-4654, INRIA, available at http://hal.inria.fr/inria-00071931, 2002.
[38] R. Cools, A. Haegemans, Algorithm 824: CUBPACK: a package for automatic cubature; framework description, ACM Trans. Math. Software 29 (3) (2003) 287–296.
[39] M. Korenblit, E. Shmerling, Algorithm and software for integration over a convex polyhedron, in: A. Iglesias, N. Takayama (Eds.), Mathematical Software – ICMS 2006, in: Lecture Notes in Comput. Sci., vol. 4151, Springer, Berlin, Heidelberg, 2006, pp. 273–283, http://dx.doi.org/10.1007/11832225_28.
[40] E. Gawrilow, M. Joswig, Polymake: A framework for analyzing convex polytopes, in: G. Kalai, G.M. Ziegler (Eds.), Polytopes – Combinatorics and Computation, Birkhäuser, 2000, pp. 43–74.
[41] G. Ziegler, Database of 0–1 polytopes, http://www.math.tu-berlin.de/polymake/examples/, accessed 07/19/2011.
[42] J. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, J. Wu, Software for exact integration of polynomials over polyhedra: Online supplement, available from URL http://www.math.ucdavis.edu/~latte/theory/SoftwareExactIntegrationPolynomialsPolyhedraOnlineSupplement.pdf, 2012.
[43] V. Shoup, NTL, a library for doing number theory, available from URL http://www.shoup.net/ntl/, 2005.
[44] The GNU multiple precision arithmetic library, http://gmplib.org/.
[45] M. Schechter, Integration over a polyhedron: An application of the Fourier–Motzkin elimination method, Amer. Math. Monthly 105 (3) (1998) 246–251.
[46] D. Lepelley, A. Louichi, H. Smaoui, On Ehrhart polynomials and probability calculations in voting theory, Soc. Choice Welf. 30 (2008) 363–383, http://dx.doi.org/10.1007/s00355-007-0236-1.
[47] A. Schürmann, Exploiting polyhedral symmetries in social choice, available from URL http://front.math.ucdavis.edu/1109.1545, 2012.