Exact Volume Computation for Polytopes: A Practical Study

B. Büeler, A. Enge and K. Fukuda
Institute for Operations Research
Swiss Federal Institute of Technology
CH-8092 Zurich, Switzerland

September 26, 1997

Abstract

We discuss some known volume computation algorithms for convex d-polytopes. By incorporating the detection of simplicial faces and a storing/reusing scheme for face volumes we propose practical and theoretical improvements for two of the algorithms. Finally we present a hybrid method. The behaviour of the algorithms is theoretically analysed for hypercubes and practically tested on a wide range of polytopes, where the new hybrid method proves to be superior.

1 Introduction

A *convex polytope* P is the convex hull $\text{conv}(V)$ of a finite set $V = \{v_1, v_2, \ldots, v_n\}$ of points in $\mathbb{R}^d$. Equivalently, it is a bounded subset of $\mathbb{R}^d$ which is the intersection of a finite set of half spaces. We shall omit 'convex' since we only deal with such polytopes. When $P = \text{conv}(V)$, V is called a *vertex representation* or simply *$\mathcal{V}$–representation* of P. When $P = \{x | A\ x \leq b\}$ for some $m \times d$ real matrix A and m-vector b, the pair (A, b) is called a *halfspace representation* or simply *$\mathcal{H}$–representation* of P. In this paper we treat the *volume computation problem* as to compute the volume $\text{Vol}(P)$ of a polytope P given by both $\mathcal{V}$– and $\mathcal{H}$–representations. We also briefly discuss the problem when only one representation is provided.

The relevance of volume computation is nicely demonstrated in the excellent survey [6], where also several basic approaches for volume computation are discussed in depth. In this paper we consider five methods and classify them into two groups. Triangulation methods on the one hand decompose the polytope into simplices for which the volume is easily computed and summed up; members of this group are the boundary triangulation, Delaunay triangulation and Cohen & Hickey's triangulation. The signed decomposition methods, on the other hand, decompose a given polytope into signed simplices such that the signed sum of their volumes is the volume of the polytope; in this group we consider Lasserre's and Lawrence's method. As explained in Section 2, these methods are in a sense dual to the triangulation methods. A basic algorithm excluded from our investigations is the incremental construction of a triangulation using the beneath-beyond method [10, 4] for convex hull computation (see also [6, Section 4]). Concerning random approximation algorithms (see [7]) we know so far of no implementation and have therefore ignored this highly interesting new approach.

By implementing Cohen & Hickey's and Lasserre's method we found a significant potential for improvements, and consequently those versions which exploit part of this potential are labelled 'revised'. All methods mentioned so far, revised or not, can be called 'pure', meaning that they are either a pure triangulation or pure signed decomposition method. In view of the strengths and weaknesses of both groups we developped hybrid approaches combining strong sides from triangulation and signed-decomposition methods.

While experiments showed differences among the algorithms in CPU-time and memory up to several orders of magnitude, the theoretical analysis is difficult. One major source is related to the difficulty of classifying polytopes, for example by a satiesfying measure for 'how simple' or 'how simplicial' a polytope is. Another difficulty is introduced by the recursive or iterative structure of the algorithms; it is usually not only the structure of the original polytope, but of all intermediate polytopes generated in the solution process which determine the complexity. Despite these difficulties we give complexity estimates for some concrete algorithms and polytopes accepting the limited relevance and hoping that it is a first step in the explanation of the real algorithmic behavior for a wider class of polytopes. In this state experiments may prove useful to gain a better understanding of polytopal structures and the behaviour of different algorithmic concepts.

Even though the experimental time and space complexity dependents highly on the quality of implementation, it can be expected that triangulation algorithms work best on near simplicial polytopes, whereas signed decomposition algorithms are preferable on near simple polytopes. It is interesting, however, *how* poor the algorithms behave in the opposite cases. A second experimental finding is the overall convincing quality of the hybrid approach, to which the laurels can be awarded in our competition. Notice, however, that the algorithms differ in the input they require; because the transformation $\mathcal{V} \rightleftarrows \mathcal{H}$ can be prohibitively costly for some polytopes, this can influence decisively the choice of algorithms. Though highly depending on the concrete structure of the polytope under consideration, one can expect tractability on workstations for polytopes with up to 10^3 hyperplanes, 10^4 vertices and 15 dimensions, requiring memory up to the range of 10^2– 10^3 MB.

The paper is organised as follows. In Section 2 we discuss two basic ideas of volume computation together with their dual relation, allowing a deeper understanding of the complexity observed. Section 3 is devoted to the five pure algorithms we have chosen from the literature. Then, in Section 4, we propose a number of improvements for Cohen & Hickey's and Lasserre's method and present a hybrid algorithm. We conclude in Section 5 by presenting experimental results for seven classes of polytopes. Finally, as a byproduct of our work, the code of all algorithms discussed are publicly available; the sources are given in the Appendix.

2 The Basic Idea of the Pure Methods and their Dual Relation

All known algorithms for exact volume computation decompose a given polytope into simplices, and thus they all rely, explicitly or implicitly, on the volume formula of a simplex:

$$\mathrm{Vol}(\Delta(v_0,\dots,v_d)) = \frac{|\det(v_1 - v_0,\dots,v_d - v_0)|}{d!},$$

where $\Delta(v_0,\dots,v_d)$ denotes the simplex in $\mathbb{R}^d$ with vertices $v_0,\dots,v_d \in \mathbb{R}^d$. There are two types of methods for volume computation, depending on how a given polytope P is decomposed into simplices. As we explain later, these two algorithm classes are related by polarity of convex polytopes.

2.1 Triangulation methods

Let P be a given polytope in $\mathbb{R}^d$. When P is triangulated into simplices Δ_i ($i = 1, \ldots, s$), we have $P = \cup_{i=1}^{s} \Delta_i$, and the volume of P is simply the sum of the volumes of the simplices:

$$\mathrm{Vol}(P) = \sum_{i=1}^{s} \mathrm{Vol}(\Delta_i). \tag{1}$$

A triangulation method for volume computation generates a triangulation, explicitly or implicitly, and computes the volume by (1). Figure 1 illustrates a triangulation using an interior point e; it is often used when the boundary of a polytope is easily triangulated or already triangulated as in case of simplicial polytopes. Besides such a boundary triangulation method we will also consider the Delaunay triangulation and Cohen & Hickey's combinatorial triangulation by dimensional recursion. A first important difference is that the former two methods need only a $\mathcal{V}$–representation while the last method requires both the $\mathcal{V}$– and $\mathcal{H}$–representations.

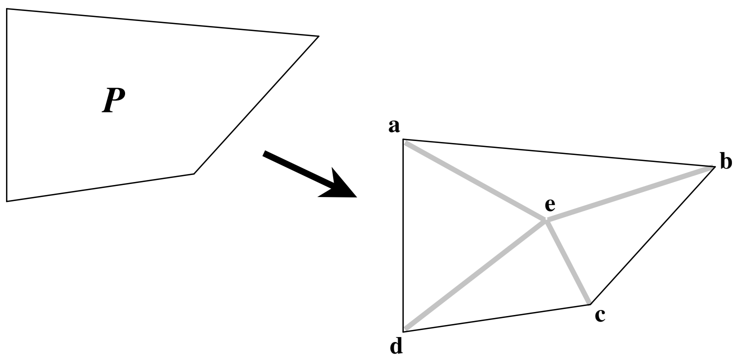

Figure 1: Boundary triangulation

2.2 Signed decomposition methods

Instead of triangulating a polytope P, one can decompose P into 'signed' simplices whose signed union is exactly P. More specifically, we represent P as a signed union of simplices Δ_i, $i = 1, \ldots, s$,

$$P = \biguplus_{i=1}^{s} \sigma_i \Delta_i, \tag{2}$$

where σ_i is either $+1$ or -1. Equation (2) means that each point $x \in P$ appears in exactly one more positive Δ_i than in negative Δ_i's, and each point x not in P appears equally often in positive and in negative Δ_i's. Then the volume of P is

$$\mathrm{Vol}(P) = \sum_{i=1}^{s} \sigma_i \, \mathrm{Vol}(\Delta_i). \tag{3}$$

In this paper we consider two signed decomposition methods, Lawrence's decomposition as illustrated in Figure 2, and Lasserre's decomposition. Fundamental to Lawrence's method for simple polytopes is the introduction of an extra hyperplane e which is not parallel to any edge of the polytope. The simplices in the signed decomposition are generated by passing once through each vertex, and taking the simplex built by the vertex-touching hyperplanes together with e. In the example shown in Figure 2 the decomposition contains two positive simplices, ade and cbe, and two negative simplices, abe and cde, where notationally xyz indicates the unique simplex determined by the hyperplanes x, y and z.

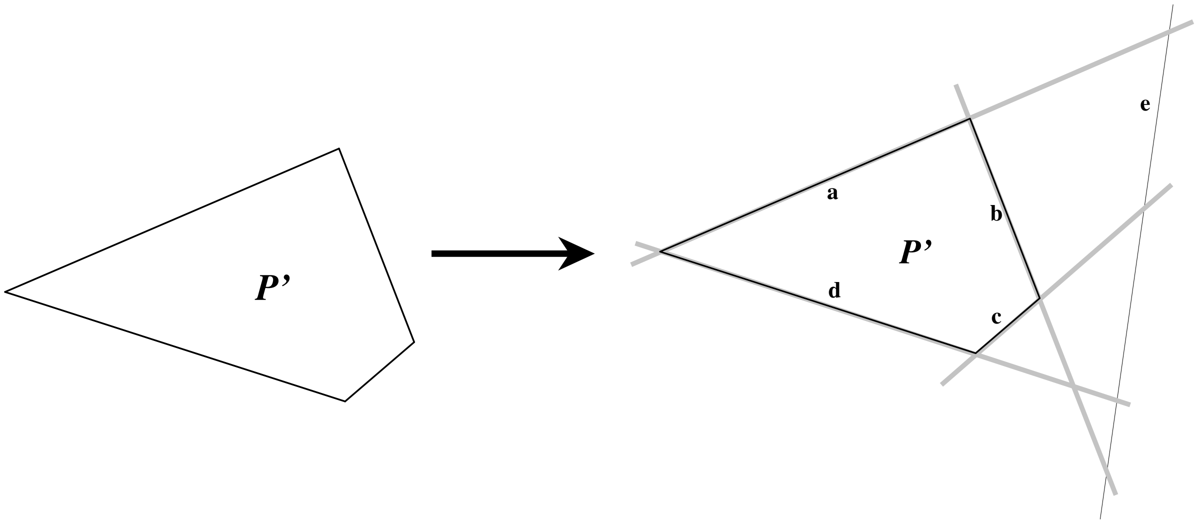

Figure 2: Lawrence's decomposition scheme

Note that Lasserre's method requires only an $\mathcal{H}$–representation, while Lawrence' method requires both $\mathcal{V}$– and $\mathcal{H}$–representations.

2.3 Duality

Assume that the origin lies in the interior of a polytope P; then the polar polytope P' is defined by $P' := \{x \mid y^T x \leq 1 \ \forall y \in P\}$. Note that the vertices are mapped onto the facets and vice versa as is illustrated in Figure 3. Filliman showed that there is a dual relation between a triangulation of a polytope and a signed decomposition of its polar P' (see [5]). While the left polytope is triangulated into the four simplices $\Delta(abe)$, $\Delta(bce)$, $\Delta(cde)$ and $\Delta(ade)$, the polar P' is decomposed into signed simplices determined by the same triples. The sign of each simplex xyz in the decomposition of P' is determined by the *separation parity* of the corresponding simplex $\Delta(xyz)$ in the triangulation of P, which is the parity of the number of facet-defining hyperplanes of $\Delta(xyz)$ which separates the simplex from the origin. The simplex $\Delta(bce)$ has even parity $(+1)$ since it has two facet-defining lines be and ce separating it from the origin. The simplex cde has odd parity (-1) since it has only one such line de. When the parity is odd, the corresponding simplex in the signed decomposition of the polar is negative. Thus we have two positive simplices cbe, ade and two negative simplices cde, abe.

More generally Filliman showed:

> Let P be a d-polytope containing the origin in its interior and let $\mathcal{T}$ be a triangulation of P such that the origin is not contained in the

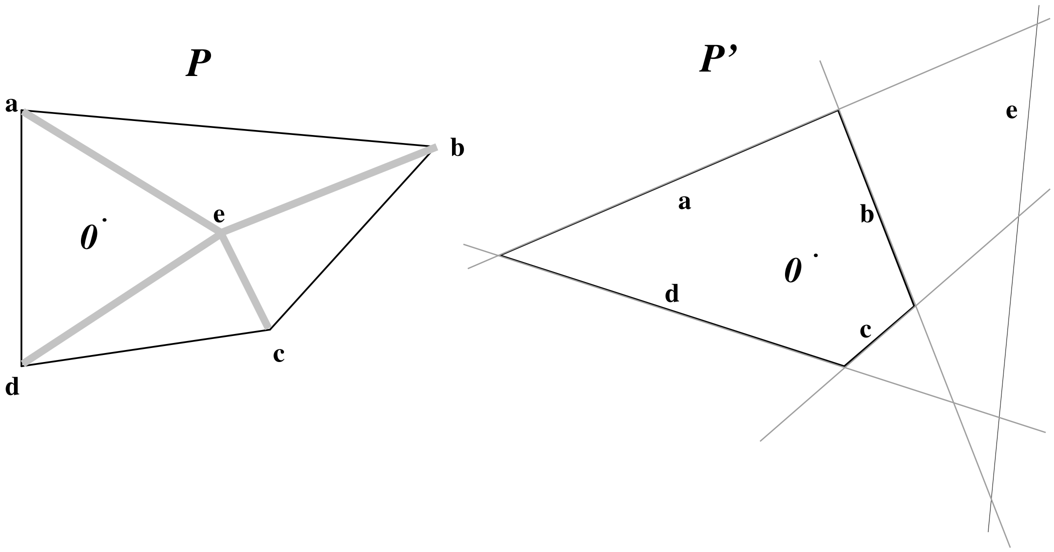

Figure 3: Filliman's duality

> union of hyperplanes spanned by vertices of the triangulation. Then the triangulation induces a signed decomposition of the polar P' of P in such a way that each simplex $\Delta(a_0, a_1, \ldots, a_d)$ is in $\mathcal{T}$ if and only if the unique bounded simplex determined by the corresponding hyperplanes in the polar appears as a simplex in the decomposition. The sign of each simplex in the signed decomposition is determined by its separation parity in the triangulation.

Observe that the assumptions on the location of the origin is important in defining the separation parity properly.

2.4 Should one triangulate or signed decompose?

Whichever type of algorithm one selects, the number of simplices in a triangulation or a signed decomposition serves as a guideline for the computational complexity since any algorithm generating it explicitely will depend at least linearly on this number. From the considerations above we know that for each polytope P there is a polytope P' such that the number of simplices are the same for both a triangulation algorithm and its corresponding signed decomposition method. As a first consequence, this indicates that neither of the two methods can outperform the other method on a sufficiently broad class of polytopes. However, for a given polytope the number of simplices generated in a triangulation or signed decomposition can drastically differ, motivating a second consequence: Choosing the right method can determine the tractability of volume computation for a concrete polytope. To illustrate this claim the unit d-hypercube C^d of volume 1 and its polar, the cross polytope, can serve as an example. C^d has 2^d vertices and $2d$ facets, and consequently Lawrence's decomposition produces 2^d simplices. To estimate a lower bound for the number of simplices in a triangulation we can use Hadamard's upper bound for the largest volume simplex in the unit hypercube, $d^{d/2}/d!$. Although the resulting lower bound for the number of simplices, $O((d/2)!)$, might be far from the unknown tight bound, the growth in the number of simplices is much faster for any triangulation method compared to Lawrence's signed decomposition.

Indeed, in the case of pure methods we experimentally found a behaviour following exactly these hypotheses: triangulation methods fail for hypercubes and convince for cross polytopes, while signed decomposition methods behave in an opposite way. This was a strong motivation to develop a hybrid method.

As a prerequisite for the complexity discussion below we assume a computer device with arbitrary precision. Namely we assume that all arithmetic operations require an amount of time which is independent of the number of digits needed in the representation of the numbers. In such a computational model the time complexity is proportional to the number of elementary arithmetic operations. In view of real world computers with fixed accuracy architectures the following remarks are in order.

(i) For triangulation methods and Lasserre's method the number of digits required for a simplex-volume can be bounded by a polynomial in the problem size and d. The time complexity of real-world computers is therefore bounded by the time complexity in our computational model times a polynomial.

(ii) In case of Lawrence's signed decomposition method this is not true for our implementation. Hence it is open how to bound the time complexity of real-world computers based on the time complexity in our computational model.

3 Pure Methods

In this section representatives of triangulation methods and signed decomposition methods described in the literature are presented. We call them 'pure' meaning that they perform exactly either a triangulation or a signed decomposition. To allow a theoretical comparison we give some complexity analyses. Although the equal number of simplices for both a triangulation of P and a signed decomposition of its polar P' only holds for specific pairs of algorithms (e.g. boundary triangulation on the one hand and Lawrence's formula on the other hand) and only under conditions on the location of the origin, we can expect that the qualitative behaviour of a signed decomposition applied to cross polytopes equals the behaviour of a triangulation of the corresponding polar hypercubes, and equally well is the behavior of triangulation methods for cross polytopes close to signed-decomposition methods for hypercubes. Therefore the complexity analysis concentrates on hypercubes treated by Cohen & Hickey's method as a representative for triangulation methods and Lasserre's method standing for signed decomposition methods. In the subsequent section this is continued for the revised methods and the new hybrid approach. Nevertheless, due to duality, this should well enlighten the behaviour of all algorithms tested on a broad class of polytopes ranging from being simple to simplicial.

3.1 Triangulation methods

Delaunay triangulation

The geometric idea behind a Delaunay triangulation of a d-polytope is to 'lift' it on a paraboloid in dimension $d+1$. This is done on the level of vertices $v \to \bar{v}$, $(x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_d, \sum_{i=1}^{d} x_i^2)$. If the resulting vertices are in general position, the convex hull of $\bar{V}$ yields a triangulation, whose 'lower half' is a Delaunay triangulation. In our experiments the underlying convex hull algorithm uses the 'beneath-beyond' method (see [6]). This method requires only the $\mathcal{V}$-representation.

Boundary triangulation

Even if P itself is not simplicial, lexicographic perturbation (either symbolically or numerically) of the vertices leads to a simplicial polytope. Computing the convex hull of the perturbed points and interpreting the result in terms of the original vertices leads to a triangulation of the boundary which, by linking with a fixed interior point, yields a triangulation of P. As convex hull algorithm we chose reverse search, see [1]. Only the $\mathcal{V}$-representation is required as input.

Triangulation by Cohen & Hickey

This recursive scheme triangulates a d-polytope P by choosing any vertex $v \in P$ as apex and connecting it with the $d-1$-dimensional simplices resulting from a triangulation of all facets of P not containing v, see [3]. To be precise, denote by e^k, $0 \leq k \leq d$, k-dimensional faces of P, and let η be a map which associates to each face one of its vertices. Then the pyramids with apex $\eta(e^d)$ and bases among the facets e^{d-1} with $\eta(e^d) \notin e^{d-1}$ form a dissection of the polytope. Applying this scheme recursively to all e^{d-1} results in a set of decreasing chains of faces, $e^d \supset e^{d-1} \supset \cdots \supset e^1 \supset e^0$ such that $\eta(e^i) \neq \eta(e^j)$ for $1 \leq i, j \leq d$ and $i \neq j$. Then the set of corresponding simplices $\Delta(\eta(e^0), \ldots, \eta(e^d))$ is a triangulation of P. To implement this elegant recursive method extensive use of the double description as $\mathcal{V}-$ and $\mathcal{H}-$polytope is made by representing all faces as sets of vertices. Based on this representation we pass from a face e^k to e^{k-1} by intersecting the set of vertices of e^k with the facets of P not containing the vertex $\eta(e^k)$. To prevent in degenerate cases the multiple generation of the same face, a list is maintained containing all faces of e^k generated so far; only faces which are not in this list are accepted. In the sequel we call this basic algorithmic scheme 'C&H'.

Note that in the case of C&H compared to a boundary triangulation all simplices in the facets containing the apex v are eliminated and thereby the number of simplices is usually reduced.

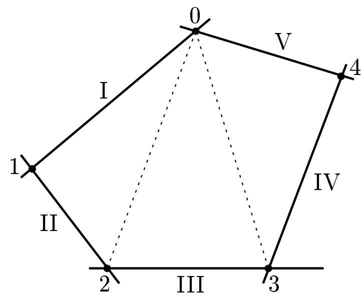

Figure 4: Cohen & Hickey's triangulation scheme

As an example consider Figure 4 where η assigns to each face its vertex with the lowest number, so $\eta(P) = 0$. Now all facets which do not contain the vertex 0 are examined, that is, II, III and IV. The scheme is now applied to facet II with $\eta(\text{II}) = 1$. II is intersected with all facets not containing the vertex 1, these are III, IV and V. The intersections with IV and V are empty, so these recursion branches are unsuccessful. The intersection with III yields the vertex 2, and the fixed vertices 0, 1 and 2 form a first simplex. The other simplices obtained from III and IV are also marked in the figure.

Proposition 1 *The time complexity of C&H applied to a d-hypercube is $O(d^3 d!)$.*

Proof: There are $2^{d-k}\binom{d}{k}$ faces of dimension k, each of which contains 2^k vertices. The recursion tree resembles the face lattice where some faces do not appear whereas others are present multiple times. The (unique) root is the original d-polytope P, and in each recursion level k *some* of the incident $(k-1)$-dimensional faces e^{k-1} are included. Denote by s_k the number of faces of dimension k appearing in the recursion tree; from the construction we know $s_d = 1$. In recursion k we intersect a face e^k with all facets not containing $\eta(e^k)$, and since there are k such facets we have $s_{k-1} = ks_k$, implying $s_k = \frac{d!}{k!}$. Hence the number of simplices, s_0, equals $d!$. Because for each simplex a determinant has to be computed, the overall complexity bound derived so far is $O(d^3 d!)$, assuming conventional matrix computation techniques. In the rest of the proof we verify that the additional computational burden of the algorithm per simplex is in $O(d^3)$. Suppose that the intersection of e^k with a facet e^{d-1} is done by a binary look-up of the vertices of e^k in e^{d-1}, resulting in $O(|e^k| \log |e^{d-1}|) = O((d-1)2^k) \subseteq O(d2^k)$ comparisons, where the notation $|e^k|$ designates the number of vertices in the face e^k. To check further whether the newly determined face e^{k-1} appears in the list of at most $k-1$ faces already derived from e^k requires another $O((k-1)|e^{k-1}|) \subseteq O(d2^k)$ comparisons. The facet-intersection and the maintenance of the list have to be done for each facet, resulting in a total complexity of $O(\sum_{k=1}^{d} s_k k(d2^k + d2^k))$, where

$$\sum_{k=1}^{d} s_k k d 2^k = dd! \sum_{k=1}^{d} \frac{2^k}{k!} k < 2dd! \sum_{k=0}^{\infty} \frac{2^k}{k!} k = 2e^2 dd!.$$

Thus the additional computational burden per simplex is in $O(d)$ which can be neglected compared to the determinant computation. □

3.2 Signed decomposition methods

Lasserre's recursive algorithm

The volume is computed via a recursive scheme based on Euler's formula for homogeneous functions. Denote by $\mathrm{Vol}(d, A, b)$ the volume sought, by a_i the i^{th} row of A, and by $\mathrm{Vol}_i(d-1, A, b)$ the volume in the appropriate $\mathbb{R}^{d-1}$ subspace defined by the face $P \cap \{a_i^T x = b_i\}$. Then the volume of P can be expressed by the following recursive structure:

$$\mathrm{Vol}(d, A, b) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{\|a_i\|} \mathrm{Vol}_i(d-1, A, b). \tag{4}$$

Now (4) is not yet an implementable recursion scheme because the terms in the sum have to be evaluated in appropriate lower dimensional spaces. To compute $\mathrm{Vol}_i(d-1, A, b)$ Lasserre [8] suggested a projection scheme as follows. For the constraint to be fixed choose a pivot element $a_{ij} \neq 0$ and set $x_j = (b_i - \sum_{k \neq j} a_{ik} x_{ik})/a_{ij}$. Substituting x_j in this way defines a new system $\tilde{A}\tilde{x} \leq \tilde{b}$ with $m-1$ constraints and $\tilde{d} := d-1$ variables for which we have $\mathrm{Vol}_i(d-1, A, b) = (\|a_i\|/a_{ij}) \mathrm{Vol}(\tilde{d}, \tilde{A}, \tilde{b})$. Geometrically $\mathrm{Vol}(\tilde{d}, \tilde{A}, \tilde{b})$ is the volume of $P \cap \{a_i^T x = b_i\}$ projected onto the subspace $\{x_j = 0\}$. Finally, using (4) gives the recursive scheme we implemented:

$$\mathrm{Vol}(d, A, b) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{a_{ij}} \mathrm{Vol}(\tilde{d}, \tilde{A}, \tilde{b}). \tag{5}$$

At the bottom of the recursion tree, the volume of a line segment bounded by (at most) $m-d+1$ constraints has to be computed. It is of critical importance in this

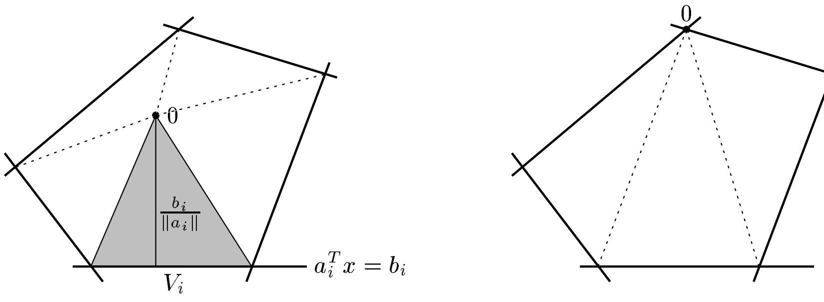

Figure 5: Triangulation induced by Lasserre's method if the origin is inside a facet (left) or if it coincides with a vertex (right). The volume $\frac{b_i}{d\|a_i\|}\,\mathrm{Vol}_i(d-1, A, b)$ of a single triangle is gray shaded in the left graph.

formula that no constraint appears more than once in any (recursively generated) face. Even though this might seem to be usually given it turns out that in many interesting examples this situation occurs in lower dimensional faces. In the process of eliminating identical constraints non-identical parallel constraints are treated simultaneously, where two cases are possible: either the intersection of two parallel cuts is non-empty or empty. While we can stop the recursion branch in the latter case, we check in the former case wheter the normal vectors point in the same direction and if so drop the 'outer' of the two parallel cuts. For hypercubes this implies that in dimension k there are at most $2k$ constraints in the reduced system.

Looking at (5) another simple improvement is possible by making as many components of b as possible equal to zero and thereby suppressing the recursion for the related branch. This could be done by shifting the most degenerate vertex of the face under consideration into the origin. In practice, however, we do not want to use information about vertices in order to preserve a pure $\mathcal{H}$-based algorithm. Thus we shift a face in dimension k only into a basis solution by solving an arbitrary, non-degenerate $k \times k$ subsystem of equations. In the case of hypercubes this decreases the number of constraints with non-zero b-component in the reduced systems of dimension k from $2k$ to k. The resulting algorithm, including parallel face detection and shifting, is called 'LAS'.

Proposition 2 *The time complexity of LAS applied to a d-hypercube is $O(d!)$.*

Proof: A similar analysis as in Lemma 1 can be made. To pass from e^k to e^{k-1} we choose among k faces with non-zero b-component, resulting in the recursive equation $s_{k-1} = ks_k$. With $s_d = 1$ it follows that $s_k = \frac{d!}{k!}$. Fixing one constraint together with shifting requires $O(k^3)$ operations for solving a system of linear equations, and the subsequent check for multiple restrictions (which detects the hyperplane parallel to the newly fixed one) needs $O(k^3)$ steps, yielding a complexity bounded by $O(\sum_{k=1}^{d} s_k k k^3) = O(d! \sum_{k=1}^{d} \frac{k^3}{(k-1)!})$. Now $\sum_{k=1}^{d} \frac{k^3}{(k-1)!}$ is bounded by $22.5 + 11 \sum_{k=0}^{d-4} \frac{1}{k!}$, and with $\sum_{k=0}^{d} \frac{1}{k!} < e$ we derive a total complexity bound of $O(d!)$. □

Note that no extra determinant computation is needed because this is implicitly done in the course of fixing constraints. LAS improves on C&H by a factor of d^3.

Lawrence's volume formula

Assume P is *simple* and choose a vector $c \in \mathbb{R}^d$ and a scalar q such that the function $x \mapsto c^T x + q$ is not constant along any edge. For each vertex $v \in V$ let A_v be the

$d \times d$-matrix composed by the rows of A which are binding at v. Then A_v is invertible and $\gamma^v := (A_v^T)^{-1} c$ is well defined up to permutations of the entries. The assumption imposed on c assures that none of the entries of γ^v is zero. Lawrence [9] shows that then

$$\mathrm{Vol}(P) = \sum_{v \in V} \frac{(c^T v + q)^d}{d! \, |\det A_v| \, \prod_{i=1}^{d} \gamma_i^v}. \tag{6}$$

Lawrence's formula easily generalises to the non-simple case using standard lexicographic perturbation techniques. Consequently the summation in (6) must be done over all lexicographically feasible cobases of all $v \in V$.

An open question is how to choose c and q efficiently; even if $c^T x + q$ is not constant on any edge, a 'nearly constant' choice results in very small entries γ_i^v in the denominator, causing potentially severe numerical problems. Indeed, such numerical instabilities were regularly found in our experiments.

The time complexity for the simple case is $O(d^3 n)$, where n denotes the number of vertices, for cubes this implies a complexity of $O(d^3 2^d)$. As mentioned at the end of Section 2 this bound can not be directly carried over to real world computers due to the potentially unbounded number of digits involved.

4 Revised Methods and a Hybrid Method

The pure methods described in the previous section can be improved by a few simple ideas. In this section we present such improvements for C&H and LAS, calling the resulting algorithms r-C&H and r-LAS respectively, where 'r' abbreviates 'revised'. After discussing r-C&H and r-LAS we present a hybrid approach called HOT for 'hybrid orthonormalisation technique'. To underline the qualitative differences of the methods a few complexity estimates are given.

4.1 Revised Cohen & Hickey's method (r-C&H)

The major advantage of a triangulation method using both the $\mathcal{H}$- and $\mathcal{V}$-representation is the possibility for a computationally cheap, but powerful test on empty or simplicial faces. Consider in the recursive process a face e^k; then we know that the volume of e^k in the appropriate k-dimensional affine embedding is zero if the number of vertices in e^k is smaller than $k+1$. And if the number of vertices is exactly $k+1$ the recursive process can stop with a (potentially empty) simplex. Even though this does not change the behaviour on simple polytopes like hypercubes, i.e. Lemma 1 still applies, simplicial polytopes have a reduced complexity of $O(md^3)$, where m denotes the number of facets. The practical gain for neither simple nor simplicial polytopes is well demonstrated by the metric polytope Fm-6 described in Section 5. Here the running time without empty/simplicial face detection is 4600 seconds, dropping to 25 seconds if this detection is activated.

Another improvement uses degeneracy information: If the vertices are sorted decreasingly by the number of incident hyperplanes, the number of e^{k-1}'s not containing v_k drops significantly. If the vertices are ordered reversely Fm–6 needs 8400 CPU-seconds, whereas in the well-sorted case only 25 seconds are spent.

4.2 Revised Lasserre's method (r-LAS)

Observing that when the same constraints are fixed in a different order the same face may appear several times in (4), it is natural for Lasserre's method to store and subsequently reuse the (projected) face volumes, e.g. in a balanced tree. Due to the projection, not only the indices of the fixed constraints I characterizing the face e^k have to be stored, but also the fixed variables $\bar{J}$ representing the projection space $\Pi_J = \mathbb{R}^d \cap \{x \mid x_j = 0 \; \forall j \in \bar{J}\}$, where $J = \{1, \ldots, d\} \setminus \bar{J}$ denotes the set of free variables. Once the volume of a face e^k projected onto Π_J is known, its projection onto $\Pi_{J'}$, where $J \neq J'$ but $|J| = |J'|$, can be found directly by computing the determinant of the transformation $\Pi_J \to \Pi_{J'}$. Denote by A_{IJ} the matrix consisting of the rows and columns of A indexed by I and J respectively. In a first step, we 'deproject' $\Pi_J \to \mathbb{R}^d$, $x_J \mapsto x = \left[\begin{smallmatrix} x_{\bar{J}} \\ x_J \end{smallmatrix}\right]$, with $x_{\bar{J}} = A_{I\bar{J}}^{-1}(b_I - A_{IJ}x_J)$. In a second step x is projected on $\Pi_{J'}$, $x \mapsto x_{J'}$, by dropping the components related to $\bar{J}'$. Therefore the rows J' of the linear part of the map $x_J \mapsto x$ yield a nonsingular $(d-k) \times (d-k)$ matrix whose determinant s fixes the projective effect, i.e. $\mathrm{Vol}(\Pi_{J'}(e^k)) = |s| \cdot \mathrm{Vol}(\Pi_J(e^k)) =$.

Proposition 3 *The time complexity of r-LAS applied to a d-hypercube is $O(d^4 3^d)$.*

Proof: (Cf. the proof of Lemma 2) Again, let s_k denote the number of faces considered in dimension k. Starting with any of the k-dimensional faces, we have to fix k constraints and subsequently check for identical and parallel constraints. Each of these steps can be done in time $O(k^3)$. We obtain ks_k faces of dimension $k-1$. Each of these faces has to be looked up in the balanced tree containing the projected face volumes. Taking into account that at most all 3^d faces of the cube are stored in the tree, and that one key comparison can be done with $O(m-k) \subseteq O(d)$ index comparisons, the effort for one look-up is in $O(d^2)$. Notice now that some of the ks_k faces are found in the tree, causing each an overhead of $O(d^3)$ for the deprojection. So the total complexity is in $O(ks_k(k^2 + d^2 + d^3)) \subseteq O(d^4 s_k)$. We now claim that s_k, the number of k-dimensional faces stored in the tree, is bounded above by the total number $2^{d-k}\binom{d}{k}$ of k-dimensional faces. This is true since in a hypercube, which is simple, each non-empty k-dimensional face is the intersection of a unique set of $d-k$ facets. Moreover, empty faces are not stored, for this would involve an intersection of parallel hyperplanes. Hence the overall arithmetic complexity is $O(d^4 \sum_{k=0}^{d} 2^{d-k}\binom{d}{k}) = O(d^4 3^d)$. □

It is interesting that the real code exhibits empirically a behavior following 3^d when applied on hypercubes for $d = 8, \ldots, 14$.

4.3 A hybrid method (HOT)

In the course of our experiments we identified two outstanding factors explaining the different behaviour of triangulations versus signed decomposition methods: empty face detection and storing/reusing intermediate results.

Triangulation methods using vertex and incidence information permit a strong test on simplicial or empty faces by simply counting the number of vertices. Signed decomposition methods like Lasserre's method are usually facet-based where a test on empty faces, e.g. by means of linear programming, is significantly more costly. The situation is reversed when storing/reusing intermediate results is considered. Here triangulations like Cohen & Hickey's algorithm suffer from the fact that they are based on simplex volumes and not on face volumes. A signed decomposition like

Lasserre's scheme on the other hand works directly with the (projected) volume of faces, offering a simple way of storing/reusing face volumes in the recursion process.

Among the possibilities to construct a hybrid method, one can include an empty face detection in Lasserre's method or a storing/reusing scheme in Cohen & Hickey's method. Concerning the first possibility we included an empty face detection in Lasserre's method, but the LP-based direct approach turned out to be not competitive in most cases.[1] On the other hand, storing the whole subtriangulation of a face in Cohen & Hickey's algorithm would require by far too much memory.

So we adopted a different approach, which nevertheless is based on Cohen & Hickey's enumeration of faces. Notice that the volume of P can be derived directly from the volume of its facets by the formula

$$\mathrm{Vol}_d(e^d) \quad = \quad \frac{1}{d} \sum_{e^{d-1}: v_d \notin e^{d-1}} \mathrm{Vol}_{d-1}(e^{d-1})\, \mathrm{dist}(v_d, \mathrm{aff}(e^{d-1})), \tag{7}$$

where Vol_d denotes the d-dimensional volume, aff the affine hull of a set, and $v_d = \eta(e^d)$. This corresponds to a subdivision of P into pyramids with apex v_d and bases e^{d-1} instead of simplices. The distance between v_d and $\mathrm{aff}(e^{d-1})$ is easily computed if an orthonormal basis of the affine subspace embedding e^{d-1} is known, which can be computed using Housholder transformations. The face volumes $\mathrm{Vol}_k(e^k)$ are stored in a balanced tree with the vertices as key; they are reused whenever the same face reappears during the recursion. The resulting algorithm is called *hybrid orthonormalisation technique*, or abbreviated HOT. Since storing the orthonormal bases would require an enormous amount of memory, they are recomputed from scratch when a $\mathrm{Vol}_k(e^k)$ is reused.

Proposition 4 *The time complexity of HOT applied to a d-hypercube is $O(d^2 4^d)$.*

Proof: (Cf. the proof of Lemma 3) In dimension k, a number of s_k faces are intersected by k facets each which are then looked up in the tree, requiring $O(k\, s_k\, 2^k d)$ comparisons, cf. the related complexity analysis for r-C&H in Lemma 1. By s_{k-1} we denote the number of $k-1$-dimensional faces where the recursion proceeds because the corresponding volume is not yet known. After these recursions are finished, $O(d^2 s_{k-1})$ steps are needed for adding one vector to the orthonormal basis. For the remaining $ks_k - s_{k-1}$ faces, which are retrieved, orthonormal bases are computed from scratch, taking $O(d^3(ks_k - s_{k-1}))$ steps. Adding together, the total complexity is bounded by $O(\sum_{k=1}^d (ks_k 2^k d + d^2 s_{k-1} + d^3(ks_k - s_{k-1}))) \subset O(d^2 \sum_{k=0}^d s_k 2^k + d^4 \sum_{k=0}^d s_k)$. To estimate s_k note that a face is uniquely determined by the set of its vertices, so the same face can be stored only once in the tree. As before, it folllows that $s_k \leq 2^{d-k}\binom{d}{k}$. We obtain therefore an overall arithmetic complexity of $O(d^2 2^d \sum_{k=0}^d \binom{d}{k} + d^4 \sum_{k=0}^d 2^{d-k}\binom{d}{k}) = O(d^2 4^d + d^4 3^d)$. □

5 Experimental Results

The implemented algorithms have been tested on a broad range of examples, varying from polytopes created with random vertices or hyperplanes to specially structured polytopes as they arise in combinatorics. We denote by d the dimension, by m the number of hyperplanes and by n the number of vertices. The groups of examples are ordered from large to small ratio n/m:

[1] Our tests are based on the efficient callable library of cplex using restarting techniques.

- cube–d: Hypercube in dimension d.
- rh–d–m: d-dimensional problems with m randomly generated constraints. The constraints are constructed by generating integral vectors on a sphere with radius 1000. The resulting polytopes are (most probably) simple.
- $\mathrm{CC}_d(k)$: The product of two cyclic polytopes over k points in dimension d.
- Fm–d: One facette of the metric polytope in dimension d.
- ccp–v: Complete cut polytope on v vertices.
- rv–d–n: d-dimensional convex hulls of n randomly generated vertices on a sphere with radius 1000. The resulting polytopes are (most probably) simplicial.
- cross–d: cross polytope of dimension d; it is the polar of the d-dimensional cube.

A discussion of these examples can be found in [1]. We emphasise that our main concern is the comparison of the general behaviour of the methods and not of specific implementations; furthermore, most of the codes are experimental in nature leaving room for improvements. The observed time should therefore be interpreted cautiously.

Unless otherwise stated the computed volume is identical for all methods for at least six digits and is therefore given only once. The general abbreviations for the codes used in the sequel are given below in the first column; in the second column the shorter ones used in Table 2 are stated.

BND	Bnd	boundary triangulation using 'lrs'
DEL	Del	Delaunay triangulation using 'qhull'
C&H		original Cohen & Hickey triangulation
r-C&H	rCH	revised Cohen & Hickey triangulation
HOT	HOT	hybrid orthonormalisation technique
LAW-nd	Lnd	Lawrence's formula in the non-degenerate case
LAW-d	Ld	Lawrence's formula in the general case using 'lrs' for computing all lexicographically feasible cobases
LAS		original Lasserre
r-LAS	rL	revised Lasserre

Sources for the codes are given in the appendix. The only code using rational arithmetic is 'lrs'. All computations have been done on an HP 7000/735–99. Among the many parameters influencing the behaviour of the codes the one determining the level of storing face volumes is of major importance in case of HOT and r-LAS. The choice was done to take most advantage of the memory available which is limited to approximately 500 MB. Obviously, for larger problems there is a decisive trade-off between memory and CPU usage if face volumes are stored/reused. In case of r-LAS another influencing parameter is the minimal absolute value MIN_PIVOT accepted for pivoting; the strategy is to go through a row and pick the first element which exceeds MIN_PIVOT with absolute value. If none is found the absolute largest element is chosen. The larger MIN_PIVOT, the slower r-LAS because the probability for projecting the same face onto the same subspace decreases. In our experiments we set MIN_PIVOT = 0.01. A too small MIN_PIVOT, however, contains the risk of numerical instabilities.

In the following two sections we first compare the pure methods with our revised versions; here we consider only cubes and cross polytopes. Then a comparison of the revised methods with all other codes on the full set of test problems is presented. Here we drop the non-revised C&H and LAS because they are not competitive.

5.1 C&H versus r-C&H

5.2 LAS versus r-LAS

In Table 1 the effect of shifting and storing/reusing face volumes is demonstrated for cubes and cross polytopes. While in the first column, marked by $(-, -)$, neither is possible, in a first step shifting without storing/reusing $(+, -)$ is activated representing LAS, then storing/reusing alone is allowed $(-, +)$, and finally—the r-LAS case—both is possible $(+, +)$. For cubes the effect of those two factors reduces convincingly the CPU-time required, where the effect of storing/reusing is slightly more important than shifting. In case of cross polytopes, however, the facet-based approach of r-LAS has severe difficulties exploiting the structure. First of all storing/reusing face volumes has no effect because no face ever appears twice, thus (?, $-$) equals (?, $+$) where ? is a wildcard. Secondly shifting is only done once at the very beginning because in every recursion level $k > 0$ there are already at least $d-k$ components of the right hand side equal zero. We observe therefore in CPU-time $(-, ?) \approx (+, ?)$.

		r-LAS [CPU-seconds]			
Problem	d/m/n	$(-, -)$	$(+, -)$	$(-, +)$	$(+, +)$
cube–6	6/12/64	0.2	0.0	0.1	0.0
cube–7	7/14/124	3.4	0.1	0.2	0.0
cube–8	8/16/256	53	1.1	1.1	0.0
cube–9	9/18/512	946	9.7	4.9	0.1
cube–10	10/20/1024	18993	97.5	16.3	0.2
cross–6	6/64/12	1.1	0.6		
cross–7	7/124/14	17.3	8.6		
cross–8	8/256/16	258.4	136		
cross–9	9/512/18	4300[a]	2177		
cross–10	10/1024/20	70000[a]	37117		

Table 1: r-LAS with $(+)$ or without $(-)$ shifting and storing/reusing face volumes respectively. The case $(+, -)$ represents LAS whereas $(+, +)$ is r-LAS. [a]is estimated by twice the time of case $(+, -)$, i.e. with shifting but without storing.

5.3 Comparing the main codes

In Table 2 the timing in CPU-seconds for all codes and examples is given; because the transformation $\mathcal{V} \to \mathcal{H}$ or $\mathcal{H} \to \mathcal{V}$ can be as demanding as the volume computation itself, and furthermore we used both representations for certain methods, we give in the last two columns the transformation time when using 'cdd'.

Table 2 reveals huge differences in the CPU time among the different codes for the same polytope. Some problems are even intractable with certain methods whereas they are quite efficiently solved with others, demonstrating the relevance of a good choice. Moreover, we faced regularly practical problems ranging from insufficient memory to numerical instabilities. In the following we briefly comment each code (column) separately, and include also information going beyond Table 2.

The boundary triangulation BND, done by 'lrs' using rational (exact) arithmetic, requires in general more CPU-time but can nevertheless be competitive for some near-simplical polytopes with small ratio n/m. In all cases tested this method produced the largest number of simplices compared to the other triangulation methods.

						Triangulation				Signed decmp.				
						$\mathcal{V}$–r.		$\mathcal{V}$– and $\mathcal{H}$–r.			$\mathcal{H}$–r.			
Exp.	$\frac{n}{m}$	d	m	n	vol.	Bnd	Del	rCH	HOT	Lnd	Ld	rL	$\mathcal{H}\to\mathcal{V}$	$\mathcal{V}\to\mathcal{H}$
cube-9	28.4	9	18	512	512	73e3	440	82	4.0	0.3	2.6	0.2	0.5	1.3
cube-10	51.2	10	20	1024	1024	$***^{c}$	$***^{a}$	940	19	0.6	5.4	0.4	0.9	3.6
cube-14	585	14	28	16384	16384	$***^{c}$	$***^{a}$	$***^{c}$	3300	13	140	9.5	120	1200
rh-8-20	56	8	20	1115	37576	$***^{c}$	$***^{b}$	93	7.7	1.5	51	12	2.5	76e3^{b}
rh-8-25	104	8	25	2596	785.9	$***^{c}$	$***^{b}$	440	27	3.4	140	110	9.0	19e3^{b}
rh-10-20	109	10	20	2180	13883	$***^{c}$	$***^{b}$	2900	62	5.4	160	27	6.4	83e4
rh-10-25	309	10	25	7724	5729.5	$***^{c}$	$***^{a}$	37e3	390	15	650	600	70	$***^{c}$
$CC_8(9)$	1.5	8	54	81	13340	420	44	11	3.3	$***^{c}$	120	130	0.5	0.9
$CC_8(10)$	1.43	8	70	100	156816	1300	100	28	7.9	$***^{c}$	320	840	1.1	1.8
$CC_8(11)$	1.38	8	88	121	1.39e6	3200	180	65	16	$***^{c}$	820	5140	2.9	3.9
Fm-6	3	15	59	177	286113	21e4	$***^{a}$	25	12	$***^{c}$	2e4	4500	1.6	$***^{c}$
ccp-5	0.3	10	56	16	2.26e-3	1.2	0.4	0.1	0.1	$***^{c}$	230	3564	0.2	0.3
ccp-6	0.09	15	368	32	1.3458	3800	160	43	23	$***^{c}$	$***^{c}$	$***^{c}$	28	2.9
rv-8-10	0.42	8	24	10	1.41e19	2.4	0.2	0.1	0.1	$***^{c}$	160	0.2	3.7^{b}	0.2
rv-8-11	0.2	8	54	11	3.05e18	3.8	0.3	0.1	0.1	$***^{c}$	4000	80	0.6	0.2
rv-8-30	0.007	8	4482	30	7.35e21	260	4.4	7.1	6.9	$***^{c}$	$***^{c}$	$***^{c}$	$***^{c}$	23
rv-10-12	0.34	10	35	12	2.14e22	6.4	0.3	0.1	0.1	$***^{c}$	2300	0.2	630^{b}	0.2
rv-10-14	0.08	10	177	14	2.93e23	13.5	0.3	0.2	0.1	$***^{c}$	$***^{c}$	$***^{c}$	15e5	0.3
cross-8	0.063	8	256	16	6.35e-3	1.1	0.4	0.2	0.2	$***^{c}$	4000	170	0.5	0.3
cross-9	0.035	9	512	18	1.41e-3	2.5	0.5	0.2	0.3	$***^{c}$	83e3	2700	1.2	0.4

Table 2: Time in CPU-seconds; notes: $***^{a}$ beyond memory limit, $***^{b}$ numerical errors, $***^{c}$ problem is intractable with this method.

The Delaunay triangulation DEL produced by 'qhull' exhibits numerical problems in many examples. For specially structured polytopes, where the vertices are not in general position, numerical perturbation is needed, which may cause a break–down as observed with hypercubes. Moreover, the memory space needed for the quick hull algorithm is not bounded above by a polynomial in the input and output size, and quite often a memory overflow occurs. The ratio 'largest/smallest simplex-volume', in the sequel called condition number, is comparable to C&H's method; also the number of generated simplices is comparable to r-C&H's

r-C&H turns out to be numerically very robust due to its combinatorial nature. For special problems it profits a lot from degeneracy information, e.g. for Fm-6 where one vertex lies in all but one facet. (This information is not used by r-LAS.) The bigger the ratio n/m, however, the more r-C&H is slowed down by the increasing number of simplices.

The new hybrid code, HOT, is uniformly faster than r-C&H. Compared to r-LAS the situation is slightly different. Leaving apart hypercubes, the hybrid method is faster in most cases, but suffers from the burden of computing orthonormal bases for retrieved faces. Considering the memory requirement, HOT is in most problems more modest than r-LAS. To explain the tremendous effect of reusing face-volumes in HOT, we observed for increasing $\frac{n}{m}$ an increased average reuse of already computed face volumes. Thus HOT handles just the examples efficiently where r-C&H fails while making still use of r-C&H's strong point in the other examples, namely its detection of simplicial faces.

Numerical instabilities related to the unresolved problem of choosing the objective function are a serious drawback of Lawrence's approach presented in the columns labelled LAW-nd (non-degenerate) and LAW-d (degenerate: 'lrs' is used for finding all cobases). However, for most of the examples the result is correct, even though in some examples positive and negative summands occured with absolute value up to 10^{20}, where extinction of digits can drastically destroy the accuracy of the final result. For simple polytopes LAW-nd is the fastest method exploiting efficiently the double description.

r-LAS is quite efficient for near-simple polytopes with a high ratio $\frac{n}{m}$. It profits a lot from memory for storing intermediate results. Due to the detection of parallel facets it works extremely quickly on hypercubes. If the number of hyperplanes increases, however, time and/or memory requirement can grow exceedingly large, see ccp-6, rv-8-30 or rv-10-14.

In cases where only one description is given, the effort presented in the columns $\mathcal{H}\rightarrow\mathcal{V}$ and $\mathcal{V}\rightarrow\mathcal{H}$ to produce the other description using 'cdd' can be higher than that for the volume computation itself. Therefore, it is worthwhile to take the given representation into account when deciding which volume code to use. We have to mention, though, that specifically in the time consuming examples an approach based on reverse search could much better handle such degenerate polytopes; in that sense the timings for transformation should be interpreted very carefully. As indicated by our experiments, volume computation is specifically hard in two cases: (near-)simple polytopes where only the $\mathcal{V}$-representation is given, and (near-)simplicial polytopes given by an $\mathcal{H}$-representation only. This (dual and primal respectively) degeneracy makes both the direct volume computation and the transformation using 'cdd' fail. Because HOT requires the double description this otherwise convincing code can not be exploited in such a situation neither. Let us report, however, an important new finding [2] which claims to change this situation. Basically it is proved that if one direction of transformation is 'easy' (polynomial in input and output), then the other direction of transformation is easy, too. The underlying idea is to use the 'easy' direction as an oracle to help solve the seemingly hard direction.

6 Conclusions

The experimental results indicate that vertex based triangulations are superior for small ratio n/m, namely for simplicial polytopes, whereas constraint based signed decomposition codes behave favourably for large n/m, e.g. for simple polytopes. Outstanding is — except for hypercubes — the behaviour of the hybrid code HOT; it convinces on both near-simple and near-simplicial polytopes, and exhibits even a modest need for memory on our set of examples. Thus, in the general case, given both representations or an efficient transformation code, HOT seems to be the method of choice.

Appendix A: Availability of the Codes

All algorithms described in Section 3 and 4 have been implemented in C by the authors, using three publicly available additional programs, namely 'cdd' (double describtion in C, transfers between $\mathcal{H}$- and $\mathcal{V}$-representations), 'qhull' (quick hull, computes convex hulls for a given set of vertices), and 'lrs' (lexicographic reverse search, constructs convex hulls and cobases using lexicographic perturbation). The resulting volume computation package called 'vinci' provides a common framework for the different methods and codes. While r-C&H, HOT and r-LAS are implemented directly in 'vinci', the remaining algorithms — Delaunay triangulation, boundary triangulation and Lawrence's method — are realised using 'qhull' for the first and 'lrs' for the other two methods; in the special case of a simple polytope the incidence information suffices to generate Lawrence's signed decomposition and hence a direct summation, omitting 'lrs', can be performed. The necessary communication is done via text files. In these cases 'vinci' checks the input and calls the appropriate external code. After the termination of 'qhull' or 'lrs', 'vinci' reads the resulting triangulation or signed decomposition and makes the appropriate summation of simplex volumes. Concerning Lawrence's method we use random values for c and q. A thorough discussion of 'cdd', 'qhull' and 'lrs' can be found in [1, 2]. 'vinci' and the additional programs are freely available at the following sites:

vinci	authors:	Benno Büeler (bueeler@ifor.math.ethz.ch) and Andreas Enge (enge@ifor.math.ethz.ch)
	www page:	http://www.mathpool.uni-augsburg.de/~enge
	ftp site;	ftp.ifor.math.ethz.ch (129.132.154.13)
	directory:	pub/volume
	file name:	vinci-*.tar.gz, where '*' stands for the actual version number
lrs	author:	David Avis (avis@cs.mcgill.ca)
	ftp site:	mutt.cs.mcgill.ca (132.206.3.13)
	directory:	pub/C
	file name:	lrs*.c.gz, where '*' stands for the actual version number
qhull	authors:	Brad Barber (bradb@geom.umn.edu) and Hannu Huhdanpaa (hannu@geom.umn.edu)
	www page:	http://www.geom.umn.edu/locate/qhull
	ftp site:	geom.umn.edu (128.101.25.35)
	directory:	priv/hannu
	file name:	qhull-*.*.tar.Z, where '*.*' stands for the actual version number
cdd+	author:	Komei Fukuda (fukuda@ifor.math.ethz.ch)
	www page:	http://www.ifor.math.ethz.ch/staff/fukuda/fukuda.html
	ftp site:	ftp.ifor.math.ethz.ch (129.132.154.13)
	directory:	pub/fukuda/cdd
	file name:	cdd+-***.tar.gz, where '***' stands for the actual version number

Acknowledgment: We are grateful to Jean-Bernard Lasserre for encouraging and fruitful discussions related to his volume computation method; we also want to thank Paul A. Vixie[2] for giving us his efficient avl-tree implementation.

References

[1] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms. *Computational Geometry: Theory and Applications*, to appear. Available via anonymous ftp from ftp://mutt.cs.mcgill.ca/pub/doc/hgch.ps.gz.

[2] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. Extended abstract, 13th ACM symposium on computational geometry, March 1997. Full paper available from ftp://www.ifor.math.ethz.ch/pub/fukuda/reports/primal-dual.ps.gz.

[2] Internet Software Consortium, Star Route Box 159A, Woodside, CA 94062 USA, vixie@vix.com, http://www.isc.org/isc/

[3] Jaques Cohen and Timothy Hickey. Two algorithms for determining volumes of convex polyhedra. *Journal of the ACM*, 26(3):401–414, July 1979.

[4] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[5] P. Filliman. The volume of duals and sections of polytopes. *Mathematika*, 39:67–80, 1992.

[6] P. Gritzmann and V. Klee. On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In T. Bisztriczky, P. McMullen, R. Schneider, and A.I. Weiss, editors, *Polytopes: Abstract, convex and computational (Scarborough, ON, 1993)*, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., 440, pages 373–466. Kluwer Acad. Publ., Dordrecht, 1994.

[7] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithms for convex bodies. Manuscript, January 1996.

[8] J.B. Lasserre. An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$. *J. of Optimization Theory and Applications*, 39(3):363–377, 1983.

[9] Jim Lawrence. Polytope volume computation. *Mathematics of Computation*, 57(195):259–271, 1991.

[10] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.