Algorithms for Analytic Combinatorics in Several Variables

by

Josip Smolčić

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Abstract

Given a multivariate rational generating function we are interested in computing asymptotic formulas for the sequences encoded by the coefficients. In this thesis we apply the theory of analytic combinatorics in several variables (ACSV) to this problem and build algorithms which seek to compute asymptotic formulas automatically, and to aid in understanding of the theory. Under certain assumptions on a given rational multivariate generating series, we demonstrate two algorithms which compute an asymptotic formula for the coefficients. The first algorithm applies numerical methods for polynomial system solving to compute minimal points which are essential to asymptotics, while the second algorithm leverages the geometry of a so-called height map in two variables to compute asymptotics even in the absence of minimal points. We also provide software for computing gradient flows on the height maps of rational generating functions. These flows are useful for understanding the deformations of integral contours which are present in the analysis of rational generating functions.

Acknowledgment

I would like to give my thanks and appreciation to the following people that helped make this thesis possible:

Stephen Melczer for granting me the opportunity to study mathematics at the graduate level, for bringing this thesis up to a presentable standard and for inspiring confidence in me. It has been a great experience working with you.

Eric Schost and Kevin Purbhoo for reading and providing feedback on this thesis.

The staff and faculty at the Combinatorics and Optimization department at UW for providing an exciting environment to study Mathematics and for supporting me throughout my time here.

My Mom, Dad and Sister for supporting and believing in me, I love you all.

Table of Contents

Chapter 1

Introduction

A generating function is a mathematical object which encodes a sequence. For example, consider the Fibonacci sequence f_n which begins

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ \ldots$$

and is often described by its initial values $f_0 = 1$, $f_1 = 1$ and the recurrence $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$. The recursive description describes the entire Fibonacci sequence, however, it can be difficult to analyze. The generating function $F(z)$

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

of the sequence f_n is the power series in z whose nth coefficient is the nth Fibonacci number. At first, the introduction of a generating function seems superfluous, however it is a powerful tool for analyzing f_n. For instance, the Fibonacci generating function satisfies

$$\begin{aligned}
(1 - z - z^2)F(z) &= F(z) - zF(z) - z^2F(z) \\
&= \sum_{n \geq 0} f_n z^n - z \sum_{n \geq 0} f_n z^n - z^2 \sum_{n \geq 0} f_n z^n \\
&= \sum_{n \geq 0} f_n z^n - \sum_{n \geq 0} f_n z^{n+1} - \sum_{n \geq 0} f_n z^{n+2} \\
&= f_0 + (f_1 - f_0)z + \sum_{n \geq 2} (f_n - f_{n-1} - f_{n-2}) z^n = f_0 = 1,
\end{aligned}$$

and thus $F(z) = \frac{1}{1-z-z^2}$. Since $F(z)$ is a rational function, we may apply partial fraction decomposition, and geometric series identities to derive the formula

$$f_n = A\varphi^n + B\varphi^{-n}$$

where $\varphi = \frac{\sqrt{5}+1}{2}$, $A = \frac{1}{1+1/\varphi^2}$ and $B = 1 - A$.

The field of analytic combinatorics uses generating functions to derive behavior of univariate sequences. It is now classical. In this thesis we are interested in the behavior of multivariate generating functions, an area still being developed.

1.1 Overview of Topics

Analytic combinatorics in several variables (ACSV) is the subject that applies the theory of complex analysis to analyze multivariate generating functions and the sequences which they encode. The ACSV framework draws on many areas of mathematics, from algebra and topology to geometry and complex analysis. An introduction to the theory and methods in one and several variables is given in [13]. Throughout this thesis we build on the theory to develop software which applies the results of ACSV to automatically analyze generating functions in several variables.

In Chapter 2 of this thesis we establish the necessary background, notation and machinery needed to describe the methods in the following chapters. We begin with an overview of the necessary tools for studying systems of multivariate polynomials, and in particular computing the solutions to these systems. Following this we review background on formal power series and generating functions. Generating functions are important in combinatorics and computing asymptotic formulas for the sequences they encode is the primary application of the software developed in this thesis. In the final section of Chapter 2 we introduce the methods of analytic combinatorics with several examples, stating the key results that we use in our analysis and introducing the problems we seek to solve.

In Chapter 3 we discuss our results and contributions. Section 3.1 builds on the work of Melczer and Salvy in [14] to reduce the problem of computing the asymptotic formula for a sequence with a given rational generating function, satisfying certain conditions, to the problem of analyzing the solutions to a particular system of polynomial equations. We apply the method of homotopy continuation through the Julia software implementation [4] by Breiding and Timme to solve these polynomial equations. In Section 3.2 we discuss an important tool in the theory of ACSV called the height function, and show a method of numerically computing a flow on the surface which is defined by this height function. A Julia implementation of this is provided to aid in visualization of ACSV problems. In Section 3.3 we provide a `SageMath` [18] implementation of an algorithm of DeVries [5]. This algorithm leverages the geometry of height functions in two variables to provide

an effective characterization of points necessary in computing asymptotic formulas for coefficients of bivariate generating functions using ACSV. We conclude with a discussion of future work in Section 3.4.

1.2 Original Contributions

The software developed in this thesis can be found on

https://github.com/JSmol/acsv-algorithms

and is original. Further contributions include

- ◇ The Julia software package `ACSVHomotopy` discussed and used in Section 3.1, and an accompanying paper [11] with Lee and Melczer. The software implements both the combinatorial, non-combinatorial and heuristic methods, and it is the only practical software to study multivariate generating functions that are non-combinatorial.
- ◇ The Julia software for computing flows on height function in Section 3.2.
- ◇ To our knowledge the first implementation of DeVries' algorithm for bivariate ACSV discussed in Section 3.3.

Chapter 2

Background and Motivating Examples

In this chapter we give an overview of the definitions, theorems and concepts which we rely on in later chapters. The first section discusses the background on polynomial system solving that is heavily relied upon in the theory of ACSV. The following section discusses our main object of study: generating functions. The final section introduces the ACSV framework as currently established, which is built upon in the later chapters of this thesis. Several examples are given to illustrate the concepts, and references are given for further reading when applicable.

2.1 Polynomials

Let $\mathbb{K}$ be a field. We denote the ring of n-variate polynomials over $\mathbb{K}$ with indeterminants $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ as $R = \mathbb{K}[\mathbf{z}]$. We use the notation $\mathbf{z}^{\mathbf{j}} = z_1^{j_1} \cdots z_n^{j_n}$ for multivariate monomials. Given any polynomial

$$f = \sum_{\mathbf{j} \in \mathbb{N}^d} c_{\mathbf{j}} \mathbf{z}^{\mathbf{j}} \in \mathbb{K}[\mathbf{z}]$$

we call the set $\text{supp}(f) = \{\mathbf{j} : c_{\mathbf{j}} \neq 0\}$ the *support* of f. Given any finite set of polynomials $F = \{f_1, f_2, \ldots, f_m\} \subseteq R$ the *ideal* generated by F is

$$\langle F \rangle = \langle f_1, f_2, \ldots, f_n \rangle = \left\{ \sum_{j=0}^{m} a_j f_j : a_j \in R \right\}.$$

A set F generating an ideal I is called a *basis* for the ideal I. When considering sets of polynomials we are often interested in the vanishing set of F, that is the points $\mathbf{w} \in \mathbb{K}^n$ such that $f(\mathbf{w}) = 0$ for each $f \in F$. The vanishing set of F is denoted $\mathcal{V}(F)$, and the

vanishing set of any set of polynomials is called an *algebraic set*. Since any point vanishing on F also vanishes on $\langle F \rangle$, we also consider vanishing sets of entire ideals $\mathcal{V}(\langle F \rangle) = \mathcal{V}(F)$.

Example 1. When $f(x, y) = x^2 + y^2 - 1$, the real part of the vanishing set denoted $\mathcal{V}(f) \cap \mathbb{R}^2$ is the set of all points on the unit circle in $\mathbb{R}^2$.

Example 2. When $F = \{x^2 + y^2 - 1, x - 1/2\}$ the real part of the vanishing set has precisely two elements, $\mathcal{V}(F) = \left\{\left(1/2, \pm\sqrt{3/4}\right)\right\}$. We understand these elements as the intersection of a circle and a line in $\mathbb{R}^2$.

Notice that a system of polynomial equations may have infinitely many solutions, as in Example 1. For our applications, we are interested in polynomial systems which have only finitely many solutions, and call a polynomial system F *zero dimensional* if $\mathcal{V}(F)$ is finite. The system $\{x^2 + y^2 - 1, x - 1/2\}$ in Example 2 is zero dimensional. An important question we seek to answer is: given a zero dimensional system F, how do we compute the set $\mathcal{V}(F)$?

Example 3. Let $f_1 = 1+x+y+z$, $f_2 = 1+x^2+y^2$, $f_3 = 1-x-y+z^2$ and $F = \{f_1, f_2, f_3\}$. The ideal $I = \langle F \rangle$ has an alternate generating set,

$$I = \langle x + y + z + 1, y^2 + yz + y + z/2, z^2 + z + 2 \rangle.$$

From here we can solve for $\mathcal{V}(I)$ as follows. Notice that $z^2 + z + 2$ is univariate and has solutions $\sigma_1 = \frac{1+\sqrt{-7}}{2} \approx -0.500+1.323i$ and $\sigma_2 = \frac{1-\sqrt{-7}}{2} \approx -0.500-1.323i$. Next, $y^2+yz+y+z/2$ only has y and z variables, so substituting $z = \sigma_1$ gives a univariate polynomial in y with approximate solutions $-0.588 + 0.172i$ and $0.088 + 1.151i$, and substituting $z = \sigma_2$ gives approximate solutions $-0.588 - 0.172i$ and $0.088 - 1.151i$. The x coordinate of a solution is obtained by substituting z and y with σ_1 and its corresponding y solutions, into $x + y + z + 1 = 0$ and the same is done with σ_2. The result is 4 solutions, with approximations

$$\begin{array}{lll}
z = -0.500 - 1.323i & y = -0.588 + 0.172i & x = 0.088 + 1.151i \\
z = -0.500 - 1.323i & y = 0.088 + 1.151i & x = -0.588 + 0.172i \\
z = -0.500 + 1.323i & y = -0.588 - 0.172i & x = 0.088 - 1.151i \\
z = -0.500 + 1.323i & y = 0.088 - 1.151i & x = -0.588 - 0.172i.
\end{array}$$

In Example 3 the solutions to the polynomial system F are difficult to read off, however given a different basis for $I = \langle F \rangle$ we are able to explicitly compute the solutions easily. The alternative basis given in Example 3 is an example of a special basis called a **Gröbner basis**. Gröbner bases offer an approach to computing vanishing sets of zero dimensional systems.

2.1.1 Gröbner Bases

For our purposes, Gröbner bases are a useful computational tool for computing vanishing sets of special polynomial systems, however the theory of Gröbner bases and their applications is much deeper. For completeness we define the machinery required for our application. The book [6] provides further reading of Gröbner bases and various other applications can be found. We follow [19, Chapter 21] which provides a computational introduction to Gröbner bases.

Definition 1. A *monomial order* is a total order $\preceq$ on the set of monomials $\mathbf{z}^{\mathbf{a}}$ such that $1 \preceq \mathbf{z}^{\mathbf{a}}$ for all monomials $\mathbf{z}^{\mathbf{a}}$ and $\mathbf{z}^{\mathbf{a}+\mathbf{c}} \preceq \mathbf{z}^{\mathbf{b}+\mathbf{c}}$ whenever $\mathbf{z}^{\mathbf{a}} \preceq \mathbf{z}^{\mathbf{b}}$.

Example 4. Let $\mathbf{z}^{\mathbf{a}} \neq \mathbf{z}^{\mathbf{b}}$ be monomials. The order $\preceq_{\text{lex}}$ defined by

$$\mathbf{z}^{\mathbf{a}} \preceq_{\text{lex}} \mathbf{z}^{\mathbf{b}} \iff \text{the leftmost nonzero entry of } \mathbf{b}-\mathbf{a} \text{ is positive}$$

is a monomial order. For example, in $\mathbb{Q}[z_1, z_2, z_3]$ we order the monomials z_1, z_2, z_1z_2, $z_1^3z_3$, $z_2^4z_3$ by $\preceq_{\text{lex}}$ as

$$z_2 \preceq_{\text{lex}} z_2^4z_3 \preceq_{\text{lex}} z_1 \preceq_{\text{lex}} z_1z_2 \preceq_{\text{lex}} z_1^3z_3.$$

Definition 2. Let f be a d-variate polynomial such that $f = \sum_{\mathbf{a}\in\mathbb{N}^d} f_{\mathbf{a}}\mathbf{z}^{\mathbf{a}}$ where only finitely many $f_{\mathbf{a}} \in \mathbb{K}$ are nonzero. The *leading term* of f with respect to a monomial order $\preceq$ is denoted $\text{lt}_{\preceq}(f)$, where $\text{lt}_{\preceq}(f) = f_{\mathbf{a}}\mathbf{z}^{\mathbf{a}}$ if and only if $\mathbf{z}^{\mathbf{a}} \succeq \mathbf{z}^{\mathbf{b}}$ whenever $f_{\mathbf{b}} \neq 0$. We also employ the notation

$$\text{lt}_{\preceq}(I) = \langle \text{lt}_{\preceq}(f) : f \in I \rangle$$

for ideals $I = \langle f_1, f_2, \ldots, f_m \rangle$ and monomial orders $\preceq$. When the monomial order is clear from context, we write $\text{lt}(f)$ instead of $\text{lt}_{\preceq}(f)$ and $\text{lt}(I)$ instead of $\text{lt}_{\preceq}(I)$.

Example 5. Suppose we order the variables as $x \preceq_{\text{lex}} y \preceq_{\text{lex}} z$ in $\mathbb{Q}[x, y, z]$, then for $f = x + y + xy + x^3z + y^4z$ we have $\text{lt}(f) = x^3z$ and $\text{lt}(\langle f \rangle) = \langle x^3z \rangle$.

Definition 3 (Gröbner Basis)**.** Fix a monomial order $\preceq$ and consider a polynomial ideal $I = \langle f_1, f_2, \ldots, f_n \rangle$. A basis $g_1, g_2, \ldots, g_m$ for I is a *Gröbner basis* for I if $I = \langle g_1, g_2, \ldots, g_m \rangle$ and $\text{lt}(I) = \langle \text{lt}(g_1), \text{lt}(g_2), \ldots, \text{lt}(g_m) \rangle$.

Gröbner bases have many nice properties which are useful in theory and practice. Here we are primarily interested in computing Gröbner bases explicitly for solving zero dimensional polynomial systems. The following theorem illustrates how Gröbner basis are used for explicit computations when working with multivariate polynomials.

Theorem 1 (Elimination Theorem; Theorem 3.3 of [6])**.** Let I be an ideal in $K[\mathbf{z}]$, let $R_t = \mathbb{K}[z_{t+1}, \ldots, z_d]$ and let $I_t = I \cap R_t$. If G is a Gröbner basis for I with respect to the lexicographic order then

$$G_t = G \cap R_t$$

is a Gröbner basis for I_t with respect to the induced lexicographic order on R_t.

It is interesting to note that Theorem 1 does not work for arbitrary term orders. Term orders for which the elimination theorem holds are called *elimination orders*. The elimination theorem allows vanishing sets of zero dimensional ideals to be computed one coordinate at a time, as illustrated in Example 3, and Theorem 1 implies that this method applies to any zero dimensional ideal. Unfortunately, computing Gröbner bases with respect to the lexicographic order is at times not practical due to the computational complexity of the task. Alternative orders which are more efficient to compute exist, but come at the cost of not being an elimination order. Although there are methods to convert between Gröbner bases with different monomial orders for zero dimensional systems, we consider an alternative method for computing the zeros of polynomial systems.

2.1.2 Homotopy Continuation

Homotopy continuation is a method for numerically solving equations by analyzing maps between polynomial systems called homotopies in $R = \mathbb{C}[\mathbf{z}]$. Given continuous functions F and G mapping from $\mathbb{C}^d$ to $\mathbb{C}^m$, a continuous map $H : \mathbb{C}^d \times [0,1] \to \mathbb{C}^m$ where $H(\mathbf{z}, 0) = G(\mathbf{z})$ and $H(\mathbf{z}, t) = F(\mathbf{z})$ is called a *homotopy* between G and F. In our case the maps F and G are m-tuples of d-variate polynomials being evaluated at points in $\mathbb{C}^d$, that is, we identify the polynomial systems F with the continuous map $F : \mathbb{C}^d \to \mathbb{C}^m$ defined by $F(\mathbf{z}) = (f_1(\mathbf{z}), f_2(\mathbf{z}), \ldots, f_m(\mathbf{z}))$.

Let $F = \{f_1, f_2, \ldots, f_m\} \subseteq R$ and let $G = \{g_1, g_2, \ldots, g_m\} \subseteq R$ be polynomial systems, and suppose that both G and F are zero dimensional. Further suppose that the set $\mathcal{V}(G)$ is known. The method at a high level is to construct a differentiable homotopy map $H(\mathbf{z}, t)$ such that $H(\mathbf{z}, 0) = G(\mathbf{z})$ and $H(\mathbf{z}, 1) = F(\mathbf{z})$, then track the zeroes of $H(\mathbf{z}, t)$ in the $\mathbf{z}$ variables as t varies smoothly from $0 \to 1$. Under certain assumptions on the map H, all the unknown solutions $F(\mathbf{z})$ can be recovered at $t = 1$.

The polynomial system G is called the *start system* in the homotopy continuation method. The choice of a start system G and homotopy H is important in determining whether the homotopy continuation method will find all possible solutions. In practice, start systems $G(\mathbf{z})$ and homotopies $H(\mathbf{z}, t)$ that meet the requirements for homotopy continuation exist *generically*. We say that a property holds generically if it holds for all polynomials (or systems of polynomials) of fixed support except for those which have coefficients in a fixed algebraic set.

The simplest example of a start system is called a *total degree start system*. Given a square polynomial system $F = \{f_1, \ldots, f_m\}$, Bézout's theorem states that there are at most $N = \prod_{j=1}^{m} \deg(f_j)$ isolated solutions to $F(\mathbf{z})$. Let $g_j(\mathbf{z}) = z_j^{\deg(f_j)} - 1$ for $j \in [m]$. The system $G = \{g_1, \ldots, g_m\}$ has exactly N solutions in $\mathbb{C}^m$. We construct a homotopy

$H_\zeta(\mathbf{z}, t) = \zeta t F(\mathbf{z}) + (1-t)G(\mathbf{z})$ where ζ is a random nonzero complex number so that the solutions of $H_\zeta(\mathbf{z}, 1)$ are the same as those of $F(\mathbf{z})$. Multiplying by ζ is done in practice to decrease the likelihood of a root vanishing or two roots colliding as t goes from $0 \to 1$.

Example 6. We begin with a simple one variable illustration of the idea. Suppose we wish to use this method to calculate the zeroes of the quintic equation

$$F(x) = x^5 - 6x^4 + 16x^2 - 4.$$

Let $G = (x-2)(x-1)x(x+1)(x+2)$, the solutions of which are $\{0, \pm 1, \pm 2\}$ by construction. The map $H(x,t) = tF(x) + (1-t)G(x)$ is a homotopy between F and G. Moreover, the map $H(x,t)$ has exactly 5 real[1] solutions for all t between 0 and 1. Let N be a natural number and for all $n \leq N$, approximate the solutions to $H(x, (n+1)/N)$ by M iterations of Newtons method. That is, the solution to $H(x, (n+1)/N)$ is approximated by x_M in the sequence

$$x_{m+1} = x_m - \frac{H(x_m, (n+1)/N)}{H_x(x_m, (n+1)/N)}$$

where x_0 is an approximate (or exact in the case $n = 0$) solution to $H(x, n/N)$. Starting from the solutions $0, \pm 1, \pm 2$ of $H(x, 0)$ we build the up all the solutions to $H(x, n/N)$ where $1 \leq n \leq N$. Figure 6 shows the plot of all the solutions when $N = 50$ and $M = 20$. The final result is the 5 points

$$-1.3719, \qquad -0.5316, \qquad 0.5254, \qquad 1.9084, \qquad 5.4697,$$

which numerically match the roots of F. Moreover, the solutions can be approximated arbitrarily closely by choosing M to be larger.

The `HomotopyContinuation.jl` [4] Julia library is complete with sophisticated start systems and homotopies which consider the structure of the given polynomial systems, further increasing the probability that the algorithm successfully finds all the roots without wasting computational effort. While other implementations of the method exist, the Julia library is easy to download and there is minimal code required to build and solve equations. Additionally, it is well documented and there are many examples available online demonstrating the software. The software is available online at

https://www.juliahomotopycontinuation.org.

As we are working with polynomial systems, we also leverage the multivariate polynomial library `DynamicPolynomials.jl` for constructing polynomials and performing basic operations, available online at

[1] We only require that $H(x,t)$ has 5 complex solutions, however, for visualization purposes, it is better to study an example with real roots.

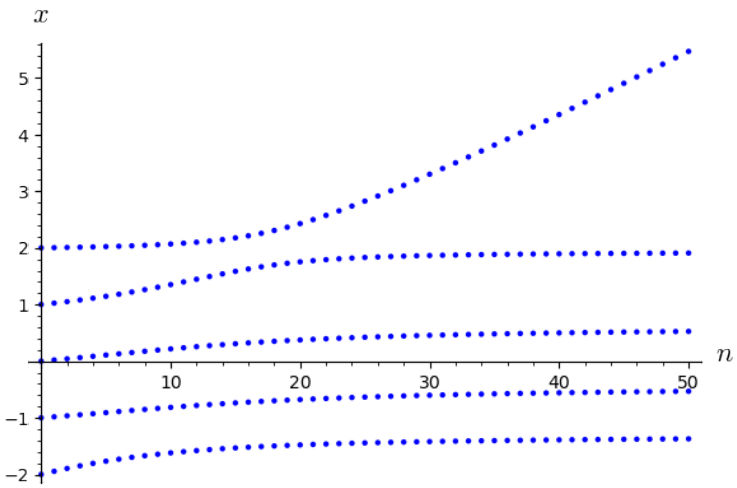


Figure 2.1: Plot of the solutions computed in Example 6. As n goes from $0 \to N$ the solutions of $H(x, n/N)$ go from the solutions of G to the solutions of F.

https://github.com/JuliaAlgebra/DynamicPolynomials.jl.

These packages are imported into Julia with the following code.

```
using HomotopyContinuation
using DynamicPolynomials
```

We omit this code in future examples for succinctness.

Example 7. Suppose we want to solve for the intersection points of two curves in $\mathbb{R}^2$, defined by the solutions of $f = x^3+y^3-xy^2-yx^2-24xy+16$ and $g = -24xy-x+12y+10$. We do this in Julia by writing the following code.

```
@polyvar x y
f = x^3 + y^3 - x*y^2 - y*x^2 - 24*x*y + 16
g = -24*x*y - x + 12*y + 10
F = System([f, g])
result = solve(F)
```

```
Result with 6 solutions
=======================
  6 paths tracked
  6 non-singular solutions (4 real)
  random_seed: 0xe0f2ef76
  start_system: :polyhedral
```

The output shows that there are 6 solutions, 4 of which are real. The package also provides the random seed that was used internally for any random number generations, and which

polynomial system was used as a start system. Often we are interested only in the real solutions, which are recovered using the command `real_solutions(result)`. In this case we have 4 real solutions.

```
4-element Vector{Vector{Float64}}:
 [0.6101580387872538, 3.551655597338044]
 [1.095352969283962, 0.6232050208121189]
 [0.3871275309385142, -3.548574388493044]
 [-1.9435708172691493, -0.2036563788272101]
```

Figure 2.1.2 shows the plot of the curves where we can clearly visualize the 4 real solutions as intersection points.

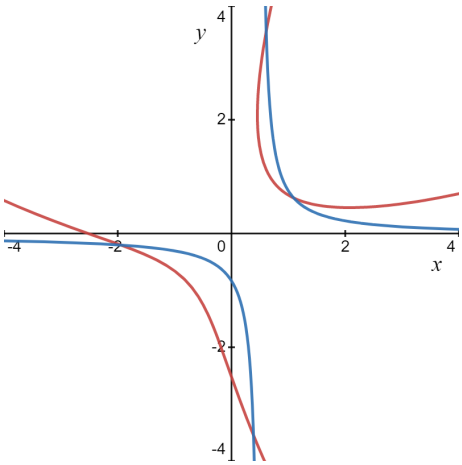


Figure 2.2: Plot of curves defined in Example 7.

A consequence of using numerical methods for polynomial system solving is that we cannot recover the algebraic solutions without outside knowledge, instead using floating point approximations to true algebraic solutions. In general it is not possible to obtain exact algebraic solutions from the homotopy continuation method, however it is possible to compute small subsets of $\mathbb{C}^d$ which are guaranteed to contain distinct algebraic solutions. The `HomotopyContinuation.jl` library accomplishes this *certification* by way of interval arithmetic [3].

Certification of solutions

Define the set of *compact real intervals* to be

$$\mathbb{IR} = \{x : a \leq x \leq b \text{ where } x, a, b \in \mathbb{R}\}$$

with binary operations $+, -, \times, \div$ defined by

$$X \circ Y = \{x \circ y : x \in X, y \in Y\},$$

except that division by 0 is undefined. The set of *complex intervals* is then

$$\mathbb{IC} = \{X + iY : X, Y \in \mathbb{IR}\}.$$

Let $F = \{f_1, \ldots, f_n\}$ be an n-variate square polynomial system. The certification routine of the homotopy continuation library takes the system F and returns a sequence of disjoint tuples $I^{(k)} = \left(I_1^{(k)}, \ldots, I_n^{(k)}\right) \in \mathbb{IC}^n$ such that for each k there exists a unique solution $\mathbf{a} \in \mathbb{C}^d$ of F with $a_j \in I_j^{(k)}$ for all $j \in [n]$. Homotopy continuation also offers methods for certifying whether solutions are real, or real and positive, as this is often an important feature in applications [3].

Example 8. To demonstrate the certification functionality, consider the simple bivariate square system $F = \{x^2 - 1, y^3 - 1\}$. We solve this system in Julia and certify it in Julia with the following code.

```
@polyvar x y
F = System([x^2 - 1, y^3 - 1], variables=[x, y])
sols = solutions(solve(F))
certs = certificates(certify(F, sols))
```

Each certificate corresponds to a solution of F and carries with it metadata about that solution. The following certificate corresponds to the solution $(1, 1)$ of F.

```
SolutionCertificate:
solution_candidate = [
  1.0 + 0.0im,
  1.0 + 0.0im,
]
is_certified = true
certified_solution_interval = [
  [1.000000000000 +/- 1.14e-13] + [+/- 1.14e-13]im,
  [1.000000000000 +/- 1.52e-13] + [+/- 1.52e-13]im,
]
precision = 53
is_real = true
```

The metadata of each certificate shows: an approximate solution, whether the solution was successfully certified, the complex intervals which contain the true algebraic solution, the precision of the floating point numbers, and whether or not the solution is real.

The system F has solutions $\left(e^{2k\pi i/2}, e^{2\ell\pi i/3}\right)$ for $k \in \{0, 1\}$ and $\ell \in \{0, 1, 2\}$. There are exactly 2 real solutions and 1 real positive solution, and we recover them with the following code in Julia.

```
real = filter(is_real, certs) # returns 2 solutions: (-1, 1) and (1, 1)
real_positive = filter(is_positive, certs) # returns 1 solution: (1, 1)
```

Our main algorithm for ACSV presented in Section 3.1 leverages all of these features.

As a final example to demonstrate the power of the method, and the Julia implementation, we walk through an example from [2].

Example 9. A *conic* is the real part of a vanishing set of a degree two polynomial in two variables. How many circles are tangent to 3 conic curves in $\mathbb{R}^2$?

Let $F = \{f_1, f_2, f_3\}$ be degree two bivariate polynomials where f_j is bivariate in variable x_j and y_j. We wish to find values a, b, r such that the circle

$$\left\{(x, y) \in \mathbb{R}^2 : (x-a)^2 + (y-b)^2 = r\right\}$$

is tangent to each conic defined in F. The conic f_j is tangent to the circle at a point $(x_j, y_j) \in \mathbb{R}^2$ if

$$\frac{\frac{\partial}{\partial x_j}\left((x_j-a)^2 + (y_j-b)^2 - r\right)}{\frac{\partial}{\partial y_j}\left((x_j-a)^2 + (y_j-b)^2 - r\right)} = \frac{\frac{\partial}{\partial x_j} f_j}{\frac{\partial}{\partial y_j} f_j}.$$

Therefore, the real solutions to the square system of equations

$$\begin{aligned}
0 &= f_j(x_j, y_j) \\
0 &= (x_j-a)^2 + (y_j-b)^2 - r \\
0 &= 2(x_j-a)\frac{\partial}{\partial x_j} f_j(x_j, y_j) - 2(y_j-b)\frac{\partial}{\partial y} f_j(x_j, y_j)
\end{aligned}$$

in the variables a, b, r, x_1, x_2, x_3, y_1, y_2 and y_3 describe circles of radius r centered at (a, b) tangent to f_j at (x_j, y_j). The system is square with 9 equations and variables. The following code generates random polynomials, builds the corresponding system and solves it in Julia.

```
@polyvar x[1:3] y[1:3] a b r
f = [sum(randn(6) .* [1, x[j], y[j], x[j]*y[j], x[j]^2, y[j]^2]) for j in 1:3]
F = System([
    f; # conics
    [(x[j]-a)^2 + (y[j]-b)^2 - r for j in 1:3]; # circles
    [2*(x[j] - a)*differentiate(f[j], y[j]) - # tangent conditions
     2*(y[j] - b)*differentiate(f[j], x[j]) for j in 1:3]
])
res = solve(F)
```

```
Result with 184 solutions
=========================
  256 paths tracked
  184 non-singular solutions (14 real)
  random_seed: 0xee0b55b6
  start_system: :polyhedral
```

Generically, there are 184 complex solutions with 3 conics, and in our example we find this to be the case. Additionally, we find there are 14 real solutions. Figure 2.3 plots the results.

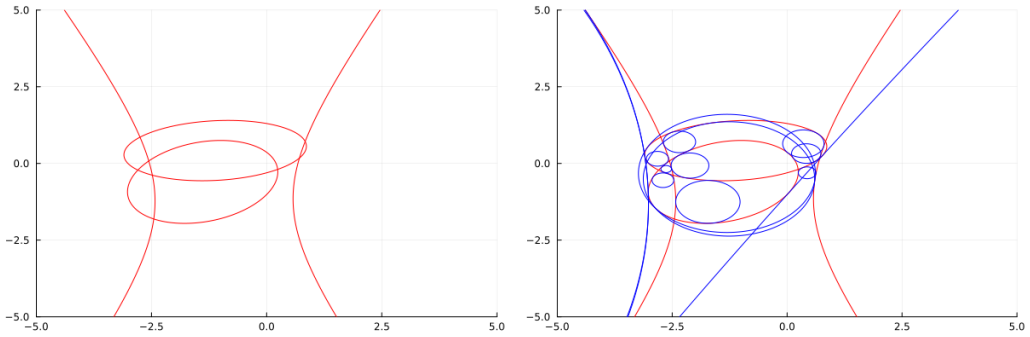


Figure 2.3: Three random conic curves and the 14 circles which are tangent to all of them in $\mathbb{R}^2$.

2.2 Generating Functions

In this section we briefly establish notation and language for working with generating functions, then proceed with several examples of generating functions with direct application to combinatorics. A comprehensive first introduction to generating functions for the purpose of solving enumerative problems is available online at [12]. In this thesis we are not focused on the derivation of generating functions, and in most cases we assume that a generating function has been given in an appropriate form for analysis. However, some examples will be done from scratch for completeness and to demonstrate the power of the methodology.

2.2.1 Formal Series

Let $d \in \mathbb{N}$, let $\mathbb{K}$ be a field, and let

$$A(z_1, z_2, \ldots, z_d) = \sum_{j_1, j_2, \ldots, j_d \geq 0} a_{j_1, j_2, \ldots, j_d} z_1^{j_1} z_2^{j_2} \cdots z_d^{j_d}$$

where $a_{j_1,j_2,\ldots,j_d} \in \mathbb{K}$. We call the summation $A(z_1, z_2, \ldots, z_d)$ a *formal power series* over the field $\mathbb{K}$. For notational convenience we often write $A(\mathbf{z}) = \sum_{\mathbf{j}\in\mathbb{N}^d} a_{\mathbf{j}}\mathbf{z}^{\mathbf{j}}$ to denote multivariate series. The *ring of formal power series* over $\mathbb{K}$ is written

$$K[[\mathbf{z}]] = \left\{ \sum_{\mathbf{j}\in\mathbb{N}^d} a_{\mathbf{j}}\mathbf{z}^{\mathbf{j}} : a_{\mathbf{j}} \in \mathbb{K} \right\},$$

with binary operations $+, -, \times, \div$ defined as usual for convergent series.

In many applications it is not enough to consider only formal power series, as there is often a need for negative powers of variables. Therefore, we construct a larger class of series. The field of univariate *formal Laurent series* over $\mathbb{K}$ is the set

$$\mathbb{K}((z)) = \left\{ \sum_{j\geq N} a_j z^j : N \in \mathbb{Z},\ a_j \in \mathbb{K} \right\}.$$

The field of *multivariate formal Laurent series* is defined iteratively as

$$\mathbb{K}((z_1, z_2, \ldots, z_k)) = \mathbb{K}((z_1, z_2, \ldots, z_{k-1}))((z_k))$$

for all $k \in [d]$. Note that changing the order of the variables changes the elements of the field.

We call these series *formal* because we are unconcerned with the convergence of the series, for example the series $\sum_{n\geq 0} n!z^n$ does not converge for any nonzero z but is still a valid element of $\mathbb{C}[[z]]$. Operations such as addition, multiplication or substitution are done formally in these rings by manipulating the symbols abiding by the distribution and associativity rules. The purpose of a formal series for us is to represent the sequence or multidimensional array of numbers which is induced by its coefficients. When we are studying the coefficients $a_{\mathbf{j}}$ of the formal series $A(\mathbf{z})$ we call $A(\mathbf{z})$ the *generating function* of $a_{\mathbf{j}}$. We employ the standard coefficient extraction notation $[\mathbf{z}^{\mathbf{j}}]A(\mathbf{z}) = a_{\mathbf{j}}$ when working with generating functions. In enumerative combinatorics, the coefficients of generating functions are what is important.

While we think of a generating function as a potentially infinite series, it is often infeasible to write down the coefficients in practice, as this often requires great knowledge of the coefficients. Instead, we strive to find a representation of the generating series through some other object, such as the solution to a functional equation or a power series expansion of an analytic function around the origin. For example, we saw at the start of Chapter 1 the generating series of the Fibonacci sequence was the power series expansion of $\frac{1}{1-z-z^2}$ around $z = 0$. In Section 2.2.2 we will see the Catalan generating function, which is the power series solution to the functional equation $A(z) = 1 + zA(z)^2$. In this

thesis, the focus is on multivariate generating functions which are described as power series expansions of rational functions around the origin.

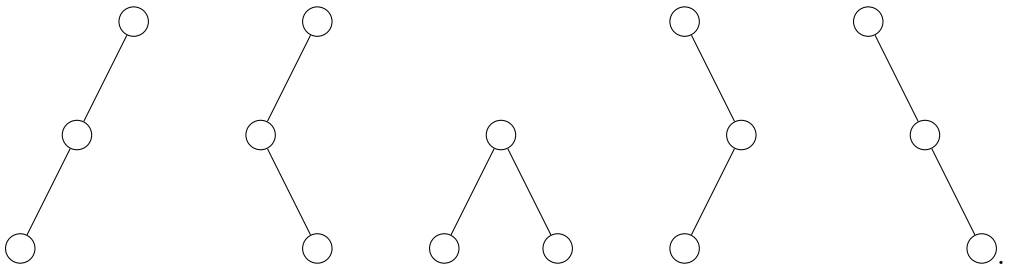
2.2.2 Examples and Constructions of Generating Functions

To motivate the analysis that follows below, we give several examples constructing generating functions. We begin our discussion with a construction of the well-known Catalan generating function.

Example 10 (Catalan generating function)**.** Suppose we wish to enumerate rooted binary trees with n vertices, where we distinguish between left and right children. Let $\mathcal{T}$ denote the set of all such trees. A tree in $\mathcal{T}$ is either empty or has the form (L, R) for two smaller trees L and R in $\mathcal{T}$, for example the empty tree is $\varnothing$, the tree of size 1 is $(\varnothing, \varnothing)$, and trees of size three are

$$(((\varnothing,\varnothing),\varnothing),\varnothing),\ ((\varnothing,(\varnothing,\varnothing)),\varnothing),\ ((\varnothing,\varnothing),(\varnothing,\varnothing)),\ (\varnothing,((\varnothing,\varnothing),\varnothing)),\ (\varnothing,(\varnothing,(\varnothing,\varnothing))).$$

Pictorially, we draw the trees of size three as



The univariate generating function whose coefficients count the number of trees in $\mathcal{T}$ with n vertices is called the Catalan generating function, and satisfies the functional equation $T(z) = 1 + zT(z)^2$. To see this, denote the coefficients of $T(z)$ by t_n, so

$$t_n = \sum_{k=0}^{n-1} t_k t_{n-1-k} \tag{2.1}$$

for $n \geq 1$ since any non-empty tree of size n is a vertex and two smaller trees whose number of vertices sum to $n-1$. Multiplying Equation (2.1) by z^n and summing over all $n \geq 1$, after adding $1 = t_0$ to both sides, we obtain

$$T(z) = 1 + \sum_{n\geq 1}\left(\sum_{k=0}^{n-1} t_k t_{n-1-k}\right) z^n = 1 + z\sum_{n\geq 0}\left(\sum_{k=0}^{n} t_k t_{n-k}\right) z^n = 1 + zT(z)^2,$$

as claimed. From here the analysis continues. We apply the quadratic formula and conclude that $T(z)$ is the power series expansion of $\frac{1+\sqrt{1-4z}}{2z}$ or $\frac{1-\sqrt{1-4z}}{2z}$ at the origin. Expanding both functions around $z = 0$ we have

$$\frac{1-\sqrt{1-4z}}{2z} = 1 + z + 2z^2 + 5z^3 + \cdots$$

and

$$\frac{1+\sqrt{1-4z}}{2z} = \frac{1}{z} - 1 - z - 2z^2 - 5z^3 - \cdots .$$

Notice that the expansion of $\frac{1+\sqrt{1-4z}}{2z}$ is not a power series. Therefore, since there are no rooted binary trees with a negative number of vertices, we conclude that $T(z) = \frac{1-\sqrt{1-4z}}{2z}$ and thus there are $[z^n]\frac{1-\sqrt{1-4z}}{2z}$ rooted binary trees with n vertices.

A generating function $F(z)$ which satisfies $P(F(z), z) = 0$ where $P(y, z)$ is a bivariate polynomial is called an *algebraic generating function*. The Catalan generating function is an example of an algebraic generating function. Extracting asymptotic formulas of univariate algebraic generating functions is automatic in most applications using the analytic combinatorics framework, and the methods for doing this are covered in [7].

While univariate generating functions are interesting and useful, this thesis is focused on multivariate generating functions which are series expansions at the origin of rational functions. For multivariate generating functions, the coefficients form a multidimensional array, rather than a univariate sequence. We are typically interested in univariate sequences, so we introduce the following definition to obtain univariate sequences from multivariate generating functions.

Definition 4. Let $F(\mathbf{z})$ be a multivariate power series with coefficients $f_{\mathbf{j}}$. The *main diagonal* of $F(z)$ is the univariate power series

$$\Delta F(\mathbf{z}) = \sum_{n \geq 0} f_{n\mathbf{r}} z^n.$$

When $\mathbf{r}$ is any vector in $\mathbb{N}^d$, we call

$$\Delta_{\mathbf{r}} F(\mathbf{z}) = \sum_{n \geq 0} f_{n\mathbf{r}} z^n$$

the $\mathbf{r}$-*diagonal* of $F(\mathbf{z})$.

While multivariate generating functions arise naturally when counting combinatorial objects with respect to multiple parameters (such as walks by length and end point or trees by internal vertices and leaves), Definition 4 hints that we may encode univariate sequences

as diagonals of multivariate generating functions. The following theorem demonstrates how a sequence encoded by an algebraic generating function can be encoded as the diagonal of a bivariate rational generating function.

Theorem 2 (Proposition 3.8 of [13])**.** Suppose $f(z) = \sum_{n\geq 0} f_n z^n$ satisfies $f(0) = 0$ and $P(f(z), z) = 0$, where $P(y, z)$ is a polynomial such that $\frac{\partial P}{\partial y}(0, 0) \neq 0$. Then the sequence f_n is the diagonal of the bivariate generating function

$$F(y, z) = \frac{y^2 P_y(y, yz)}{P(y, yz)}.$$

Example 11 (Catalan generating function as a diagonal)**.** The catalan generating function does not satisfy the conditions of Theorem 2, however, this is easily fixed by multiplying by z shifting the coefficients. If $f(z) = zT(z)$ then $f(z)$ satisfies $P(y, z) = y^2 - y + z$ and the conditions of Theorem 2 are satisfied, so

$$f(z) = \Delta\left(\frac{y(2y-1)}{y-1+z}\right).$$

This means we can analyze the bivariate rational function $\frac{y(2y-1)}{y-1+z}$ instead of the univariate functional equation $T(z) = 1 + zT(z)^2$.

Example 12 (non-crossing configurations, Example VII.16 of [7])**.** Non-crossing configurations are graphs where the vertices are ordered in a circle and each edge between vertices is a straight line connection between them such that no pair of edges cross. Figure 2.4 shows 4 different non-crossing configurations. Non-crossing configurations come in various flavours, for example, we may be interested in those non-crossing graphs which are only trees and discard all other configurations. In [7], Flajolet and Sedgewick study the enumeration of non-crossing configurations which are trees, forests, connected graphs and general graphs by number of vertices. In each case, the generating function is univariate and satisfies an algebraic equation. In this example we apply Theorem 2 to build bivariate rational functions which have diagonals counting non-crossing configurations. In future sections, we will analyze these multivariate generating functions.

The enumeration of non-crossing graphs results in the simplest bivariate rational generating function, so we begin by considering the generating function $G(z)$ counting those. The generating function $G(z) = 1 + z + 2z^2 + 8z^3 + \cdots$ satisfies the algebraic equation

$$P(y, z) = y^2 + (2z^2 - 3z - 2)y + 3z + 1.$$

We do not have $G(0) = 0$ which is required for Theorem 2. To resolve this we let $Q(y, z) = P(1 + z + 2z^2 + z^2 y, y)/z^3$ and let $G'(z) = (G(z) - 1 - z - 2z^2)/z^2$, then $G'(0) = 0$ and

$Q_y(0,0) \neq 0$, so Theorem 2 holds. As a result we have

$$G'(z) = \Delta \frac{y^2 P_y(y,yz)}{P(y,yz)} = \Delta \frac{2y^4z^2 - 3y^3z + 2y^3 - 2y^2}{2y^3z^2 - 3y^2z + y^2 + 3yz - 2y + 1}. \tag{2.2}$$

The generating function $T(z)$ of non-crossing configurations which are trees does not satisfy Theorem 2 either, since $T(z) = z + z^2 + 3z^2 + \cdots$ satisfies

$$P(y,z) = y^3 - zy + z^2$$

where $P_y(0,0) = 0$. In general we remedy this by following the method demonstrated by Adamczewski and Bell in [1]. Let $T(z) = a(z) + z^r b(z)$ where a has degree r, $b(0) = 0$ and r is chosen such that the discriminant $D(z)$ of P in y is $D(z) = z^r d(z)$ for some polynomial $d(z)$ where $d(0) \neq 0$. Then the generating function $b(z)$ satisfies the polynomial

$$Q(y,z) = P(a(z) + z^r y, z)/z^{r+1}$$

and $Q_y(y,z) \neq 0$ by construction. The discriminant of P in y is $-27z^4 + 4z^3$, so

$$\begin{aligned} Q(y,z) = {} & y^3z^5 + 9y^2z^5 + 3y^2z^4 + 27yz^5 + 3y^2z^3 + 18yz^4 + 27z^5 \\ & + 21yz^3 + 27z^4 + 6yz^2 + 36z^3 + 3yz + 19z^2 - y + 12z. \end{aligned}$$

The generating function $b(z)$ is thus encoded by

$$b(z) = \Delta \frac{y^2 Q(y,yz)}{Q_y(y,yz)},$$

from which we have $T(z)$ as well. The analysis is similar in the case of the generating functions $C(z)$ enumerating non-crossing connected graphs and $F(z)$ enumerating non-crossing forests. In Chapter 3 we revisit this example and explicitly compute the exponential growth rate of each of the non-crossing configurations.

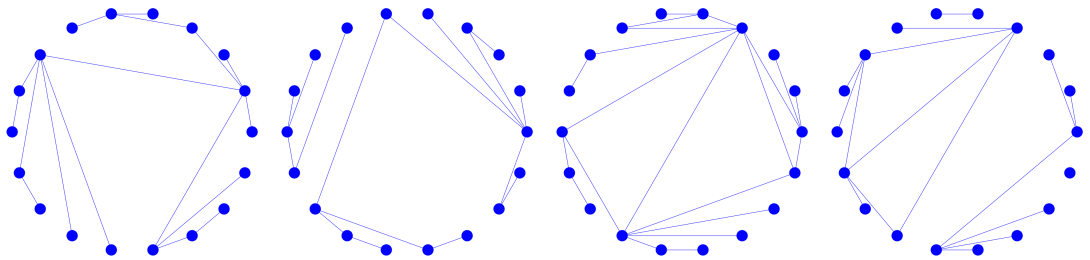


Figure 2.4: Plots of the different types of non-crossing configurations. From left to right: tree, forest, connected graph and graph.

A rich source of multivariate generating functions comes from the enumeration of walks in lattices. There are several variations, each with different levels of difficulty providing a sandbox for exploring the theory of generating functions.

Example 13 (lattice walk enumeration)**.** Suppose we wish to count walks in $\mathbb{Z} \times \mathbb{Z}$ which only take steps from $\mathcal{S} = \{(1,1), (1,-1), (-1,1), (-1,-1)\}$. We denote the steps in the set $\mathcal{S}$ pictorially as $\mathcal{S} = \{\nearrow, \searrow, \nwarrow, \swarrow\}$, and walks are sequences of steps $s = s_1, s_2, \ldots, s_n$ where $s_j \in \mathcal{S}$, see Figure 2.5. If $a(i,j,k)$ is the number of length k walks which end at coordinate $(i,j) \in \mathbb{Z}^2$, the function

$$A(x,y,t) = \sum_{i,j,k} a(i,j,k) x^i y^j t^k \in \mathbb{C}((x,y))[[t]]$$

is the generating function for the number of such walks. Let $S(x,y) = \sum_{(a,b)\in\mathcal{S}} x^a y^b$ and $A_k(x,y) = [t^k]A(x,y)$. Then $A_k(x,y)$ is a polynomial in x, y, x^{-1}, y^{-1} such that

$$A_{k+1}(x,y) = \sum_{(a,b)\in\mathcal{S}} x^a y^b A_k(x,y) = S(x,y) A_k(x,y)$$

for all k. Multiplying by t^{k+1} and summing we find $A(x,y,t) - 1 = S(x,y)A(x,y,t)$. so the generating function is

$$A(x,y,t) = \frac{1}{1 - tS(x,y)} = \frac{1}{1 - t(xy + xy^{-1} + x^{-1}y + x^{-1}y^{-1})}.$$

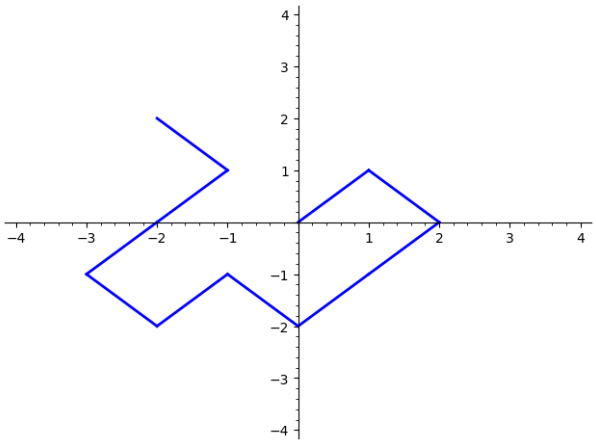


Figure 2.5: The walk $s = \nearrow\searrow\swarrow\swarrow\nwarrow\swarrow\nwarrow\nearrow\nearrow\nwarrow$ plotted in $\mathbb{Z} \times \mathbb{Z}$.

Example 14 (restricted quadrant lattice walk)**.** Let $\mathcal{Q}$ denote the set of all walks in the lattice $\mathbb{N} \times \mathbb{N}$ which take steps from $\mathcal{S} = \{(1,1), (1,-1), (-1,1), (-1,-1)\}$ and begin at

$(0,0)$. Walks in $\mathcal{Q}$ are confined the the north east quadrant of the integer lattice $\mathbb{Z}\times\mathbb{Z}$ and we call such walks *restricted quadrant lattice walks* or *quadrant lattice walks*. The main diagonal of the generating function

$$Q(x,y,t) = -\frac{(x+1)(y+1)}{tx^2y^2+tx^2+ty^2+t-1}$$

counts the number of of quadrant lattice walks in $\mathcal{Q}$ by length. The methods for deriving these generating functions are sophisticated and beyond the scope of this thesis, but the methods generalize to many different steps sets each with different rational generating functions. For example, changing the step set to $\{\downarrow, \nearrow, \nwarrow\}$, yields the generating function

$$-\frac{(xy^2-x^2-1)(x+1)}{(txy^2+tx^2+t-1)(x^2+1)(y-1)}.$$

Chapter 4 of [13] covers much of the methods used to construct the generating functions enumerating lattice walks which we will analyze in Chapter 3.

2.3 Analytic Combinatorics

We begin this section by briefly reviewing the techniques for generating functions of a single variable. Following the univariate analysis, we illustrate the differences that arise when considering rational multivariate generating functions before discussing the methods of ACSV and applying them to an explicit example. The goal of our analysis will be to obtain an asymptotic formula for a sequence of numbers a_n, so we begin with the following definition.

Definition 5. Let a_n define a sequence of numbers in $\mathbb{C}$. We say that a_n is *asymptotically equal* to $f(n)$, and write $a_n \sim f(n)$, if

$$\lim_{n\to\infty} \frac{a_n}{f(n)} = 1.$$

When $a_n \sim f(n)$ the function $f(n)$ is useful as an approximation for a_n, which gets more accurate as n grows. We also make use of big-O notation, writing $|a_n| = O(f(n))$ if there exists $N > 0$ and $C \in \mathbb{R}_{>0}$ such that $|a_n| \leq C|f(n)|$ for all $n \geq N$.

2.3.1 The Method in One Variable

It is helpful to understand the process in one variable before attempting to attack the problem in the multivariate setting. Consider the following combinatorial object.

Definition 6 (Alternating Permutation)**.** A permutation $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$ is said to be *alternating* if

$$\sigma_1 < \sigma_2,\ \sigma_2 > \sigma_3,\ \sigma_3 < \sigma_4,\ \sigma_4 > \sigma_5,\ \ldots.$$

Consider the function $\sec(z)$ and its power series expansion around the origin,

$$\sec(z) = \sum_{n\geq 0} s_n z^n = \sum_{n\geq 0} \frac{a_n}{n!} z^n.$$

It is known that the coefficients a_n count even length alternating permutations [17]. The idea of analytic combinatorics is the characterization of the relationship between the singularities of $\sec(z)$ and the asymptotic behavior of the coefficients s_n. Leveraging this relationship, we arrive at an explicit asymptotic formula for the number of alternating permutations a_n. In order to proceed with the analysis, we recall the following essential results from complex analysis. Let D be an open connected subset of $\mathbb{C}$.

Theorem 3 (Cauchy Integral Formula)**.** If $F(z) = \sum_{n\geq 0} f_n z^n$ is analytic in D then, for a closed curve γ inside D containing the origin, we have

$$f_n = \frac{1}{2\pi i} \oint_\gamma \frac{F(z)}{z^{n+1}} dz. \tag{2.3}$$

Theorem 4 (Cauchy Residue Theorem)**.** If $F(z)$ is meromorphic on D and γ is a simple closed curve inside D then

$$\frac{1}{2\pi i} \oint_\gamma F(z) dz = \sum_\rho \operatorname*{Res}_{z=\rho} F(z),$$

where the summation is over the singularities ρ of $F(z)$ which are inside γ and $\operatorname{Res}_{z=\rho} F(z)$ denotes the complex residue.

The function $\sec(z) = \frac{1}{\cos(z)}$ is analytic within the disc $D = \{z \in \mathbb{C} : |z| < \frac{\pi}{2}\}$, therefore by Theorem 3, we have

$$s_n = I_n = \frac{1}{2\pi i} \oint_\gamma \frac{\sec(z)}{z^{n+1}} dz$$

where $\gamma = \{z \in \mathbb{C} : |z| = 1\}$. Notice that by Theorem 4 and the fact that $\sec(z)$ is analytic inside γ, we know that $I_n = \operatorname*{Res}_{z=0} \frac{\sec(z)}{z^{n+1}}$. The next step in the analysis is to compute the integral I_n. We proceed by introducing the integral

$$I'_n = \frac{1}{2\pi i} \oint_{\gamma'} \frac{\sec(z)}{z^{n+1}} dz,$$

where $\gamma' = \{z \in \mathbb{C} : |z| = 2\}$. Note that the only difference between I_n and I'_n is that the domain of integration has larger radius. Let $M = \max\{|\sec(z)| : z \in \gamma'\}$ and observe that

$$|I'_n| = \left| \frac{1}{2\pi i} \oint_{\gamma'} \frac{\sec(z)}{z^{n+1}} dz \right| \leq \frac{\text{length}(\gamma')}{2\pi} \cdot \frac{M}{2^{n+1}} = \frac{M}{2^n} = O(2^{-n}).$$

Moreover, by Theorem 4 and the fact that $\sec(z)$ has precisely two singularities inside γ' at $\pm\frac{\pi}{2}$, we have

$$I'_n = \underbrace{\operatorname*{Res}_{z=0} \frac{\sec(z)}{z^{n+1}}}_{I_n} + \operatorname*{Res}_{z=\pi/2} \frac{\sec(z)}{z^{n+1}} + \operatorname*{Res}_{z=-\pi/2} \frac{\sec(z)}{z^{n+1}}.$$

The residues of $\frac{\sec(z)}{z^{n+1}}$ at $\frac{\pi}{2}$ and $-\frac{\pi}{2}$ are $-(-\pi/2)^{-n+1}$ and $(\pi/2)^{-n+1}$. It follows by taking absolute values that

$$|I'_n| = \left| I_n - (2/\pi)^{n+1} + (-2/\pi)^{n+1} \right| = O(2^{-n}).$$

Thus

$$s_n \sim (2/\pi)^{n+1} - (-2/\pi)^{n+1}$$

and the number of alternating permutations is given by $a_n \sim n!\,((2/\pi)^{n+1} - (-2/\pi)^{n+1})$. Let $f_n = n!((2/\pi)^{n+1} - (-2/\pi)^{n+1})$, Table 2.1 shows the approximation to 5 significant figures for $n \leq 20$.

n	a_n	f_n	a_n/f_n
0	1.00000e+0	1.27324e+0	7.85398e-1
2	1.00000e+0	1.03205e+0	9.68946e-1
4	5.00000e+0	5.01928e+0	9.96158e-1
6	6.10000e+1	6.10272e+1	9.99555e-1
8	1.38500e+3	1.38507e+3	9.99950e-1
10	5.05210e+4	5.05213e+4	9.99994e-1
12	2.70276e+6	2.70277e+6	9.99999e-1
14	1.99361e+8	1.99361e+8	1.00000e+0
16	1.93915e+10	1.93915e+10	1.00000e+0
18	2.40488e+12	2.40488e+12	1.00000e+0
20	3.70371e+14	3.70371e+14	1.00000e+0

Table 2.1: The asymptotic formula f_n converges to the actual number a_n of alternating permutations as $n \to \infty$.

The process above generally applies to a wide range of meromorphic functions that are analytic around the origin. This illuminates an intimate connection between the location of singularities of a meromorphic function and the asymptotic growth rates of the coefficients

of its power series expansion around the origin. For large n, the integrand on the right hand side of (2.3) gets smaller as the curve γ is pushed further away from the origin. However, as γ is pushed beyond singular points of $F(z)$ it picks up residues at those singular points. Those residues are then used to approximate the coefficients f_n.

The approach when considering generating functions of several variables attempts to mirror that of the approach in a single variable. There is, however, a significant difference between the singularities of multivariate functions and univariate functions which will ultimately complicate the method: even bivariate generating functions have infinitely large sets of singularities.

2.3.2 The Method in Several Variables

In this thesis we restrict to the case where the generating function is expressed as the power series expansion at the origin of a rational function, as we would like to apply the techniques of computing with multivariate polynomials to analytic combinatorics. Let

$$F(\mathbf{z}) = \frac{G(\mathbf{z})}{H(\mathbf{z})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} f_{\mathbf{n}}\mathbf{z}^{\mathbf{n}}$$

denote a multivariate rational function in d variables over $\mathbb{C}$ with a power series expansion around the origin. As we have seen, there are many different sequences which are encoded in the coefficients of $F(\mathbf{z})$. Fix a direction $\mathbf{r} \in \mathbb{N}^d$ and study the $\mathbf{r}$-diagonal of $F(\mathbf{z})$, however in our examples, and in practice, the most important diagonal is the main diagonal. Our main example for illustrating the methods is the rational generating function $F(x, y) = \frac{1}{1-x-y}$, with power series expansion

$$F(x, y) = \frac{1}{1-(x+y)} = \sum_{i,j\geq 0} \binom{i+j}{j} x^i y^j.$$

The main diagonal of $F(x, y)$ is the sequence of central binomial coefficients $f_{n,n} = \binom{2n}{n}$. As in the univariate case, the objective of ACSV is to extract asymptotic formulas for sequences encoded by generating functions. This is done by applying the theory of multivariate complex analysis to the generating function $F(\mathbf{z})$. We begin by reviewing the language, definitions and theorems from complex analysis in several variables. A thorough and introductory review of the necessary background is given in Chapter 3 of [13].

Given $\mathbf{a} \in \mathbb{C}^d$ and $\mathbf{r} \in \mathbb{R}^d_{>0}$, the set

$$D_{\mathbf{a}}(\mathbf{r}) = \left\{\mathbf{z} \in \mathbb{C}^d : |z_j - a_j| < r_j,\ j \in [d]\right\}$$

is the open *polydisk* of radius $\mathbf{r}$ centered at $\mathbf{a}$. The closure of $D_{\mathbf{a}}(\mathbf{r})$ is

$$\overline{D_{\mathbf{a}}(\mathbf{r})} = \left\{\mathbf{z} \in \mathbb{C}^d : |z_j - a_j| \leq r_j,\ j \in [d]\right\},$$

and the set

$$T_{\mathbf{a}}(\mathbf{r}) = \left\{\mathbf{z} \in \mathbb{C}^d : |z_j - a_j| = r_j,\ j \in [d]\right\}$$

is the *polytorus* of radius $\mathbf{r}$ centered at $\mathbf{a}$. For polydisks and polytori centered at the origin, we omit the subscript $\mathbf{a}$ since this is the case we consider most often.

A series $\sum_{n\geq 0} c_n$ is said to be *absolutely convergent* if $\sum_{n\geq 0} |c_n|$ converges, and a function $F(\mathbf{z})$ is *analytic* at $\mathbf{a} \in \mathbb{C}^d$ if there exists $\mathbf{r} \in \mathbb{R}^d_{>0}$ such that

$$F(\mathbf{z}) = \sum_{\mathbf{n}\in\mathbb{N}^d} f_{\mathbf{n}}(\mathbf{z}-\mathbf{a})^{\mathbf{n}}$$

is absolutely convergent for each $\mathbf{z} \in D_{\mathbf{a}}(\mathbf{r})$. The set $\mathcal{D}$ containing the points $\mathbf{z} \in \mathbb{C}^d$ such that the series $\sum_{\mathbf{n}\in\mathbb{N}^d} f_{\mathbf{n}}(\mathbf{z}-\mathbf{a})^{\mathbf{n}}$ converges absolutely is the *domain of convergence* of F. Notice that polynomials are analytic at every point in $\mathbb{C}^d$ since the summation is finite. Moreover, rational functions $G(\mathbf{z})/H(\mathbf{z})$ are analytic at points $\mathbf{a} \in \mathbb{C}^d$ such that $H(\mathbf{z}) \neq 0$, since we can recover the coefficients iteratively from the equation

$$G(\mathbf{z}) = H(\mathbf{z}) \sum_{\mathbf{n}\in\mathbb{N}^d} f_{\mathbf{n}}(\mathbf{z}-\mathbf{a})^{\mathbf{n}}.$$

Given a rational function $F(\mathbf{z}) = G(\mathbf{z})/H(\mathbf{z})$ where $G(\mathbf{z})$ and $H(\mathbf{z})$ are coprime, the *singular set* of $F(\mathbf{z})$ is the set $\mathcal{V}(H)$. Elements of $\mathcal{V}(H)$ are called *singularities* of $F(\mathbf{z})$. A key difference between univariate and multivariate complex analysis is the singular behavior of functions which are not analytic on all of $\mathbb{C}$. In one variable, the singular sets of rational functions are finite and consist of the isolated zeros of univariate polynomials. Moving to several variables, the singular sets contain positive dimensional algebraic sets. The following theorem, which generalizes the univariate Cauchy integral formula, allows us to compute asymptotic formulas for multivariate generating functions.

Theorem 5 (multivariate Cauchy integral formula: Theorem 3.1 of [13])**.** If $F(\mathbf{z})$ is analytic on a connected open subset $\Omega \subseteq \mathbb{C}^d$ and $\overline{D_{\mathbf{a}}(\mathbf{r})} \subseteq \Omega$ such that

$$F(\mathbf{z}) = \sum_{\mathbf{n}\in\mathbb{N}^d} f_{\mathbf{n}}(\mathbf{z}-\mathbf{a})^{\mathbf{n}}$$

in $D_{\mathbf{a}}(\mathbf{r})$ then for all $\mathbf{n} \in \mathbb{N}^d$ we have

$$f_{\mathbf{n}} = \frac{1}{(2\pi i)^d} \int_{T_{\mathbf{a}}(\mathbf{r})} \frac{F(\mathbf{z})}{(\mathbf{z}-\mathbf{a})^{\mathbf{n}+\mathbf{1}}} d\mathbf{z},$$

where $\mathbf{1} \in \mathbb{C}^d$ is the all ones vector.

We now return to the rational generating function $F(x,y) = \frac{1}{1-x-y}$, whose singular points are the points in $\mathbb{C}^d$ defined by $x+y=1$. There are three different Laurent series expansions of $F(x,y)$ centered at the origin:

- $F(x,y) = \dfrac{1}{1-(x+y)} = \displaystyle\sum_{i,j\geq 0} \binom{i+j}{j} x^i y^j$
- $F(x,y) = \dfrac{-1/x}{1-(1-y)/x} = \displaystyle\sum_{i,j\geq 0} \binom{i}{j} (-1)^{j+1} y^j x^{-i-1}$
- $F(x,y) = \dfrac{-1/y}{1-(1-x)/y} = \displaystyle\sum_{i,j\geq 0} \binom{i}{j} (-1)^{j+1} x^j y^{-i-1}.$

Each expansion is centered at the origin, however each expansion has a different domain of convergence. The power series expansion has domain of convergence $|x|+|y|<1$, therefore Theorem 5 implies

$$\binom{2n}{n} = I_n = \frac{1}{(2\pi i)^2} \int_{T(1/4,1/4)} \frac{F(x,y)}{x^{n+1}y^{n+1}} dxdy. \tag{2.4}$$

Chapter 5 of [13] illustrates the process of manually deforming the domain of integration and approximating the integral (2.4) explicitly to extract a asymptotic formula for $\binom{2n}{n}$. The method mirrors that of the univariate analysis in that the domain of integration is expanded past the singular points of $F(x,y)$ then a residue calculation is made to approximate the value of I_n. The result in this case is that $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$. The remaining expansions of $\frac{1}{1-x-y}$ require additional tools to analyze. Here we restrict to analyzing only the power series expansions of rational generating functions, as these are most often the expansions with combinatorial significance.

Let $F(\mathbf{z}) = G(\mathbf{z})/H(\mathbf{z})$ with coprime polynomials $G(\mathbf{z})$ and $H(\mathbf{z})$, and fix a diagonal vector $\mathbf{r} \in \mathbb{N}^d$. Further suppose that $F(\mathbf{z})$ admits a power series expansion at the origin.

Definition 7 (critical points)**.** The solutions $\mathbf{w} \in \mathbb{C}_*^d$ to the system of equations

$$\begin{aligned} H(\mathbf{w}) &= 0 \\ z_j \frac{\partial H}{\partial z_j}(\mathbf{w}) &= r_j \lambda, \quad j \in [d] \end{aligned} \tag{2.5}$$

are called the *critical points* with respect to the function $F(\mathbf{z})$ and direction $\mathbf{r}$. When $\frac{\partial H}{\partial z_j}(\mathbf{w}) \neq 0$ for some j, and thus all j when $\mathbf{r}$ has no zero coordinate, the point $\mathbf{w}$ is a

smooth critical point. The variable λ in the critical point equations is used in the formulas for asymptotics, however when the critical points are smooth it is possible to eliminate λ from the equations so that there is one less equation and variable by instead considering the system

$$H(\mathbf{w}) = 0$$
$$r_j z_1 \frac{\partial H}{\partial z_1}(\mathbf{w}) - r_1 z_j \frac{\partial H}{\partial z_j}(\mathbf{w}) = 0, \quad j \in \{2, 3, \ldots, d\}.$$

At times we will solve this system when we do not need the value λ.

Definition 8 (minimal points)**.** Suppose $\mathcal{D} \subseteq \mathbb{C}^d$ is the domain of convergence of the power series expansion of $F(\mathbf{z})$ centered at the origin. A point $\mathbf{w} \in \mathcal{D} \cap \mathcal{V}(H)$ is called a *minimal point* of $F(\mathbf{z})$ with respect to the power series expansion. Equivalently, a minimal point $\mathbf{w} \in \mathcal{V}(H)$ is a point such that there does not exist a point $\mathbf{z} \in \mathcal{V}(H)$ with $|z_j| < |w_j|$ for all $j \in [d]$. Furthermore, if $T(\mathbf{w}) \cap \mathcal{V}(H)$ is finite then we say that $\mathbf{w}$ is *finitely minimal.* If $T(\mathbf{w}) \cap \mathcal{V}(H) = \{w\}$ then we say that $\mathbf{w}$ is *strictly minimal.*

Theorem 6 (smooth multivariate asymptotics; Theorem 5.2 of An Invitation to Analytic Combinatorics in Several Variables, Melczer 2021)**.** Let

$$F(\mathbf{z}) = G(\mathbf{z})/H(\mathbf{z}) = \sum_{\mathbf{n} \in \mathbb{N}^d} f_{\mathbf{n}} \mathbf{z}^{\mathbf{n}}$$

be a rational generating function and suppose $\mathbf{w}$ is a nondegenerate strictly minimal smooth critical point $\mathbf{w}$ in the direction $\mathbf{r}$ such that $\frac{\partial}{\partial z_d} H(\mathbf{w}) \neq 0$. Then

$$f_{n\mathbf{r}} = \mathbf{w}^{-n\mathbf{r}} \frac{(2\pi n)^{(1-d)/2}}{\sqrt{\det(r_d \mathcal{H})}} \left(C_0 + O(n^{-1})\right) \tag{2.6}$$

where $C_0 = \frac{-G(\mathbf{w})}{w_d H_{z_d}(\mathbf{w})}$ and $\mathcal{H}$ is the $(d-1) \times (d-1)$ matrix

$$\mathcal{H}_{ij} = \begin{cases} V_i V_j + U_{ij} - V_j U_{id} - V_i U_{jd} + V_i V_j U_{dd} & i \neq j \\ V_i + V_i^2 + U_{ii} - 2V_i U_{id} - V_i U_{jd} + V_i^2 U_{dd} & i = j \end{cases}$$

where

$$U_{ij} = \frac{w_i w_j H_{z_i z_j}(\mathbf{w})}{H_{z_d}(\mathbf{w})} \text{ and } V_i = \frac{w_i H_{z_i}(\mathbf{w})}{w_d H_{z_d}(\mathbf{w})} = \frac{r_i}{r_d}$$

and the point $\mathbf{w}$ is nondegenerate if and only if $\det \mathcal{H} \neq 0$.

Theorem 6 is a powerful tool for analysing power series expansions of multivariate rational generating functions. When $\mathbf{w}$ is finitely minimal and each point in $T(\mathbf{w}) \cap \mathcal{V}(H)$ satisfies the conditions, asymptotics are obtained by summing (2.6) over all points in

$T(\mathbf{w}) \cap \mathcal{V}(H)$. Each component of Theorem 6 is explicit except for the value of $\mathbf{w}$ and verification of minimality. Therefore, the problem of computing minimal critical points explicitly is of great importance when working with multivariate generating functions. In the later chapters, we demonstrate algorithms which offer a solution this problem.

Example 15. We conclude the section by applying Theorem 6 to $F(x, y) = \frac{1}{1-x-y}$ to compute an asymptotic formula for the central binomial coefficients. On the main diagonal, the second set of critical equations from Definition 7 are $1 = x + y$ and $x = y$ from which we deduce there is exactly one critical point at $(1/2, 1/2)$. Moreover, this critical point is smooth since

$$-1/2 = H_x(1/2, 1/2) = H_y(1/2, 1/2) \neq 0.$$

The domain of convergence $\mathcal{D}$ of the power series expansion of $F(x, y)$ is $|x| + |y| \leq 1$, therefore the point $(1/2, 1/2) \in \mathcal{D}$ is also a minimal point. The 1×1 matrix $\mathcal{H}$ in Theorem 6 is $[2]$, so $(1/2, 1/2)$ is non-degenerate. The constant C_0 is 1 and therefore we have the well known asymptotic formula

$$\binom{2n}{n} = (1/2)^{-n}(1/2)^{-n}\frac{(2\pi n)^{-1/2}}{\sqrt{2}}(1 + O(1/n)) = \frac{4^n}{\sqrt{\pi n}}(1 + O(1/n)).$$

n	$\binom{2n}{n}$	$\frac{4^n}{\sqrt{\pi n}}$	$\binom{2n}{n}/\frac{4^n}{\sqrt{\pi n}}$
1	2.00000e+00	2.25676e+0	8.86227e-1
2	6.00000e+00	6.38308e+0	9.39986e-1
3	2.00000e+01	2.08470e+1	9.59369e-1
4	7.00000e+01	7.22163e+1	9.69311e-1
5	2.52000e+02	2.58369e+2	9.75350e-1
6	9.24000e+02	9.43429e+2	9.79406e-1
7	3.43200e+03	3.49378e+3	9.82316e-1
8	1.28700e+04	1.30725e+4	9.84506e-1
9	4.86200e+04	4.92996e+4	9.86214e-1
10	1.84756e+05	1.87079e+5	9.87583e-1
11	7.05432e+05	7.13491e+5	9.88705e-1
12	2.70416e+06	2.73246e+6	9.89640e-1
13	1.04006e+07	1.05011e+7	9.90433e-1
14	4.01166e+07	4.04763e+7	9.91113e-1
15	1.55118e+08	1.56415e+8	9.91703e-1
16	6.01080e+08	6.05794e+8	9.92219e-1
17	2.33361e+09	2.35083e+9	9.92675e-1
18	9.07514e+09	9.13837e+9	9.93080e-1
19	3.53453e+10	3.55785e+10	9.93443e-1
20	1.37847e+11	1.38711e+11	9.93770e-1

Table 2.2: The asymptotic formula converges to $\binom{2n}{n}$ with error $O(1/n)$

Chapter 3

Algorithms for Analytic Combinatorics in Several Variables

In this chapter we survey algorithms for ACSV. We begin with an application of the homotopy continuation method discussed in Section 2.1 to the problem of computing minimal critical points of rational generating functions. The primary goal is to develop an algorithm that takes a rational generating function specified by its numerator and denominator, and a direction vector, and outputs the asymptotic formula obtained by an application of Theorem 6 if and when it applies. Section 3.1 builds on the work of Melzcer and Salvy in [14] to achieve this. Next, we introduce the "height function" of a generating function with respect to a direction. Understanding the geometry of the singular variety under the height function is important in ACSV because it is connected to deformations of the Cauchy integral formula for coefficients. In Section 3.2 we demonstrate the process of computing a flow on the height function. In Section 3.3 we discuss the implementation details of an alternative algorithm for computing critical points determining asymptotics that leverages the geometry of the height function for bivariate rational generating functions.

Partial code examples are given throughout and links to complete implementations are provided to the interested reader. The examples demonstrate cases when the algorithms are successful and also illustrate what shortcomings are present. We omit some code from examples to avoid redundancy and encourage the interested reader to see the full code available on GitHub at

https://github.com/JSmol/acsv-algorithms.

Below we refer to notebooks available inside the repository.

3.1 Numerically Computing Minimal Points

In this section, we demonstrate a method for computing minimal critical points in order to apply Theorem 6. To accomplish this, we encode the minimal points of a rational generating function as the solutions of a zero dimensional polynomial system, then solve the system using the numerical methods discussed in Chapter 2. We provide an implementation of this in the Julia programming language built on the `HomotopyContinuation.jl` [4] library, which provides methods for solving polynomial systems using homotopy continuation techniques.

Let $F(\mathbf{z}) = G(\mathbf{z})/H(\mathbf{z})$ be a d-variate rational generating function with a power series expansion around the origin and fix a vector $\mathbf{r} \in \mathbb{N}^d$. The most expensive computation required to apply Theorem 6 is typically finding the minimal critical points of $F(\mathbf{z})$. Following the work of Melczer and Salvy in [14], we begin by considering a special type of rational generating function that simplifies the method before moving to a more general class.

3.1.1 The Combinatorial Case

A power series is called *combinatorial* if only finitely many of its coefficients are negative. If a rational generating function F has a combinatorial power series expansion, then we say F is *combinatorial*. Generating functions which are obtained through combinatorial processes are often combinatorial and combinatorial generating functions have minimal critical points that are easier to compute. This allows for a simpler algorithm when computing minimal points.

Suppose that $F(\mathbf{z})$ is a combinatorial rational function, let $\mathbf{w}$ be a critical point and consider the equation

$$H(t\mathbf{z}) = 0 \tag{3.1}$$

in the complex variables $\mathbf{z}$ and real variable t. Lemma 21 of [14] shows that a critical point $\mathbf{w} \in \mathcal{V}(H)$ is minimal if and only if $\mathbf{w}' = (|w_1|, \ldots, |w_d|)$ is a minimal critical point. The resulting system is square in the variables $\mathbf{w}, \lambda, t$, and is generically zero dimensional [13]. To check whether a known critical point $\mathbf{w} = (w_1, \ldots, w_d)$ is minimal, we can check if there is a solution $(|w_1|, \ldots, |w_d|, \lambda, t)$ to Equations (2.5) and (3.1) where $t \in (0, 1)$.

This introduces two complications which arise when considering numerical solutions. The first problem comes from verifying $(\mathbf{w}', \lambda, t) = (w_1', \ldots, w_d', \lambda, t)$ is a solution to equations (2.5) and (3.1) where $\mathbf{w}' = (|w_1|, \ldots, |w_d|)$ and $\mathbf{w} = (w_1, \ldots, w_d)$ is a critical point. Since the homotopy method produces open intervals containing solutions, rather than solutions themselves, it is difficult to verify that $\mathbf{z} = \mathbf{w}$ for vectors of algebraic numbers. In theory this is resolved with *root separation bounds* for polynomials. A root separation

bound is a bound m which has the property that $|\alpha - \beta| > m$ for all distinct pairs of roots α, β of a polynomial. Unfortunately, the bounds which are available are exponential in the degrees of the polynomials involved. The arguments presented in [14] show that an interval width of $2^{-O(D^3 \log^k(D^3))}$ where $D = \deg(H)^d$ and k is some natural number is sufficient to rigorously verify $\mathbf{w}' = (|w_1|, \ldots, |w_d|)$. The exponent in this bound is large, and therefore computing the roots to this precision is the most expensive operation of computing the minimal points in the combinatorial case.

The second problem comes from verifying when a solution of (3.1) has $t = 1$ exactly. Given any interval around 1 containing a solution, it is not possible to determine whether the solution is equal to 1 without external knowledge. To resolve this, we extend the system by adding the additional equation $s(1-t)-1$ with a new variable s. This equation removes the solutions where $t = 1$ exactly while all other solutions remain.

We now describe the method for computing minimal points in the combinatorial case. Define the *combinatorial case system* to be

$$\begin{aligned} 0 &= H(\mathbf{w}) \\ 0 &= w_j H_{w_j}(\mathbf{w}) - r_j \lambda, \quad \forall j \in [d] \\ 0 &= H(t\mathbf{w}) \\ 0 &= s(1-t) - 1. \end{aligned} \tag{3.2}$$

Section 3 of [14] gives the following sufficient conditions under which the combinatorial case system correctly identifies minimal critical points, and dominant asymptotics can be effectively computed:

(A0) $F(\mathbf{z}) = G(\mathbf{z})/H(\mathbf{z})$ admits at least one minimal critical point,

(A1) $\nabla H(\mathbf{z})$ does not vanish at the minimal critical points,

(A2) $G(\mathbf{z})$ is nonzero at at least one minimal point,

(A3) all minimal points are non-degenerate,

(J1) the Jacobian matrix of the combinatorial case system is non-singular at its solutions.

Suppose $F = G/H$ satisfies (A0), (A1), (A2), (A3), (J1) and let S be the set of solution to (3.2). Let $\mathbf{w}' \in \mathbb{R}^d_{>0}$ be a critical point such that for all solutions $(\mathbf{w}', \lambda, t) \in S$ we have $t \notin (0, 1)$ and further suppose that $\mathbf{w}'$ is the only such point. Then $\mathbf{w}$ is a minimal critical point determining the asymptotics of the $\mathbf{r}$-diagonal if $\mathbf{w}$ is a critical point and $|w_j| = w'_j$ for all $j \in [d]$. We conclude this subsection with an example applying the combinatorial-case method.

Example 16. Let $F(x,y) = \frac{1}{1-x-y}$. The first step of computing the minimal points of $F(x,y)$ using the combinatorial case method is to solve the critical point equations. This is done in Julia with the following code.

```
@polyvar x y λ s t
H = 1 - x - y
sys = System(
    [H, x*differentiate(H, x) - y*differentiate(H, y)],
    variables=[x, y]
)
result = solve(sys)
display(map(sol -> round.(sol; digits=3), solutions(result)))
```

```
1-element Vector{Vector{ComplexF64}}:
 [0.5 + 0.0im, 0.5 + 0.0im]
```

There is exactly one critical point at $(1/2, 1/2)$, as expected. The following code solves the full combinatorial case system and displays the real solutions.

```
@polyvar λ t s
sys = System([ # full comb-case system
    H;
    [x, y] .* differentiate(H, [x, y]) .- λ;
    H([x, y] => t*[x, y]);
    s*(1-t) - 1
], variables=[t; x; y; λ; s])
real_sols = real_solutions(solve(sys; show_progress=false))
```

```
Vector{Float64}[]
```

We see there is no real solutions, and hence none with $t \in (0,1)$, thus verifying that the point $(1/2, 1/2)$ is minimal. Later we will see an example with multiple critical points where System (3.2) has non-trivial solutions that require further analysis to determine minimality.

3.1.2 The Non-Combinatorial Case

There are symbolic-numeric approaches to asymptotics in the combinatorial case demonstrated in [9], but they are not efficient enough to handle non-combinatorial functions. This is a problem as verifying that an arbitrary rational generating function $F(\mathbf{z})$ is combinatorial is difficult (perhaps undecidable) in general. If we do not know that the rational function $F(\mathbf{z})$ is combinatorial, we cannot verify minimality just by looking at points with positive coordinates. Fortunately, it is still possible to construct a square polynomial system whose solutions identify the minimal points of $F(\mathbf{z})$.

We begin by decomposing into real variables. Let $\mathbf{w} = \mathbf{a} + i\mathbf{b}$ and split $H(\mathbf{w})$ so that

$$H(\mathbf{w}) = H(\mathbf{a} + i\mathbf{b}) = u(\mathbf{a}, \mathbf{b}) + iv(\mathbf{a}, \mathbf{b})$$

for real valued functions u and v. The critical point equations (2.5) in the variables $\mathbf{w}, \lambda$ become

$$\begin{aligned} 0 &= u(\mathbf{a}, \mathbf{b}) \\ 0 &= v(\mathbf{a}, \mathbf{b}) \\ r_j \lambda_R &= a_j \frac{\partial u}{\partial a_j} + b_j \frac{\partial u}{\partial b_j}, \quad j \in [d] \\ r_j \lambda_I &= a_j \frac{\partial v}{\partial a_j} + b_j \frac{\partial v}{\partial b_j}, \quad j \in [d] \end{aligned} \tag{3.3}$$

in the variables $\mathbf{a}, \mathbf{b}, \lambda_R, \lambda_I$. Section 3.3 of [14] notes that $\mathbf{a} + i\mathbf{b}$ is minimal if the system

$$\begin{aligned} 0 &= u(\mathbf{x}, \mathbf{y}) \\ 0 &= v(\mathbf{x}, \mathbf{y}) \\ 0 &= x_j^2 + y_j^2 - t(a_j^2 + b_j^2), \quad j \in [d] \end{aligned} \tag{3.4}$$

does not have a real solution $(\mathbf{x}, \mathbf{y}, t)$ where $0 < t < 1$. Together, Equations (3.3) and (3.4) form a polynomial system of $3d + 4$ equations in the $4d + 3$ variables $\mathbf{a}, \mathbf{b}, \lambda_R, \lambda_I, \mathbf{x}, \mathbf{y}, t$. The resulting system is not square, however under the assumption:

(J2) the Jacobian matrix of the system (3.3), (3.4) is non-singular at its solutions,

Melczer and Salvy prove in [14] that the real solutions of equations (3.3) and (3.4) satisfy the additional equations

$$(\nu_1 y_j - \nu_2 x_j) \frac{\partial u}{\partial x_j}(\mathbf{x}, \mathbf{y}) - (\nu_1 x_j + \nu_2 y_j) \frac{\partial u}{\partial y_j}(\mathbf{x}, \mathbf{y})$$

for $1 \leq j \leq d$ with at least one of ν_1 or ν_2 nonzero[1]. Therefore we extend the system of Equations (3.3) and (3.4) to a square system depending on whether ν_1 is zero by adding one of the following sets of d (or $d-1$) equations

$$\begin{aligned} 0 &= (y_j - \nu x_j) \frac{\partial u}{\partial x_j}(\mathbf{x}, \mathbf{y}) - (x_j + \nu y_j) \frac{\partial u}{\partial y_j}(\mathbf{x}, \mathbf{y}), \ j \in [d] \quad && \nu_1 \neq 0 \\ 0 &= -x_j \frac{\partial u}{\partial x_j}(\mathbf{x}, \mathbf{y}) - y_j \frac{\partial u}{\partial y_j}(\mathbf{x}, \mathbf{y}), \ j \in [d-1] && \nu_1 = 0 \end{aligned} \tag{3.5}$$

and in each case the result is a square polynomial system in 1 (or 0) extra variables. We call the systems built from equations (3.3), (3.4), (3.5) and $s(1-t)-1$ the *extended critical point systems*.

[1] Melczer and Salvy incorrectly state in [14] that ν_1 and ν_2 must both be nonzero: at least one is non-zero at the solutions of interest, but the other may vanish [11].

The extended critical point systems are large with many variables and the computation of solutions is expensive in general. The following heuristic, which replaces the large systems by two smaller systems where we substitute the variables $\mathbf{a}, \mathbf{b}, \lambda_R, \lambda_I$ by floating point approximations to the critical points of F, significantly improves the running time of the computation and we make use of it in our examples. For each approximation $\mathbf{w}$ to a critical point, we construct a system of equations by substituting $\mathbf{a} = \Re(w)$ and $\mathbf{b} = \Im(w)$ in each extended system. The result is square systems of size $2d + 4$ and $2d + 3$ for each critical point. In practice, this heuristic performs much faster than solving the large system outright and still correctly identifies minimal points. It is the case that replacing the critical equations by numerical approximations to critical points introduces issues pertaining to correctness of results, however this currently the only method that can practically study non-combinatorial generating functions in more than two variables.

Example 17. We conclude this subsection by solving for the minimal points of a bivariate generating function with denominator

$$H(w_1, w_2) = (1 - w_1 - w_2)(20 - w_1 - 40w_2) - 1$$

using the extended critical point systems. We begin by declaring the necessary variables in Julia.

```
@polyvar w[1:2] x[1:2] y[1:2] a[1:2] b[1:2] λ^R λ^I s t ν ν₁ ν₂
```

The critical points are computed as is in the combinatorial case.

```
H = (1-w[1]-w[2])*(20-w[1]-40*w[2])-1
sys = System([H, w[1]*differentiate(H, w[1]) - w[2]*differentiate(H, w[2])])
result = solve(sys)
display(map(sol -> round.(sol; digits=3), solutions(result)))
```

```
4-element Vector{Vector{ComplexF64}}:
 [9.997 - 0.0im, 0.253 + 0.0im]
 [0.49 + 0.281im, 0.581 - 0.162im]
 [0.49 - 0.281im, 0.581 + 0.162im]
 [0.548 + 0.0im, 0.31 + 0.0im]
```

There are 4 critical points, two of which are real. The point $(0.548, 0.31)$ is the only minimal critical point. The next step is to split $H(w_1, w_2)$ into real and imaginary components and build the extended non-combinatorial case systems.

```
@polyvar I
u = sum(filter(t -> iseven(degree(t, I)), terms(H(w => a + I.*b))))
v = sum(filter(t -> isodd(degree(t, I)), terms(H(w => a + I.*b))))
u, v = subs(u, I => im), -im*subs(v, I => im)
```

With $u(a, b)$ and $v(a, b)$ computed, we construct the extended systems with the following code.

```
# critical equations in variables a, b, λ^R λ^I after splitting
criteqs = [ u; v; a.*differentiate(u, a) + b.*differentiate(u, b) .- λ^R;
                  a.*differentiate(v, a) + b.*differentiate(v, b) .- λ^I ]
# minimality equations in variables x, y, a, b, t
circeqs = [u([a; b] => [x; y]);
    v([a; b] => [x; y]); x.^2 + y.^2 - t.*(a.^2 + b.^2)]
# balance equations
J2eqs = (ν₁.*y - ν₂.*x) .* differentiate(u([a; b] => [x; y]), x) -
        (ν₁.*x + ν₂.*y) .* differentiate(u([a; b] => [x; y]), y)
# # there are two systems to solve depending on the value of ν₁
extended_systems = [
    System([criteqs; circeqs; s*(1-t) - 1; subs(J2eqs, [ν₁, ν₂] => [1, ν])]),
    System([criteqs; circeqs; s*(1-t) - 1; subs(J2eqs[1], [ν₁, ν₂] => [0, 1])])
]
```

The first system corresponds to adding the top d equations of (3.5) and the second system corresponds to taking the bottom $d-1$ equations. We solve both equations and examine the $\mathbf{a}, \mathbf{b}$ and t coordinates of each solution in the following code.

```
# the output solutions have form (a, b, t)
sols1 = real_solutions(solve(extended_systems[1]))
sols2 = real_solutions(solve(extended_systems[2]))
display(map(sol -> round.([sol[5:8]; sol[12]]; digits=4), [sols1; sols2]))
```

```
32-element Vector{Vector{Float64}}:
 [0.4901, 0.5808, -0.2808, 0.1624, 688.0156]
 [9.9971, 0.2528, 0.0, 0.0, 0.0085]
 [0.4901, 0.5808, -0.2808, 0.1624, 0.4167]
 [0.4901, 0.5808, -0.2808, 0.1624, 1.0421]
 ⋮
```

There are 16 solutions to each system, resulting in 32 solutions total. The critical point $(9.997, 0.253)$ is not a minimal point, since the second solution in the list

$$\texttt{[9.9971, 0.2528, 0.0, 0.0, 0.0085]}$$

clearly has $t \approx 0.0085 \in (0, 1)$. We verify this rigorously by first certifying the solutions to the systems.

```
certs1 = certificates(certify(extended_systems[1], sols1))
display(certified_solution_interval(certs1[2])[12]) # t value
```

```
[0.008498195031 +/- 4.39e-13] + [+/- 1.14e-13]im
```

The solution t is certified to be real so the imaginary component is exactly 0, and the real component is inside the interval

$$0.008498195031 \pm 4.39 \cdot 10^{-13}$$

with endpoints larger than 0 and less than 1, as required. The remaining two complex

points are easily seen to be non-minimal in a similar way by analyzing the solutions and checking for $t \in (0,1)$, which is done automatically in Julia with minimal code. Finally, it is also seen that there are no solutions where the t value is between 0 and 1 corresponding to the point $(0.548, 0.31)$. It follows that $(0.548, 0.31)$ is the only minimal critical point.

3.1.3 Practice and Examples

In this section we demonstrate the effectiveness and shortcomings of the homotopy continuation method by applying it to several examples. The package `ACSVHomotopy` is available GitHub at

https://github.com/ACSVMath/ACSVHomotopy

with further documentation available in [11]. All computations in this section were completed on a machine with the following (virtual) hardware.

```
OS: Ubuntu 22.04.1 LTS on Windows 10 x86_64
Kernel: 5.15.79.1-microsoft-standard-WSL2
CPU: AMD Ryzen 5 5600X (12) @ 3.700GHz
GPU: 5c84:00:00.0 Microsoft Corporation Device 008e
Memory: 7915MiB
```

Example 18. The rational function $F(x,y) = \frac{1}{(1-x-y)(20-x-40y)-1}$ from Example 17 happens to be combinatorial. We will consider the main diagonal and compute asymptotics using the combinatorial case method. The first step is to define the rational function that we are analyzing.

```
@polyvar x y z λ t s
H = (1-x-y)*(20-x-40*y)-1
```

The following code solves the critical point system in Julia given the denominator H, and certifies the solutions.

```
sys = System([H; x*differentiate(H, x) - y*differentiate(H, y)])
sols = solutions(solve(sys; show_progress=false))
certs = certificates(certify(sys, sols))
display(map(sol -> round.(sol; digits=4), solution_approximation.(certs)))
```

```
4-element Vector{Vector{ComplexF64}}:
 [9.9971 + 0.0im, 0.2528 + 0.0im, 93.5529 + 0.0im]
 [0.4901 - 0.2808im, 0.5808 + 0.1624im, 3.5707 + 1.9223im]
 [0.4901 + 0.2808im, 0.5808 - 0.1624im, 3.5707 - 1.9223im]
 [0.5482 + 0.0im, 0.31 + 0.0im, -3.9442 + 0.0im]
```

There are 4 critical points, two of which $(9.997, 0.253)$ and $(0.548, 0.31)$, are real. In order to determine which of these two critical points is minimal, we solve System (3.2) and examine the real solutions.

```
@polyvar λ t s
sys = System([ # full comb-case system
    H;
    [x, y] .* differentiate(H, [x, y]) .- λ;
    H([x, y] => t*[x, y]);
    s*(1-t) - 1
], variables=[t; x; y; λ; s])
real_sols = real_solutions(solve(sys; show_progress=false))
certs = certificates(certify(sys, real_sols))
display(map(sol -> round.(sol[1:3]; digits=4), solution_approximation.(certs)))
```

```
2-element Vector{Vector{ComplexF64}}:
 [0.0922 + 0.0im, 9.9971 + 0.0im, 0.2528 + 0.0im]
 [1.7099 + 0.0im, 0.5482 + 0.0im, 0.31 + 0.0im]
```

The first solution has $t = 0.092$ and therefore the critical point $(9.997, 0.253)$ is not minimal. The second solution has $t = 1.71$ so it does not provide information about minimality. To be certain that the point $(9.997, 0.253)$ is not minimal we check the certified interval and see that the entire interval is between 0 and 1 as follows.

```
intervals = certified_solution_interval.(sols)
intervals = filter(I -> 0 < real(I[1]) < 1, intervals)
display(map(I -> real(I[1]), intervals))
```

```
1-element Vector{Arblib.Arb}:
 [0.092185655233 +/- 4.51e-13]
```

Indeed, the point $(9.997, 0.253)$ is not minimal. Since those are the only real solutions, and since the complex points are easily seen to be coordinate wise further from the origin than $(0.548, 0.31)$, we conclude that $(0.548, 0.31)$ is the only minimal point of $F(x, y)$. The fact that $F(x, y)$ is combinatorial simplified the analysis dramatically when compared to the non-combinatorial case.

The code in the following examples is from the worksheet ACSVHomotopy.ipynb in the GitHub repository.

Example 19 (non-crossing configurations)**.** We return to the generating functions for non-crossing configurations discussed in Example 12. Applying Theorem 2, we are able to obtain bivariate rational generating functions T, F, C, G whose diagonals count non-crossing trees, forests, connected graphs and graphs respectively. The notebook non-crossing-configurations.ipynb on GitHub shows how these generating functions are derived and computes the first 20 coefficients of each. As we have obtained these generating functions through algebraic means, it is not clear whether or not they are combinatorial generating series. Therefore, we solve this series using the non-combinatorial method through the `ACSVHomotopy` package. Once the code is downloaded and included in Julia, the following code declares the denominator of G which we computed in Example 12 and computes the minimal point.

```
@polyvar x y
H = (y^2*x + 6*y*x + 8*x - 1)
@time find_min_crits(H)
```

```
 46.263916 seconds (87.24 M allocations: 4.335 GiB, 2.74% gc time,
85.05% compilation time)
1-element Vector{Vector{Float64}}:
 [0.03033008588991066, 2.828427124746189]
```

The output shows that the machine used over 4 gigabytes of memory and took 46 seconds to compile and solve the systems. The minimal point is found to be approximately $(0.03033, 2.828)$. Applying Theorem 6 to extract the exponential growth, we find that the number of non-crossing graphs on n vertices grows exponentially with base 11.656854249492378, matching the univariate analysis to 15 decimal places. We can verify minimality heuristically with the approach above by running.

```
@time find_min_crits(H; approx_crit=true)
```

```
 12.545423 seconds (28.35 M allocations: 1.335 GiB, 2.00% gc time,
98.31% compilation time)
1-element Vector{Vector{Float64}}:
 [0.03033008588991066, 2.828427124746189]
```

With the heuristic method, the algorithm uses over 1 gigabyte of memory and completes in 12 seconds.

The remaining non-crossing configuration generating functions C, F, T have denominators of degree 8, 12 and 12 respectively, and the computations are more involved. Solving the systems without the heuristic is possible, but far less reasonable than the heuristic approach method that approximates critical points. The following code computes the minimal points for each of the generating series C, F, T using the heuristic method.

```
# non-crossing connected graphs denominator
H = y^5*x^3 + 3*y^4*x^3 + 3*y^3*x^3 + 3*y^3*x^2 + y^2*x^3 + 6*y^2*x^2 + y^2*x
  + 3*y*x^2 + 5*y*x + 4*x - 1
find_min_crits(H; approx_crit=true)
```

```
1-element Vector{Vector{Float64}}:
 [0.05391937820329315, 1.7846096908265279]
```

```
# non-crossing forests denominator
H = y^7*x^5 + 21*y^6*x^5 + 147*y^5*x^5 + 7*y^5*x^4 + 343*y^4*x^5 + 98*y^4*x^4
  + 2*y^4*x^3 + 343*y^3*x^4 + 44*y^3*x^3 + 210*y^2*x^3 + 10*y^2*x^2 + 82*y*x^2
  + 3*y*x + 33*x - 1
find_min_crits(H; approx_crit=true)
```

```
1-element Vector{Vector{Float64}}:
 [0.0043199830940064176, 28.144810807432684]
```

```
# non-crossing trees denominator
H = y^7*x^5 + 9*y^6*x^5 + 27*y^5*x^5 + 3*y^5*x^4 + 27*y^4*x^5 + 18*y^4*x^4
  + 3*y^4*x^3 + 27*y^3*x^4 + 21*y^3*x^3 + 36*y^2*x^3 + 6*y^2*x^2 + 19*y*x^2
  + 3*y*x + 12*x - 1
find_min_crits(H; approx_crit=true)
```

```
1-element Vector{Vector{Float64}}:
 [0.011368682831512572, 13.031249999999993]
```

Each generating function admits one minimal critical point, all of which have positive and real coordinates. Applying Theorem 6 to each we find that the exponential growth of the number of non-crossing connected graphs, forests and trees to 5 decimal places is 10.39230^n, 8.22469^n and 6.75000^n as expected from known univariate analysis.

Note that the leading coefficient C_0 in Theorem 6 is zero in these examples. Asymptotics can still be derived from these points, but higher order terms are necessary [13].

Example 20 (lattice walks)**.** To conclude this section we apply the homotopy method to generating functions pertaining to lattice walk enumeration. The generating functions in this example are all combinatorial 3-variate generating functions. Recall from Chapter 2 that the number of lattice walks in $\mathbb{N} \times \mathbb{N}$ of length n with step set $\mathcal{S} = \{\nearrow, \searrow, \nwarrow, \swarrow\}$ is counted by the coefficients of

$$F(x, y, z) = \frac{(x+1)(y+1)}{1 - z(x^2y^2 + x^2 + y^2 + 1)}$$

on the main diagonal. The generating function has 4 critical points,

```
4-element Vector{Vector{ComplexF64}}:
 [1.0 + 0.0im, -1.0 + 0.0im, 0.25 + 0.0im, -1.0 + 0.0im]
 [1.0 + 0.0im, 1.0 + 0.0im, 0.25 + 0.0im, -1.0 + 0.0im]
 [-1.0 + 0.0im, -1.0 + 0.0im, 0.25 + 0.0im, -1.0 + 0.0im]
 [-1.0 + 0.0im, 1.0 + 0.0im, 0.25 + 0.0im, -1.0 + 0.0im]
```

which represent the points $(\pm 1, \pm 1, 1/4) \in \mathbb{C}^3$. The function $F(x, y, z)$ is a combinatorial generating series, therefore to test minimality we check the critical points with real and positive coordinates. There is only one such point $(1, 1, 1/4)$ and this point is found to be minimal by solving the combinatorial case system. As a result the exponential growth of these types of lattice walks is 4^n. To compute the sub-exponential growth and coefficients in the asymptotic formula, we need the remaining minimal critical points, which have the same coordinate-wise modulus. Here it happens that all 4 of the critical points are minimal, however, to rigorously verify this we require precision on the order of 2^{-4096} which is unreasonable to compute without powerful hardware. The `SageMath` package demonstrated in [9] is available online and it provides an alternative symbolic-numeric approach to solving for minimal points of combinatorial generating functions which is less prone to this weakness.

For now we turn to the non-combinatorial case algorithm to verify the minimal points and recover asymptotics. The following code computes minimal critical points and gives asymptotics using Theorem 6.

```
H = 1 - (z*x^2*y^2 + z*x^2 + z*y^2 + z)
@time find_min_crits(H; approx_crit=true)
println("minimal points: ")
display(map(pnt -> round.(pnt; digits=4), MCP))
asm = asymptotics((1+x)*(1+y), H, MCP)
println("result: ", asm(n))
println("expect: ", 2/π*(4^n/n))
```

```
minimal points:
4-element Vector{Vector{Float64}}:
 [1.0, -1.0, 0.25]
 [1.0, 1.0, 0.25]
 [-1.0, -1.0, 0.25]
 [-1.0, 1.0, 0.25]
result: (0.6366197723675815(4.000000000000001^n)) / n
expect: (0.6366197723675814(4^n)) / n
```

Our calculation matches the known asymptotic growth $\frac{2}{\pi n}4^n$ to 15 decimal places.

We can apply the same analysis to other generating functions pertaining to these types of lattice walks. The generating function counting restricted quadrant walks with step set $\{\nwarrow, \uparrow, \nearrow, \swarrow, \downarrow, \searrow\}$ by length is the main diagonal of

$$-\frac{(x+1)(y+1)}{tx^2y^2 + txy^2 + tx^2 + ty^2 + tx + t - 1}.$$

The known asymptotic growth is $\frac{\sqrt{6}}{\pi n}6^n$, which the following code computes to 15 decimal places.

```
G = -(x + 1)*(y + 1)
H = (t*x^2*y^2 + t*x*y^2 + t*x^2 + t*y^2 + t*x + t - 1)
MC = find_min_crits(H; approx_crit=true)
println("minimal points: ")
display(map(pnt -> round.(pnt; digits=4), MCP))
asm = asymptotics(G, H, MCP)
println("result: ", asm(n))
println("expect: ", sqrt(6)/π*(6^n/n))
```

```
minimal points:
2-element Vector{Vector{Float64}}:
 [1.0, -1.0, 0.1667]
 [1.0, 1.0, 0.1667]
result: (0.7796968012336761(6.0^n)) / n
expect: (0.779696801233676(6^n)) / n
```

The step set $\{\nwarrow, \uparrow, \nearrow, \leftarrow, \rightarrow, \swarrow, \downarrow, \searrow\}$ yields the generating function

$$-\frac{(x+1)(y+1)}{tx^2y^2 + tx^2y + txy^2 + tx^2 + ty^2 + tx + ty + t - 1},$$

and the asymptotic growth $\frac{8}{3\pi n}8^n$ is computed the same way.

```
G = -(x + 1)*(y + 1)
H = (t*x^2*y^2 + t*x^2*y + t*x*y^2 + t*x^2 + t*y^2 + t*x + t*y + t - 1)
MCP = find_min_crits(H; approx_crit=true)
println("minimal points:")
display(map(pnt -> round.(pnt; digits=4), MCP))
asm = asymptotics(G, H, MCP)
println("result: ", asm(n))
println("expect: ", 8/3π*(8^n/n))
```

```
minimal points:
1-element Vector{Vector{Float64}}:
 [1.0, 1.0, 0.125]
result: (0.8488263631567748(7.999999999999998^n)) / n
expect: (0.8488263631567752(8^n)) / n
```

3.2 Flows on the Height Function

The code above proves minimality at critical points. In the absence of minimal critical points it is hard to know how to compute asymptotics. One approach uses the *height function* $h : \mathcal{V}(H) \to \mathbb{R}^d$ in the direction $\mathbf{r}$ defined by

$$h(\mathbf{z}) = h(z_1, \ldots, z_d) = -\sum_{j=1}^{d} r_j \log |z_j|.$$

The importance of the height function comes from the fact that it represents the terms in the Cauchy integral for coefficients

$$\frac{1}{(2\pi i)^d}\int_T \frac{F(\mathbf{z})}{\mathbf{z}^{n\mathbf{r}+\mathbf{1}}}d\mathbf{z} = \frac{1}{(2\pi i)^d}\int_T \frac{F(\mathbf{z})}{\mathbf{z}^{\mathbf{1}}}e^{-n\mathbf{r}\log(\mathbf{z})}d\mathbf{z} \tag{3.6}$$

that vary with n. Points where the height function is large are those where the integrand is large, and points where the height function is small are those where the integral is small. When using the Cauchy integral formula to approximate the coefficients $f_{n\mathbf{r}}$, points of lower height will have exponentially smaller contributions. Analysis of the integral (3.6) is done by deforming the contour T smoothly and continuously past the singular set $\mathcal{V}(H)$. As the contour T is pushed passed points on the singular set, the Cauchy integral is reduced to an integral on $\mathcal{V}(H)$ using residues and intersection cycles [15, Chapter 7]. For this reason, it

is helpful to analyze the process of continuously and smoothly deforming a contour T' of dimension $d-1$ which lives in $\mathcal{V}(H)$ instead of deforming T directly. This motivates the concept of a *flow* on $\mathcal{V}(H)$, which moves points to lower heights.

Returning to the main diagonal of the generating function

$$F(x,y) = \frac{1}{H(x,y)} = \frac{1}{1-x-y}$$

we are able to plot the height function in 3 real dimensions by parameterizing $\mathcal{V} = \mathcal{V}(H)$ in terms of x as

$$h(x, y(x)) = -\log|x| - \log|y(x)| = -\log|x| - \log|1-x|\,.$$

Figure 3.1 depicts the height function $h : \mathbb{C}\backslash\{0,1\} \to \mathbb{R}$ as a surface in $\mathbb{R}^3$. We visualize a contour of integration $T' \subseteq \mathcal{V}(H)$ as a curve which rests on the surface defined by the height map. The goal is to deform a given T' smoothly and continuously to a contour with smaller height, when possible.

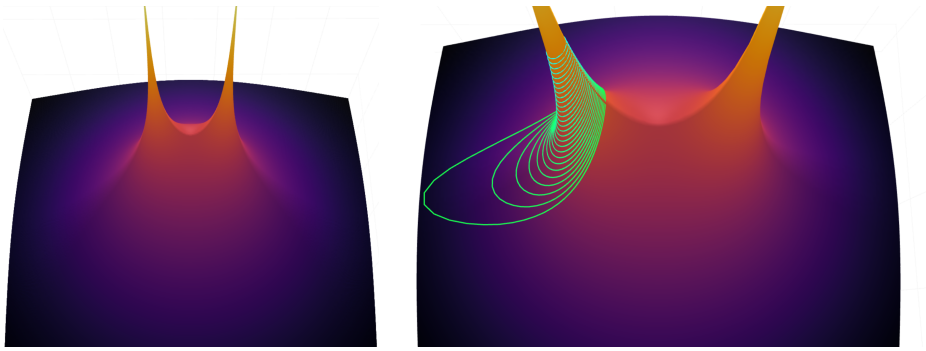


Figure 3.1: Plot of height function $h(x, 1-x)$. The critical point is the saddle point at $x = 1/2$. On the right we plot rings where the top ring corresponds to a contour where $|x| = 1/18$ and each lower ring is the result of computing a gradient flow for each point on the initial contour. We see that the rings continue to flow towards the critical point $(1/2, 1/2)$ on one side, and $-\infty$ on the other.

3.2.1 Computation of Flows

We now demonstrate how to compute a flow on the height map of a given rational generating function with respect to the main diagonal. Formally for our purposes, a *flow* (often called a *gradient flow*) is a function from $f(\mathbf{z}, t) = \mathcal{V} \times [0, T] \to \mathcal{V}$ for some fixed T such

that at each value $t \in [0, T]$ we have

$$\frac{d}{dt} f(\mathbf{z}, t) = -\nabla \tilde{h}(\mathbf{z})$$

where $\tilde{h} = \prod_{j=1}^{d} |z_j|^2$. The reason we consider $\tilde{h}$ instead of h is to avoid unnecessary rational functions or logarithms in our computations, and h and $\tilde{h}$ give points the same relative heights. We begin by illustrating the process of computing flows here in two variables.

Let $H \in \mathbb{Q}[z, w]$ where $H_w(z_0, w_0) \neq 0$ for some point $(z_0, w_0) \in \mathcal{V}(H)$. The implicit function theorem implies the existence of an open set D containing z_0 such that $H(z, w(z)) = 0$ for all $z \in D$. Splitting the variables into real and complex components, we write $z = x + iy$ and $w(z) = u(x, y) + iv(x, y)$. Our goal is to construct functions $x(t)$ and $y(t)$ such that $z_0 = x(0) + iy(0)$, and as t goes from $0 \to 1$ the path on the surface $h'(x(t), y(t), u(x(t), y(t)), v(x(t), y(t)))$ moves along the path of steepest decent.

We begin with the defining equations

$$\begin{aligned} \frac{dx}{dt} &= -\frac{\partial}{\partial x} \tilde{h}'(x, y, u(x, y), v(x, y)) \\ \frac{dy}{dt} &= -\frac{\partial}{\partial y} \tilde{h}'(x, y, u(x, y), v(x, y)) \end{aligned}$$

which we require the flow to satisfy. Then we compute

$$\frac{dx}{dt} = \frac{\partial}{\partial x} \tilde{h}'(x, y, u(x, y), v(x, y)) = -2x(u^2 + v^2) + (x^2 + y^2) \left(2u \frac{\partial u}{\partial x} + 2v \frac{\partial v}{\partial x} \right)$$

and, in a similar fashion,

$$\frac{dy}{dt} = -2y(u^2 + v^2) + (x^2 + y^2) \left(2u \frac{\partial u}{\partial y} + 2v \frac{\partial v}{\partial y} \right).$$

The result is the differential-algebraic system of equations

$$\begin{aligned} 0 &= \Re(H(x, y, u, v)) \\ 0 &= \Im(H(x, y, u, v)) \\ x' &= -2x(u^2 + v^2) + (x^2 + y^2) \left(2u \frac{\partial u}{\partial x} + 2v \frac{\partial v}{\partial x} \right) \\ y' &= -2y(u^2 + v^2) + (x^2 + y^2) \left(2u \frac{\partial u}{\partial y} + 2v \frac{\partial v}{\partial y} \right), \end{aligned} \tag{3.7}$$

in the unknowns x, y, u and v as functions of t. In order to compute a solution to (3.7) we first need to compute the partials $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial v}{\partial x}$ and $\frac{\partial v}{\partial y}$. Since $H_w(z_0, w_0) \neq 0$ the implicit

function theorem yields the equality

$$\frac{dw}{dz} = -\frac{H_z(z,w)}{H_w(z,w)},$$

valid on an open set containing the point (z_0, w_0). The Cauchy-Riemann equations are

$$-\frac{H_z(z,w)}{H_w(z,w)} = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} - i\frac{\partial v}{\partial y},$$

from which we are able to compute all required partials, and therefore the system (3.7). To compute initial conditions for the differential-algebraic system, we take

$$x_0 = \Re(z_0), \qquad y_0 = \Im(z_0), \qquad u_0 = \Re(w_0), \qquad v_0 = \Im(w_0),$$

and the first order derivatives with respect to t are

$$\begin{aligned} x_0' &= \frac{dx}{dt}(x_0, y_0, u_0, v_0), & y_0' &= \frac{dy}{dt}(x_0, y_0, u_0, v_0), \\ u_0' &= \left(\frac{\partial u}{\partial x}\frac{dx}{dt} + \frac{\partial u}{\partial y}\frac{dy}{dt}\right)(x_0, y_0, u_0, v_0), & v_0' &= \left(\frac{\partial v}{\partial x}\frac{dx}{dt} + \frac{\partial u}{\partial y}\frac{dy}{dt}\right)(x_0, y_0, u_0, v_0), \end{aligned}$$

where each function to be evaluated is explicit in terms of H and derivatives of H.

To solve the differential-algebraic system (3.7) we employ the extensive Julia library `DifferentialEquations.jl` [16] with the Sundials [10, 8] differential equation solver through Julia interface `Sundials.jl`. The code referenced in this section uses both of these these packages.

Example 21. Let $H(z,w) = 1 - z - w$, $\mathbf{r} = (1,1)$ and note that $H_w(z,w) = -1 \neq 0$. The function $w(z) = 1 - z$ satisfies $H(z, w(z)) = 0$ on all of $\mathbb{C}$. Since $\frac{dw}{dz} = -1$ and therefore $\frac{\partial u}{\partial x} = -1$, $\frac{\partial u}{\partial y} = 0$, $\frac{\partial v}{\partial x} = 0$ and $\frac{\partial v}{\partial y} = -1$, the system (3.7) is

$$\begin{aligned} 0 &= 1 - x - u, & 0 &= -y - v, \\ x' &= -2x(u^2 + v^2) + (x^2 + y^2)2u, & y' &= -2y(u^2 + v^2) + (x^2 + y^2)2v. \end{aligned}$$

If we take the points on $\mathcal{V}$ where $|z| = 1/18$ as our initial conditions, we are able to compute a numerical solution to the system. The result is used to produce images such as the right of Figure 3.1 and the full code is in the notebook flows-on-binom.ipynb on the GitHub repository. Note that it is possible to compute flows with many different sets of initial conditions using this method. For example, we may instead be interested in the contour

where $|w| = 1/18$, which is this case is symmetric.

Example 22. Consider the generating function $1/H(z, w)$ where

$$H(z, w) = 2 + z - w(1 + z)^2.$$

Figure 3.2 depicts the height function $h(z, w(z))$, where $w(z) = \frac{2+z}{(1+z)^2}$. The implicit function theorem yields $\frac{dw}{dz} = -\frac{H_z}{H_w}$ valid around points where $z \neq -1$. The notebook cpai-flow.ipynb builds and solves System (3.7) automatically. A flow with initial points on $\mathcal{V}(H)$ where $|z_0| = 1/12$ was used to generate the plot on the left in Figure 3.2. Notice that in Figure 3.2 the flow appears to drift off on the real positive axis, while the height is not decreasing significantly. The generating function $1/H$ has a *critical point at infinity* which is apparent in the graphic as the height function is not decaying as $z \to \infty$. In fact one can prove that the maximum value of the height function on a curve around the orgin in z is bounded below, even as $z \to \infty$. Critical points at infinity are complicated to analyze, and are actively being researched.

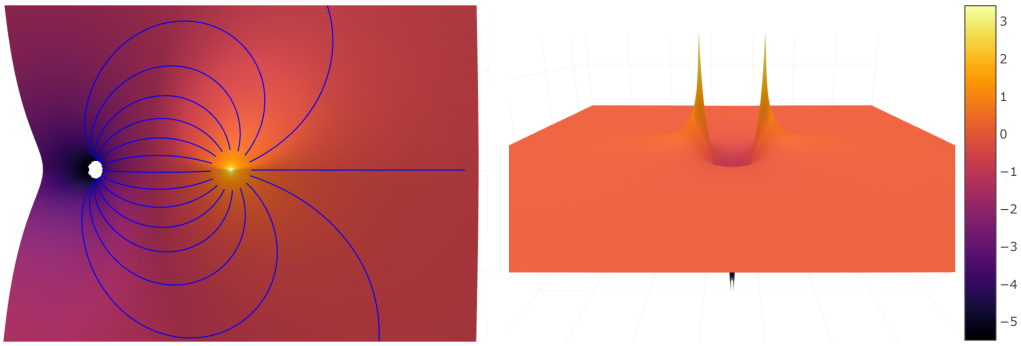


Figure 3.2: Plots of flows on $\mathcal{V}(H)$ with $H = 2+z-w(1+z)^2$ using the parameterization $w(z) = \frac{2+z}{(1+z)^2}$. The plot on the left shows a flow computed with initial points on $|z| = 1/12$. Rather than plot the contours, we plot the trajectories of each sampled point as it flows down the gradient.

Flows in many variables

Flows in many variables are computed similarly, but the process is more computationally expensive. Moreover, higher dimensional varieties are not amenable to simple visualizations such as the ones presented in the bivariate case.

Let $\mathbf{z}^\circ = (z_1, z_2 \ldots, z_{d-1})$ denote the first $d-1$ coordinates of $\mathbf{z}$. Let $d > 1$ and let $H(z_1, z_2, \ldots, z_{d-1}, w) = H(\mathbf{z}^\circ, w) \in \mathbb{Q}[\mathbf{z}^\circ, w]$. Fix a point $(\mathbf{a}^\circ, w_0) \in \mathcal{V}(H)$ and assume that

$H_w(\mathbf{a}^\circ, w_0) \neq 0$. The implicit function theorem implies the existence of a parameterization $w(\mathbf{z}^\circ)$ valid on an open set D around $(\mathbf{a}^\circ)$ such that $H(\mathbf{z}^\circ, w(\mathbf{z}^\circ)) = 0$ and

$$\nabla w(\mathbf{z}^\circ, w(\mathbf{z}^\circ)) = -\frac{1}{H_w(\mathbf{z}^\circ, w(\mathbf{z}^\circ))} \left(H_{z_1}(\mathbf{z}^\circ, w(\mathbf{z}^\circ)), \ldots, H_{z_{d-1}}(\mathbf{z}^\circ, w(\mathbf{z}^\circ)) \right),$$

for all $\mathbf{z} \in D$. As in the bivariate case, we split into real and imaginary components so that $\mathbf{z}^\circ = \mathbf{x}^\circ + i\mathbf{y}^\circ$ and $w = u(\mathbf{x}^\circ, \mathbf{y}^\circ) + iv(\mathbf{x}^\circ, \mathbf{y}^\circ)$, and

$$\frac{\partial u}{\partial x_j} = -\Re\left(\frac{H_{z_j}}{H_w}\right), \qquad \frac{\partial v}{\partial x_j} = -\Im\left(\frac{H_{z_j}}{H_w}\right), \qquad \frac{\partial u}{\partial y_j} = -\frac{\partial u}{\partial y_j}, \qquad \frac{\partial v}{\partial y_j} = \frac{\partial u}{\partial x_j}$$

for each $j \in [d-1]$. The final differential-algebraic system is thus

$$\begin{aligned} 0 &= \Re(H(\mathbf{x}^\circ, \mathbf{y}^\circ, u, v)), \\ 0 &= \Im(H(\mathbf{x}^\circ, \mathbf{y}^\circ, u, v)), \\ x_j' &= \frac{\partial}{\partial x_j} \tilde{h}(\mathbf{x}^\circ, \mathbf{y}^\circ, u, v), \\ y_j' &= \frac{\partial}{\partial y_j} \tilde{h}(\mathbf{x}^\circ, \mathbf{y}^\circ, u, v). \end{aligned}$$

for all $j \in [d-1]$ and $\tilde{h}(\mathbf{x}^\circ, \mathbf{y}^\circ, u, v) = -(u^2+v^2)\prod_{j=1}^{d-1}(x_j^2+y_j^2)$. The initial conditions are computed by fixing a point $(\mathbf{a}^\circ, w_0)$ and evaluating $\frac{dx_j}{dt}, \frac{dy_j}{dt}$ and

$$\begin{aligned} \frac{du}{dt} &= \nabla u \cdot \left(\frac{dx_1}{dt}, \frac{dy_1}{dt}, \ldots, \frac{dx_n}{dt}, \frac{dy_n}{dt} \right) \\ \frac{dv}{dt} &= \nabla v \cdot \left(\frac{dx_1}{dt}, \frac{dy_1}{dt}, \ldots, \frac{dx_n}{dt}, \frac{dy_n}{dt} \right) \end{aligned}$$

at $(\mathbf{a}^\circ, w_0)$. As in the bivariate case, the entire process is automatic in Julia and the code is available in the notebook flows-in-many-vars.ipynb.

3.3 Bivariate Generating Functions and DeVries Algorithm

Non-minimal critical points are hard to analyze in general. However, in the bivariate case the PhD thesis [5] describes an algorithm that provides a general analysis. To the best of our knowledge, we provide the first implementation of this algorithm, using `SageMath`

[18].

Fix a denominator $H(z, w) \in \mathbb{Q}[z, w]$ and consider the main diagonal. For simplicity, we assume that the partial derivative $H_w(z, w)$ never vanishes, so that the implicit function theorem implies the existence of a paramaterization of $\mathcal{V}(H)$ in terms of z. Let C denote the set of critical points of H. Around each critical point $(z_0, w_0) \in C$ there exists a disk split into m regions which have heights above $h(z_0, w_0)$ and m regions which have height below $h(z_0, w_0)$, where m is the smallest number greater than 0 such that

$$\frac{d^m}{dz^m} h(z_0, w(z_0)) \neq 0.$$

The number m is called the *degree of degeneracy* of the point (z_0, w_0). Figure 3.3 depicts the typical geometry of the height function around a critical point with degree of degeneracy 4. Since the point (z_0, w_0) is critical, the degree of degeneracy is always at least 2. When the degree of degeneracy at (z_0, w_0) is 2 we call (z_0, w_0) *nondegenerate*. When the degeneracy is greater than two Theorem 6 does not apply, but it is still possible to extract asymptotic formulas for generating functions with minimal points that have degeneracy greater than 2.

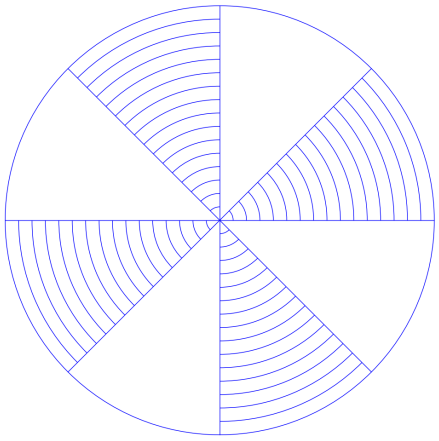


Figure 3.3: The typical geometry of the height map around a point of degeneracy 4. The surface is split into 8 regions, 4 of which are monotone ascending on rays pointing out of the critical point and 4 of which form monotone descending rays pointing out of the critical point.

Following [5], we begin by solving the critical point equations. For each critical point (z_0, w_0) in descending order of height we check each ascending region for paths which approach a pole where $z = 0$ or $w = 0$, such that the path is strictly ascending in height. For a fixed critical point (z_0, w_0), if there exists strictly ascending paths which approach

$z = 0$ and strictly ascending paths which approach $w = 0$, and there are no other critical points of higher height which satisfy this property, then the point (z_0, w_0) is a critical point of H that determines asymptotic behavior [5, Chapter 3]. Figure 3.4 illustrates the strictly ascending paths out of the critical point $(1/2, 1/2)$ for the generating function $\frac{1}{1-z-w}$. Once a critical point with ascending paths to poles where $z = 0$ and $w = 0$ is found, it is sufficient to check all other critical points which are at equal height and then determine an asymptotic formula for coefficients. Critical points of lower height contribute exponentially smaller asymptotic growth.

The formal details of implementing these computations rigorously, and all the necessary and sufficient conditions for successful completion, are outlined in [5, Chapter 4]. Here we focus on the details of implementing the stepping function which is responsible for determining how points move discretely along the surface defined by the height function such that their paths are strictly ascending in height. This is the most costly and interesting operation in the algorithm.

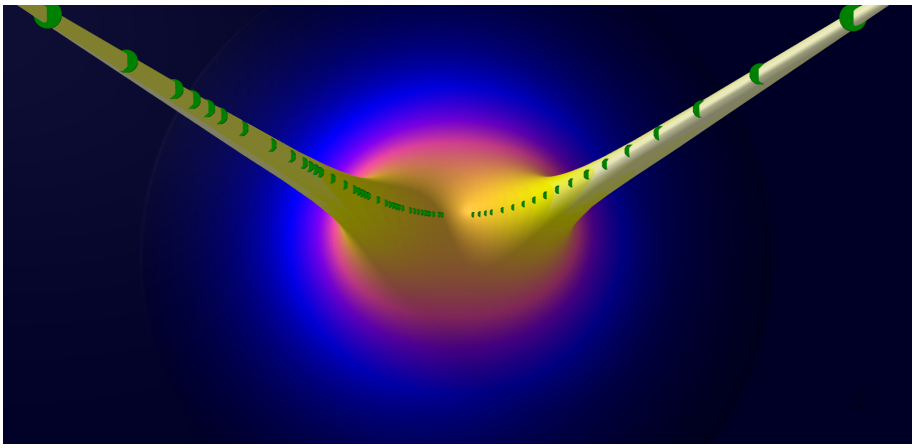


Figure 3.4: The zero set of $H = 1 - x - y$. The critical point $(1/2, 1/2)$ has strictly ascending paths to the poles $z = 0$ and $w = 0$ and therefore it detemines asymptotics. Plotted in green are some of the discrete steps taken by of DeVries algorithm implemented in `SageMath`.

The stepping function

Let $(z_0, w_0) \in \mathcal{V}(H)$ and suppose that (z_0, w_0) is not a critical point, so $\frac{dh}{dz}(z_0, w_0) \neq 0$. Let $w = w(z)$ be a parameterization of $\mathcal{V}(H)$ inside a open set D given by the implicit function theorem. We wish to step to a point $(z_1, w(z_1)) \in \mathcal{V}(H)$ such that along the ray $z_0 \to z_1$ the height function strictly increases. We begin by fixing a direction v with $|v| = 1$ and determining the maximal δ such that $z + \delta v \in D$, and thus $H(z + \delta v, w(z + \delta v)) = 0$.

We accomplish this by computing disjoint open balls $B_{\epsilon_j}(w_j)$ such that

$$\bigcup_{j=1}^{\deg(H(z_0,w))} B_{\epsilon_j}(w_j) \supseteq \{w : \exists z \in B_\delta(z_0),\ H(z,w) = 0\},$$

where δ meets the following criteria.

- ◇ For any $z' \in B_\delta(z_0)$ the polynomial $H(z', w)$ has the same number of roots as $H(z_0, w)$.
- ◇ There exists $\deg(H(z_0, w))$ unique disjoint solutions of $H(B_\delta(z_0), w)$, contained in the balls $B_{\epsilon_j}(w_j)$.

With these conditions met we are able to discretely step from any point (z_0, w_0) to $(z_0 + \delta v, w(z_0 + \delta v))$ with $w(z_0 + \delta v) \in B_{\epsilon_j}(w_j)$ for only one value of j between 1 and $\deg(H(z_0, w))$. Let $B_{\epsilon_1}(w_1)$ denote the unique solution to $H(B_\delta(z_0), w)$ such that $w_0 \in B_{\epsilon_1}(w_1)$. Then $w(z_0 + \delta v) \in B_{\epsilon_1}(w_1)$ is computed by solving for the roots of the univariate polynomial $H(z_0 + \delta v, w)$ up to sufficient accuracy and checking which root is in $B_{\epsilon_1}(w_1)$.

Next we determine a suitable δ and direction v so that the curve on the height map

$$h(z_0 + tv, w(z_0 + tv)),\ t \in [0, \delta] \tag{3.8}$$

is increasing as t continuously moves from $0 \to \delta$. The number δ is the called the *step size* and Section 4.12 of the PhD thesis [5] gives a method of computing a neighborhood of z_0 such that the ray (3.8) is monotone for all choices of v. With this δ we are able to discretely step on the height function such that the piece-wise linear functions connecting the discrete jumps are strictly ascending on the height map by choosing

$$v = \frac{dh}{dz}(z_0, w_0) = \Re\left(\left[-\frac{1}{z} + \frac{1}{w}\frac{H_z(z,w)}{H_w(z,w)}\right]_{\substack{z=z_0\\w=w_0}}\right).$$

When (z_0, w_0) is a critical point we have $\frac{dh}{dz}(z_0, w_0) = 0$ and there are $n \geq 2$ distinct strictly ascending paths out of (z_0, w_0), where n is the degree of degeneracy of (z_0, w_0). In this case, the direction vectors are rotations of $\frac{d^n h}{dz^n}(z_0, w_0)$ as shown in [5, Chapter 4].

Implementation

Our `SageMath` implementation of DeVries algorithm is available on GitHub inside the notebook DeVries-algorithm.ipynb. The implementation reports various statistics pertaining to the algorithm during execution. Returning to the bivariate generating function

$F(z,w) = \frac{1}{1-z-w}$ we verify that the the lone critical point $(1/2, 1/2)$ determines asymptotics by executing DeVries algorithm. Execution of the code yields two strictly ascending paths which exist the point $(1/2, 1/2)$ leading towards the poles at $(0, 1)$ and $(1, 0)$ of the height function. These are easily visualized in Figure 3.4. The first path approaches $w = 0$ and takes 2643 steps total to do so, the second path approaches $z = 0$ and takes 1021 steps total. We conclude that $(1/2, 1/2)$ is a critical point determining asymptotics as expected. Examining the output we find that on average the step sizes lay inside the interval $[10^{-5}, 10^{-4}]$. In this case it is clear that that larger step sizes work to produce the same results with less computations, hinting that the sufficient conditions for guaranteeing an ascending path on the height map are not necessary. A more optimal step size would significantly improve the efficiency of this algorithm.

As a final example, we show a bivariate generating function to which the methods of Section 3.1 do not apply. The methods of DeVries' algorithm are valid in this case, however the computation of strictly ascending paths is infeasible due to small step sizes.

Example 23 (Supertrees)**.** The rational generating function

$$F(z,w) = \frac{G(z,w)}{H(z,w)} = \frac{2z^2w(2z^5w^2 - 3z^3w + x + 2z^2w - 1}{z^5w^2 + 2z^2w - 2z^3w + 4w + z - 2}$$

has as its main diagonal the generating function counting the combinatorial class of *supertrees*, analyzed by Flajolet and Sedgewick in Example VI.10 of [7]. In their analysis they show that the class of supertree satisfies an algebraic equation and proceed with a univariate analysis. Here we consider a multivariate approach to analyzing this combinatorial class. The methods of Section 3.1 do not apply as the function $F(z, w)$ has a degenerate critical point determining asymptotics. There are 3 solutions to the critical point equations which, ordered by height in decreasing order are

$$(3.236,\ 0.0477), \qquad (2.000,\ 0.125), \qquad (-1.236,\ 0.3273).$$

The point $(2, 1/8)$ is the critical point with degree of degeneracy 4. When computing ascent paths from the critical points the average step size δ which guarantees that the step is strictly ascending is between 10^{-16} and 10^{-17}. As a result it is infeasible to compute paths which we know rigorously are strictly ascending. We may proceed with the analysis by scaling the step size up several orders of magnitude. Doing so we find numerically that the steps appear to be strictly increasing, even though we cannot say so rigorously. Regardless, proceeding with the algorithm we find that the critical point $(3.236, 0.0477)$ has two paths which both lead to $w = 0$, and the critical point $(2, 1/8)$ has three paths which lead to $w = 0$ and one path which leads to $z = 0$. The point $(-1.236, 0.3273)$ is strictly lower in height and therefore the algorithm correctly identifies the strictly minimal point $(2, 1/8)$.

3.4 Future Work and Continuations

We conclude this thesis with a discussion of directions of future work relating to the algorithms discussed in this thesis. In Section 3.1 we saw that the paper [14] uses the structure of the critical point system to analyze the complexity of the system using general techniques of polynomial system solving. The combinatorial case and non-combinatorial case systems are sparse and have a particular structure depending only on the denominator of the corresponding generating function, which naturally leads to the following question.

Is there some way to leverage the structure of the combinatorial and non-combinatorial systems to further improve the efficiency of the solver?

In [11] the authors discuss a heuristic which employs the *monodromy method* of the homotopy continuation library [4]. The monodromy method is potentially a more efficient method of solving polynomial systems, but has more strict necessary conditions that the standard homotopy continuation method. For example, the monodromy method requires a solution to the polynomial system as input. The monodromy method also requires that the *monodromy group* of the polynomial system be transitive if all the solutions are to be computed.

Can we apply the monodromy method to more efficiently solve the combinatorial and non-combinatorial systems?

When building the non-combinatorial case systems we introduced the additional assumption (J2). In [14] the authors show that assumptions (A0), (A1), (A2), (A3) and (J1) all hold generically, and conjecture that assumption (J2) holds generically as well.

Does assumption (J2) hold generically?

In Section 3.3 we saw that DeVries algorithm applies in cases when the methods of Section 3.1 do not. Unfortunately, the step sizes required to ensure that the discrete steps are strictly increasing are extremely small, and in our examples, not optimal.

Can we find better bounds so that discrete steps taken in DeVries algorithm are strictly increasing?

References

[1] Boris Adamczewski and Jason P. Bell. Diagonalization and rationalization of algebraic Laurent series. *Annales scientifiques de l'École Normale Supérieure*, Ser. 4, 46(6):963–1004, 2013.

[2] Paul Breiding. The number of circles that are tangent to 3 given conics. https://www.JuliaHomotopyContinuation.org/examples/circles-conics/. Accessed: June 27, 2022.

[3] Paul Breiding, Kemal Rose, and Sascha Timme. Certifying zeros of polynomial systems using interval arithmetic, 2020.

[4] Paul Breiding and Sascha Timme. HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia. In *International Congress on Mathematical Software*, pages 458–465. Springer, 2018.

[5] Timothy DeVries. *Algorithms for Bivariate Singularity Analysis*. ProQuest LLC, Ann Arbor, MI, 2011. Thesis (Ph.D.)–University of Pennsylvania.

[6] Viviana Ene and Jürgen Herzog. *Gröbner Bases in Commutative Algebra*. American Mathematical Soc., 2011, 01 2011.

[7] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[8] David J Gardner, Daniel R Reynolds, Carol S Woodward, and Cody J Balos. Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 2022.

[9] Benjamin Hackl, Andrew Luo, Stephen Melczer, Jesse Selover, and Elaine Wong. Rigorous analytic combinatorics in several variables in sagemath, 2023.

[10] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[11] Kisun Lee, Stephen Melczer, and Josip Smolčić. Homotopy techniques for analytic combinatorics in several variables. In *Proceedings of the 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 27 – 34, 2022.

[12] Stephen Melczer. An invitation to enumeration. https://enumeration.ca/. Accessed: March 19, 2023.

[13] Stephen Melczer. *Algorithmic and symbolic combinatorics—an invitation to analytic combinatorics in several variables*. Texts and Monographs in Symbolic Computation. Springer, Cham, [2021] ©2021. With a foreword by Robin Pemantle and Mark Wilson.

[14] Stephen Melczer and Bruno Salvy. Effective coefficient asymptotics of multivariate rational functions via semi-numerical algorithms for polynomial systems. *J. Symbolic Comput.*, 103:234–279, 2021.

[15] Robin Pemantle, Mark C. Wilson, and Stephen Melczer. *Analytic Combinatorics in Several Variables, Second Edition.* Cambridge University Press, 2023.

[16] Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[17] Richard P. Stanley. A survey of alternating permutations. In *Combinatorics and graphs*, volume 531 of *Contemp. Math.*, pages 165–196. Amer. Math. Soc., Providence, RI, 2010.

[18] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5)*, 2022. `https://www.sagemath.org`.

[19] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.