On the Volume of the Birkhoff Polytope

by

Alexandria Vassallo

B.Sc., McMaster University, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Mathematics
Faculty of Science

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Alexandria Vassallo**
Degree:	**Master of Science**
Thesis title:	**On the Volume of the Birkhoff Polytope**
Committee:	**Chair:** Nilima Nigam Professor, Mathematics
	Marni Mishna Supervisor Professor, Mathematics
	Amarpreet Rattan Committee Member Associate Professor, Mathematics
	Nathan Ilten Examiner Associate Professor, Mathematics

Abstract

The Birkhoff polytope was introduced in 1946 to facilitate the study of doubly stochastic matrices. The volume of the n^{th} Birkhoff polytope is an essential characteristic but methods to compute this are computationally complex and the volume is only known up to $n = 10$. In this thesis, we apply a novel automated method to calculate the relative volume using analytic combinatorics in several variables. An implementation using Maple and Sage computes this volume up to $n = 6$ and we compare to existing methods including interpolation, Euler's generating function, complex analytic techniques, and Barvinok's Algorithm (`LattE`). The key advantage of this method is its robustness and adaptability to variants of the initial problem.

Keywords: Birkhoff polytope, doubly stochastic matrices, semi-magic squares, asympotics, polytopes

Acknowledgements

I would like to thank Dr. Marni Mishna for her guidance, support, and encouragement throughout this process. You have aided in my love of math in so many ways. I would also like to thank Stefan Trandafir for his invaluable contributions, it has been great working alongside you these last three years.

I would like to thank my committee, Dr. Amarpreet Rattan and Dr. Nathan Ilten, for their time to read and examine my thesis.

I would like to thank my parents, Anita and Tony Vassallo, for their unconditional love, support, and encouragement even though they would much rather have me back home in Ontario! I would like to thank my best friend Meagan Curtis and her partner Alex Becchetti, even though we are on opposite ends of the country they never let me feel like I was on this journey alone.

I would like to thank the amazing friends I have made during my time at SFU including Hannah Sutton, Jas Dhahan, Trevor Hearty, Dana Mraz, Sam Simon, Danielle Rogers, Jesse Campion Loth, Alexandra Wesolek, Tabriz Popatia, and Khalil Shivji.

Last but not least I would like to thank my right hand cat, Charlie. His steps across my keyboard provided critical grammatical corrections.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

The study of polytopes has a rich history dating back to 2000 B.C. where it was believed that the Egyptians knew how to calculate the volume of a truncated square pyramid [37, 32]. A *polytope*, $\mathcal{P}$, is a geometric object with flat sides existing in any number of dimensions, with a *convex polytope* being the convex hull of finitely many points.

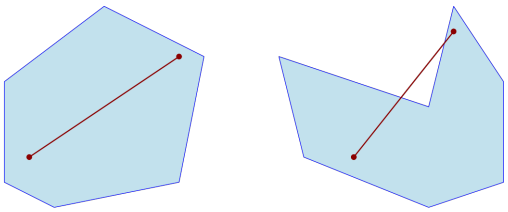


Figure 1.1: The polytope on the left is convex while the one on the right is not.

Early mathematicians and philosophers such as Democritus, Eudoxus, Plato, Euclid, and Archimedes [37], paved the way for the thriving study of polytopes that we see today in many areas of mathematics including combinatorics, algebraic geometry, linear programming, and graph theory. While there are many characteristics of the polytope being studied, we will take a specific interest in the volume. In the 1960's, French mathematician Eugéne Ehrhart established the Ehrhart polynomial, $L_{\mathcal{P}}(t)$, a counting function for the number of lattice points in a dilated polytope [12, 29]. He provided two invaluable results: the leading term of $L_{\mathcal{P}}(t)$ is, up to a multiplicative constant, the volume of $\mathcal{P}$, and the degree of $L_{\mathcal{P}}(t)$ is the dimension of $\mathcal{P}$ [29, 28]. The first of these results forms a basis for this thesis as we aim to compute the relative volume of the Birkhoff polytope. The Birkhoff polytope, named after Garrett Birkhoff, is denoted by $\mathcal{B}_n$ and formed from the set of $n \times n$ doubly stochastic

matrices,

$$\mathcal{B}_n = \left\{ \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{pmatrix} \in \mathbb{R}^{n^2} : x_{jk} \geq 0, \begin{matrix} \sum_j x_{jk} = 1 \ \forall \ 1 \leq k \leq n \\ \sum_k x_{jk} = 1 \ \forall \ 1 \leq j \leq n \end{matrix} \right\}.$$

The study of the Birkhoff polytope has since spread to many fields in the sciences and has applications extending to enumerative combinatorics, optimization, statistics, quantum mechanics, and representation theory. One interpretation of note that we will consider is that the number of lattice points in a dilated Birkhoff polytope is equal to the number of semi-magic squares.

There exists a lot of work dedicated to computing the relative volume of $\mathcal{B}_n$. Due to groundbreaking work done by Beck and Pixton [14] the relative volume of $\mathcal{B}_n$ is known for $n \leq 10$ with an asymptotic formula presented by Canfield and McKay [18]. Existing methods to calculate the relative volume of $\mathcal{B}_n$ use a range of different mathematical concepts. Many methods focus solely on building the Ehrhart polynomial, either by means of interpolation [15, 26, 41, 47, 20], generating functions [15, 24, 9, 23, 31, 7], or complex analysis [14, 11]. An open source software named `LattE` [4] employs Barvinok's Algorithm which utilizes multivariate generating functions and is designed to output the Ehrhart polynomial and the volume for any given polytope. Though this method is simple and easy to use, it proves inefficient for the Birkhoff Polytope, due to a very long computation time. The complex analytic techniques employed by Beck and Pixton are presently the most successful.

In this thesis, we introduce a novel method to compute the relative volume of $\mathcal{B}_n$ as well as a faster implementation of `LattE`. In their 2004 paper, Baryshnikov and Pemantle [10] question the feasibility of applying the method of analytic combinatorics in several variables when compared to complex analytic techniques, though they had not done any conclusive tests. With this question in mind, our method utilizes existing work involving complex analytic techniques and analytic combinatorics in several variables to asymptotically compute the leading term of the Ehrhart polynomial. We use complex analytic techniques and `LattE` to provide a more efficient use of the open source software. The remainder of this chapter will expand on introduced concepts and give necessary notation and definitions. Chapter 2 will provide a survey of four existing methods: interpolation, generating functions, complex analytic techniques, and Barvinok's Algorithm. Chapter 3 will showcase two new methods to calculate the relative volume of $\mathcal{B}_n$, with success for $n \leq 6$. We conclude with results from a series of calculations done in Maple and Sage, along with a comparison, shown in Table 4.1, to results obtained from various existing methods.

1.2 Convex Polytopes

The work of Beck and Robins [15] and Stanley [46] are used as guides for the definitions, notation, theorems and proofs given in this section. We use the notation $\mathbf{z}$ to represent the real d-dimensional vector $(z_1, ..., z_d)$ and define the multivariate notation $\mathbf{z}^{\mathbf{a}} := z_1^{a_1} \cdot z_2^{a_2} \cdots z_d^{a_d}$ for $\mathbf{z}, \mathbf{a} \in \mathbb{R}^d$. Let $\langle \mathbf{z}, \mathbf{a} \rangle$ denote the dot product for $\mathbf{z}, \mathbf{a} \in \mathbb{R}^d$. There are two well known equivalent representations of a convex polytope both of which we use throughout this thesis depending on the problem.

Definition (vertex representation)**.** A convex polytope $\mathcal{P} \subset \mathbb{R}^d$ is formed by taking the convex hull of the finite point set $\{v_1, v_2, ..., v_n\} \in \mathbb{R}^d$, that is

$$\mathcal{P} = \text{conv}(v_1, v_2, ..., v_n) := \left\{ \lambda_1 v_1 + \lambda_2 v_2 + ... + \lambda_n v_n : \text{ for all } \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1 \right\}. \quad (1.1)$$

One can think of taking the convex hull of a finite set of points as tightening shrink wrap around the outside of the points. This ensures that the shape formed has flat sides and the finite set of points are convex.

Definition (half-space representation)**.** A convex polytope $\mathcal{P} \subset \mathbb{R}^d$ is formed by the bounded intersection of finitely many half-spaces. That is

$$\mathcal{P} := \{\mathbf{x} \in \mathbb{R}^d : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}, \quad (1.2)$$

for $\mathbf{A} \in \mathbb{R}^{m \times d}$ and $\mathbf{b} \in \mathbb{R}^m$.

The matrix $\mathbf{A}$ and vector $\mathbf{b}$ come from a set of linear inequalities. Each half-space that bounds $\mathcal{P} \subset \mathbb{R}^d$ can be written as

$$a_1 x_1 + a_2 x_2 + \cdots + a_d x_d \leq b_1,$$

thus the polytope is formed by the finite set of solutions to the system of linear equations,

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1d}x_d & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2d}x_d & \leq & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{md}x_d & \leq & b_m. \end{array} \quad (1.3)$$

A polytope can be translated into a non-negative orthant without changing the number of lattice points contained in $\mathcal{P} \subset \mathbb{R}^d$, therefore we can assume that all solutions to Equation (1.2) are non-negative. Each inequality in Equation (1.3) can be converted into an equality

using what is known as a *slack variable*. For example, the inequality

$$a_1x_1 + a_2x_2 + \cdots + a_dx_d \leq b_1$$

can be rewritten as

$$a_1x_1 + a_2x_2 + \cdots + a_dx_d + x_{d+1} = b_1$$

for $x_{d+1} \in \mathbb{R}_{\geq 0}$. We now assume for the remainder of the thesis without loss of generality that

$$\mathcal{P} := \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{1.4}$$

for $\mathbf{A} \in \mathbb{R}^{m\times d}$ and $\mathbf{b} \in \mathbb{R}^m$.
The *dimension* of a polytope is the dimension of the affine space spanned by $\mathcal{P} \subset \mathbb{R}^d$, that is

$$\operatorname{span} \mathcal{P} := \{x + \lambda(y - x) : x, y \in \mathcal{P}, \lambda \in \mathbb{R}\}.$$

If $\mathcal{P}$ has dimension d then we say that $\mathcal{P}$ is a d-polytope. We say that the d-polytope $\mathcal{P}$ is *full-dimensional* if $\mathcal{P} \subset \mathbb{R}^d$. A hyperplane, $H = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : \langle \mathbf{a}, \mathbf{x}\rangle = b\}$, intersecting $\mathcal{P} \subset \mathbb{R}^d$ is called a *supporting hyperplane* if the polytope lies entirely on one side of H. A *face* of $\mathcal{P}$ is a subset of $\mathcal{P}$ of the form $\mathcal{P} \cap H$ for H a supporting hyperplane. From this definition we see that a face of a polytope is itself a polytope. A *facet* of a d-polytope is a face of dimension $d - 1$, an *edge* has dimension 1, and a *vertex* 0. A vertex $v \in \mathcal{P}$ is also called an extreme point of a polytope and given a supporting hyperplane H is a subset of $\mathcal{P}$ of the form $\mathcal{P} \cap H = \{v\}$. A convex d-polytope with exactly $d + 1$ vertices is called a *d-simplex*.
A polytope $\mathcal{P} \subset \mathbb{R}^d$ is called *integral* if all of its vertices have integer coordinates, that is

$$\mathcal{P} = \operatorname{conv}\{v_1, v_2, ..., v_n\} \text{ for } v_i \in \mathbb{Z}^d.$$

Furthermore, $\mathcal{P}$ is called *rational* if all of its vertices have rational coordinates, that is

$$\mathcal{P} = \operatorname{conv}\{v_1, v_2, ..., v_n\} \text{ for } v_i \in \mathbb{Q}^d.$$

Example 1.2.1. The line segment in blue in Figure 1.2 is a 1-polytope, however it lies in $\mathbb{R}^2$. A line in $\mathbb{R}^2$ is not full dimensional. This line segment is an example of a 1-simplex with two vertices, (0,0), and (4,2).

A convex polytope can be *dilated*, meaning that we can either shrink or stretch the polytope uniformly by a factor $t \in \mathbb{Z}$. Written in terms of the vertex representation
$t\mathcal{P} = \operatorname{conv}(tv_1, tv_2, ..., tv_n)$, or in half-space representation $t\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : \mathbf{A}\mathbf{x} = t\mathbf{b}\}$. Given a dilated polytope $t\mathcal{P}$, we say we are looking at the t^{th} dilate of the polytope.

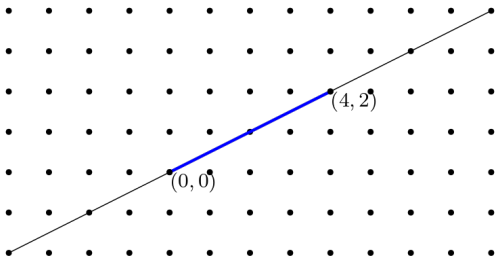


Figure 1.2: The line segment from $(0, 0)$ to $(4, 2)$ shown in blue and its affine space.

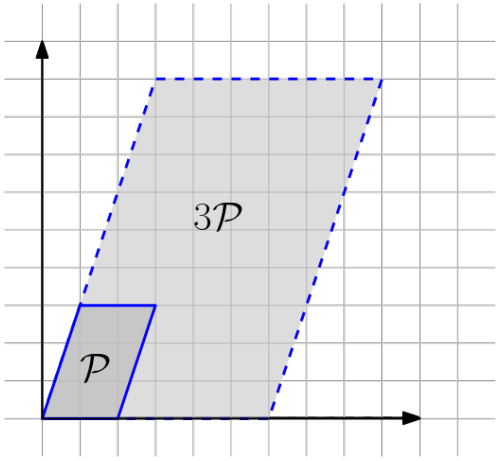


Figure 1.3: A polytope and its 3^{rd} dilate.

We define

$$L_{\mathcal{P}}(t) := \#(t\mathcal{P} \cap \mathbb{Z}^d) = \#\{\mathbf{x} \in \mathbb{Z}^d_{\geq 0} : \mathbf{A}\mathbf{x} = t\mathbf{b}\}$$

to be the *lattice-point enumerator* for the t^{th} dilate of $\mathcal{P} \subset \mathbb{R}^d$. Its purpose, as the name says, is to count the number of lattice points contained in $t\mathcal{P}$. The infinite sequence $\{L_{\mathcal{P}}(t)\}_{t\geq 1}$ can be encoded in a formal power series as coefficients, this is known as a *generating function*. The generating function associated to the infinite sequence $\{L_{\mathcal{P}}(t)\}_{t\geq 1}$ is commonly referred to as the *Ehrhart series* of $\mathcal{P}$,

$$\text{Ehr}_{\mathcal{P}}(z) := 1 + \sum_{t\geq 1} L_{\mathcal{P}}(t) z^t.$$

We now state an important theorem regarding $L_{\mathcal{P}}(t)$ presented by Eugéne Ehrhart in [29].

Theorem 1.2.1 (Ehrhart's Theorem)**.** *If $\mathcal{P}$ is an integral convex d-polytope, then $L_{\mathcal{P}}(t)$ is a polynomial in t of degree d.*

The polynomial $L_{\mathcal{P}}(t)$ is known as the Ehrhart Polynomial. The next two subsections will not only provide the background needed to prove Ehrhart's Theorem but also present concepts required for Barvinok's algorithm discussed in Section 2.4. We will see that the Ehrhart

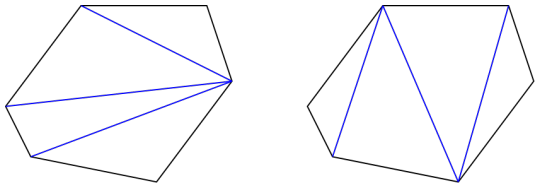


Figure 1.4: Two possible triangulations of a polytope.

polynomial given by Ehrhart's Theorem is critical in computing the volume of a polytope therefore the proof is presented in Subsection 1.2.3.

1.2.1 Triangulating a polytope

Recall that a d-simplex is a convex d-polytope with exactly $d+1$ vertices. A method called *triangulation* dissects a non-simplex convex d-polytope $\mathcal{P}$ into a finite set $T = \{\Delta_1, ..., \Delta_m\}$ of d-simplices such that:

1. $\mathcal{P} = \bigcup_{\Delta \in T} \Delta$,
2. for every $\Delta_k, \Delta_j \in T$, $\Delta_j \cap \Delta_k$ is a common face of both Δ_k and Δ_j or empty,
3. if $\Delta_k \in T$ then every face of Δ_k is also in T.

If there exists a triangulation T of a polytope $\mathcal{P}$ such that the vertices of every $\Delta \in T$ are vertices of $\mathcal{P}$ then it is said that $\mathcal{P}$ can be triangulated using no new vertices.

Theorem 1.2.2 (Existence of triangulations [15])**.** *Every convex polytope can be triangulated using no new vertices.*

Theorem 1.2.2 tells us that we are able to dissect any non-simplicial integral convex d-polytope into a finite set of integral d-simplices. Ehrhart's Theorem ensures that each one of these integral convex d-simplices will have an Ehrhart polynomial. Taking the union of the finite set of d-simplices and summing their Ehrhart polynomials will give the Ehrhart polynomial for the entire integral convex d-polytope. Therefore, the proof of Ehrhart's theorem will only need to be shown for d-simplicies. It is important to address the over counting when computing $L_{\mathcal{P}}(t)$ as, by definition, many d-simplices will share a common face, one must take caution and use the principle of inclusion and exclusion to ensure no lattice point is over counted.

1.2.2 Coning over a polytope

We now discuss how to cone over a convex polytope. The *integer point transform* of $\mathcal{P} \subset \mathbb{R}^d$ is defined as the multivariate generating function

$$\sigma_{\mathcal{P}}(\mathbf{z}) = \sigma_{\mathcal{P}}(z_1, z_2, ..., z_d) = \sum_{a \in \mathcal{P} \cap \mathbb{Z}^d} \mathbf{z}^{\mathbf{a}}.$$

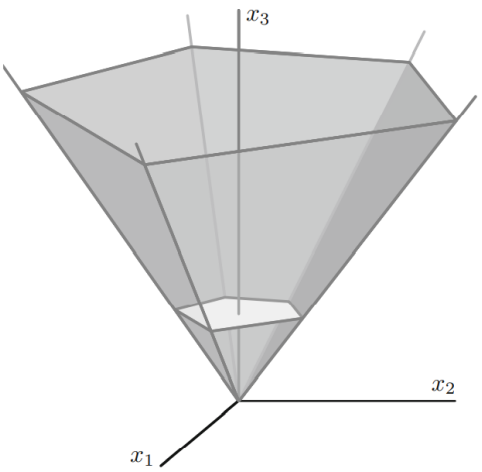


Figure 1.5: Coning over the polytope shown in the slice of the cone at the hyperplane $x_3 = 1$. Figure taken directly from [15].

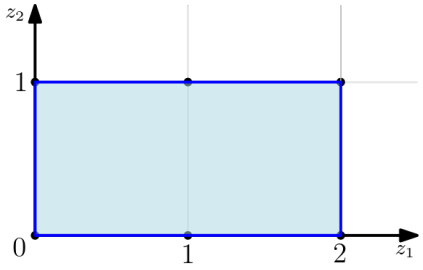


Figure 1.6: Polytope with vertices $(0,0), (0,1), (2,1)$ and $(2,0)$

Evaluating $\sigma_{\mathcal{P}}(\mathbf{z})$ at the all 1's vector gives the number of lattice points in $\mathcal{P}$, hence

$$\sigma_{\mathcal{P}}(1,1,...,1) = \#(\mathcal{P} \cap \mathbb{Z}^d) = L_{\mathcal{P}}(1).$$

Example 1.2.2. Consider the polytope $\mathcal{P} \subset \mathbb{R}^2$ with vertices $(0,0), (0,1), (2,1), (2,0)$ shown in Figure 1.6. The integer point transform is given by

$$\begin{aligned}\sigma_{\mathcal{P}}(\mathbf{z}) &= (z_1, z_2)^{(0,0)} + (z_1, z_2)^{(0,1)} + (z_1, z_2)^{(1,1)} + (z_1, z_2)^{(2,1)} + (z_1, z_2)^{(2,0)} + (z_1, z_2)^{(1,0)} \\ &= 1 + z_2^2 + z_1 z_2 + z_1^2 z_2 + z_1^2 + z_1.\end{aligned}$$

Therefore the number of integer points in $\mathcal{P}$ is $L_{\mathcal{P}}(1) = \sigma_{\mathcal{P}}(\mathbf{1}) = 1+1+1+1+1+1 = 6$.

Before discussing what it means to cone over a polytope, we first state two important definitions.

Definition. For $\mathbf{v}, \mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n \in \mathbb{R}^d$ a *cone* $C \subset \mathbb{R}^d$ is defined as the set of points

$$C = \{\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2 + ... + \lambda_n \mathbf{w}_n : \lambda_1, \lambda_2, ..., \lambda_n \geq 0\},$$

and a *pointed cone* is defined as

$$\mathbf{v} + C.$$

We call $\mathbf{v}$ the apex and $\mathbf{w}_i$ for $i = 1, ..., n$ the generators. If $C \subset \mathbb{R}^d$ is generated by at most d linearly independent generators then C is a *simplicial* cone.

Definition. The *fundamental parallelepiped* of a cone C is given by

$$\pi = \{\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2 + ... + \lambda_n \mathbf{w}_n : 0 \leq \lambda_1, ..., \lambda_n < 1\}$$

where $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n \in \mathbb{R}^d$ are the generators of C.

We now have the necessary tools to describe how to cone over a polytope. Given a convex polytope $\mathcal{P} = \text{conv}(v_1, v_2, ..., v_n) \subset \mathbb{R}^d$, we aim to lift $\mathcal{P}$ to dimension $d + 1$. We set the origin as the apex and append the value 1 to each vertex of $\mathcal{P}$ to build the new generators:

$$\mathbf{w}_1 = (v_1, 1), \mathbf{w}_2 = (v_2, 1), ..., \mathbf{w}_n = (v_n, 1).$$

We define the *cone over* $\mathcal{P}$ as

$$\text{cone}(\mathcal{P}) = \{\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2 + ... + \lambda_n \mathbf{w}_n : \lambda_1, \lambda_2, ..., \lambda_n \geq 0\} \subset \mathbb{R}^{d+1}.$$

The original polytope can be obtained by slicing $\text{cone}(\mathcal{P})$ at the hyperplane $x_{d+1} = 1$, and any dilate of $\mathcal{P}$ by slicing at $x_{d+1} = t$:

$$t\mathcal{P} = \text{cone}(\mathcal{P}) \cap \{x_{d+1} = t\}.$$

We define the integer point transform of $\text{cone}(\mathcal{P})$ as

$$\begin{aligned} \sigma_{\text{cone}(\mathcal{P})}(z_1, ..., z_d, z_{d+1}) &= 1 + \sigma_{\mathcal{P}}(z_1, ..., z_d) z_{d+1} + \sigma_{2\mathcal{P}}(z_1, ..., z_d) z_{d+1}^2 + ... \\ &= 1 + \sum_{t \geq 1} \sigma_{t\mathcal{P}}(x_1, ..., z_d) z_{d+1}^t. \end{aligned} \tag{1.5}$$

We evaluate $\sigma_{\text{cone}(\mathcal{P})}(z_1, ..., z_{d+1})$ at $(1, ..., 1, z_{d+1})$ to get

$$\begin{aligned}\sigma_{\text{cone}(\mathcal{P})}(1, ..., 1, z_{d+1}) &= 1 + \sum_{t\geq 1} \sigma_{t\mathcal{P}}(1, ..., 1) z_{d+1}^t \\ &= 1 + \sum_{t\geq 1} \#(t\mathcal{P} \cap \mathbb{Z}^d) z_{d+1}^t \\ &= 1 + \sum_{t\geq 1} L_{\mathcal{P}}(t) z_{d+1}^t.\end{aligned} \tag{1.6}$$

Lemma 1.2.1 ([15, Lemma 3.10])**.** *For a polytope $\mathcal{P} \subset \mathbb{R}^d$*

$$\sigma_{\text{cone}(\mathcal{P})}(1, ..., 1, z) = 1 + \sum_{t\geq 1} L_{\mathcal{P}}(t) z^t = \text{Ehr}_{\mathcal{P}}(z).$$

Proof. Follows from Equation (1.5) and (1.6). □

We now have the necessary insight for the following theorem.

Theorem 1.2.3 ([15, Theorem 3.5])**.** *Given a d-simplex $\mathcal{P} = \text{conv}(v_1, v_2, ..., v_{d+1}) \subset \mathbb{R}^d$, and the generators $\boldsymbol{w}_i = (v_i, 1)$ for $i = 1, ..., d+1$, the integer point transform for cone($\mathcal{P}$) is given by*

$$\sigma_{\text{cone}(\mathcal{P})}(\boldsymbol{z}) = \frac{\sigma_\pi(\boldsymbol{z})}{(1 - \boldsymbol{z}^{\boldsymbol{w}_1})(1 - \boldsymbol{z}^{\boldsymbol{w}_2}) \dots (1 - \boldsymbol{z}^{\boldsymbol{w}_{d+1}})},$$

where

$$\pi = \{\lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2 + ... + \lambda_{d+1} \boldsymbol{w}_{d+1} : 0 \leq \lambda_1, ..., \lambda_{d+1} < 1\}.$$

Proof. The integer point transform of cone($\mathcal{P}$) can be written as

$$\sigma_{\text{cone}(\mathcal{P})}(\mathbf{z}) = \sum_{\mathbf{a} \in \text{cone}(\mathcal{P}) \cap \mathbb{Z}^{d+1}} \mathbf{z}^{\mathbf{a}}.$$

As $\mathbf{a}$ is an integer point in cone($\mathcal{P}$), $\mathbf{a} = \lambda_1 \mathbf{w}_1 + ... + \lambda_{d+1} \mathbf{w}_{d+1}$ for $\lambda_1, ..., \lambda_{d+1} \geq 0$. Knowing that the vectors $\mathbf{w}_i$ for $i = 1, ..., d+1$ form a basis of $\mathbb{R}^{d+1}$ this representation is unique. Given $\lambda_i = \lfloor \lambda_i \rfloor + \{\lambda_i\}$ let

$$\mathbf{a} = (\{\lambda_1\}\mathbf{w}_1 + ... + \{\lambda_{d+1}\}\mathbf{w}_{d+1}) + \lfloor \lambda_1 \rfloor \mathbf{w}_1 + ... + \lfloor \lambda_{d+1} \rfloor \mathbf{w}_{d+1}.$$

As $0 \leq \{\lambda_i\} < 1$ then $\mathbf{p} = \{\lambda_1\}\mathbf{w}_1 + ... + \{\lambda_{d+1}\}\mathbf{w}_{d+1} \in \pi$. Again the vectors $\mathbf{w}_i$ for $i = 1, ..., d+1$ form a basis of $\mathbb{R}^{d+1}$ so this representation is unique. Therefore we have

$\mathbf{a} = \mathbf{p} + k_1\mathbf{w}_1 + ... + k_{d+1}\mathbf{w}_{d+1}$ for $\mathbf{p} \in \pi$ and $k_1, ..., k_{d+1} \in \mathbb{Z}_{\geq 0}$. Hence,

$$\begin{aligned}\sigma_{\text{cone}(\mathcal{P})}(\mathbf{z}) &= \sum_{\mathbf{a}\in\text{cone}(\mathcal{P})\cap\mathbb{Z}^{d+1}} \mathbf{z}^{\mathbf{p}+k_1\mathbf{w}_1+...+k_{d+1}\mathbf{w}_{d+1}} \\ &= \left(\sum_{\mathbf{p}\in\pi\cap\mathbb{Z}^{d+1}} \mathbf{z}^{\mathbf{p}}\right)\left(\sum_{k_1\geq 0} \mathbf{z}^{k_1\mathbf{w}_1}\right)\cdots\left(\sum_{k_{d+1}\geq 0} \mathbf{z}^{k_{d+1}\mathbf{w}_{d+1}}\right) \\ &= \frac{\sigma_\pi(\mathbf{z})}{(1-\mathbf{z}^{\mathbf{w}_1})(1-\mathbf{z}^{\mathbf{w}_2})\dots(1-\mathbf{z}^{\mathbf{w}_{d+1}})}.\end{aligned}$$

□

1.2.3 Proof of Ehrhart's Theorem

The following lemma will provide the last piece of the puzzle to prove Theorem 1.2.1.

Lemma 1.2.2 ([15, Lemma 3.9])**.** *Let f, g be polynomials in $\mathbb{R}[t]$ satisfying*

$$\sum_{t\geq 0} f(t)z^t = \frac{g(t)}{(1-z)^{d+1}}.$$

Then f is of degree d if and only if g is of degree at most d and $g(1) \neq 0$.

The proof can be done by algebraic manipulation.

Proof of Theorem 1.2.1. By Lemma 1.2.2 and the method of triangulations it suffices to prove for an integral d-simplex Δ that

$$\sum_{t\geq 0} L_\Delta(t)z^t = \frac{g(t)}{(1-z)^{d+1}},$$

for $g(t)$ a polynomial of degree at most d and $g(1) \neq 0$.
By definition Δ has $d+1$ vertices and cone(Δ) is simplicial with the origin as the apex and the generators

$$\mathbf{w}_1 = (v_1, 1), \mathbf{w}_2 = (v_2, 1), \cdots, \mathbf{w}_{d+1} = (v_{d+1}, 1).$$

By Theorem 1.2.3 we have the integer point transform

$$\sigma_{\text{cone}(\Delta)}(z_1, z_2, ..., z_{d+1}) = \frac{\sigma_\pi(z_1, z_2, ..., z_{d+1})}{(1-\mathbf{z}^{\mathbf{w}_1})(1-\mathbf{z}^{\mathbf{w}_2})\dots(1-\mathbf{z}^{\mathbf{w}_{d+1}})},$$

with $\pi = \{\lambda_1\mathbf{w}_1 + \lambda_2\mathbf{w}_2 + ... + \lambda_{d+1}\mathbf{w}_{d+1} : 0 \leq \lambda_1, ..., \lambda_{d+1} < 1\}$. The z_{d+1} coordinate of $\mathbf{w}_i$ for $i = 1, ..., d+1$ is 1, hence the z_{d+1} coordinate of π is $\lambda_1 + \lambda_2 + ... + \lambda_{d+1}$. Each $\lambda_i < 1$ therefore $\lambda_1 + \lambda_2 + ... + \lambda_{d+1} < d+1$ with an integer sum of at most d. Hence, the degree of z_{d+1} in $\sigma_\pi(z_1, z_2, ..., z_{d+1})$ is at most d and $\sigma_\pi(1, ..., 1, z_{d+1})$ is a polynomial in z_{d+1} of degree at most d. The apex of π is a lattice point (the origin) so $\sigma_\pi(1, ..., 1, 1) = \#(\pi \cap \mathbb{Z}^{d+1}) \neq 0$.

By Lemma 1.2.1 we have

$$\sum_{t\geq 0} L_\Delta(t)z^t = \sigma_{\text{cone}(\Delta)}(1,...,1,z) = \frac{\sigma_\pi(1,...,1,z)}{(1-z)^{d+1}},$$

and therefore, by Lemma 1.2.2, $L_\Delta(t)$ is a polynomial in t of degree d. □

1.3 The Birkhoff Polytope $\mathcal{B}_n$

The Birkhoff polytope is one of the most intensively studied polytopes and goes by many names including the assignment polytope, polytope of doubly stochastic matrices, and the perfect matching polytope of the complete bipartite graph $K_{n,n}$. We now formally introduce the Birkhoff polytope. A *stochastic vector* is composed of non-negative, real numbers such that the sum of all of its elements equates to 1. A *doubly stochastic matrix*, $\mathbf{A} = (a_{jk})$, is an $n \times n$ matrix with both column and row vectors being stochastic,

$$\sum_{j=1}^{n} a_{jk} = \sum_{k=1}^{n} a_{jk} = 1 \text{ and } a_{jk} \geq 0 \text{ for all } 1 \leq j,k \leq n.$$

The convex hull of all $n \times n$ doubly stochastic matrices forms the n^{th} *Birkhoff Polytope*,

$$\begin{aligned}\mathcal{B}_n &:= \left\{ \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{pmatrix} \in \mathbb{R}^{n^2} : x_{jk} \geq 0, \begin{matrix} \sum_{j=1}^n x_{jk} = 1 \text{ for all } 1 \leq k \leq n \\ \sum_{k=1}^n x_{jk} = 1 \text{ for all } 1 \leq j \leq n \end{matrix} \right\} \\ &= \{\mathbf{x} \in \mathbb{R}^{n^2}_{\geq 0} : \mathbf{Ax} = \mathbf{b}\} \text{ for } \mathbf{x} = (x_{11}, x_{12}, ..., x_{nn}),\end{aligned}$$

where, for $\mathbf{1}$ being the all 1's row vector in $\mathbb{R}^n$ and I_n the $n \times n$ identity matrix,

$$\mathbf{A} = \begin{pmatrix} \mathbf{1} & & & \\ & \mathbf{1} & & \\ & & \ddots & \\ & & & \mathbf{1} \\ I_n & I_n & \cdots & I_n \end{pmatrix} \in \mathbb{Z}^{2n \times n^2}, \text{ and } \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{Z}^{2n}. \tag{1.7}$$

The remaining entries are all 0.

Example 1.3.1. The 2nd Birkhoff polytope is formed by the convex hull of all 2×2 doubly stochastic matrices,

$$\mathcal{B}_2 = \left\{ \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \in \mathbb{R}^4 : x_1, x_2, x_3, x_4 \geq 0, \begin{matrix} x_1 + x_2 = x_3 + x_4 = 1 \\ x_1 + x_3 = x_2 + x_4 = 1 \end{matrix} \right\}. \tag{1.8}$$

The polytope $\mathcal{B}_2$ can be stated in half-space representation as

$$\mathcal{B}_2 := \left\{ \mathbf{x} \in \mathbb{R}^4_{\geq 0} : \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

Problems surrounding the Birkhoff polytope include, but are not limited to: construction and enumeration of doubly stochastic matrices, and the volume of such a polytope. Birkhoff [16] and von Neumann [50] presented the following theorem.

Theorem 1.3.1 (Birkhoff-von Neumann Thoerem)**.** *The vertices of $\mathcal{B}_n$ are the $n!$ permutation matrices.*

The proof is out of the scope of this thesis but can be found in [46, 21, 45, 33, 34].

Example 1.3.2. The two vertices of B_2 are the extremal points of Equation (1.8) given by $(1, 0, 0, 1)$ and $(0, 1, 1, 0)$. These two vertices correspond to the 2! possible 2×2 permutation matrices.

We now state an important theorem regarding the dimension of $\mathcal{B}_n$.

Theorem 1.3.2. *$\mathcal{B}_n$ is an $(n-1)^2$-polytope in $\mathbb{R}^{n^2}$.*

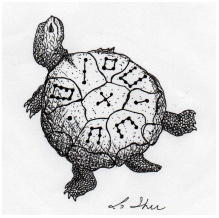
We denote the lattice point enumerator for the t^{th} dilate of $\mathcal{B}_n$ as

$$H_n(t) = \#(t\mathcal{B}_n \cap \mathbb{Z}^{n^2}) = \#\{\mathbf{x} \in \mathbb{Z}^{n^2}_{\geq 0} : \mathbf{A}\mathbf{x} = t\mathbf{b}\} = L_{\mathcal{B}_n(t)}.$$

By Ehrhart's Theorem 1.2.1 and Theorem 1.3.2 we get that $H_n(t)$ is a polynomial in t of degree $(n-1)^2$.

1.4 Semi-Magic Squares

We now present a nice interpretation for the lattice points in $\mathcal{B}_n$. A *magic square* is an $n \times n$ array of positive, distinct integers such that the sum across each row, column, and diagonal equals the magic sum or magic constant, denoted by t. They display the beauty of mathematics accompanied with a very rich history. It is believed that sometime around 2200 BCE there was a village in China that experienced huge floods destroying the crops and land. The village arranged sacrifices for the River God in hopes that it would calm their anger. Every time the river would flood a turtle would emerge and walk around the sacrifice. A pattern was noticed on the back of this turtle; dots had been arranged in a three by three grid such that the sum across any row, column or diagonal equaled fifteen [3]. This is believed to be the first magic square and is commonly known as the the Lo Shu Square.



(a) The Lo Shu turtle [40]

4	9	2
3	5	7
8	1	6

(b) The Lo Shu Square

Figure 1.7: The oldest known magic square

A *semi-magic square* is an $n \times n$ array of positive integers such that the sum across each row and column equals t. We are able to interpret the number of integral points in the t^{th} dilate of the n^{th} Birkhoff polytope as the number of $n \times n$ semi magic squares with magic sum t.

Example 1.4.1. Recall the set of 2×2 doubly stochastic matrices given in Equation (1.8). The two extreme points in the undilated $\mathcal{B}_2$ are $(1, 0, 0, 1)$ and $(0, 1, 1, 0)$. We can see the correspondence to semi-magic squares, these two integral points correspond to the only 2×2 semi-magic square with magic sum 1:

$$\begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \quad \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}.$$

Example 1.4.2. We will show in Chapter 2 that the Ehrhart Polynomial for $\mathcal{B}_2$ is $H_2(t) = t + 1$. Dilating $\mathcal{B}_2$ by a factor of 2 gives $H_2(2) = 3$. These 3 lattice points can be visualized by the following 2×2 semi-magic squares with magic sum 2:

$$\begin{array}{|c|c|}\hline 2 & 0 \\ \hline 0 & 2 \\ \hline \end{array}, \quad \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}, \quad \begin{array}{|c|c|}\hline 0 & 2 \\ \hline 2 & 0 \\ \hline \end{array}.$$

Therefore the Ehrhart Polynomial for $\mathcal{B}_n$ can be used to count the number of $n \times n$ semi-magic squares with magic sum t, a useful and fun application. Lastly, we notice that any doubly stochastic matrix with rational entries can be transformed into a semi-magic square by multiplying by a non-unique positive integer.

Example 1.4.3. Consider the 3×3 doubly stochastic matrix

$$\begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.5 & 0.3 \end{pmatrix}.$$

We can transform this particular matrix into a semi-magic square by multiplying by a dilate of 10, producing the following semi-magic square with magic sum of 10,

2	3	5
6	2	2
2	5	3

.

Using this interpretation, we identify semi-magic squares to the lattice points of the Birkhoff polytope.

1.5 Volume of $\mathcal{B}_n$

There are two notions of volume for a polytope that we consider. The *discrete volume* of a polytope $\mathcal{P} \subset \mathbb{R}^d$ is the number of points in $\mathcal{P} \cap \mathbb{Z}^d$. Thus, $L_{\mathcal{P}}(t)$ computes the discrete volume for the t^{th} dilation of $\mathcal{P}$. In particular, the number of $n \times n$ semi-magic squares with magic sum t is given by the discrete volume of $\mathcal{B}_n$. The *continuous volume* of a polytope $\mathcal{P} \subset \mathbb{R}^d$ is given by

$$\text{vol}(\mathcal{P}) := \int_{\mathcal{P}} 1 \; dm,$$

where dm is the *integral Lebesgue measure* on the affine hull of $\mathcal{P}$. The continuous volume of $\mathcal{P}$ can be thought of intuitively as the volume that we attach to everyday objects. In the case that $\mathcal{P}$ is full dimensional we obtain the following equivalent expression for continuous volume

$$\text{vol}(\mathcal{P}) = \lim_{t\to\infty} \frac{1}{t^d} \#(t\mathcal{P} \cap \mathbb{Z}^d). \tag{1.9}$$

It is clear that when computing the continuous volume using Equation (1.9) $\mathcal{P}$ must be full-dimensional, if not we will get a continuous volume of 0. As $\mathcal{B}_n$ is not full dimensional we must extend our notion of volume and consider the relative volume. This is the volume relative to the sublattice $(\text{span}\,\mathcal{P}) \cap \mathbb{Z}^d$ where $\text{span}\,\mathcal{P}$ is the affine span of $\mathcal{P}$ [15]. Instead of dividing by t to the power of the dimension $\mathcal{P}$ lives in, we instead divide by t to the power of the dimension of $\mathcal{P}$. The *relative volume* of an m-dimensional polytope $\mathcal{P} \subset \mathbb{R}^d$ is given by

$$\nu(\mathcal{P}) = \lim_{t\to\infty} \frac{1}{t^m} \#(t\mathcal{P} \cap \mathbb{Z}^d), \tag{1.10}$$

for $m < d$. Therefore the leading coefficient of the Ehrhart polynomial $H_n(t)$ is the relative volume of $\mathcal{B}_n$.

Example 1.5.1. We will show in Chapter 2 that the Ehrhart Polynomial for the 4-dimensional polytope $\mathcal{B}_3$ is $H_3(t) = \frac{1}{8}t^4 + \frac{3}{4}t^3 + \frac{15}{8}t^2 + \frac{9}{4}t + 1$. Using Equation (1.10) the relative volume for $\mathcal{B}_3$ is

$$\nu(\mathcal{B}_3) = \lim_{t\to\infty} \frac{1}{t^4} \cdot \left(\frac{1}{8}t^4 + \frac{3}{4}t^3 + \frac{15}{8}t^2 + \frac{9}{4}t + 1\right) = \frac{1}{8}.$$

Relative volume is in units equal to the volume of the fundamental domain of the affine subspace spanned by $\mathcal{P}$ [14, 24] and can be converted back to continuous volume. For the

case of the Birkhoff polytope the continuous volume is equivalent to

$$\text{vol}(\mathcal{B}_n) = n^{n-1}\nu(\mathcal{B}_n). \tag{1.11}$$

The details of this equivalence are in [25] and the Appendix of [20].

In 2007, Canfield and McKay [18, 19] were able to provide the first asymptotic estimate of the relative volume of $\mathcal{B}_n$ by expressing the contingency tables as an integral and then estimating its value using the saddle-point method [19]. For any $\epsilon > 0$ they give the asymptotic estimate equivalent to

$$\nu(\mathcal{B}_n) = \frac{1}{(2\pi)^{n-\frac{1}{2}} n^{n(n-1)}} \exp\left(\frac{1}{3} + n^2 + \mathcal{O}(n^{\frac{-1}{2}+\epsilon})\right) \tag{1.12}$$

as $n \to \infty$.
As the relative volume of the Birkhoff polytope is only known for $n \leq 10$ the accuracy of such a formula can only be tested for a small set. For example, using Table 2.1 when $n = 3$ Equation (1.12) is off by a factor of 1.25 whereas in when $n = 10$ Equation (1.12) is off by a factor of 1.11. The asymptotic estimate given by Canfield and McKay proves to be quite accurate making Equation (1.12) invaluable when estimating the relative volume of $\mathcal{B}_n$ for $n > 10$.

Chapter 2

Existing methods to compute the relative volume of $\mathcal{B}_n$

This chapter will survey four existing methods used to compute the Ehrhart polynomial and the relative volume of the Birkhoff polytope given by

$$\mathcal{B}_n = \{x \in \mathbb{R}^{n^2}_{\geq 0} : \mathbf{A}\mathbf{x} = \mathbf{b}\},$$

for the matrix $\mathbf{A} \in \mathbb{Z}^{2n \times n^2}$ and vector $\mathbf{b} \in \mathbb{Z}^{2n}$ given in Equation (1.7). Section 2.1 will discuss the straightforward implementation of polynomial interpolation and Section 2.2 will discuss the use of Euler's generating function to extract the Ehrhart polynomial. The use of generating functions will prove more time consuming for smaller values of n but a more efficient method than polynomial interpolation for $n \geq 4$. Both of these sections will take work from Beck and Robins [14, 15]. We then discuss the groundbreaking work of Beck and Pixton [11, 14] in Section 2.3. They were able to use complex analytic techniques, in particular the residue theorem, to calculate $\nu(\mathcal{B}_9)$ and $\nu(\mathcal{B}_{10})$, shown in Table 2.1. We end this chapter with a look at Barvinok's algorithm which is implemented in the open-source computer program `LattE` [4]. This algorithm will be presented in Section 2.4 using the work of De Loera et al. [22, 23]. For the case of the Birkhoff polytope, `LattE` proves more efficient than polynomial interpolation and the use of generating functions, nevertheless, it cannot compete with complex analytic techniques. The new methods that we will present in Chapter 3 will prove more efficient than interpolation, generating functions and Barvinok's algorithm. This comes down to their ability to handle the complexity of the Birkhoff polytope as n gets larger. We will see that complex analytic techniques prove to be the most efficient. It is important to note that there are other published methods in circulation to compute the Ehrhart polynomial of both the Birkhoff polytope and general polytopes, including but not limited to [6, 47, 20, 27].

2.1 Polynomial interpolation

Since we know a priori that $H_n(t)$ is a polynomial, interpolation is a straightforward method to compute the Ehrhart polynomial but can be inefficient as n gets larger due to the dimension of $\mathcal{B}_n$. Given $d+1$ dilates we can interpolate the polynomial

$$H_n(t) = c_d t^d + c_{d-1} t^{d-1} + ... + c_1 t + c_0$$

using the well known Vandermonde matrix (2.3) [15]. We describe the mechanics behind polynomial interpolation and provide an applicable theorem to aid in the computation of the Ehrhart polynomial of $\mathcal{B}_n$. The method of interpolation requires that the polynomial pass through all given points thus

$$H_n(t_i) = c_d t_i^d + c_{d-1} t_i^{d-1} + ... + c_1 t_i + c_0 = f_i \text{ for } i = 1, ..., d+1. \tag{2.1}$$

Equation (2.1) can be expressed as the linear system

$$\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{d+1} \end{pmatrix} = \mathbf{V} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{pmatrix}, \tag{2.2}$$

where $\mathbf{V}$ is the well known Vandermonde matrix

$$\mathbf{V} = \begin{pmatrix} 1 & t_1 & \cdots & t_1^{d-1} & t_1^d \\ 1 & t_2 & \cdots & t_2^{d-1} & t_2^d \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & t_{d+1} & \cdots & t_{d+1}^{d-1} & t_{d+1}^d \end{pmatrix}. \tag{2.3}$$

We isolate the coefficients of $H_n(t)$ to retrieve what is commonly referred to as the Lagrange interpolation formula [15],

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{pmatrix} = \mathbf{V}^{-1} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{d+1} \end{pmatrix}. \tag{2.4}$$

Corollary 2.1.1. *Given $d+1$ distinct real numbers $t_1, t_2, ..., t_{d+1}$ the inverse of the Vandermonde matrix exists.*

Proof. It is well know that

$$\det(V) = \prod_{1 \leq k < j \leq d+1} (t_j - t_k) \neq 0.$$

□

Let us now prove the existence and uniqueness of a polynomial found through interpolation.

Theorem 2.1.1 (Existence and Uniqueness)**.** *Given $d+1$ distinct real numbers $x_1, x_2, ..., x_{d+1}$ and arbitrary values $f_1, f_2, ..., f_{d+1}$ there exists a unique polynomial $p(x)$ of degree less than or equal to d such that $p(x_i) = f_i$ for $i = 1, .., d+1$.*

Sketch of Proof. Let $p_1(x)$ and $p_2(x)$ be polynomials of degree less than or equal to d with

$$p_1(x_i) = p_2(x_i) = f_i \text{ for } i = 1, ..., d+1.$$

Then there exists a polynomial $q(x) = p_1(x) - p_2(x)$ of degree less than or equal to d satisfying $q(x_i) = 0$ for $i = 1, ..., d+1$. By the Fundamental Theorem of Algebra we know that the number of roots of a nonzero polynomial is equal to its degree, therefore $q(x)$ must be the zero polynomial and hence $p_1(x) = p_2(x)$. Existence is proven by finding such a polynomial as demonstated at the beginning of this section. □

Recall that $H_n(t)$ is a polynomial of degree $(n-1)^2$ therefore one needs to evaluate $H_n(t)$ for $(n-1)^2 + 1$ values in order to interpolate. Through symmetry (Equation (2.5)) and trivial dilates of $H_n(t)$ (Equation (2.6), $H_n(0) = 1$, and $H_n(1) = n!$) the following theorem reduces the number of evaluations to

$$\left\lfloor \frac{1}{2} \left((n+1)^2 + 1 - (n-1) - 2 \right) \right\rfloor = \left\lfloor \frac{1}{2} \left(n^2 - 3n + 1 \right) \right\rfloor.$$

Theorem 2.1.2 ([15, Theorem 6.3])**.** *The polynomial $H_n(t)$ satisfies*

$$H_n(-n-t) = (-1)^{(n-1)^2} H_n(t) \tag{2.5}$$

and

$$H_n(-1) = H_n(-2) = ... = H_n(-n+1) = 0. \tag{2.6}$$

Proof. The proof is based on the Ehrhart-Macdonald reciprocity law

$$L_{\mathcal{P}}(-t) = (-1)^{\dim(\mathcal{P})} L_{\mathcal{P}^\circ}(t),$$

where $\mathcal{P}^\circ$ denotes the interior of $\mathcal{P}$. The proof of this law can be found in [15, 36]. Therefore we have

$$H_n(-t) = (-1)^{(n-1)^2} H_n^\circ(t). \tag{2.7}$$

In the case of $\mathcal{B}_n$ the points of the interior are those $n \times n$ doubly stochastic matrices in which all entries are greater than 0 [13]. Consider an $n \times n$ semi-magic square with all integer entries greater than 0 and magic sum t, removing 1 from each entry gives an $n \times n$ semi-magic square with magic sum $t - n$. This bijection proves that $H_n^\circ(t) = H_n(t - n)$. Plugging this into the Equation (2.7) and substituting $t = n + t$ proves Equation (2.5). When considering the relative interior of $\mathcal{B}_n$, the construction of the $n \times n$ semi-magic squares (and hence the interior lattice points of $\mathcal{B}_n$) requires that the row and column sum be greater than or equal to n. Therefore $H_n^\circ(t) = 0$ for $t = 1, ..., n-1$, and by Equation (2.7), $H_n(-1) = H_n(-2) = ... = H_n(-n+1) = 0$. □

Example 2.1.1. To interpolate $H_3(t) = c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0$, for c_i unknown for $i = 0, ..., 4$, we evaluate $H_3(t)$ at 5 different values. Using Theorem 2.1.2 and the trivial solutions we have $H_3(-3) = (-1)^{(2)^2} H_3(0) = 1$, $H_3(-2) = H_3(-1) = 0$, $H_3(0) = 1$, and $H_3(1) = 3!$. Substituting the t_i values into Equation (2.4) gives

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ \frac{-1}{12} & \frac{1}{2} & \frac{-3}{2} & \frac{5}{6} & \frac{1}{4} \\ \frac{-1}{24} & \frac{1}{6} & \frac{1}{4} & \frac{-5}{6} & \frac{11}{24} \\ \frac{1}{12} & \frac{-1}{2} & 1 & \frac{-5}{6} & \frac{1}{4} \\ \frac{1}{24} & \frac{-1}{6} & \frac{1}{4} & \frac{-1}{6} & \frac{1}{24} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 6 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{9}{4} \\ \frac{15}{8} \\ \frac{3}{4} \\ \frac{1}{8} \end{pmatrix}. \tag{2.8}$$

Therefore

$$H_3(t) = \frac{1}{8}t^4 + \frac{3}{4}t^3 + \frac{15}{8}t^2 + \frac{9}{4}t + 1 \text{ and } \nu(\mathcal{B}_3) = \frac{1}{8}.$$

This method, along with some ingenuity, was used by Chan and Robbins [20] to calculate $\nu(\mathcal{B}_n)$ for $n \leq 8$. Nevertheless the time needed to evaluate $H_n(t)$ for arbitrary values of t when $n \geq 4$ makes polynomial interpolation an inefficient method to calculate the Ehrhart polynomial and relative volume of $\mathcal{B}_n$.

2.2 Euler's Generating Function

Generating functions are a better suited tool than polynomial interpolation to study larger cases of $\mathcal{B}_n$. Specifically, we use Euler's generating function to compute $H_n(t)$ and in turn the relative volume of the Birkhoff polytope. We denote by $F(z)$ the series

$$F(z) = \sum_{n \geq 0} f_n z^n,$$

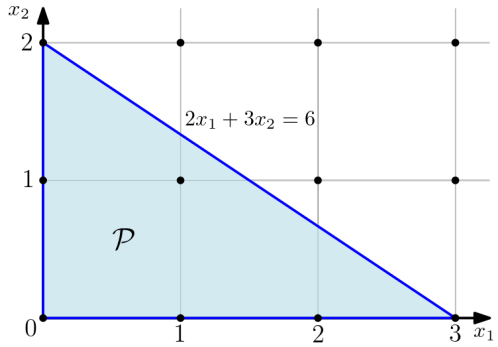


Figure 2.1: 2-dimensional polytope $\mathcal{P} = \{(x_1, x_2) \in \mathbb{R}^2_{\geq 0} : 2x_1 + 3x_2 \leq 6\}$.

and use the convention that $[z^n]F(z)$ denotes the coefficient of z^n in the Taylor series expansion of $F(z)$,

$$[z^n]F(z) = [z^n]\left(\sum_{n=0}^{\infty} f_n z^n\right) := f_n.$$

Let $\text{const}_z(F(z))$ denote $[z^0]F(z)$.

Example 2.2.1. Consider the 2-dimensional polytope

$$\mathcal{P} = \{(x_1, x_2) \in \mathbb{R}^2_{\geq 0} : 2x_1 + 3x_2 \leq 6\},$$

shown in Figure 2.1. We aim to determine the lattice point enumerator for $\mathcal{P}$:

$$\begin{aligned} L_{\mathcal{P}}(t) &= \#\{(x_1, x_2, x_3) \in \mathbb{Z}^3_{\geq 0} : 2x_1 + 3x_2 + x_3 = 6t\} \\ &= \#\{\mathbf{x} \in \mathbb{Z}^3_{\geq 0} : (2, 3, 1)\mathbf{x} = 6t\}. \end{aligned} \tag{2.9}$$

To do this, we take the product of the geometric sequences for each term in the expression $2x_1 + 3x_2 + x_3$,

$$\begin{aligned} \left(\frac{1}{1-z^2}\right)\left(\frac{1}{1-z^3}\right)\left(\frac{1}{1-z}\right) &= \sum_{n_1 \geq 0} z^{2n_1} \sum_{n_2 \geq 0} z^{3n_2} \sum_{n_3 \geq 0} z^{n_3} \\ &= \sum_{n_1 \geq 0} \sum_{n_2 \geq 0} \sum_{n_3 \geq 0} z^{2n_1 + 3n_2 + n_3} \\ &= \sum_{n \geq 0} L_{\mathcal{P}}(n) z^n. \end{aligned} \tag{2.10}$$

The left side of the Equation (2.10) is the generating function for the infinite sequence $\{L_{\mathcal{P}}(n)\}_{n \geq 1}$. We care specifically about the coefficient when $n = 6t$ so we rewrite Equation (2.10) as

$$\sum_{n \geq 0} L_{\mathcal{P}}(n) z^{n-6t} = \left(\frac{1}{(1-z^2)(1-z^3)(1-z)}\right)\frac{1}{z^{6t}}.$$

Taking the constant with respect to z of both sides gives

$$L_{\mathcal{P}}(6t) = \text{const}_z \left(\frac{1}{(1-z^2)(1-z^3)(1-z)z^{6t}} \right).$$

When dealing with the Birkhoff polytope, we require *multivariate generating functions* which are multivariate Taylor series of the form

$$G(z_1, ..., z_d) = \sum_{n_1,...,n_d \geq 0} g_{n_1,...,n_d} z_1^{n_1} ... z_d^{n_d},$$

where $g_{n_1,...,n_d} \in \mathbb{N}$. A multivariate generating function allows us to encode $H_n(t)$ in the ring $\mathbb{Z}[[z_1, z_2, ..., z_d]]$. Now that we have a general understanding of the generating function manipulations relevant to our problem we state the theorem of Euler's generating function.

Theorem 2.2.1 (Euler's generating function [15])**.** *Suppose the convex polytope $\mathcal{P}$ is given by $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^d_{\geq 0} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ for some matrix $\boldsymbol{A} \in \mathbb{Z}^{m \times d}$ and vector $\boldsymbol{b} \in \mathbb{Z}^m$. Then the lattice point enumerator of $\mathcal{P}$ can be computed as follows:*

$$L_{\mathcal{P}}(t) = const_{\boldsymbol{z}} \left(\frac{1}{(1-\boldsymbol{z}^{\boldsymbol{c}_1})(1-\boldsymbol{z}^{\boldsymbol{c}_2}) \cdots (1-\boldsymbol{z}^{\boldsymbol{c}_d}) \boldsymbol{z}^{t\boldsymbol{b}}} \right), \tag{2.11}$$

where $\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_d$ denotes the columns of $\boldsymbol{A}$.

Proof. Expand the generating function given in Equation (2.11) in terms of its geometric series,

$$\begin{aligned} \frac{1}{(1-\mathbf{z}^{\mathbf{c}_1})(1-\mathbf{z}^{\mathbf{c}_2}) \cdots (1-\mathbf{z}^{\mathbf{c}_d}) \mathbf{z}^{t\mathbf{b}}} &= \left(\sum_{n_1 \geq 0} \mathbf{z}^{n_1 \mathbf{c}_1} \right) \left(\sum_{n_2 \geq 0} \mathbf{z}^{n_2 \mathbf{c}_2} \right) \cdots \left(\sum_{n_d \geq 0} \mathbf{z}^{n_d \mathbf{c}_d} \right) \frac{1}{z^{t\mathbf{b}}} \\ &= \sum_{n_1 \geq 0} \sum_{n_2 \geq 0} \cdots \sum_{n_d \geq 0} \mathbf{z}^{n_1 \mathbf{c}_1 + n_2 \mathbf{c}_2 + ... + n_d \mathbf{c}_d - t\mathbf{b}} \\ &= \sum_{n_1 \geq 0} \sum_{n_2 \geq 0} \cdots \sum_{n_d \geq 0} \mathbf{z}^{\mathbf{A}\mathbf{n} - t\mathbf{b}}. \end{aligned} \tag{2.12}$$

Extracting the constant term with respect to z_i for $i = 1, .., d$ gives the number of integer vectors $\mathbf{n}$ such that $\mathbf{A}\mathbf{n} = t\mathbf{b}$, hence the number of lattice points contained in $t\mathcal{P}$. □

Beck and Robins [15] show how to use partial fraction decomposition and singularity analysis to decompose Theorem 2.2.1 into a clean and concise generating function expression for $H_n(t)$ presented in Theorem 2.2.2. Note that a *singularity* is a point where a functions fails to be defined. Recall the Birkhoff polytope given by $\mathcal{B}_n = \{\mathbf{x} \in \mathbb{R}^{n^2} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ for the

matrix $\mathbf{A} \in \mathbb{Z}^{2n \times n^2}$ and vector $\mathbf{b} \in \mathbb{Z}^{2n}$ given in Equation (1.7). Let the variables $z_1, ..., z_n$ denote the first n rows of $\mathbf{A}$ and $w_1, ..., w_n$ the last n rows. Using Theorem 2.2.1 we get the following generating function for $H_n(t)$,

$$H_n(t) = \text{const}_{z_1 \cdots z_n w_1 \cdots w_n} \left(\frac{1}{\prod_{1 \leq j,k \leq n} (1 - z_j w_k)(\prod_{1 \leq j \leq n} z_j \prod_{1 \leq k \leq n} w_k)^t} \right). \tag{2.13}$$

Example 2.2.2. Consider the polytope

$$\mathcal{B}_2 = \left\{ \mathbf{x} \in \mathbb{R}^4_{\geq 0} : \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

Using Theorem 2.11 and Equation (2.13) we have the following generating function for $H_2(t)$:

$$H_2(t) = \text{const}_{z_1 z_2 w_1 w_2} \left(\frac{1}{(1 - z_1 w_1)(1 - z_1 w_2)(1 - z_2 w_1)(1 - z_2 w_2)(z_1 z_2 w_1 w_2)^t} \right) \tag{2.14}$$

where z_1, z_2 denotes the first two rows of $\mathbf{A}$ and w_1, w_2 the last two.

Placing an ordering on the constant term computation simplifies Equation (2.14) to

$$H_2(t) = \text{const}_{z_1 z_2} \left(\frac{1}{(z_1 z_2)^t} \text{const}_{w_1} \left(\frac{1}{(1 - z_1 w_1)(1 - z_2 w_1) w_1^t} \right) \right.$$
$$\left. \times \ \text{const}_{w_2} \left(\frac{1}{(1 - z_1 w_2)(1 - z_2 w_2) w_2^t} \right) \right).$$

Note that w_1, w_2 are "dummy" variables so we set $w = w_1 = w_2$ and simplify,

$$H_2(t) = \text{const}_{z_1 z_2} \left(\frac{1}{(z_1 z_2)^t} \left(\text{const}_w \frac{1}{(1 - z_1 w)(1 - z_2 w) w^t} \right)^2 \right).$$

The ordering and simplification used in Example (2.2.2) works for all n therefore Equation (2.13) can be written as,

$$H_n(t) = \text{const}_{z_1 \cdots z_n} \left(\frac{1}{(z_1 z_2 \cdots z_n)^t} \left(\text{const}_w \frac{1}{(1 - z_1 w)(1 - z_2 w) \cdots (1 - z_n w) w^t} \right)^n \right). \tag{2.15}$$

The innermost expression can be rewritten using partial fraction decomposition and isolating the singularities with respect to w as follows,

$$\begin{aligned}\text{const}_w \frac{1}{(1-z_1w)(1-z_2w)\cdots(1-z_nw)w^t} &= \text{const}_w \left(\frac{A_1}{w-\frac{1}{z_1}} + \frac{A_2}{w-\frac{1}{z_2}} + ... + \frac{A_n}{w-\frac{1}{z_n}} + \sum_{k=1}^{t} \frac{B_k}{w^k} \right) \\ &= \text{const}_w \left(\frac{A_1}{w-\frac{1}{z_1}} + \frac{A_2}{w-\frac{1}{z_2}} + ... + \frac{A_n}{w-\frac{1}{z_n}} \right).\end{aligned} \tag{2.16}$$

To solve for A_k we multiply both sides of Equation (2.16) by $w-\frac{1}{z_k}$ resulting in

$$\begin{aligned}-\frac{1}{(1-z_1w)(1-z_2w)\cdots z_k \cdots(1-z_nw)w^t} = \\ \frac{\left(w-\frac{1}{z_k}\right)A_1}{w-\frac{1}{z_1}} + \frac{\left(w-\frac{1}{z_k}\right)A_2}{w-\frac{1}{z_2}} + ... + A_k + ... + \frac{\left(w-\frac{1}{z_k}\right)A_n}{w-\frac{1}{z_n}},\end{aligned}$$

and take the limit as $w \to \frac{1}{z_k}$ to isolate A_k and obtain

$$A_k = -\frac{1}{\left(1-\frac{z_1}{z_k}\right)\cdots z_k \cdots \left(1-\frac{z_n}{z_k}\right)\frac{1}{z_k^t}} = -\frac{z_k^{t-1}}{\left(1-\frac{z_1}{z_k}\right)\cdots\left(1-\frac{z_n}{z_k}\right)} = -\frac{z_k^{t+n-2}}{\prod_{j\neq k}(z_k-z_j)}. \tag{2.17}$$

For $k = 1, ..., n$ substitute the values of A_k into Equation (2.16) to get

$$\begin{aligned}\text{const}_w \frac{1}{(1-z_1w)(1-z_2w)\cdots(1-z_nw)w^t} &= \text{const}_w \left(\frac{A_1}{w-\frac{1}{z_1}} + \frac{A_2}{w-\frac{1}{z_2}} + ... + \frac{A_n}{w-\frac{1}{z_n}} \right) \\ &= -A_1z_1 - A_2z_2 - ... - A_nz_n \\ &= \sum_{k=1}^{n} \frac{z_k^{t+n-1}}{\prod_{j\neq k}(z_k-z_j)}.\end{aligned} \tag{2.18}$$

Plugging Equation (2.18) into Equation (2.15) gives the following theorem.

Theorem 2.2.2 ([15, Theorem 6.6])**.** *The Ehrhart polynomial $H_n(t)$ of the n^{th} Birkhoff polytope satisfies*

$$H_n(t) = const_{z_1\cdots z_n} \left(\frac{1}{(z_1z_2\cdots z_n)^t} \left(\sum_{k=1}^{n} \frac{z_k^{t+n-1}}{\prod_{j\neq k}(z_k-z_j)} \right)^n \right).$$

Proof. Follows directly from Equation (2.15) and Equation (2.18). □

Example 2.2.2 (Continued)**.** Using Theorem 2.2.2 we have

$$H_2(t) = \text{const}_{z_1 z_2} \left((z_1 z_2)^{-t} \left(\frac{z_1^{t+1}}{(z_1 - z_2)} + \frac{z_2^{t+1}}{(z_2 - z_1)} \right)^2 \right).$$

Taking the constant with respect to z_1 then z_2 gives the Ehrhart polynomial for $H_2(t)$:

$$\begin{aligned} H_2(t) &= \text{const}_{z_2} \left(\text{const}_{z_1} \left(\frac{z_1^{t+2} z_2^{-t}}{(z_1 - z_2)^2} - \frac{2 z_1 z_2}{(z_1 - z_2)^2} + \frac{z_1^{-t} z_2^{t+2}}{(z_2 - z_1)^2} \right) \right) \\ &= \text{const}_{z_2} \left(\text{const}_{z_1} \left(\sum_{k \geq 0} \frac{k+1}{z_2^{k+t+2}} z_1^{k+t+2} - 2 \sum_{k \geq 0} \frac{k+1}{z_2^{k+1}} z_1^{k+1} + \sum_{k \geq 0} \frac{k+1}{z_2^{k-t}} z_1^{k-t} \right) \right) \\ &= \text{const}_{z_2}(0 - 2(0) + (t+1)) \\ &= t + 1. \end{aligned}$$

Therefore the relative volume of $\mathcal{B}_2$ is

$$\nu(\mathcal{B}_2) = 1.$$

It is straightforward to see that for small values of n the method of Euler's generating function is more complex than that of polynomial interpolation. When $n = 4$ the method of interpolation requires the evaluation of $H_4(t)$ at 2 different values, and more so for larger values of n. This is a difficult task when computing by hand, whereas one could work out $H_4(t)$ by pen and paper using the method of Euler's generating function. Therefore, for $n \geq 4$ we believe this method is more efficient than polynomial interpolation.

2.3 Complex Analytic Techniques

The presentation of this strategy was originally given by Beck [11] and again in terms of its application to Birkhoff polytopes alongside Pixton [14]. They were able to use complex analysis, specifically the residue theorem, to compute the Ehrhart polynomial for $\mathcal{B}_9$ and the relative volume of $\mathcal{B}_{10}$. The main result presented and proved by Beck in [11] will now be stated.

Theorem 2.3.1 ([11, Theorem 8])**.** *Suppose the convex polytope $\mathcal{P}$ is given by $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^d_{\geq 0} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ for a matrix $\boldsymbol{A} \in \mathbb{Z}^{m \times d}$ and vector $\boldsymbol{b} \in \mathbb{Z}^m$, and denote the columns of $\boldsymbol{A}$ by $\boldsymbol{c}_1, ..., \boldsymbol{c}_d$. Then*

$$L_p(t) = \frac{1}{(2\pi i)^m} \int_{|z_1| = \epsilon_1} \cdots \int_{|z_m| = \epsilon_m} \frac{z_1^{-tb_1 - 1} \cdots z_m^{-tb_m - 1}}{(1 - \boldsymbol{z}^{\boldsymbol{c}_1}) \cdots (1 - \boldsymbol{z}^{\boldsymbol{c}_d})} d\boldsymbol{z}, \tag{2.19}$$

where $0 < \epsilon_1, ..., \epsilon_m < 1$ are distinct real numbers.

Corollary 2.3.1 will show why we require $\epsilon_1, ..., \epsilon_m$ to be distinct real numbers. Before we prove Theorem 2.3.1 we cover some key definitions.

Definition. A function is *analytic* in a region if it is complex differentiable at every point in that region.

If a function $f(z)$ is analytic everywhere inside of a simple, closed, positively oriented contour γ in the complex plane, then

$$\int_\gamma f(z)dz = 0.$$

Let $f(z)$ be analytic except for the points $z_1, z_2, ..., z_n$ inside of γ, these points are called *poles*. $f(z)$ has a pole of order 1 at z_0 if its Laurent series centred at z_0 can be written as $f(z) = \sum_{n=0}^{\infty} a_n(z-z_0)^n + \frac{b_1}{(z-z_0)}$ for $b_1 \neq 0$. Then the *residue* of $f(z)$ at z_0 can be computed as

$$\text{Res}(f(z); z_0) = \lim_{z \to z_0} (z - z_0)f(z) = b_1.$$

Furthermore, $f(z)$ has a pole of order k at z_0 if its Laurent series centred at z_0 can be written as $f(z) = \sum_{n=0}^{\infty} a_n(z - z_0)^n + \frac{b_1}{(z-z_0)} + ... + \frac{b_k}{(z-z_0)^k}$ for $b_k \neq 0$. Then

$$\text{Res}(f(z); z_0) = \frac{1}{(k-1)!} \lim_{z \to z_0} \frac{d^{k-1}}{dz^{k-1}} \{(z - z_0)^k f(z)\} = b_1.$$

Theorem 2.3.2 (Residue Theorem)**.** *If γ is a simple, closed, positively oriented contour in the complex plane and $f(z)$ is analytic except for the points $z_1, z_2, ..., z_n$ inside the contour γ, then*

$$\int_\gamma f(z)dz = 2\pi i \sum_{k=1}^{n} Res(f(z); z_k).$$

These definitions can be extended naturally to the multivariate case.

Proof of Theorem 2.3.1. Let $\mathbf{c}_j$ denote the j^{th} column of $\mathbf{A}$ and $\mathbf{r}_k$ the k^{th} row. The hyperplane equalities of the a dilated polytope $t\mathcal{P}$ can be written as

$$\begin{aligned} \langle \mathbf{r}_1, \mathbf{x} \rangle &= tb_1 \\ \langle \mathbf{r}_2, \mathbf{x} \rangle &= tb_2 \\ &\vdots \\ \langle \mathbf{r}_m, \mathbf{x} \rangle &= tb_m. \end{aligned} \tag{2.20}$$

where $\mathbf{x} \in \mathbb{R}^d_{\geq 0}$. Consider

$$f(\mathbf{z}) = f(z_1, ..., z_m) = \frac{z_1^{-tb_1-1} \cdots z_m^{-tb_m-1}}{(1-\mathbf{z}^{\mathbf{c}_1}) \cdots (1-\mathbf{z}^{\mathbf{c}_d})},$$

and integrate $f(\mathbf{z})$ with respect to each variable over small, simple, closed, positively oriented contours:

$$\int_{|z_1|=\epsilon_1} \cdots \int_{|z_m|=\epsilon_m} f(\mathbf{z}) d\mathbf{z}. \tag{2.21}$$

Choose $0 < \epsilon_1, ..., \epsilon_m < 1$ so that we can expand each $\frac{1}{1-\mathbf{z}^{\mathbf{c}_j}}$ into its power series around 0. Expanding $f(\mathbf{z})$ around 0 gives the following Laurent series,

$$\frac{z_1^{-tb_1-1} \cdots z_m^{-tb_m-1}}{(1-\mathbf{z}^{\mathbf{c}_1}) \cdots (1-\mathbf{z}^{\mathbf{c}_d})} = z_1^{-tb_1-1} \cdots z_m^{-tb_m-1} \sum_{v_1 \geq 0} \mathbf{z}^{v_1 \mathbf{c}_1} \sum_{v_2 \geq 0} \mathbf{z}^{v_2 \mathbf{c}_2} \cdots \sum_{v_d \geq 0} \mathbf{z}^{v_d \mathbf{c}_d}.$$

Upon expansion, each term will have the form

$$z_1^{\langle \mathbf{v}, \mathbf{r}_1 \rangle - tb_1 - 1} \cdots z_m^{\langle \mathbf{v}, \mathbf{r}_m \rangle - tb_1 - 1},$$

where $\mathbf{v} = (v_1, ..., v_d)$. Therefore, by the residue theorem, the integral given in Equation (2.21) will not evaluate to 0 only for those terms in which $\mathbf{v}$ satisfies the equalities in (2.20). □

We will present how Beck and Pixton used the residue theorem to simplify Theorem 2.3.1 to a more efficient integral, presented in Theorem 2.3.3. We will then use Theorem 2.3.3 to calculate $H_3(t)$. We then discuss their important result, Corollary 2.3.1, as this corollary will form the basis for the novel methods discussed in Chapter 3. We end this section by discussing the methods Beck and Pixton implemented to increase efficiency and present their computation times.

To begin, recall the Birkhoff polytope given by $\mathcal{B}_n = \{\mathbf{x} \in \mathbb{R}^{n^2} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ for the matrix $\mathbf{A} \in \mathbb{Z}^{2n \times n^2}$ and vector $\mathbf{b} \in \mathbb{Z}^{2n}$ given in Equation (1.7). Let the variables $z_1, ..., z_n$ denote the first n rows of $\mathbf{A}$ and $w_1, ..., w_n$ the last n rows and apply Theorem 2.3.1, taking each integral over a circle of radius less than 1 centered at 0:

$$H_n(t) = \frac{1}{(2\pi i)^{2n}} \int \cdots \int \frac{(z_1 \cdots z_n w_1 \cdots w_n)^{-t-1}}{(1-z_1w_1) \cdots (1-z_1w_n) \cdots (1-z_nw_1) \cdots (1-z_nw_n)} d\mathbf{w} d\mathbf{z}. \tag{2.22}$$

Just as in Section 2.2 we place an ordering on the integrals, noting the symmetry that presents itself,

$$H_n(t) = \frac{1}{(2\pi i)^n} \int \cdots \int (z_1 \cdots z_n)^{-t-1} \left(\left(\frac{1}{2\pi i} \int \frac{w_1^{-t-1}}{(1 - z_1 w_1) \cdots (1 - z_n w_1)} dw_1 \right) \cdots \left(\frac{1}{2\pi i} \int \frac{w_n^{-t-1}}{(1 - z_1 w_n) \cdots (1 - z_n w_n)} dw_n \right) \right) d\mathbf{z}. \quad (2.23)$$

Recall that w_k is a dummy variable and simplify

$$H_n(t) = \frac{1}{(2\pi i)^n} \int \cdots \int (z_1 \cdots z_n)^{-t-1} \left(\frac{1}{2\pi i} \int \frac{w^{-t-1}}{(1 - z_1 w) \cdots (1 - z_n w)} dw \right)^n d\mathbf{z}. \quad (2.24)$$

The innermost integral of Equation (2.24) is taken around the smallest simple, closed, positively oriented curve and the integrand

$$f(w) = \frac{w^{-t-1}}{(1 - z_1 w) \cdots (1 - z_n w)},$$

is analytic everywhere inside the curve except for $w = 0$, therefore by the residue theorem

$$\int f(w) dw = 2\pi i \ \mathrm{Res}(f(w); 0).$$

The residue of $f(w)$ at 0 is equal to the negative of the sum of the residues of $f(w)$ at the singularities outside of the curve: $w_1 = z_1^{-1}, .., w_n = z_n^{-1}$. Therefore,

$$\begin{aligned}
\mathrm{Res}(f(w); 0) &= -\left(\mathrm{Res}\left(f(w); \frac{1}{z_1} \right) + ... + \mathrm{Res}\left(f(w); \frac{1}{z_n} \right) \right) \\
&= -\left(\lim_{w \to \frac{1}{z_1}} \left(w - \frac{1}{z_1} \right) f(w) + ... + \lim_{w \to \frac{1}{z_n}} \left(w - \frac{1}{z_n} \right) f(w) \right) \\
&= -\left(\lim_{w \to \frac{1}{z_1}} \frac{w - \frac{1}{z_1}}{1 - z_1 w} \frac{w^{-t-1}}{(1 - z_2 w) \cdots (1 - z_n w)} + ... \right. \\
&\qquad \left. + \lim_{w \to \frac{1}{z_n}} \frac{w - \frac{1}{z_n}}{1 - z_n w} \frac{w^{-t-1}}{(1 - z_1 w) \cdots (1 - z_{n-1} w)} \right) \\
&= \frac{1}{z_1} \frac{z_1^{t+1}}{\left(1 - \frac{z_2}{z_1}\right) \cdots \left(1 - \frac{z_n}{z_1}\right)} + ... + \frac{1}{z_n} \frac{z_n^{t+1}}{\left(1 - \frac{z_1}{z_n}\right) \cdots \left(1 - \frac{z_{n-1}}{z_n}\right)} \\
&= \frac{z_1^{t+n-1}}{(z_1 - z_2) \cdots (z_1 - z_n)} + ... + \frac{z_n^{t+n-1}}{(z_n - z_1) \cdots (z_n - z_{n-1})} \\
&= \sum_{k=1}^{n} \frac{z_k^{t+n-1}}{\prod_{j \neq k} (z_k - z_j)}.
\end{aligned}$$

Substituting this result back into Equation (2.24) gives the following theorem.

Theorem 2.3.3 ([14, Theorem 2]). *The Ehrhart polynomial $H_n(t)$ of the n^{th} Birkhoff polytope satisfies*

$$H_n(t) = \frac{1}{(2\pi i)^n} \int_{|z_1|=\epsilon_1} \cdots \int_{|z_n|=\epsilon_n} (z_1 ... z_n)^{-t-1} \left(\sum_{k=1}^{n} \frac{z_k^{t+n-1}}{\prod_{j \neq k}(z_k - z_j)} \right)^n d\boldsymbol{z}$$

for distinct $0 < \epsilon_1, ..., \epsilon_n < 1$.

Proof. Follows from Theorem 2.3.2, 2.24, and 2.3. □

Example 2.3.1. Consider $\mathcal{B}_3$, by Theorem 2.3.3 with $0 < \epsilon_3 < \epsilon_2 < \epsilon_1 < 1$

$$H_3(t) = \frac{1}{(2\pi i)^3} \int_{|z_1|=\epsilon_1} \int_{|z_2|=\epsilon_2} \int_{|z_3|=\epsilon_3} (z_1 z_2 z_3)^{-t-1} \left(\frac{z_1^{t+2}}{(z_1 - z_2)(z_1 - z_3)} + \frac{z_2^{t+2}}{(z_2 - z_1)(z_2 - z_3)} + \frac{z_3^{t+2}}{(z_3 - z_1)(z_3 - z_2)} \right)^3 d\mathbf{z}.$$

After expanding the cubic polynomial we have

$$H_3(t) = \frac{1}{(2\pi i)^3} \iiint \left(\frac{z_1^{2t+5} z_2^{-t-1} z_3^{-t-1}}{(z_1 - z_2)^3 (z_1 - z_3)^3} - \frac{3 z_1^{t+3} z_2 z_3^{-t-1}}{(z_1 - z_2)^3 (z_1 - z_3)^2 (z_2 - z_3)} \right) d\mathbf{z}, \quad (2.25)$$

as the rest of the terms evaluate to 0. For example, the term

$$\frac{z_1^{-t-1} z_2^{-t-1} z_3^{2t+5}}{(z_3 - z_2)^3 (z_3 - z_1)^3}$$

is analytic at the z_3-origin and $|z_1|, |z_2| > \epsilon_3$ therefore integrating this term with respect to z_3 gives 0. Similar explanations are given for the rest of the terms not in the integrand of Equation (2.25). We can evaluate the first integral in Equation (2.25) using symmetry and the residue theorem as follows,

$$\frac{1}{(2\pi i)^3} \iiint \frac{z_1^{2t+5} z_2^{-t-1} z_3^{-t-1}}{(z_1 - z_2)^3 (z_1 - z_3)^3} d\mathbf{z} = \frac{1}{(2\pi i)^2} \iint z_1^{2t+5} \left(\frac{z^{-t-1}}{(z_1 - z)^3} \right)^2 dz dz_1$$

$$= \frac{1}{2\pi i} \int z_1^{2t+5} \left(-\frac{1}{2}(-t-1)(-t-2) z_1^{-t-3} \right)^2 dz_1 = \binom{t+2}{2}^2.$$

Using similar tactics we calculate the second integral:

$$-\frac{3}{(2\pi i)^3} \iiint \frac{z_1^{t+3} z_2 z_3^{-t-1}}{(z_1 - z_2)^3 (z_1 - z_3)^2 (z_2 - z_3)} d\mathbf{z} = -3 \binom{t+3}{4}.$$

Adding the two integrals together gives

$$H_3(t) = \binom{t+2}{2}^2 - 3\binom{t+3}{4} = \frac{1}{8}t^4 + \frac{3}{4}t^3 + \frac{15}{8}t^2 + \frac{9}{4}t + 1,$$

and therefore

$$\nu(\mathcal{B}_3) = \frac{1}{8}.$$

We noted in Example (2.3.1) that not all the terms in the integrand will contribute to the final polynomial. This idea can be expressed more rigorously with the following proposition.

Proposition 2.3.1 ([14]). *Assume $p_1, ..., p_n$ are integers, q_{jk} are nonnegative integers $(1 \leq j, k \leq n)$, $0 < \epsilon_n < \cdots < \epsilon_1 < 1$, and*

$$f(z_1, ..., z_n) = \frac{\prod_{1 \leq j \leq n} z_j^{p_j}}{\prod_{1 \leq j,k \leq n} (z_j - z_k)^{q_{jk}}}. \tag{2.26}$$

If $1 \leq r \leq n$ and $d_r(f) < -r$ where $d_r(f)$ represents the degree of f in the variables $z_1, ..., z_r$, then

$$\int_{|z_1|=\epsilon_1} \cdots \int_{|z_n|=\epsilon_n} f(\boldsymbol{z}) d\boldsymbol{z} = 0.$$

Proposition 2.3.1 can be proved by induction on r. As n gets larger the number of terms that evaluate to 0 increases, this in turn increases computation time as one must detect each one of them beforehand or use time to calculate them out to 0. The following corollary ensures that integrals that evaluate to 0 are omitted.

Corollary 2.3.1 ([14, Corollary 4]). *For $0 < \epsilon_n < \cdots < \epsilon_1 < 1$ and $t \geq 0$,*

$$H_n(t) = \frac{1}{(2\pi i)^n} \int_{|z_1|=\epsilon_1} \cdots \int_{|z_n|=\epsilon_n} (z_1 \cdots z_n)^{-t-1} \sum_{a_1+...+a_n=n}^{*} \binom{n}{a_1, ..., a_n} \prod_{k=1}^{n} \left(\frac{z_k^{t+n-1}}{\prod_{j \neq k}(z_k - z_j)} \right)^{a_k} d\boldsymbol{z}.$$

where $\sum^*$ denotes the sum over those n-tuples of non-negative integers satisfying $a_1 + ... + a_n = n$ and $a_1 + ... + a_r > r$ for $1 \leq r < n$.

The constraint placed on the values of $a_1, .., a_n$ are known as *lecture hall partitions*. They can be visualized as a lattice path from $(0, 0)$ to (n, n) taking the steps $(1, a_1)$ to $(1, a_n)$ so the path stays strictly above the diagonal.

Proof. By Theorem 2.3.3 we have

$$H_n(t) = \frac{1}{(2\pi i)^n} \int \cdots \int (z_1 \cdots z_n)^{-t-1} \left(\sum_{k=1}^{n} \frac{z_k^{t+n-1}}{\prod_{j \neq k}(z_k - z_j)} \right)^n d\mathbf{z}.$$

which is equivalent to

$$H_n(t) = \frac{1}{(2\pi i)^n} \int \cdots \int (z_1 \cdots z_n)^{-t-1} \sum_{a_1+\ldots+a_n=n} \binom{n}{a_1, ..., a_n} \prod_{k=1}^{n} \left(\frac{z_k^{t+n-1}}{\prod_{j \neq k}(z_k - z_j)} \right)^{a_k} d\mathbf{z}. \tag{2.27}$$

We use Proposition 2.3.1 to show that the integral of a term in Equation (2.27) is 0 unless $\sum_{j=1}^{r} a_j > r$ for the partition $a_1 + \ldots + a_n = n$. Using the notation given in Proposition 2.3.1 a term with partition $a_1 + ... + a_n = n$ will have the following exponent for the numerator and denominator respectively for $1 \leq j, k \leq n$:

$$p_j = t(a_j - 1) + (n-1)a_j - 1 \text{ and } q_{jk} = a_j + a_k.$$

Suppose $1 \leq r < n$ and calculate the degree of the given term in $z_1, ..., z_r$. The degree of the numerator is

$$\sum_{j=1}^{r} p_j = t \sum_{j=1}^{r} (a_j - 1) + (n-1) \sum_{j=1}^{r} a_j - r,$$

and the denominator,

$$\begin{aligned} \sum_{j=1}^{r} \sum_{k=j+1}^{n} (a_j + a_k) &= (n-1) \sum_{j=1}^{r} a_j + r \sum_{j=r+1}^{n} a_j \\ &= (n-1) \sum_{j=1}^{r} a_j + r \sum_{j=1}^{n} a_j - r \sum_{j=1}^{r} a_j \\ &= (n-1) \sum_{j=1}^{r} a_j - r \sum_{j=1}^{r} (a_j - 1) + rn - r^2. \end{aligned}$$

Therefore

$$\begin{aligned} d_r(f) &= t \sum_{j=1}^{r} (a_j - 1) + (n-1) \sum_{j=1}^{r} a_j - r - (n-1) \sum_{j=1}^{r} a_j + r \sum_{j=1}^{r} (a_j - 1) - rn + r^2 \\ &= (t + r) \sum_{j=1}^{r} (a_j - 1) - r - r(n - r). \end{aligned}$$

It is clear that $r(n-r) > 0$ and $t \geq 0$, hence by Proposition 2.3.1 the integral of a given term in Equation (2.27) is 0 unless $\sum_{j=1}^{r} (a_j - 1) > 0$. □

Example 2.3.2. The only lecture hall partitions of 3 are $(3,0,0)$ and $(2,1,0)$. Using Corollary 2.3.1 we have

$$H_3(t) = \frac{1}{(2\pi i)^3}\int_{|z_1|=\epsilon_1}\cdots\int_{|z_3|=\epsilon_3}(z_1z_2z_3)^{-t-1}$$
$$\left(\binom{3}{3,0,0}\frac{z_1^{3t+6}}{(z_1-z_2)^3(z_1-z_3)^3}+\binom{3}{2,1,0}\frac{z_1^{2t+4}z_2^{t+2}}{(z_1-z_2)^2(z_1-z_3)^2(z_2-z_1)(z_2-z_3)}\right)d\mathbf{z},$$

which agrees with our findings in Example 2.3.1.

Corollary 2.3.1 is more efficient than Theorem 2.3.3, but as n gets larger the use of lecture hall partitions does little to cut down on computation time. That is because the number of lecture hall partitions equals the $(n-1)^{th}$ Catalan number

$$\frac{1}{n}\binom{2(n-1)}{n-1}=\frac{(2n-2)!}{n!(n-1)!},$$

which grows exponentially with n: $\sim 4^n n^{-\frac{3}{2}}\pi^{-\frac{1}{2}}$. Each integral must be evaluated a variable at a time, for larger values of n this has adverse effects on computation time. Beck and Pixton feel that once $n=7$ or 8 that this is equivalent to interpolating the polynomial. There are four implementations put forth and applied by Beck and Pixton in their C++ program to increase efficiency:

1. Identify when a function is analytic at the z_k origin.
2. Choose the most efficient order of integration.
3. Factor the integral if some of the variables appear in a symmetric fashion.
4. Suppress a particular integral if it does not contribute to the leading term (useful if you only want the relative volume).

With these implementations, they were able to compute the relative volume of the Birkhoff polytopes for $n \leq 10$, a value that was unattainable before. Their results for $n \leq 9$ are given in Table 2.1, note that the volumes provided are the relative volume of $\mathcal{B}_n$ multiplied by the constant n^{n-1}. The relative volume of $\mathcal{B}_{10}$ multiplied by 10^9 is equal to

$$\frac{7272912840167864209775084579901218625488232600525573333866078 89}{828160860106766855125676318796872729344622463533089422677980721388055739956270293750883504892820848640000000}$$

which, scaled down to a 1GHz processor, took 6160 days to compute [14]. We believe this method to be superior to interpolation due to its automated properties. The use of generating functions could be close in computation time given similar implementations to automate and speed up the constant value computations, though this was not tested conclusively.

n	volume of $\mathcal{B}_n$	time
1	1	<.01 sec
2	2	<.01 sec
3	$\frac{9}{8}$	<.01 sec
4	$\frac{176}{2835}$	<.01 sec
5	$\frac{23590375}{167382319104}$	<.01 sec
6	$\frac{9700106723}{1319281996032 \cdot 10^6}$	.18 sec
7	$\frac{77436678274508929033}{13730296368223523839986892 8 \cdot 10^8}$	15 sec
8	$\frac{5562533838576105333259507434329}{125890362600954779500814809426933398033089 28 \cdot 10^{10}}$	54 min
9	$\frac{55949812970279602224689568637276605247549669 1}{92692623409952636498965146712806984296051951329202419606108477153345536 \cdot 10^{14}}$	317 hr

Table 2.1: Volume and computation time in seconds for calculation of $\mathcal{B}_n$ for $n \leq 9$ presented in [14]

2.4 Barvinok's Algorithm and `LattE`

The final method to compute the relative volume of the Birkhoff polytope we will survey is the mechanics behind the open source software `LattE` (**Latt**ice Point **E**numeration) [4]. In 2001 De Loera and his team released the first version of `LattE` in which Barvinoks algorithm is used to count the number of integral points in a fixed dimensional polytope [8]; this was the first ever implementation of Barvinoks algorithm. Not only could `LattE` count the number of integral points in a polytope but was also able to compute the Ehrhart polynomial among other features. In 2010 `LattE integrale` was released with the ability to compute the volume of a rational polytope [2]. This calculation can be done in two ways: the triangulation-determinant method and the cone decomposition method.

This section will first discuss how `LattE` implements the triangulation-determinant method followed by the cone decomposition method to compute the volume of a d-polytope $\mathcal{P} \subset \mathbb{R}^d$. We then discuss how `LattE` computes the Ehrhart polynomial. This may seem valueless but when $n \geq 5$ our results lead us to believe it is computationally faster to compute the Ehrhart polynomial to obtain the volume of $\mathcal{P}$ than the aforementioned volume methods. It is important to note that the following calculations are done for full dimensional polytopes. When given a non full dimensional polytope $\mathcal{P} \subset \mathbb{R}^d$, `LattE` will first transform $\mathcal{P}$ into a full dimensional polytope of fewer variables such that there exists a one-to-one correspondence between integer points to preserve the characteristics of $\mathcal{P}$ [23]. For us, this means that `LattE` will compute the relative volume of $\mathcal{B}_n$. To aid in our understanding we will instead use the quadrilateral polytope $\mathcal{Q}$, presented in Figure 2.2, as a running example.

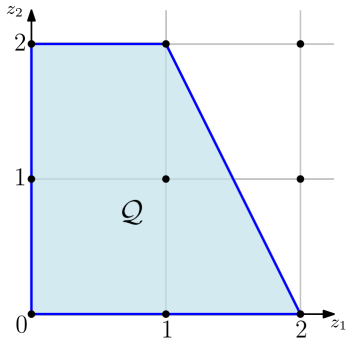


Figure 2.2: Quadrilateral polytope with vertices $(0,0),(2,0),(0,2)$ and $(1,2)$.

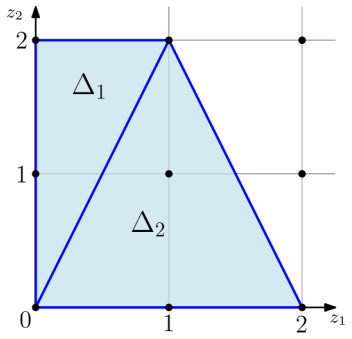


Figure 2.3: A possible triangulation of the polytope $\mathcal{Q}$.

2.4.1 Triangulation-determinant method

It is well known [5, 22, 42] that the volume of a d-simplex, Δ, is given by

$$\mathrm{vol}(\Delta) = \frac{1}{d!}\left|\det\begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_{d+1} \end{pmatrix}\right|,$$

where $v_i \in \mathbb{R}^d$ are the vertices of Δ for $i = 1, ..., d+1$. For a non-simplex d-polytope $\mathcal{P} \subset \mathbb{R}^d$ we have

$$\mathrm{vol}(\mathcal{P}) = \sum_{\Delta \in T} \mathrm{vol}(\Delta),$$

for T the finite set of triangulations of the polytope $\mathcal{P}$ as defined in Section 1.2.1.

Example 2.4.1. The quadrilateral polytope is not a simplex but can be triangulated into two 2-simplices, Δ_1 and Δ_2, shown in Figure 2.3. Therefore

$$\begin{aligned}
\mathrm{vol}(\mathcal{Q}) &= \mathrm{vol}(\Delta_1) + \mathrm{vol}(\Delta_2) \\
&= \frac{1}{2!}\left|\det\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 2 & 2 \end{pmatrix}\right| + \frac{1}{2!}\left|\det\begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}\right| \\
&= 3.
\end{aligned}$$

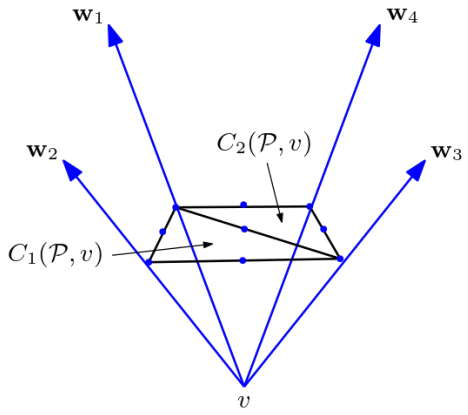


Figure 2.4: A possible triangulation of a cone $C(\mathcal{P}, v)$ with generators $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4 \in \mathbb{R}^3$.

The efficiency of this method is directly correlated to how complex the triangulation of the polytope is.

2.4.2 Cone decomposition method

De Loera et al. [22] describe how `LattE Integrale` computes the integral of a polynomial $f \in \mathbb{Q}[z_1, ..., z_n]$ over a rational convex d-polytope $\mathcal{P} \subset \mathbb{R}^d$. This is denoted by

$$\text{vol}(\mathcal{P}) = \int_{\mathcal{P}} f \; dm,$$

where dm is the *integral Lebesgue measure* on the affine hull of $\mathcal{P}$. We are interested solely in the volume of a polytope therefore we take the integral of 1 over the polytope $\mathcal{P} \subset \mathbb{R}^d$ as presented in Section 1.5. We let $V(\mathcal{P})$ denote the vertex set of a polytope $\mathcal{P}$. Recall the definition of a cone C and let $C(\mathcal{P}, v)$ denote the *supporting cone* of $\mathcal{P}$ at vertex v such that for all generators $\mathbf{w}_i$, $v + \epsilon \mathbf{w}_i \in \mathcal{P}$ for some $\epsilon > 0$. Let π_C denote the fundamental parallelepiped of the cone C. We will work with simplicial cones, i.e. a d-dimensional cone with d linearly dependent generators, so if $C(\mathcal{P}, v)$ is not simplicial we triangulate the cone to form a finite set T_v of simplicial d-cones.

Theorem 2.4.1 ([15, Theorem 3.2])**.** *Every pointed cone can be triangulated into simplicial cones using no new generators.*

Figure 2.4 depicts one possible triangulation of the supporting cone $C(\mathcal{P}, v)$ with 4 generators in $\mathbb{R}^3$. We now state two important corollaries without proof.

Corollary 2.4.1 ([22, Corollary 6])**.** *Let $\mathcal{P}$ be a bounded polytope with vertex set $V(\mathcal{P})$ and T_v a triangulation of the cone $C(\mathcal{P}, v)$. Then*

$$\text{vol}(\mathcal{P}) = \sum_{v \in V(\mathcal{P})} \sum_{C \in T_v} \text{vol}(C(\mathcal{P}, v)).$$

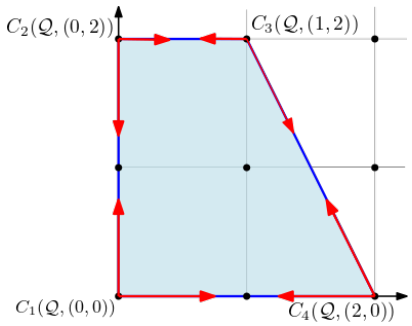


Figure 2.5: Four supporting cones of $\mathcal{Q}$.

Corollary 2.4.2 ([22, Corollary 5])**.** *For a simplicial cone C generated by $\boldsymbol{w}_1, ..., \boldsymbol{w}_d$, vertex $v \in \mathbb{R}^d$ and a linear form ℓ such that $\langle \ell, \boldsymbol{w}_i \rangle \neq 0$ for $i = 1, ..., d$ we have*

$$\operatorname{vol}(C(\mathcal{P}, v)) = \frac{1}{d!} \operatorname{vol}(\pi_C) \frac{(\langle \ell, v \rangle)^d}{\prod_{i=1}^{d} \langle -\ell, \boldsymbol{w}_i \rangle}, \tag{2.28}$$

where

$$\operatorname{vol}(\pi_C) = \left| \det \begin{pmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \cdots & \boldsymbol{w}_d \end{pmatrix} \right|.$$

We will not consider case when $\langle \ell, \mathbf{w}_i \rangle = 0$.

Example 2.4.2. To compute the volume of the polytope $\mathcal{Q}$ using Equation (2.28) we choose the vector $\ell = (1, 1)$ so that $\langle \ell, \mathbf{w}_i \rangle \neq 0$ for $i = 1, 2$. The four supporting cones of $\mathcal{Q}$ shown in Figure 2.5 are simplicial so we do not need to triangulate them. Therefore,

$$\begin{aligned}
\operatorname{vol}(\mathcal{Q}) &= \int_{\mathcal{Q}} \langle \ell, x \rangle^0 dm \\
&= L(C_1(\mathcal{P}, (0,0))) + L(C_2(\mathcal{P}, (0,2))) + L(C_3(\mathcal{P}, (1,2))) + L(C_4(\mathcal{P}, (2,0))) \\
&= \frac{0!}{2!} \left| \det \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| \frac{0}{1} + \frac{0!}{2!} \left| \det \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \right| \frac{2^2}{(-1)} + \frac{0!}{2!} \left| \det \begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix} \right| \frac{3^2}{1} \\
&\qquad + \frac{0!}{2!} \left| \det \begin{pmatrix} -1 & -1 \\ 0 & 2 \end{pmatrix} \right| \frac{2^2}{(-1)}. \\
&= 3
\end{aligned}$$

As `LattE Integrale` can use either the triangular-determinant method or the cone decomposition method to compute the volume of a polytope it is important to discuss when one method is more efficient than another. A d-polytope is *simple* if every vertex in $V(\mathcal{P})$ is connected by an edge to exactly d other vertices and *simplicial* if every facet of $\mathcal{P}$ is a d-simplex. If $\mathcal{P}$ is a simple polytope then is it more efficient to use the cone decomposition method. If $\mathcal{P}$ is almost simplicial then it is more efficient to use the triangular-determinant method. In the

case of the Birkhoff polytope Table 2.2 leads us to believe that the triangular-determinant method is more efficient.

2.4.3 The Ehrhart polynomial

The following method describes Barvinok's algorithm [8] to compute the multivariate generating function of a d-polytope $\mathcal{P} \subset \mathbb{R}^d$ and, through a change of variables, the Ehrhart polynomial. This section will follow the work presented by De Loera et al. [23]. Recall the lattice point enumerator of $\mathcal{P} \subset \mathbb{R}^d$ is

$$\sigma_{\mathcal{P}}(\mathbf{z}) = \sum_{a \in \mathcal{P} \cap \mathbb{Z}^d} \mathbf{z}^{\mathbf{a}},$$

which computes the multivariate generating function of $\mathcal{P}$. One could stop here but this method is infeasible as the number of integer points in $\mathcal{P}$ can be extremely large. We instead call upon a theorem put forth by Brion [17] and independently Lawrence [35].

Theorem 2.4.2 (Brion [17], Lawrence [35])**.** *Let $\mathcal{P} \subset \mathbb{R}^d$ be a bounded rational polyhedron and let $V(\mathcal{P})$ be the vertex set of $\mathcal{P}$. Then*

$$\sigma_{\mathcal{P}}(\boldsymbol{z}) = \sum_{v \in V(\mathcal{P})} \sigma_{C(\mathcal{P},v)}(\boldsymbol{z}),$$

for $C(\mathcal{P}, v)$ the supporting cone of $\mathcal{P}$ at vertex v.

Again, we would like to work with simplicial cones, so if $C(\mathcal{P}, v)$ is not simplicial we triangulate the cone to form a finite set of simplicial d-cones. Therefore, just as with the cone decomposition method, the first step of Barvinok's algorithm is to decompose all supporting cones at vertex $v \in V(\mathcal{P})$ into simplicial supporting cones by triangulation. Stanley [46] gave the following well known equation for the lattice point enumerator of a simplicial cone $C \subset \mathbb{R}^d$ with generators $\mathbf{w}_1, ..., \mathbf{w}_d$,

$$\sigma_C(\mathbf{z}) = \sum_{\beta \in \mathcal{C} \cap \mathbb{Z}^d} \mathbf{z}^{\beta} = \left(\sum_{\tau \in \pi_{\mathcal{C}} \cap \mathbb{Z}^d} \mathbf{z}^{\tau} \right) \prod_{i=1}^{d} \frac{1}{1 - \mathbf{z}^{\mathbf{w}_i}},$$

where π_C is again the fundamental parallelepiped of the cone C.

The number of lattice points in the fundamental parallelepiped $\pi_{C(\mathcal{P},v)}$ can be quite large, therefore Barvinok's algorithm then decomposes each supporting cone into a signed sum of unimodular cones. A simplicial cone is said to be *unimodular* if its generators $\mathbf{w}_1, ..., \mathbf{w}_d$ form a basis for $\mathbb{Z}^d$ [15]. This means that there is exactly one lattice point in $\pi_{C(\mathcal{P},v)}$ and hence the numerator of $\sigma_{C(\mathcal{P},v)}(\mathbf{z})$ is a monomial.

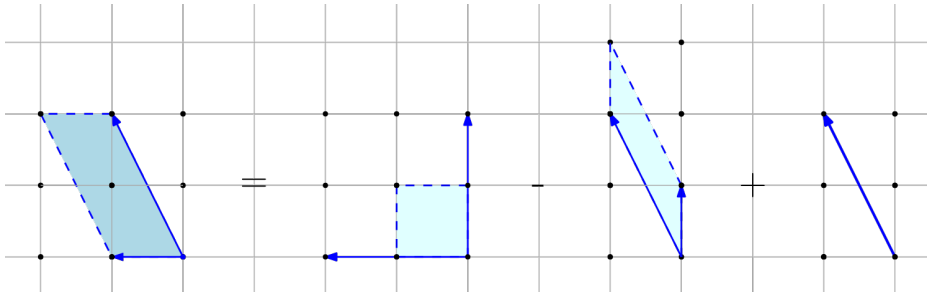


Figure 2.6: Decomposition of vertex 4 of polytope $\mathcal{Q}$ into unimodular cones.

Example 2.4.3. We see in Figure 2.5 that the supporting cone of $C_4(\mathcal{Q}, (2,0))$ is not unimodular as both the points (1,1) and (2,0) are contained in its corresponding parallelepiped. This simplicial cone can be decomposed as follows,

$$\text{cone}\left\{\begin{pmatrix}-1\\0\end{pmatrix},\begin{pmatrix}-1\\2\end{pmatrix}\right\} = \text{cone}\left\{\begin{pmatrix}-1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix}\right\} \ominus \text{cone}\left\{\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}-1\\2\end{pmatrix}\right\} \oplus \text{cone}\left\{\begin{pmatrix}-1\\2\end{pmatrix}\right\}.$$

This decomposition can be seen in Figure 2.6.

We notice a pressing issue in Example 2.4.3, the vector $(-1, 2)$ was subtracted and thus was re-added. One needs to keep track of these lower-dimensional cones to apply the principle of inclusion and exclusion in the decomposition of $C(\mathcal{P}, v)$ into a signed sum of unimodular cones which can be computationally infeasible. `Latte` applies *Brion's polarization trick* to remedy this issue, the details of this trick can be found in [23, 9]. Therefore the generating function for each supporting cone is of the form

$$\sigma_{C(\mathcal{P},v)} = \sum_{k\in K} E[k] \frac{\mathbf{z}^v}{\prod_{i=1}^{d} 1 - \mathbf{z}^{\mathbf{w}_{k,i}}} \tag{2.29}$$

where K is the set of unimodular cones decomposed from the supporting cone, $w_{k,i}$ the generators for each unimodular cone k, and $E[k] = \{-1, 1\}$.

We convert the multivariate generating function $\sigma_{\mathcal{P}}(\mathbf{z})$ to the Ehrhart polynomial, $L_{\mathcal{P}}(t)$, by first multiplying each vertex v in the exponent of the numerator of Equation (2.29) by a factor of $t \in \mathbb{Z}$ to represent the dilation of the polytope. Then, for each term in Equation (2.29) apply the following exponential change of variables

$$\mathbf{z} = e^{\varphi \mathbf{c}}$$

for $\mathbf{c} = (c_1, c_2, ..., c_d)$ such that for each supporting cone in $\mathcal{P}$ $\langle \mathbf{c}, \mathbf{w}_i \rangle \neq 0$ for $i = 1, ..., d$ [9]. Each term in $\sigma_{\mathcal{P}}(t, \varphi)$ is then written in terms of its power series around $\varphi = 0$ and the constant terms with respect to φ are summed to give the Ehrhart polynomial.

n	# of vertices	# of unimodular cones at a vertex cone	Triangular-determinant method	Cone decomposition method	Barvinoks algorithm
3	6	3	<0.01	<0.01	0.01
4	24	16	0.02	0.18	0.19
5	120	125	16198.33 $-K$	22249 $-K$	24.76
6	720	1296			17302.30

Table 2.2: Computation time in seconds for the relative volume of $\mathcal{B}_n$ for $n \leq 6$ using `LattE`. We use $-K$ to denote when `LattE` killed the computation.

Example 2.4.4. Applying Barvinok's algorithm to the polytope $\mathcal{Q}$ gives the following Ehrhart polynomial,

$$L_{\mathcal{Q}}(t) = \text{const}_{\varphi}(\sigma_{\mathcal{Q}}(t, \varphi)) = 3t^2 + 3t + 1.$$

Again, we get a volume of 3.

2.4.4 Computation times of `LattE`

We present the computation time to compute the relative volume for the Birkhoff polytope up to $n \leq 6$ in Table 2.2. We have also included the number of vertices and total unimodular cones at each vertex. For $n \geq 5$ `LattE Integrale` is unable to compute the relative volume via triangular-decomposition or the cone decomposition method, we note in the table by $-K$ how long the software ran before killing the computation. Therefore, based off of this small sample set, computing the Ehrhart polynomial by the use of Barvinok's algorithm seems to be the most efficient method to compute the relative volume for larger n.

Chapter 3

New methods to compute the relative volume of $\mathcal{B}_n$

So far, we have explored four methods to calculate the relative volume of the Birkhoff polytope for a specified n, given by

$$\mathcal{B}_n = \{x \in \mathbb{R}^{n^2}_{\geq 0} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

for the matrix $\mathbf{A} \in \mathbb{Z}^{2n \times n^2}$ and vector $\mathbf{b} \in \mathbb{Z}^{2n}$ given in Equation (1.7). How can we make this more efficient so as to succeed for values of n greater than 10? This section will describe two new methods to obtain the leading term of the Ehrhart polynomial, and hence the relative volume of $\mathcal{B}_n$. Pemantle and Wilsons book, "Analytic Combinatorics in Several Variables" [44], among others [30, 39, 1] summarizes comprehensive machinery to compute the coefficient asymptotics of multivariate generating functions. This is commonly known as the analytic combinatorics in several variables (ACSV) methodology. Section 3.1 presents the first method which asymptotically estimate the leading term of the Ehrhart polynomial for $\mathcal{B}_n$. We provide a detailed description of this method, implementations to speed up computation time, and our results. The second method, presented in Section 3.2, uses complex analytic techniques and `LattE` to calculate the leading term of the Ehrhart polynomial exactly. Though this method is not faster than the first it demonstrates an efficient use of Barvinok's algorithm.

3.1 Analytic Combinatorics in Several Variables

In Section 2.3 we explored a method for computing polytope volume that involved complex analytic techniques. We recall that the largest allocation of time was spent on integral evaluation. It turns out we can use this integral, but rather than evaluate it, we can estimate it. Since we can control the error bound there is an efficient method to compute the asymptotic

evaluation of the Ehrhart polynomial that agrees on the first term. Hence, it computes the relative volume of $\mathcal{B}_n$ exactly. We use an estimate for integrals of the Fourier-Laplace type; that is we find a multivariate function so that

$$\frac{1}{(2\pi i)^d}\int_T \mathbf{z}^{-t\mathbf{b}-1}F(\mathbf{z})dz \sim \Phi(t). \tag{3.1}$$

In this expression, $F(\mathbf{z})$ is rational, T is a d-dimensional torus in $\mathbb{C}^d$, and $\Phi(t)$ is elementary. In the ACSV methodology, this is the case of multiple point asymptotics for complete intersections. The first part of Section 3.1 will describe a change of variables to take it to this case. We then provide a literature review of definitions, theorems, and algorithms from multiple point asymptotics that aid specifically in our application taken from [44]. We end with the implementations applied to speed up computation time and the results obtained using our automated program, given in Table 3.1.

3.1.1 Change of Variables

Recall the final corollary from Section 2.3.

Corollary 3.1.1. *For $0 < \epsilon_n < \cdots < \epsilon_1 < 1$ and $t \geq 0$,*

$$H_n(t) = \frac{1}{(2\pi i)^n}\int_{|z_1|=\epsilon_1}\cdots\int_{|z_n|=\epsilon_n}(z_1...z_n)^{-t-1} \sum_{a_1+...+a_n=n}^{*}\binom{n}{a_1,...,a_n}\prod_{k=1}^{n}\left(\frac{z_k^{t+n-1}}{\prod_{j\neq k}(z_k-z_j)}\right)^{a_k} d\boldsymbol{z}. \tag{3.2}$$

where $\sum^*$ denotes the sum over those n-tuples of non-negative integers satisfying $a_1 + ... + a_n = n$ and $a_1 + ... + a_n > r$ for $1 \leq r < n$.

For the most straightforward application of the ACSV method we require that the function $F(\mathbf{z}) = \frac{G}{H}$ be a generating function in which H is a product of polynomials of the form $1-\mathbf{z}^{\mathbf{w}}$ for $\mathbf{w}, \mathbf{z} \in \mathbb{R}^d$, and G is analytic. In order to rewrite Equation (3.2) in this form the change of variables given in Algorithm 1 is applied.

Example 3.1.1. In Example 2.3.2 the following integral was computed:

$$H_3(t) = \frac{1}{(2\pi i)^3}\iiint (z_1z_2z_3)^{-t-1}\left(\frac{z_1^{3t+6}}{(z_1-z_2)^3(z_1-z_3)^3} + \frac{-3z_1^{2t+4}z_2^{t+2}}{(z_1-z_2)^3(z_1-z_3)^2(z_2-z_3)}\right)d\mathbf{z}.$$

We apply the change of variables given in Algorithm 1.

Algorithm 1: Change of variables

Input: Integral of the form given in Equation (3.2).

Output: Integral of the form

$$\frac{1}{(2\pi i)^{n-1}} \int_T \mathbf{x}^{-t\mathbf{b}-1} \frac{G(\mathbf{x})}{\prod_{\substack{\mathbf{w}\in\mathbf{W}\\ i=1}}^{n} (1-\mathbf{x}^{\mathbf{w}})^{m_i}} d\mathbf{x} \tag{3.3}$$

for $\mathbf{W}$, a set of $(n-1)$-dimensional vectors and $G(\mathbf{x})$ an analytic function.

1 Take the change of variables $z_1 := z_1$ and $z_r := z_1 x_1 x_2 ... x_{r-1}$ for $r \in \{2, .., n\}$ and calculate the determinate of the Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial z_1}{\partial z_1} & \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_{n-1}} \\ \frac{\partial z_2}{\partial z_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_2}{\partial x_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial z_1} & \frac{\partial z_n}{\partial x_1} & \cdots & \frac{\partial z_n}{\partial x_{n-1}} \end{bmatrix}.$$

2 Substitute the change of variables into Equation (3.2) and multiply by the absolute value of the determinant of J.

3 Simplify the denominator of the integrand to obtain the form

$$\prod_{\substack{\mathbf{w}\in\mathbf{W}\\ i=1}}^{n} (1-\mathbf{x}^{\mathbf{w}})^{m_i}$$

for $\mathbf{W}$, a set of $(n-1)$-dimensional vectors.

4 Using the residue theorem, take the integral with respect to z_1.

Figure 3.1: Algorithm for a change of variables

1) We set $z_1 := z_1$, $z_2 := z_1x_1$, $z_3 := z_1x_1x_2$ and compute the the Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial z_1}{\partial z_1} & \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} \\ \frac{\partial z_2}{\partial z_1} & \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} \\ \frac{\partial z_3}{\partial z_1} & \frac{\partial z_3}{\partial x_1} & \frac{\partial z_3}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ x_1 & z_1 & 0 \\ z_1x_2 & z_1x_2 & z_1x_1 \end{bmatrix}.$$

Therefore we have $\det(J) = z_1^2x_1$.

2) We then substitute the change of variables into $H_3(t)$ and multiply by $|\det(J)|$:

$$H_3(t) = \frac{1}{(2\pi i)^3} \iiint (z_1^3x_1^2x_2)^{-t-1} \left(\frac{z_1^{3t+6}}{(z_1 - z_1x_1)^3(z_1 - z_1x_1x_2)^3} + \frac{-3z_1^{2t+4}(z_1x_1)^{t+2}}{(z_1 - z_1x_1)^3(z_1 - z_1x_1x_2)^2(z_1x_1 - z_1x_1x_2)} \right) |z_1^2x_1| \, dz_1dx_1dx_2.$$

We can assume that $z_1, x_1 \geq 0$ so we are able to drop the absolute value sign.

3) We simplify the denominator to obtain a product of the form

$$\prod_{\substack{\mathbf{w} \in \mathbf{W} \\ i=1}}^{3} (1 - \mathbf{x}^{\mathbf{w}})^{m_i}$$

for $\mathbf{W}$ a set of 2-dimensional vectors:

$$H_3(t) = \frac{1}{(2\pi i)^3} \iiint (x_1^2x_2)^{-t-1} \left(\frac{z_1^{-1}x_1}{(1 - x_1)^3(1 - x_1x_2)^3} + \frac{-3z_1^{-1}x_1^{t+1}x_1}{(1 - x_1)^3(1 - x_1x_2)^2(1 - x_2)} \right) dz_1dx_1dx_2$$

4) Lastly, using the residue theorem, we take the integral with respect to z_1 and simplify:

$$\begin{aligned} H_3(t) &= \frac{1}{(2\pi i)^3} \iint (x_1^2 x_2)^{-t-1} \left(\frac{x_1}{(1-x_1)^3(1-x_1x_2)^3} \right. \\ &\qquad \left. + \frac{-3x_1^{t+1}x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} \right) \left(\int \frac{1}{z_1} dz_1 \right) dx_1 dx_2 \\ &= \frac{1}{(2\pi i)^3} \iint (x_1^2 x_2)^{-t-1} \left(\frac{x_1}{(1-x_1)^3(1-x_1x_2)^3} \right. \\ &\qquad \left. + \frac{-3x_1^{t+1}x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} \right) (2\pi i)\, dx_1 dx_2 \\ &= \frac{1}{(2\pi i)^2} \iint \left(x_1^{-2t-1} x_2^{-t-1} \frac{1}{(1-x_1)^3(1-x_1x_2)^3} \right) d\mathbf{x} \\ &\qquad - 3\frac{1}{(2\pi i)^2} \iint \left((x_1x_2)^{-t-1} \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} \right) d\mathbf{x}. \end{aligned} \tag{3.4}$$

Given an integral of the form presented in Equation (3.3) we can apply the ACSV method in a straightforward manner. Nevertheless, the size of intermediary expressions will grow exponentially limiting the maximum n for which the relative volume of $\mathcal{B}_n$ can be computed using this method.

3.1.2 Multiple Point Asymptotics for Complete Intersection

The estimate in Equation (3.1) is made precise in Theorem 10.3.3 of [44]. However, to verify the hypothesis we require some additional notation and background therefore this section walks through necessary results from Chapter 10 of [44] in the context of our problem. Directly following the statement of the theorem, we will define what it means for divisors to intersect transversely at a point $\mathbf{p}$ and provide an Algorithm (Algorithm 2) to ensure this transverse intersection. We end this section by defining the Lognormal matrix Γ_Ψ. All calculations are done in the local ring of germs of analytic functions at a point $\mathbf{p}$ which we denote $\mathbf{R_p}$. Let $\mathbf{m}$ be a positive, integral k-dimensional vector where

$$\mathbf{(m\text{-}1)!} = \prod_{j=1}^{k} (m_j - 1)!.$$

We now state the relevant theorem.

Theorem 3.1.1 ([44, Theorem 10.3.3]). *Let* $F = \frac{G}{H} = \frac{G}{\prod_{j=1}^{k} H_j^{m_j}}$ *in* $\boldsymbol{R_p}$ *with each* H_j *squarefree and all divisors intersecting transversely at* $\boldsymbol{p}$*. Suppose that* G *is holomorphic in a neighborhood of* $\boldsymbol{p}$ *and* $G(\boldsymbol{p}) \neq 0$*. Then*

$$\frac{1}{(2\pi i)^d} \int_T \boldsymbol{x}^{-t\boldsymbol{b}-1} F(\boldsymbol{x}) dz \sim \Phi(t) \tag{3.5}$$

with

$$\Phi(t) := \frac{1}{\boldsymbol{(m\text{-}1)!}} \frac{\boldsymbol{p}^{-t\boldsymbol{b}} G(\boldsymbol{x})}{|(\det \Gamma_\Psi)|} ((t\boldsymbol{b})\Gamma_\Psi^{-1})^{\boldsymbol{m\text{-}1}}. \tag{3.6}$$

The proof of Theorem 3.1.1 is out of the scope of this paper due to its reliance on advanced algebraic geometry. In ACSV, the simplest case reduces to the analysis of singular points of rational functions with a single irreducible polynomial factor in the denominator. In the case of $\mathcal{B}_n$ for $n \geq 3$, there are multiple polynomials of the form $(1 - \mathbf{x}^{\mathbf{w}})$, and the singular point $(1, 1, \cdots, 1)$ is a root of each of them. This is an example of multiple point asymptotics. The most straightforward case of multiple point asymptotics is when we have a complete intersection. Rather than define this, which would require a detour into algebraic geometry, we will use the results of the following proposition which identifies this case. Let $\mathcal{V}(H)$ denote the *variety* where H vanishes, and $\mathcal{V}_j = \mathcal{V}(H_j)$ denote the variety where H_j vanishes.

Proposition 3.1.1 ([44, Proposition 10.1.9]). *The point* $\boldsymbol{p} \in \mathcal{V}(H)$ *is a multiple point if and only if there is a factorization*

$$H = \prod_{j=1}^{k} H_j^{m_j} \tag{3.7}$$

in $\boldsymbol{R_p}$ *for each* H_j *irreducible with* $\nabla H_j(\boldsymbol{p}) \neq 0$ *and* $H_j(\boldsymbol{p}) = 0$*. The point* $\boldsymbol{p}$ *is a transverse multiple point of order* k *if and only if the gradient vectors* $\{\nabla H_j(\boldsymbol{p}) : 1 \leq j \leq k\}$ *are linearly independent.*

Let d be the dimension of the vector space containing the gradient vectors $\{\nabla H_j(\mathbf{p}) : 1 \leq j \leq k\}$. We then say that all divisors of $F(\mathbf{x})$ intersect transversely at $\mathbf{p}$ if $\mathbf{p}$ is a transverse multiple point of order d. If $\mathbf{p}$ is a transverse multiple point of order $k < d$ then one says that the divisors of $F(\mathbf{x})$ intersect transversely on a stratum, $S := \cap_{j=1}^{k} \mathcal{V}_j(H_j)$, containing the point $\mathbf{p}$. Figure 3.2 depicts on the left a graph with a transverse multiple point at (1,1) of order 2 in $\mathbb{R}^2$ - a complete intersection. The graph on the right shows a multiple point at (1,1) that is not transverse because the perpendicular lines to the gradient vectors of the three curves at the point (1,1), shown in red, are not linearly independent. This is a consequence of 3 lines in 2 dimensions. When $k = 1$, the divisor will, by default, intersect transversely at $\mathbf{p}$, this is the case for $\mathcal{B}_2$.

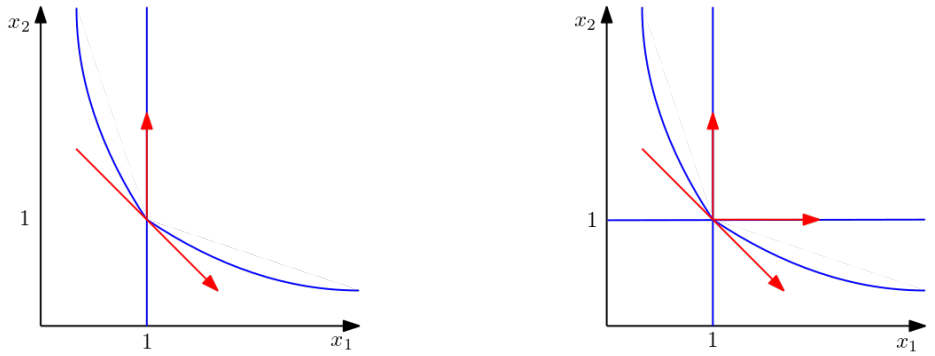


Figure 3.2: The blue curves shown on the left intersect at the transverse multiple point (1,1) of order 2, whereas the blue curves shown on the right do not.

We say that H is *square free* when $m_j = 1$ for all j in the decomposition of Equation (3.7) with all H_j's distinct.

Example 3.1.2. We treat the two terms of $H_3(t)$ separately,

$$H_3(t) = \frac{1}{(2\pi i)^2}\int\left(x_1^{-2t-1}x_2^{-t-1}\frac{1}{(1-x_1)^3(1-x_1x_2)^3}\right)d\mathbf{x} \\ -3\frac{1}{(2\pi i)^2}\int\left((x_1x_2)^{-t-1}\frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)}\right)d\mathbf{x}, \quad (3.8)$$

and let $F_1 := \frac{1}{(1-x_1)^3(1-x_1x_2)^3}$, so $H_1 = (1-x_1)$ and $H_2 = (1-x_1x_2)$. The point $\mathbf{p}$=(1,1) is the only point satisfying $H_1(\mathbf{p}) = H_2(\mathbf{p}) = 0$ and

$$\{\nabla H_j(1,1) : 1 \le j \le 2\} = \begin{bmatrix}\nabla H_1(1,1)\\ \nabla H_2(1,1)\end{bmatrix} = \begin{bmatrix}\frac{\partial H_1(1,1)}{\partial x_1} & \frac{\partial H_1(1,1)}{\partial x_2}\\ \frac{\partial H_2(1,1)}{\partial x_1} & \frac{\partial H_2(1,1)}{\partial x_2}\end{bmatrix} = \begin{bmatrix}-1 & 0\\ -1 & -1\end{bmatrix}.$$

Not only is the point (1,1) a multiple point but the gradient vectors are linearly independent, making (1,1) a transverse multiple point of order 2 in $\mathbb{R}^2$. The graph on the left of Figure 3.2 shows the varieties for each H_j in blue and in red are the perpendicular lines to the gradient vectors, ∇H_j, at (1,1) which span the 2-dimensional space.
Let $F_2 := \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)}$, so $H_1 = (1-x_1)$, $H_2 = (1-x_1x_2)$, and $H_3 = (1-x_2)$. Again, $\mathbf{p}$=(1,1) is the only point that satisfies $H_1(\mathbf{p}) = H_2(\mathbf{p}) = H_2(\mathbf{p}) = 0$ and

$$\{\nabla H_j(1,1) : 1 \le j \le 3\} = \begin{bmatrix}\nabla H_1(1,1)\\ \nabla H_2(1,1)\\ \nabla H_3(1,1)\end{bmatrix} = \begin{bmatrix}\frac{\partial H_1(1,1)}{\partial x_1} & \frac{\partial H_1(1,1)}{\partial x_2}\\ \frac{\partial H_2(1,1)}{\partial x_1} & \frac{\partial H_2(1,1)}{\partial x_2}\\ \frac{\partial H_3(1,1)}{\partial x_1} & \frac{\partial H_3(1,1)}{\partial x_2}\end{bmatrix} = \begin{bmatrix}-1 & 0\\ -1 & -1\\ 0 & -1\end{bmatrix}$$

Therefore (1,1) is a multiple point of F_2 but the gradient vectors are not linearly independent thus $\mathbf{p}$=(1,1) is not a transverse point. The graph on the right of Figure 3.2 shows the varieties in blue for each H_j and in red are the perpendicular lines to the gradient vectors,

∇H_j, at (1,1). It is easy to see that the gradient vectors do not span the 2-dimensional space, but instead form a linearly dependent set.

The asymptotic evaluation provided by Theorem 3.1.1 requires $\mathbf{p}$ be a transverse multiple point of order d. We describe a procedure (Algorithm 2) to reduce $F(\mathbf{x})$ down to a sum of *summands*, $F(\mathbf{z}) = \sum_j f_j$, for which $\mathbf{p}$ is a transverse multiple point of order d. To begin, let $\mathrm{col}_j(\mathbf{D})$ denote the j'th column of a matrix $\mathbf{D}$, and define

$$F(\mathbf{x}) = \frac{G}{\prod_{j=1}^{k} H_j^{m_j}} = \frac{G}{\prod_{j=1}^{k} (1 - \mathbf{x}^{\mathrm{col}_j(\mathbf{D})})^{m_j}}. \tag{3.9}$$

Hence, each H_j in $F(\mathbf{x})$ corresponds to a column vector of the matrix $\mathbf{D}$.

Lemma 3.1.1. *Given the factorization*

$$H = \prod_{j=1}^{k} H_j^{m_j} = \prod_{\substack{\boldsymbol{w} \in \boldsymbol{W} \\ j=1}}^{k} (1 - \boldsymbol{x}^{\boldsymbol{w}})^{m_j}$$

*for $\boldsymbol{W}$ a set of d-dimensional vectors, the point $\boldsymbol{p}$=**1** is a transverse multiple point of order k if and only if the column vectors of the matrix $\boldsymbol{D}$ given by Equation* (3.9) *are linearly independent.*

Proof. Given $H_j(\mathbf{x}) = (1 - x_1^{w_1} x_2^{w_2} ... x_d^{w_d})$ we calculate the gradient vector

$$\nabla H_j(\mathbf{x}) = (-w_1 x_1^{w_1 - 1} x_2^{w_2} \cdots x_d^{w_d}, -w_2 x_1^{w_1} x_2^{w_2 - 1} \cdots x_d^{w_d}, ..., -w_d x_1^{w_1} x_2^{w_2} \cdots x_d^{w_d - 1}).$$

Hence, evaluated at $\mathbf{1}$:

$$\nabla H_j(\mathbf{1}) = (-w_1, -w_2, ..., -w_d) = -(w_1, w_2, ..., w_d) = -col_j(\mathbf{D}). \tag{3.10}$$

By Proposition 3.1.1, if $\mathbf{1}$ is a transverse multiple point of order k then the gradient vectors $\nabla H_j(\mathbf{1})$ for $1 \leq j \leq k$ are linearly independent. Thus, by Equation (3.10) the column vectors of the matrix $\mathbf{D}$ are linearly independent. By the same reasoning, if the column vectors of matrix $\mathbf{D}$ are linearly independent then by Equation (3.10) the gradient vectors $\nabla H_j(\mathbf{1})$ for $1 \leq j \leq k$ are linearly independent and $\mathbf{1}$ is a transverse multiple point of order k. □

Given Lemma 3.1.1, our objective is now to decompose the set of linearly dependent column vectors of $\mathbf{D}$ into maximally independent subsets because of the central importance of linear independence. We can use matroid theory to facilitate this decomposition. A *matroid* is an ordered pair consisting of a ground set, $\mathbf{E}$, and the set of independent subsets of $\mathbf{E}$, denoted by $\mathbf{I}$ [43]. They satisfy the following properties:

1) $\emptyset \in \mathbf{I}$.

2) Every subset of an independent set is independent.

The ground set, $\mathbf{E}$, are the column vectors of the matrix $\mathbf{D}$ labeled using the column indices. A **circuit**, denoted $\mathbf{C}$, is a minimally dependent subset of $\mathbf{E}$ and a **base** is a maximally independent subset of $\mathbf{E}$.

Definition. We let the *broken circuits*, $\mathbf{BC}$, be the set formed by removing the largest element, i, from each set in $\mathbf{C}$, $\mathbf{BC} = \mathbf{C}\backslash i$.

Example 3.1.3. Let F be the following multivariate generating function and $\mathbf{D}$ be the corresponding matrix as defined in Equation (3.9):

$$F = \frac{1}{(1-x_1)(1-x_2^2)(1-x_1x_2^2)(1-x_1x_2^2x_3^3)^2(1-x_3)^3} \implies \mathbf{D} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

Let $\mathbf{E}$ be the columns of $\mathbf{D}$ numbered using the column indices, $\mathbf{E}$=$\{1,2,3,4,5\}$. The circuits of $\mathbf{E}$ are $\mathbf{C}$=$\{1,2,3\},\{1,2,4,5\},\{3,4,5\}$ and $\mathbf{I} = \{1,2\},\{1,3\},\{1,4\},\{1,5\},\{2,3\},\{2,4\},\{2,5\},\{3,4\},\{3,5\},\{4,5\},\{1,2,4\},\{1,2,5\},\{2,3,4\},\{2,3,5\}$. For the purpose of this thesis we will ignore the empty set. By removing the last element from each set in $\mathbf{C}$ we get $\mathbf{BC} = \{1,2\},\{1,2,4\},\{3,4\}$. Hence, ($\mathbf{E}$, $\mathbf{I}$) forms a matroid, denoted by $\mathbf{M}[\mathbf{D}]$.

Example 3.1.4. Consider F_2 from $H_3(t)$,

$$F_2 = \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} = \frac{1}{H_1^3H_2^2H_3} \implies \mathbf{D}_2 = \begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array}.$$

The only circuit is $\mathbf{C}$=$\{1,2,3\}$ and the broken circuit is $\mathbf{BC}$=$\{1,2\}$.

Definition. The *support* of a summand is the set of j's in the order they appear from $H = \prod_{j=1}^{k} H_j^{m_j}$ for which $m_j > 0$.

Example 3.1.5. The summand $f = \frac{1}{H_1^2H_3H_6}$ has support $\{1,3,6\}$.

For the largest element i in each set of $\mathbf{C}$ one is guaranteed to find a set of p_j's such that $H_i = \sum_{j\in\mathbf{BC}} p_jH_j$, this is called the *Nullstellensatz certificate*.

Lemma 3.1.2 ([44, Lemma 10.2.9])**.** *There is a collection $\{p_i : i \in \boldsymbol{C}\}$ of invertible elements of $\boldsymbol{R_p}$ such that $\sum_{i\in\boldsymbol{C}} p_iH_i = 0$.*

Sketch of Proof. We first prove the existence of such elements. This can be done using the fact that H_i, for $i \in \mathbf{C}$, is in the radical of the ideal generated by $\{H_j : j \neq i\}$ in $\mathbf{R_p}$, and that this ideal is also radical [44]. Then we prove invertibility by showing that $p_j(\mathbf{p}) \neq 0$ for $j \neq i$ and $\mathbf{p} \in S = \cup_{i \in \mathbf{C}} \mathcal{V}_i(H_i)$. □

The Nullstellensatz certificate provides the basis of the algebraic reduction, as one uses this certificate to rewrite $F(\mathbf{x})$ until the support of each summand, f, no longer contains a broken circuit. Define $\mathcal{I}(f)$ to be the ideal generated by $\langle H_1^{m_1}, ..., H_n^{m_n} \rangle$ of $\mathbf{R_p}$ for each denominator H of f.

Algorithm 2: Algebraic Reductions [44]

Input: A function $F(\mathbf{x}) = \frac{G}{\prod_{j=1}^{k} H_j^{m_j}}$ and a point $\mathbf{p}$ being a multiple point in the stratum S of $\mathcal{V}$.

Output: An expression of $F(\mathbf{x})$ as the sum of summands with nonvanishing (at $\mathbf{p}$) numerators and $\mathbf{p}$ a transverse multiple point.

1 If any summand in the current sum has a support containing a broken circuit from **BC**, use the Nullstellensatz certificate to rewrite the summand. Repeat until no longer possible.

2 Collect summands with the same H's.

3 For each summand f, check whether the numerator is in $\mathcal{I}$(f). If there are such terms, choose one among them whose denominator has maximum degree, and replace it by a sum of terms with smaller support. Repeat until no longer possible.

4 For each summand f, check whether its numerator vanishes on the stratum defined by the support of f, and if so, write it as the sum of terms with the same support but small degrees in the denominator. Repeat until no longer possible.

Figure 3.3: Algorithm for algebraic reduction

As each summand in the reduction of a function $F(\mathbf{x})$ can only be reduced a finite number of times the algorithm is guaranteed to terminate. The Nullstellensatz certificate ensures that the support of each summand no longer contains a circuit. Hence, the column vectors of the matrix $\mathbf{D}$ formed from the divisors of the summand form a basis. By Lemma 3.1.1, the point $\mathbf{p}$ is now a transverse multiple point of order d. Step 4 ensures that each summand has nonvanishing (at $\mathbf{p}$) numerators.

Remark. The use of Lemma 3.1.1 only proves correctness of Algorithm 2 in our polytope context. A general proof can be found in [44].

Example 3.1.4 (Continued)**.** We begin by computing the Nullstellensatz certificate of the only circuit for the term F_2, $\mathbf{C}$=$\{1,2,3\}$:

$$H_3 = p_1 H_1 + p_2 H_2 = -x_2 H_1 + H_2.$$

We then apply Step 1 of Algorithm 2 until the support of each summand no longer contains the broken circuit $\{1,2\}$. For the sake of simplicity we will ignore the x_1 term from the numerator for now, so we have

$$\begin{aligned}
\frac{1}{H_1^3 H_2^2 H_3} &= \frac{p_1}{H_1^2 H_2^2 H_3^2} + \frac{p_2}{H_1^3 H_2 H_3^2} \\
&= \frac{p_1^2}{H_1 H_2^2 H_3^3} + \frac{p_1 p_2}{H_1^2 H_2 H_3^3} + \frac{p_1 p_2}{H_1^2 H_2 H_3^3} + \frac{p_2^2}{H_1^3 H_3^3} \\
&= \frac{p_1^3}{H_2^2 H_3^4} + \frac{p_1^2 p_2}{H_1 H_2 H_3^4} + \frac{p_1^2 p_2}{H_1 H_2 H_3^4} + \frac{p_1 p_2^2}{H_1^2 H_3^4} + \frac{p_1^2 p_2}{H_1 H_2 H_3^4} + \frac{p_1 p_2^2}{H_1^2 H_3^4} + \frac{p_2^2}{H_1^3 H_3^3} \\
&= \frac{p_1^3}{H_2^2 H_3^4} + \frac{p_1^3 p_2}{H_2 H_3^5} + \frac{p_1^2 p_2^2}{H_1 H_3^5} + \frac{p_1^3 p_2}{H_2 H_3^5} + \frac{p_1^2 p_2^2}{H_1 H_3^5} + \frac{p_1 p_2^2}{H_1^2 H_3^4} + \frac{p_1^3 p_2}{H_2 H_3^5} + \frac{p_1^2 p_2^2}{H_1 H_3^5} + \\
&\qquad \frac{p_1 p_2^2}{H_1^2 H_3^4} + \frac{p_2^2}{H_1^3 H_3^3}.
\end{aligned} \tag{3.11}$$

We collect terms with the same denominator leaving the following five summands, note that we multiply through by the numerator term x_1 before moving on to steps 3 and 4 of Algorithm 2,

$$\begin{aligned}
\frac{x_1}{H_1^3 H_2^2 H_3} &= \frac{x_1 p_1^3}{H_2^2 H_3^4} + \frac{3 x_1 p_1^3 p_2}{H_2 H_3^5} + \frac{3 x_1 p_1^2 p_2^2}{H_1 H_3^5} + \frac{2 x_1 p_1 p_2^2}{H_1^2 H_3^4} + \frac{x_1 p_2^2}{H_1^3 H_3^3} \\
&= \frac{-x_1 x_2^3}{H_2^2 H_3^4} + \frac{-3 x_1 x_2^3}{H_2 H_3^5} + \frac{3 x_1 x_2^2}{H_1 H_3^5} + \frac{-2 x_1 x_2}{H_1^2 H_3^4} + \frac{x_1}{H_1^3 H_3^3} \\
&= f_1 + f_2 + f_3 + f_4 + f_5.
\end{aligned} \tag{3.12}$$

Observe that the numberator of the summand $f_1 = \frac{-x_1 x_2^3}{H_2^2 H_3^4}$ is not in the ideal generated by $\langle H_2^2, H_3^4 \rangle$. The stratum defined by the support of f_1 is $\mathcal{V}_2 \cup \mathcal{V}_3$=(1,1) and $-x_1 x_2^3$ does not vanish here. We now check if $\mathbf{p}$=(1,1) is a transverse multiple point of f_1.
We have $H_2(\mathbf{p}) = H_3(\mathbf{p}) = 0$ and

$$\{\nabla H_j(1,1) : j \in \{2,3\}\} = \begin{bmatrix} \nabla H_2(1,1) \\ \nabla H_3(1,1) \end{bmatrix} = \begin{bmatrix} \frac{\partial H_2(1,1)}{\partial x_1} & \frac{\partial H_2(1,1)}{\partial x_2} \\ \frac{\partial H_3(1,1)}{\partial x_1} & \frac{\partial H_3(1,1)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 0 & -1 \end{bmatrix}.$$

The gradient vectors are linearly independent so $\mathbf{p}$=(1,1) is a transverse multiple point of order 2 in $\mathbb{R}^2$.

Let Γ_Ψ denote the augmented lognormal $d \times d$-matrix whose rows are the lognormal vectors $\nabla_{\log} H_j(p)$ for $1 \leq j \leq d$. Given our constraints and transverse multiple point $\mathbf{1}$, Γ_Ψ is defined as

$$F(\mathbf{x}) = \frac{G}{\prod_{j=1}^{k} H_j^{m_j}} = \frac{G}{\prod_{j=1}^{k} (1 - \mathbf{x}^{\mathrm{row}_j(\Gamma_\Psi)})^{m_j}}. \tag{3.13}$$

where $\mathrm{row}_j(\Gamma_\Psi)$ denotes the j'th row of the matrix Γ_Ψ.

Example 3.1.6. Consider

$$F(\mathbf{x}) = \frac{1}{(1 - x_1^2 x_2 x_3)^2 (1 - x_1 x_2^2 x_3^2)^2 (1 - x_1 x_2 x_3)^2}$$

with the multiple point (1,1,1). The corresponding augmented lognormal 3×3 matrix is

$$\Gamma_\Psi = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}.$$

We have now covered the necessary notations and definitions to understand Theorem 3.1.1. For the ACSV method there is a final condition to be satisfied regarding the vector $\mathbf{b}$ in

$$\frac{1}{(2\pi i)^d} \int_T \mathbf{x}^{-t\mathbf{b}-1} F(\mathbf{x}) d\mathbf{x}.$$

Theorem 3.1.1 is unable to distinguish whether or not the output is valid based on the vector $\mathbf{b}$. For example say we wished to find the volume of the 0-polytope

$$\mathcal{P} = \left\{ x, y \in \mathbb{R}_{\geq 0} : \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\},$$

we know that there is no possible vectors (x, y) that satisfy the given vector $\mathbf{b} = (1, 2)$ and hence no volume but Theorem 3.1.1 gives an asymptotic evaluation and volume of 1. Therefore we must use caution and ensure that the vector $\mathbf{b}$ is valid, meaning that is falls into the chamber complex given by $F(\mathbf{x})$. This concept is developed in the following section.

3.1.3 Computing the Chamber Complex

Recall the definition of a cone, $C = \{\lambda_1 \mathbf{w}_1 + ... + \lambda_d \mathbf{w}_d : \lambda_1, ..., \lambda_d \geq 0\}$. For any subset of the ground set $\mathbf{E}$, σ, we let the matrix $\mathbf{D}_\sigma$ be the submatrix of $M[\mathbf{D}]$ obtained by taking only the columns who indices are in σ. For a matrix $\mathbf{B}$, we define $pos(\mathbf{B})$ to be the cone generated by the columns of $\mathbf{B}$. We define the *chamber complex* of $\mathbf{D}$ as the common refinement of the set $\{pos(\mathbf{D}_\sigma) : \sigma \text{ a base of } M[\mathbf{D}]\}$, and a *chamber* of $\mathbf{D}$ to be a maximal element of the chamber complex of $\mathbf{D}$.

The chamber complex will be calculated for each term of $H_n(t)$ for $n \geq 3$ and the *chamber*

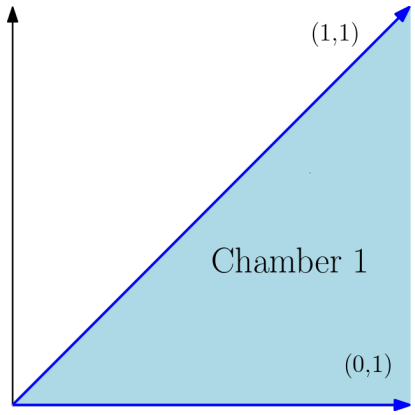


Figure 3.4: Chamber complex for F_1.

of importance will be the one that contains the vector $\mathbf{b}$. For a term of $H_n(t)$, if no such chamber exists in its chamber complex, the asymptotics of the term will not be calculated as they are not valid. For the summands of a term, their chamber complex must contain the chamber of importance from the parent term for Theorem 3.1.1 to be applied. This condition ensures we sum only the necessary and valid asymptotic evaluations. It is important to note that in the case of $\mathcal{B}_2$ we have a setting in which the output is valid based on the vector $\mathbf{b}$.

Example 3.1.7. Consider the first term for $H_3(t)$:

$$\frac{1}{(2\pi i)^2}\iint\left(x_1^{-2t-1}x_2^{-t-1}\frac{1}{(1-x_1)^3(1-x_1x_2)^3}\right)d\mathbf{x}.$$

The chamber complex corresponding to $F_1 = \frac{1}{(1-x_1)^3(1-x_1x_2)^3}$, shown in Figure 3.4, is composed strictly from the cone built by $\{pos(\mathbf{D}_\sigma) : \sigma \text{ a base of } \mathbf{D}_1\}$ where $\mathbf{D}_1 = \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$. The vector $\mathbf{b} = (2,1)$ falls into the only chamber of the chamber complex therefore we apply Theorem 3.1.1. Recall the transverse multiple point (1,1): $G = 1$ is holomorphic in the neighbourhood of (1,1) and $G(1,1) = 1 \neq 0$. The multiplicity vector is $\mathbf{m} = (3,3)$ and $\Gamma_\Psi(1,1) = \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$. Therefore,

$$\begin{aligned}\frac{1}{(2\pi i)^2}\iint\left(x_1^{-2t-1}x_2^{-t-1}\frac{1}{(1-x_1)^3(1-x_1x_2)^3}\right)d\mathbf{x} &\sim \frac{1}{2!2!}\frac{(1,1)^{(-2t,-t)}\cdot 1}{|(\det\Gamma_\Psi)|}\left([2t,t]\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}\right)^{(2,2)}\\ &= \frac{1}{4}\left([2t,t]\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}\right)^{(2,2)}\\ &= \frac{1}{4}[t,t]^{(2,2)}\\ &= \frac{1}{4}t^4\end{aligned}$$

Algorithm 3: Finding a chamber

Input: The vector $\mathbf{b}$ and a chamber complex.

Output: A chamber of the chamber complex.

1 Compute the normal vector, $\mathbf{n}$, in any direction to $\mathbf{b}$.

2 Divide the normal vector by a small factor, $\epsilon > 0$, to ensure that $\frac{1}{\epsilon}\mathbf{n} + \mathbf{b} \in \mathbf{c}$, where $\mathbf{c}$ is a chamber containing the face $\mathbf{b}$ falls onto.

3 Create a new vector that connects the base of the chamber with the end of the normal vector.

4 If this new vector falls into a chamber then terminate, else repeat the algorithm with the new vector.

Figure 3.5: Algorithm for finding a chamber in a chamber complex

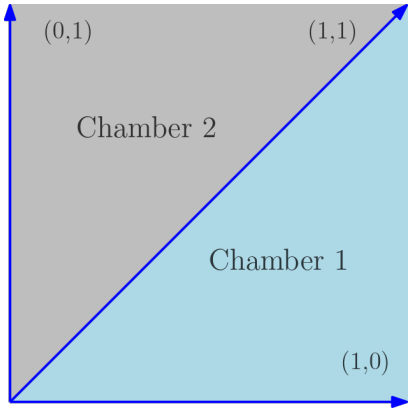


Figure 3.6: Chamber complex for F_2.

If $\mathbf{b}$ does not fall into a chamber but instead falls on a face of the chamber complex then Algorithm 3 is applied to compute the chamber of importance.

Example 3.1.8. Consider F_2 from $H_3(t)$:

$$-3\frac{1}{(2\pi i)^2}\iint\left((x_1x_2)^{-t-1}\frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)}\right)d\mathbf{x}.$$

The chamber complex corresponding to $F_2 = \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)}$, shown in Figure 3.6, is composed strictly from the cone built by $\{pos(\mathbf{D}_\sigma) : \sigma \text{ a base of } \mathbf{D}_2\}$ where $\mathbf{D}_2 = \begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}$. The vector $\mathbf{b} = (1,1)$ falls directly on a face of the chamber complex so we have to apply Algorithm 3 to calculate the chamber of importance.

We choose the normal vector (1, -1) and divide by $\frac{1}{2}$ giving the vector $(1.5, 0.5)$. This vector falls into chamber 1 shown in Figure 3.6 making chamber 1 the chamber of importance.

Recall the summands computed for F_2 given in Equation (3.12), we now check the chamber complex of each summand to ensure it contains the chamber of importance. For example,

the chamber complex corresponding to

$$f_1 = \frac{-x_1x_2^3}{H_2^2H_3^4} = \frac{-x_1x_2^3}{(1-x_1x_2)^2(1-x_2)^4}$$

does not contain chamber 1, therefore the asymptotic evaluation of f_1 is not pertinent. Upon further calculations, one can see that only the chamber complexes of f_3, f_4, and f_5 contain chamber 1.

Example 3.1.9. We now calculate the leading term of the Ehrhart polynomial for F_2 of $H_3(t)$:

$$-3\frac{1}{(2\pi i)^2}\int (x_1x_2)^{-t-1}F_2 d\mathbf{x} = -3\left(\frac{1}{(2\pi i)^2}\int (x_1x_2)^{-t-1}f_3 d\mathbf{x} + \frac{1}{(2\pi i)^2}\int (x_1x_2)^{-t-1}f_4 d\mathbf{x} + \frac{1}{(2\pi i)^2}\int (x_1x_2)^{-t-1}f_5 d\mathbf{x}\right).$$

The term containing the summand f_3 gives us the leading term $\frac{1}{8}t^4$, f_4 the leading term $\frac{-1}{3}t^4$, and f_5 gives $\frac{1}{4}t^4$ therefore,

$$-3\left(\frac{1}{8}t^4 - \frac{1}{3}t^4 + \frac{1}{4}t^4\right) = \frac{-1}{8}t^4.$$

We now have the asymptotic evaluation for both terms of $H_3(t)$, summing them will give the leading term of the Ehrhart polynomial,

$$H_3(t) = \frac{1}{4}t^4 - \frac{1}{8}t^4 = \frac{1}{8}t^4,$$

and hence the relative volume for $\mathcal{B}_3$,

$$\nu(\mathcal{B}_3) = \frac{1}{8}.$$

3.1.4 Implementations to Speed up Computation Time

Which steps of the ACSV method require the most time? It turns out that Algorithm 2 Algebraic Reductions becomes extremely inefficient for values of $n > 4$ as shown in Table 3.1. The number of summands of a given term in $H_n(t)$ grows exceedingly fast as n increases, and hence, so does the number of calculations. With this in mind, Stefan Trandafir [49] and I have implemented four techniques to speed up to Algorithm 2:

1) Presort the H_j's before Step 1 from lowest to highest multiplicities m_j.

2) Combine Steps 1 and 2 collecting terms with the same H's at each round.

3) At each round of Step 1 check if the summands chamber complex contains the necessary chamber, if it does not remove the summand.

4) Before running Steps 3 and 4 check that the summand vanishes at the multiple point.

Presorting the H_j's is one of the largest time savers. Before calculating the matroid $M[\mathbf{D}]$, the H_j's of a term should be organized based on their multiplicities, from lowest to highest. Then $M[\mathbf{D}]$ is formed with the alteration that the ground set is labeled based on the new ordering of the H_j's. The element of $\mathbf{C}$ corresponding to the column with the highest multiplicity is then removed to create the broken circuit and used for the Nullstellensatz certificate. With these changes Step 1 in Algorithm 2 will require significantly less calculations as the H_j's with the lowest multiplicities will now be the H_j's removed by the Nullstellensatz certificate. Combining steps 1 and 2 stops repeated rounds of step 1 for summands with the same denominators.

Example 3.1.10. Recall

$$F_2 = \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} = \frac{1}{H_1^3H_2^2H_3}$$

and presort the H_j's to create $\mathbf{D}_2$:

$$\frac{1}{H_3H_2^2H_1^3} \implies \mathbf{D}_2 = \begin{array}{c} \begin{matrix} 3 & 2 & 1 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}.$$

The only circuit of $\mathbf{D}_2$ is $\mathbf{C} = \{3, 2, 1\}$ with the broken circuit now being $\mathbf{BC}$=$\{3, 2\}$. The new Nullstellensatz certificate is slightly different than before,

$$H_1 = p_1H_3 + p_2H_2 = -x_1H_3 + H_2.$$

We run both Step 1 and 2 simultaneously:

$$\begin{aligned} \frac{x_1}{H_3H_2^2H_1^3} &= \frac{x_1p_1}{H_2^2H_1^4} + \frac{x_1p_2}{H_3H_2H_1^4} \\ &= \frac{x_1p_1}{H_2^2H_1^4} + \frac{x_1p_1}{H_2H_1^5} + \frac{x_1p_2}{H_3H_1^5}. \end{aligned} \tag{3.14}$$

The support of our summands no longer contain the broken circuit $\{3, 2\}$. It is hard not to notice the decrease in the number of calculations from Example 3.1.4.

Checking if a summands chamber complex contains the chamber of importance in each step can help save unnecessary computations. If the chamber complex of f does not contain the chamber of importance then there is no possibility that the summands computed from f will have a chamber complex containing the chamber of importance. This is because

with each round of Step 1 in Algorithm 2 the support either stays the same with different multiplicities, or depletes in size. Looking at Table 3.1 we see that this implementation does not improve efficiency for small n. This can be attributed to the time it takes to check that each chamber complex contains the chamber of importance. For larger n it is clear that this step improves efficiency as the time saved not applying step 1 to a summand outweighs the time taken to check if the chamber of importance is in the summands chamber complex.
As for the fourth implementation, if the summand does not vanish at the multiple point then the numerator is not in the ideal formed from the denominator of f. This saves time checking if the numerator of every summand is in the ideal.
Table 3.1 shows the computation time using the method of multiple point asymptotics with no implementations to speed up computation time (original). This is compared to the method with each of the implementations applied separately, with the final column showing the running time with all four implementations. These times were gathered by running a program we wrote in Maple [38] and Sage [48] that can be found in Appendix A. Note that these times do not include the computation time of Algorithm 1.

n	Original	Speed up #1	Speed up #2	Speed up #3	Speed up #4	All speed ups
3	0.02	0.03	0.04	0.04	0.02	0.01
4	0.70	0.51	0.68	1.98	0.25	0.19
5	17262.01	7101.97	8022.01	8739.65	13615.82	8.87
6						944.02

Table 3.1: Computation time in seconds for the relative volume of $\mathcal{B}_n$ for $n \leq 6$ using different implementations to speed up running time of ACSV method.

It can be seen from Table 3.1 that the four implementations vastly improve the computation time but were unable to compute $\mathcal{B}_n$ for $n \geq 7$ with the time and equipment at hand. Once $n \geq 7$, the computation time of the Reduction Algorithm 2 proves infeasible even with implementations to improve computation time.

3.2 Complex Analytic Techniques and `LattE`

We can also use the software `LattE` introduced in Section 2.4 to compute the Ehrhart polynomial of the Birkhoff polytope given by

$$\mathcal{B}_n = \{x \in \mathbb{R}^{n^2}_{\geq 0} : \mathbf{A}\mathbf{x} = \mathbf{b}\},$$

for the matrix $\mathbf{A} \in \mathbb{Z}^{2n \times n^2}$ and vector $\mathbf{b} \in \mathbb{Z}^{2n}$ given in Equation (1.7). Barvinok's algorithm accepts both the hyperplane and vertex description of a polytope. We will work with the hyperplane description.

We learned in Section 2.4 that using the Ehrhart polynomial seemed to be the fastest method to compute the relative volume of $\mathcal{B}_n$ when compared to other `LattE` commands but still had a high computation time. To try and speed up the computation we apply the the complex analytic techniques from Section 2.3 to reduce the integral down to the terms which contribute to the Ehrhart polynomial. These terms will then be put into the correct format and run through `LattE` where the leading coefficient of each Ehrhart polynomial will be summed to give the leading coefficient of $H_n(t)$, and hence the relative volume of $\mathcal{B}_n$. We start again from the final corollary from Section 2.3.

Corollary 3.2.1. *For $0 < \epsilon_n < \cdots < \epsilon_1 < 1$ and $t \geq 0$,*

$$H_n(t) = \frac{1}{(2\pi i)^n} \int_{|z_1|=\epsilon_1} \cdots \int_{|z_n|=\epsilon_n} (z_1 \ldots z_n)^{-t-1} \sum_{a_1+\ldots+a_n=n}^{*} \binom{n}{a_1, \ldots, a_n} \prod_{k=1}^{n} \left(\frac{z_k^{t+n-1}}{\prod_{j \neq k}(z_k - z_j)} \right)^{a_k} d\boldsymbol{z}. \quad (3.15)$$

where $\sum^*$ denotes the sum over those n-tuples of non-negative integers satisfying $a_1 + \ldots + a_n = n$ and $a_1 + \ldots + a_n > r$ for $1 \leq r < n$.

We apply the same change of variables given in Algorithm 1 of Subsection 3.1.1 to Equation (3.15).

Example 3.2.1.

$$H_3(t) = \frac{1}{(2\pi i)^2} \int \left(x_1^{-2t-1} x_2^{-t-1} \frac{1}{(1-x_1)^3(1-x_1x_2)^3} \right) d\mathbf{x} - 3\frac{1}{(2\pi i)^2} \int \left((x_1x_2)^{-t-1} \frac{x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} \right) d\mathbf{x} \quad (3.16)$$

The matrix $\mathbf{A}$ is formed from augmenting the vectors $\mathbf{w} \in \mathbf{W}$ as column vectors, each one with multiplicity m_i.

Example 3.2.1 (Continued)**.** Take the first term of $H_3(t)$,

$$\frac{1}{(2\pi i)^3} \int x_1^{-2t-1} x_2^{-t-1} \left(\frac{1}{(1-x_1)^3(1-x_1x_2)^3} \right) d\mathbf{x}.$$

We build the 2×6 matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \implies \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

and note that $\mathbf{b} = (2, 1)$. We run this polytope through `LattE` and receive the polynomial

$$1 + 3t + \frac{13}{4}t^2 + \frac{3}{2}t^3 + \frac{1}{4}t^4.$$

Take the second term of $H_3(t)$,

$$\frac{1}{(2\pi i)^3} \int (x_1 x_2)^{-t-1} \left(\frac{-3x_1}{(1-x_1)^3(1-x_1x_2)^2(1-x_2)} \right) d\mathbf{x}..$$

We build the 2×6 matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \implies \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

and note that $\mathbf{b} = (1, 1)$. We run this polytope through `LattE` and receive the polynomial

$$1 + \frac{25}{12}t + \frac{35}{24}t^2 + \frac{5}{12}t^3 + \frac{1}{24}t^4$$

We add the leading coefficient from each polynomial to give relative volume of $\mathcal{B}_3$,

$$\nu(\mathcal{B}_3) = \frac{1}{8}.$$

This vastly improves `LattE`'s computation time when $n \geq 5$ as shown in Table 3.2 where Quick `LattE` represents this more efficient method. The time recorded for quick `LattE` includes the computation time of Algorithm 1, the time to convert each integrand in Equation (3.3) to the hyperplane format necessary for `LattE`, and the time to sum up all of the leading coefficients.

n	`LattE`	Quick `LattE`
2	<0.01	-
3	0.01	0.10
4	0.19	0.41
5	24.76	13.37
6	17302.30	1413.38

Table 3.2: Computation time in seconds for relative volume of $\mathcal{B}_n$ for $n \leq 6$ using two different implementations of `LattE`

Chapter 4

Conclusion

We have now considered multiple methods to compute the relative volume of the n^{th} Birkhoff polytope. We compare the computation time of two published methods, complex analytic techniques and `Latte`, alongside our new methods titled ACSV and Quick `Latte`. The time for the ACSV method and quick `LattE` includes the computation time of Algorithm 1 Change of variables. The time recorded for quick `LattE` includes the time taken to convert each integrand to the correct format necessary for `LattE` and to sum up all of the leading coefficients.

n	Complex Analytic Techniques	`LattE`	ACSV	Quick `LattE`
2	<0.01	<0.01	-	-
3	<0.01	0.01	0.07	0.10
4	<0.01	0.19	0.30	0.41
5	<0.01	24.76	9.12	13.37
6	0.18	17302.30	945.09	1413.38

Table 4.1: Comparison of computation time in seconds for the relative volume of $\mathcal{B}_n$ for $n \leq 6$ for two published methods and two new methods.

The last three columns are results using a Windows 10 Pro with an Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz processor. As we were unable to compile the C++ code for the complex analytic method we have included the results given by Beck and Pixton in which they state were scaled down to a 1GHz machine [14].
Baryshnikov and Pemantle [10] question in their 2004 paper the feasibility of the ACSV method when compared to complex analytic techniques, though they had not done any conclusive tests. We feel comfortable stating that the ACSV method is inefficient when compared to the complex analytic method. Not only can this be seen through computation time but through the complexity of the method, namely Algorithm 2 Algebraic Reductions

and ensuring you are only summing terms with a valid vector $\mathbf{b}$. Nevertheless, we believe the ACSV method has a rightful place among the other existing methods used to compute the relative volume of the Birkhoff polytope as it provides a strong geometric insight into why the computation slows down.

Baryshnikov and Pemantle pose a different theorem to compute asymptotic evaluation in [10], this theorem was not implemented in this thesis but it would be interesting to see how the feasibility of their proposed theorem compares to the method of ACSV proposed here. Another possible direction for future work involves implementing changes to the ACSV method to be able to compute the volume of any face of the Birkhoff polytope. This problem is of specific interest because Chan and Robbins [20] state that if we can calculate the volumes of each face of $\mathcal{B}_n$, then one can generate doubly-stochastic matrices uniformly at random. De Loera, Liu, and Yoshida [24] were able to use their generating function for the Ehrhart polynomial of $\mathcal{B}_n$ to compute the Ehrhart polynomials of the facets of $\mathcal{B}_n$ for $n \leq 7$. A third possible direction is the use of the asymptotic estimate for the relative volume of $\mathcal{B}_n$ provided by Canfield and McKay, presented here in Equation (1.12), to eliminate unnecessary calculations in the ACSV method to cut down on computation time.

Bibliography

[1] Asymptotics of multivariate sequences. URL: https://www.cs.auckland.ac.nz/~mcw/Research/mvGF/asymultseq/index.html.

[2] Installing LattE. URL: https://www.math.ucdavis.edu/~latte/software.php.

[3] Magic squares: A simple webquest. Accessed: 2020-04-13. URL: http://plaza.ufl.edu/ufkelley/magic/history.htm.

[4] V. Baldoni, N. Berline, J.A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, M. Vergne, and J. Wu. A User's Guide for LattE integrale v1.7.2, 2013. URL: http://www.math.ucdavis.edu/~latte/.

[5] Velleda Baldoni, Nicole Berline, Jesús A. De Loera, Matthias Köppe, and Michele Vergne. How to integrate a polynomial over a simplex. *Math. Comp.*, 80(273):297–325, 2011. doi:10.1090/S0025-5718-2010-02378-6.

[6] Welleda Baldoni-Silva and Michèle Vergne. Residues formulae for volumes and ehrhart polynomials of convex polytopes. 2001. arXiv:math/0103097.

[7] Alexander I. Barvinok. Computing the Ehrhart polynomial of a convex lattice polytope. *Discrete Comput. Geom.*, 12(1):35–48, 1994. doi:10.1007/BF02574364.

[8] Alexander I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Math. Oper. Res.*, 19(4):769–779, 1994. doi:10.1287/moor.19.4.769.

[9] Alexander I. Barvinok and James E. Pommersheim. An algorithmic theory of lattice points in polyhedra. In *New perspectives in algebraic combinatorics (Berkeley, CA, 1996–97)*, volume 38 of *Math. Sci. Res. Inst. Publ.*, pages 91–147. Cambridge Univ. Press, Cambridge, 1999.

[10] Yuliy Baryshnikov and Robin Pemantle. Convolutions of inverse linear functions via multivariate residues. *preprint*, 48, 2004.

[11] Matthias Beck. Counting lattice points by means of the residue theorem. *Ramanujan J.*, 4(3):299–310, 2000. doi:10.1023/A:1009853104418.

[12] Matthias Beck. Stanley's major contributions to Ehrhart theory. In *The mathematical legacy of Richard P. Stanley*, pages 53–63. Amer. Math. Soc., Providence, RI, 2016. doi:10.1090//mbk/100/03.

[13] Matthias Beck, Moshe Cohen, Jessica Cuomo, and Paul Gribelyuk. The number of "magic" squares, cubes, and hypercubes. *Amer. Math. Monthly*, 110(8):707–717, 2003. doi:10.2307/3647853.

[14] Matthias Beck and Dennis Pixton. The Ehrhart polynomial of the Birkhoff polytope. *Discrete Comput. Geom.*, 30(4):623–637, 2003. doi:10.1007/s00454-003-2850-8.

[15] Matthias Beck and Sinai Robins. *Computing the continuous discretely*. Undergraduate Texts in Mathematics. Springer, New York, second edition, 2015. Integer-point enumeration in polyhedra, With illustrations by David Austin. doi:10.1007/978-1-4939-2969-6.

[16] Garrett Birkhoff. Three observations on linear algebra. *Univ. Nac. Tucumán. Revista A.*, 5:147–151, 1946.

[17] Michel Brion. Points entiers dans les polyèdres convexes. *Scientific annals of the École Normale Supérieure*, 21(4):653–663, 1988. URL: http://www.numdam.org/item/ASENS_1988_4_21_4_653_0/, doi:10.24033/asens.1572.

[18] E. Rodney Canfield and Brendan D. McKay. The asymptotic volume of the Birkhoff polytope. *Online J. Anal. Comb.*, (4):4, 2009.

[19] E. Rodney Canfield and Brendan D. McKay. Asymptotic enumeration of integer matrices with large equal row and column sums. *Combinatorica*, 30(6):655–680, 2010. doi:10.1007/s00493-010-2426-1.

[20] Clara S. Chan and David P. Robbins. On the volume of the polytope of doubly stochastic matrices. *Experiment. Math.*, 8(3):291–300, 1999.

[21] Vašek Chvátal. *Linear programming*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, 1983.

[22] J. A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, and J. Wu. Software for exact integration of polynomials over polyhedra, 2013. doi:10.1016/j.comgeo.2012.09.001.

[23] Jesús A. De Loera, Raymond Hemmecke, Jeremiah Tauzer, and Ruriko Yoshida. Effective lattice point counting in rational convex polytopes. *J. Symbolic Comput.*, 38(4):1273–1302, 2004. doi:10.1016/j.jsc.2003.04.003.

[24] Jesús A. De Loera, Fu Liu, and Ruriko Yoshida. A generating function for all semi-magic squares and the volume of the Birkhoff polytope. *J. Algebraic Combin.*, 30(1):113–139, 2009. doi:10.1007/s10801-008-0155-y.

[25] Persi Diaconis and Bradley Efron. Testing for independence in a two-way table: new interpretations of the chi-square statistic. *Ann. Statist.*, 13(3):845–913, 1985. With discussions and with a reply by the authors. doi:10.1214/aos/1176349634.

[26] Persi Diaconis and Anil Gangolli. Rectangular arrays with fixed margins. In *Discrete probability and algorithms (Minneapolis, MN, 1993)*, volume 72 of *IMA Vol. Math. Appl.*, pages 15–41. Springer, New York, 1995. doi:10.1007/978-1-4612-0801-3_3.

[27] Ricardo Diaz and Sinai Robins. The Ehrhart polynomial of a lattice polytope. *Ann. of Math. (2)*, 145(3):503–518, 1997. doi:10.2307/2951842.

[28] Eugène Ehrhart. Sur les polyèdres rationnels homothétiques à n dimensions. *C. R. Acad. Sci. Paris*, 254:616–618, 1962.

[29] Eugène Ehrhart. Sur un problème de géométrie diophantienne linéaire. I. Polyèdres et réseaux. *J. Reine Angew. Math.*, 226:1–29, 1967. doi:10.1515/crll.1967.226.1.

[30] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009. doi:10.1017/CBO9780511801655.

[31] Christoph Glanzer. Integer programming and lattice point enumeration. 2014.

[32] Peter M. Gruber. *Convex and discrete geometry*, volume 336 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer, Berlin, 2007.

[33] A. J. Hoffman and H. W. Wielandt. The variation of the spectrum of a normal matrix. *Duke Math. J.*, 20:37–39, 1953.

[34] Glenn Hurlbert. A short proof of the birkhoff-von neumann theorem. 2012.

[35] Jim Lawrence. Rational-function-valued valuations on polyhedra. In *Discrete and computational geometry (New Brunswick, NJ, 1989/1990)*, volume 6 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 199–208. Amer. Math. Soc., Providence, RI, 1991.

[36] Ian G. MacDonald. Polynomials associated with finite gell-complexes. s2-4:181–192, 1971. doi:10.1112/jlms/s2-4.1.181.

[37] Joseph Malkevitch. Milestones in the history of polyhedra. In *Shaping space*, pages 53–63. Springer, New York, 2013. doi:10.1007/978-0-387-92714-5_4.

[38] Maplesoft, a division of Waterloo Maple Inc.. Maple 2020.2. URL: https://www.maplesoft.com/.

[39] Marni Mishna. *Analytic Combinatorics: A Multidimensional Approach*. Chapman and Hall/CRC, 2019. doi:10.1201/9781351036825.

[40] Cleve Moler. *Experiments with MATLAB*. mathworks, 2011. URL: https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/exm/book.pdf.

[41] John Mount. *Application of Convex Sampling to Optimization and Contingency Table Generation/counting*. CMU-CS. School of Computer Science, Carnegie Mellon University, 1995. URL: https://books.google.ca/books?id=LcaSoAEACAAJ.

[42] H. L. Ong, H.C. Huang, and W. M. Huin. Finding the exact volume of a polyhedron. *Advances in Engineering Software*, 34(6):351–356, 2003. URL: https://www.sciencedirect.com/science/article/pii/S0965997803000309, doi:https://doi.org/10.1016/S0965-9978(03)00030-9.

[43] James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, second edition, 2011. doi:10.1093/acprof:oso/9780198566946.001.0001.

[44] Robin Pemantle and Mark C. Wilson. *Analytic combinatorics in several variables*, volume 140 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2013. doi:10.1017/CBO9781139381864.

[45] Joseph V. Romanovsky. A simple proof of the Birkhoff-von Neumann theorem on bistochastic matrices. In *A tribute to Ilya Bakelman (College Station, TX, 1993)*, volume 3 of *Discourses Math. Appl.*, pages 51–53. Texas A & M Univ., College Station, TX, 1994.

[46] Richard P. Stanley. *Enumerative combinatorics. Volume 1*, volume 49 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second edition, 2012.

[47] Bernd Sturmfels. Equations defining toric varieties. In *Algebraic geometry—Santa Cruz 1995*, volume 62 of *Proc. Sympos. Pure Math.*, pages 437–449. Amer. Math. Soc., Providence, RI, 1997. doi:10.1016/0040-9383(96)00016-x.

[48] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.2)*, 2020. URL: https://www.sagemath.org.

[49] S. Trandafir. personal communication, 2018-2021.

[50] John von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. In *Contributions to the theory of games, vol. 2*, Annals of Mathematics Studies, no. 28, pages 5–12. Princeton University Press, Princeton, N. J., 1953.

Appendix A

Code

The computations for the results presented in Table 4.1 were done in Maple [38] and SageMath [48].

The following code computes the integrands of the integral presented in Corollary 3.2 for the n^{th} Birkhoff polytope for $n \geq 3$. The terms of the integrand are placed in an array.

```
lecturehallpartitions := proc(n) local i, partition_n, lecturepartition_n, L, dummy, k, r;
# INPUT:
# n - the Birkhoff polytope we want to work with
# OUTPUT:
# lecturepartition_n - the lecture hall partitions of n
        partition_n := partition(n):

        for i to nops(partition_n) do
            if evalb(nops(partition_n[i])<n) = true then
                partition_n[i] := [op(partition_n[i]), seq(0, i = 1 .. n -
                ↪  NumElems(partition_n[i]))];
            fi;
            partition_n[i] := permute(partition_n[i]);
        od:

        L := [seq(i, i = 1 .. n)]:
        lecturepartition_n := Array([]):

        for k to nops(partition_n) do
            for i to nops(partition_n[k]) do
                 dummy := Array([]);
                 for r to n - 1 do
                     if L[r] < sum(partition_n[k][i][h], h = 1 .. L[r]) then dummy :=
                     ↪  Append(dummy, 1); else dummy := Append(dummy, 0); fi;
                 od;
            if has(dummy, 0) = false then lecturepartition_n := Append(lecturepartition_n,
            ↪  partition_n[k][i]); fi;
            od;
        od;

return lecturepartition_n:
end proc:

# Calculate the lecture hall partitions of n, put them in a sequence in a list
seqLHP:= [seq(lecturehallpartitions(n))];
```

```
size_a := numelems(seqLHP);
for i from 1 to size_a do
      a[i]:=seq(seqLHP[i][j], j = 1 .. n);
od;

# Create denominator of integrand
base:=[]:
for k from 1 to n do;
     basefactors:=1:
     for j from 1 to n do;
        if k<>j then basefactors := basefactors*(z[k]-z[j]); fi;
     od:
     base:=[op(base), basefactors];
od;

# Create z^(-bt-1) using standard 1's vector for b
outside:=z[1]:
for i from 2 to n do;
     outside := outside*z[i];
od:
outside := (outside)^(-t-1);

# Create each integrand as per Corollary 3.15
term := Array([seq((outside)*(multinomial(n,
↪  a[r])*simplify(product(((z[v]^(t+n-1))/base[v])^(a[r][v]), v=1..n))), r=1..size_a)]);
```

The following computations apply Algorithm 1 Change of Variables to each term. Only the computations of n equaling 3 are shown below.

```
x := [x1, x2, x3, x4, x5, x6]:

changeofvariables := proc(n, term) local Jacob, detJ, aterm;
# INPUT:
# n - the number of the Birkhoff polytope we want to work with
# term - a term
# OUTPUT:
# aterm - a term with the change of variables applied

        aterm:=term:
        aterm:=simplify(simplify(subs({z[2]=z[1]*x[1], z[3]=z[1]*x[1]*x[2],
        ↪  z[4]=z[1]*x[1]*x[2]*x[3], z[5]=z[1]*x[1]*x[2]*x[3]*x[4],
        ↪  z[6]=z[1]*x[1]*x[2]*x[3]*x[4]*x[5], z[7]=z[1]*x[1]*x[2]*x[3]*x[4]*x[5]*x[6]},
        ↪  aterm), power, symbolic)):

        if (n=3) then
             Jacob, detJ := Jacobian([z[1], z[1]*x[1], z[1]*x[1]*x[2]],[z[1], x[1],
             ↪  x[2]],'determinant');
        fi:

        aterm:= simplify(aterm*detJ):
        aterm:= residue(aterm, z[1]=0);

return aterm;
end proc:

changedterm := Array([seq(changeofvariables(n, term[i]), i = 1 .. size_a)]);
```

The following computations create a list of leading coefficients and another for the vector $\mathbf{b}$ for each term in the integrand.

```
# Create a list of the numerator, and one of the leading coefficients, for each term
numerator := [seq(numer(changedterm[i]), i=1..size_a)]:
lnumerator := [seq(lcoeff(numerator[i]), i=1..size_a)]:
numerator := [seq(numer(changedterm[i])/lnumerator[i], i=1..size_a)]:

powernumerator := proc(numerator, n, w) local b, a, c, j;
# INPUT:
# numerator - the numerator the term
# n - the Birkhoff polytope we want to work with
# w - the amount of times to try finding the numerator. It is set to 200 to to ensure all
↪  entries of the vector b are computed.
# OUTPUT:
# the vector b

        b:= []:
        try
            for j from 1 to w+1 do
                if type(op([j], numerator),rational) = false then
                   a:= op([j], numerator);
                   c:= op(2,a);
                   b:=[op(b), -lcoeff(c)];
                fi;
            od;
        catch: Error
        end try;
        if nops(b) < (n-1) then: b:=[op(b), 1]; fi:
return b;
end proc:

# calculate b using powernumerator. Note that b will be denoted by chambvect.
chambvect:=[]:
for i from 1 to size_a do;
   chambvect:=[op(chambvect), powernumerator(numerator[i], n, 200)];
od:
```

The following code extracts $\mathbf{x}^{-t\mathbf{b}-1}$ from Equation (3.5) in Theorem 3.1.1 leaving $F(\mathbf{x}) = \frac{G}{H}$. We then place the information regarding G into two arrays. Again we only show this for $n = 3$.

```
# remove x^bt+1 to leave G from F=G/H
if (n=3) then
   G:=[seq(simplify((x[1]^(chambvect[i][1]*t+1)*x[2]^(chambvect[i][2]*t+1))*numerator[i]),
   ↪  i=1..size_a)];
fi:

# Place the information for G in two arrays. Gbase provides the entry of the vector x and
↪  Gpower provides the power of that entry.
Gbase := Array([seq(0, i = 1 .. size_a)]):
Gpower := Array([seq(0, i = 1 .. size_a)]):

for i from 1 to size_a do:
    w := 0:
    if has(G[i], x[1]) then Gbase[i]:= [1]; fi;
    while has(G[i], x[1]) do:
         newG := G[i]/x[1];
         G[i] := newG;
```

```
          w := w+1;
    od:
    Gpower[i]:= [w]:

    w := 0:
    if has(G[i], x[2]) then Gbase[i]:= [op(Gbase[i]), 1]; fi;
    while has(G[i], x[2]) do:
          newG := G[i]/x[2];
          G[i] := newG;
          w := w+1;
    od:
    Gpower[i]:= [op(Gpower[i]), w]:

    od:
    Gpower[i]:= [op(Gpower[i]), w]:
od:

Gbase[-1]:= [Gbase[-1]]:
```

The following code calculates the multiplicity vector, $\mathbf{m}$, in the form of a list.

```
# Create a list of the denominator H
H:=seq(denom(changedterm[i]), i=1..size_a):

powerH := proc(H, w) local m, j, a;
# INPUT:
# denominator - H from F=G/H
# n - the number of the Birkhoff polytope we are working with
# w - the amount of times to try finding the numerator. It is set to 200 to to ensure all
↪  entries of the m are computed.
# OUTPUT:
# m - the multiplicity vector

        m:=[];
        try
            for j from 1 to w+1 do
                a:=op(j, H);
                if op(2,a) <> -1 then m:=[op(m), op(2,a)]
                else m:=[op(m), 1] fi;
            od;
        catch: Error
        end try;
return m;
end proc:

#calculate the multiplicity vector m for each term
multiplicity:=[]:
for i from 1 to size_a do;
   powerH(H[i], 200);
   multiplicity:=[op(multiplicity), powerH(H[i], 200)];
od:
```

The following code calculates the matrix $\mathbf{D}$ from Equation (3.9). The columns are the matrix are stored as entries in a list.

```
# INPUT:
# H - H from F=G/H
# w - the amount of times to try finding the power of each x in H_j.
```

```
# OUTPUT:
# Mat - list of the columns of the matrix D

        MatrixD := proc(H, n, w) local Mat, j, L, a, k,i;
        Mat:=[]:
        try
            for j from 1 to w+1 do;
                L:=[seq(0,i=1..n-1)]:
                a:=op(j, H):
                    for k from 1 to n-1 do;
                        if has(a, x[k]) = true then L[k]:=1 fi;
                    od:
                Mat:=[op(Mat), L]:
            od:
        catch: Error
        end try;
return Mat
end proc:

# create a list of the columns of the matrix D for each term
MatB:=[]:
for i from 1 to size_d do;
    MatD:=[op(MatD), MatrixB(denominator[i], n, nops(multiplicity[i]))];
od:
```

Example A.0.1. The following is the Maple output for $n = 3$,

```
L:=[n, MatB, multiplicity, chambvect, lnumerator, convert( Gbase, list, nested ), convert(
↪  Gpower, list, nested )];
 L := [3, [[[1, 0], [1, 1], [0, 1]], [[1, 0], [1, 1]]], [[3, 2, 1], [3, 3]], [[1, 1], [2,
 ↪  1]], [-3, 1], [[1], [0]],  [[1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]]
```

Notice that the arrays for Gbase and Gpower are converted to lists. When $n = 7$ the number of entries is too large for Maple therefore an array must be used. SageMath reads L in list form so Gbase and Gpower must be converted to lists.

The following code is done in SageMath. This work is done alongside Stefan Trandafir [49] who wrote the base of this code for us to develop for the case of the Birkhoff polytope when $n \geq 3$.

The data given as output by Maple is read by SageMath as follows.

```
Lnew = [x for x in L]
n, A, multiplicity, chambvect, lnumera, Gbase, Gpower = Lnew

# Implement the first speed up to Algorithm 2, "Presort H_j's"
def column_rearrange(A, mult):
    """
    INPUT:
    A - input matrix
    mult - the multiplicity of each column vector in A

    OUTPUT:
    Anew - the Matrix A such that the columns are ordered from lowest multiplicity to
    ↪  highest
    multnew - the multiplicity of each column vector in Anew
    """
    permutes = [Word(mult[i]).standard_permutation().inverse() for i in
    ↪  range(_sage_const_0, len(mult))]
```

```
    multnew = [sorted(mult[i]) for i in range(_sage_const_0, len(mult))]
    A = [[A[j][i - _sage_const_1] for i in permutes[j]] for j in range(len(permutes))]
    Anew = [Matrix(A[u]) for u in range(len(A))]

    return Anew, multnew

A, multiplicity = column_rearrange(A, multiplicity)
```

This following code is run for every term given in the integrand of Algorithm 1 Change of variables, each term will be denoted by the counter w. The final answer for each term is then summed up to obtain the relative volume of $\mathcal{B}_n$.

The following pre-processing is slightly different from the ASCV method discussed in Section 3.1.2. Proposition 3.1.1 will not be used to test whether $\mathbf{p}$ is a transverse multiple point, instead the circuits and broken circuits of $\mathbf{D}$ (in the code these matrices are referred to as A) are calculated, if there are no broken circuits then $\mathbf{p}$ is a transverse multiple point by Lemma 3.1.1.

```
def circuit_permute_lift(circuit, h_j, p):
    """
    INPUT:
    circuit - a set of circuits for the matrix A
    h_j - a list of H_j's that form H
    p - transverse multiple point
    OUTPUT:
    circuit_rule - a circuit rewriting rule for the Nullstellensatz certificates (a list of
    ↪  'coefficients')
    """
    polys = []

    # try to find rewriting rule in standard way (max polynomial in terms of rest)
    broken_circuit_polys = [h_j[elem-1] for elem in circuit[:-1]]
    circuit_max_poly = h_j[circuit[-1]-1]
    try:
        polys = circuit_max_poly.lift(broken_circuit_polys)
    except ValueError:
        pass
    if polys:
        good = 1
        for k in polys:
            if k.subs(p) == 0:
                good = 0
            if good == 1:
                circuit_rule = polys
                return circuit_rule

    # try rewriting for every other element in the circuit until a way is found
    for elem in circuit[:-1]:
        circuit_poly = h_j[elem-1]
        torn_circuit_polys = [h_j[c-1] for c in circuit if c != elem]
        try:
            polys = circuit_poly.lift(torn_circuit_polys)
        except ValueError:
            pass
        if polys:
            good = 1
            for k in polys:
                if k.subs(p) == 0:
                    good = 0
```

```
                if good == 1:
                    d = -polys[-1]
                    i = circuit.index(elem)
                    polys = polys[:i] + [-h_j[0]**0] + polys[i:-1]
                    circuit_rule = [p/d for p in polys]
                    return circuit_rule

    return []


def nullstellensatz_certificates(circuits, h_j, p):
    """
    INPUT:
    circuits - a set of circuits of the matrix A
    h_j - list of H_j's that form H
    p - transverse multiple point
    OUTPUT:
    circuit_rewriting_rules - a dictionary of the Nullstellensatz certificates for each
    ↪  circuit (key: list of non-negative integers in 1..NC, val: list of elements from
    ↪  ring)
    """

    circuit_rewriting_rules = {}
    problematic_circuits = []

    for circuit in circuits:
        circuit_rule = circuit_permute_lift(circuit, h_j, p)
        if not circuit_rule:
            problematic_circuits.append(circuit)
            print('There is an problem with circuit {}. The polynomial {} is not in the
            ↪  ideal generated by the polynomials {}'.format(circuit, circuit_max_poly,
            ↪  broken_circuit_polys))
                sys.exit()
        else:
            circuit_rewriting_rules[Set(circuit)] = circuit_rule

    return circuit_rewriting_rules

def pre_processing(A, multipl, gbase, gpower):
    """"
    INPUT:
    A - the matrix A formed from H
    multipl - multiplicities of the columns of A
    gbase - entries of the vector x in each H_j
    gpower - power of the entries of the vector x in each H_j
    OUTPUT:
    rank - rank of matrix A
    nr - number of rows of A
    circuits - circuits of the matrix A
    broken_circuits - broken circuits of A
    circuit_rewriting_rules - a dictionary of the Nullstellensatz certificates for each
    ↪  circuits
    p - the transverse multiple point
    g - G from F=G/H
    H - H from F= G/H
    """
    rank, nc, nr = A.rank(), A.ncols(), A.nrows()
    M = Matroid(A)
```

```
    # compute the circuits of M
    circuits1 = [list(sorted([elem+1 for elem in circuit])) for circuit in M.circuits()]
    circuits = sorted(circuits1, key=lambda c: [-c[-1]] + c[-1:0:-1])

    # compute the broken circuits of M
    broken_circuits1 = [set(circuit[:-1]) for circuit in circuits]
    broken_circuits = []
    for broken_circuit in broken_circuits1:
        if broken_circuit not in broken_circuits:
            broken_circuits.append(broken_circuit)

    # intialize ring and create the vector x
    P = PolynomialRing(QQ, nr, 'x', order='lex', implementation='singular')
    X = list(P.gens())

    # create H as a dictionary
    H = {i: multipl[k] for i, k in zip(range(_sage_const_1, nc + _sage_const_1),
    ↪  range(_sage_const_0, len(multipl)))}

    # create the H_j's and place in a list as to coincide with the dictionary H
    H_j = [_sage_const_1 - reduce(lambda p, q: p * q, [X[i] ** col[i] for i in range(nr)])
    ↪  for col in A.columns()]

    # create G
    g = _sage_const_1
    for i in range(len(gbase)):
            if gbase[i] == 1:
                g=g*(X[i] ** (gpower[i]))

    # compute Nullstellensatz certificates
    p = {y: _sage_const_1 for y in X}
    circuit_rewriting_rules = nullstellensatz_certificates(circuits, H_j, p)

    return rank, nr, circuits, broken_circuits, circuit_rewriting_rules, p, g, H, H_j, P

A_matrix = A[w].transpose()
multipl = multiplicity[w]
rank, nr, circuits, broken_circuits, circuit_rewriting_rules, p, g, H, H_j, P =
↪  pre_processing(A_matrix, Gbase[w], Gpower[w]
```

The following calculations computes the chamber of importance and if necessary uses Algorithm 3 Finding a chamber if the vector $\mathbf{b}$ falls on a face of the chamber complex.

```
def proj_a_faces(A):
    """
    INPUT:
    A - the matrix A formed from H
    OUTPUT:
    projected_a_faces - a list of all faces of the chamber complex
    """
    A_matroid = Matroid(A)
    a_faces = list(A_matroid.bases())
    projected_a_faces = [Cone([A.column(i) for i in a_face]) for a_face in a_faces]

    return projected_a_faces

def git_chamber(v, A_gamma, projected_a_faces):
```

```
    """
    INPUT:
    v - a vector
    A_gamma - the chamber complex of A
    projected_a_faces - a list of all faces of the chamber complex
    OUTPUT:
    nothing - this occurs if v is not in the chamber formed by the cones
    chamber - a chamber (in the form of a cone) containing the vector v
    """
    chamber = A_gamma

    if v not in chamber:
        return

    for projected_a_face in projected_a_faces:
        if v in projected_a_face:
            chamber = chamber.intersection(projected_a_face)
    return chamber

def b_chamber(b, A):
    """
    INPUT:
    A - the matrix A formed from H
    b - a vector
    OUTPUT:
    chamber - the  chamber of importance (hence the chamber that contains b)
    """
    b = vector(b)
    projected_a_faces = proj_a_faces(A)
    A_gamma = Cone([A.column(i) for i in range(A.ncols())])

    chamber = git_chamber(b, A_gamma, projected_a_faces)

    # set a counter so that we do not end up in an endless loop is no chamber exists
    max_iter = 1000
    counter = 0
    d = len(b)
    c = 1/512

    # Algorithm 2 Finding a chamber
    while (not chamber or chamber.dim() != chamber.lattice_dim()) and counter < max_iter:

            # compute a random normal vector.
            normal_vector = vector([0 for i in range(d)])
            for i in range(d-1):
                normal_vector[i] = randint(0,10)
            normal_vector[d-1] = (-1//b[d-1])*sum([a*b for a,b in zip(b[:-1],
            ↪  normal_vector)])

            b_new = tuple(b + c*normal_vector)
            counter = counter+1

            chamber = git_chamber(b_new, A_gamma, projected_a_faces)

    return chamber

# compute chambers complex
vec = chambvect[w]
```

```
chamber_list = [b_chamber(vec, A_matrix)]
```

The following calculations run Algorithm 2 Algebraic Reductions. Some of the smaller calculations will not be displayed.

```
def algebraic_reduction(rank, circuits, broken_circuits, circuit_rewriting_rules, p, g, h,
↪  h_polys, chamber, A):
    """
    OUTPUT:
    summands - the list summands of the term F
    """
    summands = []
    G = 1
    summand_queue = [Summand(h, G)]
    while summand_queue:
        summand = summand_queue.pop(0)

        # implementation to speed up algorithm number 2, collect terms with the same H
        ↪  before each round of reduction
        summand.sum_queue(summand_queue)

        # implementation to speed up algorithm number 3, check if summands chamber complex
        ↪  contains the chamber of importance
        if summand.contains_the_chamber(chamber, A):
            # Step 1 of Algorithm 2
            if len(summand.h) > rank or summand.contains_broken_circuit(broken_circuits):
                circuit = summand.find_circuit(circuits)
                if circuit:
                    children = summand.circuit_reduce(circuit, circuit_rewriting_rules)
                    summand_queue1 = summand_queue + children
                    summand_queue = summand_queue1
                else:
                    print('No circuits associated to {} have Nullstellensatz
                    ↪  certificates'.format(summand.support()))
                    sys.exit()

            else:
                summands.append(summand)

    # Step 2 of Algorithm 2
    new_summands = []
    i = 0
    while i < len(summands):
        summand = summands.pop(0)
        summand.sum_queue(summands)
           # multiply by our numerator g before steps 3 and 4 of Algorithm 2.
        summand.numerator = summand.numerator*g
        new_summands.append(summand)
    summands = new_summands

    for summand in summands:
        if summand.numerator == 0:
            summands.remove(summand)

    # Step 3 Of Algorithm 2
    i = 0
    while i < len(summands):
        summand = summands[i]
```

```
        # implementation to speed up algorithm number 4, check if summand vanishes at
        ↪  multiple point
        if summand.vanish(p):
            new_summands = summand.split3(h_polys)

            if new_summands:
                for summand in new_summands:
                    if summand.contains_the_chamber(chamber, A):
                        summands = summands + summand
                summands.pop(i)

            else:
                i = i + 1

        else:
            i = i + 1

    # Step 4 Of Algorithm 2
    i = 0
    while i < len(summands):
        summand = summands[i]

        if summand.vanish(p):
            new_summands = summand.split4(h_polys)

            if new_summands:
                for summand in new_summands:
                    if summand.contains_the_chamber(chamber, A):
                        summands = summands + summand
                summands.pop(i)

            else:
                i = i + 1

        else:
            i = i + 1

    return summands

# Perform algebraic reduction
finalsummands = algebraic_reduction(rank, circuits, broken_circuits,
↪  circuit_rewriting_rules, p, g, H, H_j, chamber_list[0], A_matrix)
```

The following calculations compute Equation (3.6) of Theorem 3.1.1.

```
def asymptotics(self, A_matrix, vectr, p):
    """
    INPUT:
    A_matrix: input matrix
    vectr: b*t
    p: transverse multiple point in vector form

    OUTPUT:
    phi - asymptotic evaluation of the summand
    """
        gamma_psi = A_matrix.matrix_from_columns([a - 1 for a, b in
        ↪  self.ordered_denominator()]).transpose()
```

```
        if gamma_psi.ncols() != len(vectr):
            sys.exit()
        gp_inverse = gamma_psi.inverse()

        r = vector(vectr)
        n = self.numerator
        Gp = Rational(n.subs(p))
        multiplicities = self.multiplicities()
        numerator = Gp * product([a ** (m - 1) for a, m in zip(r * gp_inverse,
        ↪  multiplicities)])
        denominator = product([factorial(m - 1) for m in multiplicities]) *
        ↪  abs(gamma_psi.determinant())
        phi = (numerator * lnumera[w]) / denominator
        return phi

T = PolynomialRing(QQ, 't', names=('t',));
(t,) = T._first_ngens(1)
vectr = [t * vec[j] for j in range(_sage_const_0, len(vec))]
asymptotics = _sage_const_0
for summand in finalsummands:
    if summand.contains_the_chamber(chamber_list[0], A_matrix):
    asymptotics = asymptotics + summand.asymptotics(A_matrix, vectr, p)
```