

# Lattice Walk Enumeration

Bryan Ek\*

April 18, 2018

## Abstract

Trying to enumerate all of the walks in a 2D lattice is a fun combinatorial problem and there are numerous applications, from polymers to sports. Computers provide a wonderful tool for analyzing these walks; we provide a Maple package for automatically describing generating functions of walks restricted to any step set in a 2D lattice. We always obtain a closed system of relations for generating functions of walks that are bounded, semi-bounded, or unbounded. For bounded walks, this leads to explicit rational solutions! For semi-bounded or unbounded walks, we may get lucky and obtain algebraic solutions; if not, we still have a short self-referential description of the generating function.

## 1 Introduction

We consider walks in the two-dimensional square lattice with an “arbitrary” set of integral steps  $(x, y)$  subject to  $x \geq 0$ . In addition to unbounded walks, we also separately constrain the walks to lie in regions bounded above and below as well as bounded only below.

One would then like to count all possible walks of a certain length possibly with a specific total change in  $y$  value. Rather than a brute-force search of the entire space, looking for 1 value, one could use generating function relations. As a bonus, one would obtain not only the initial generating function of desire, but also many related ones that may be of interest.

Studies of this kind have been done in the literature with simple step sets such as Dyck paths ( $\{[1, 1], [1, -1]\}$ ) [DR71, BORW05] and old-time basketball games [AZ07]. Philippe Duchon

---

\*Department of Mathematics, The School of Arts and Sciences, Rutgers, The State University of New Jersey, Piscataway, NJ 08854

analyzed the case of nonnegative bridges with step set  $\{[1, -2], [1, 3]\}$ ; see OEIS<sup>1</sup> sequence [A060941](#) [OEIa]. For further developments on the subject, see [vR00] and the references therein.

The accompanying Maple package is able to extend and inform on old sequences and create many new sequences. Much of the analysis, thus far, has been on steps with  $x$ -value exactly equal to 1. One of the aspects of this paper that sets it apart is the ease with which it can analyze more generic cases.

## 1.1 Motivation

We want to enumerate walking paths constrained to specific allowable steps. Most of the time we are looking for paths that begin and end on the  $x$ -axis.

An earlier motivation for bounded walks came from Physics: analyzing polymers constrained between plates [BORW05]. Ayyer and Zeilberger gave one solution in an earlier paper [AZ07] that provided the main motivation for this research.

The kernel method has received attention lately for analyzing specific cases of walks [BKK<sup>+</sup>17]. There are several advantages to describing walks using the method of this paper. The main idea is the same: writing functional equations to describe possible steps in a walk. The difference is that this method then describes “new” components of the functional equation in an iterative manner. The kernel method uses analytical number theory on the roots and can be reliant on very case-specific techniques. Compared to the kernel method, we believe our method is a lot easier to understand combinatorially, is more insightful, faster, and easier to produce.

Trying to picture an entire walk at once can be difficult. This is where the awesome powers of dynamical programming come into play. Instead of trying to think about the entirety of a walk, think about a part: either the beginning step, end step, or the middle step across the  $x$ -axis. Break the walk down into different parts (irreducible versus reducible). This is what the generating function equations accomplish.

Originally, we wrote out a single equation to describe the initial walk of interest, then the next, then the next, until it eventually became a closed system. (And the wonderful part is that our descriptions ALWAYS lead to closed systems.) Finally, we solved the system we created.

---

<sup>1</sup>Online Encyclopedia of Integer Sequences [Nei].

Since this is a very algorithmic approach to answering the question, why not have a computer work for us?

## 1.2 Definitions

I will generically use the term *walk* to indicate any sequence of points  $\{(x_0, y_0), \dots, (x_s, y_s)\}$  in the  $xy$ -plane. Though the walk will not necessarily start at the origin, if the starting point is not given, it is assumed to begin at  $(0, 0)$ . The steps of a walk are  $\{(x_1 - x_0, y_1 - y_0), \dots, (x_s - x_{s-1}, y_s - y_{s-1})\}$  and are built from some step set  $\mathcal{S}$ , a set of ordered pairs. I exclusively consider walks that are monotonically weakly increasing:  $x_{i+1} - x_i \geq 0$ . A walk is *nonnegative* (*nonpositive*) if the walk never crosses below (above) the  $x$ -axis. I will sometimes refer to the  $y$ -value as the *altitude* of the walk.

**Definition 1** (Walks). A *bridge* is an unbounded walk that begins at the origin and ends on the  $x$ -axis. I say *bounded bridge* for a bounded walk that begins at the origin and ends on the  $x$ -axis.

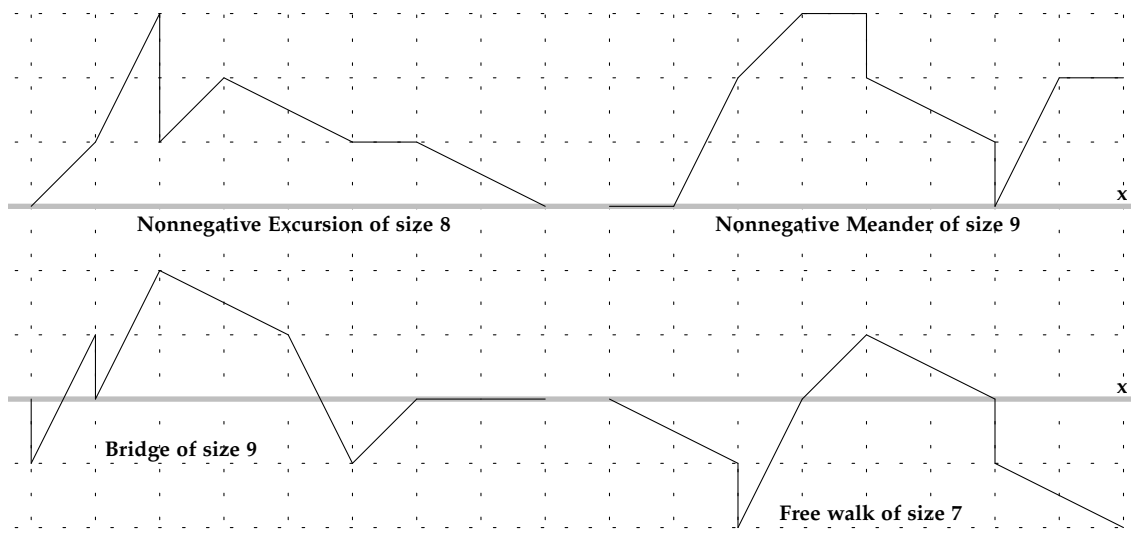
An *excursion* is a semi-bounded (not necessarily nonnegative) walk that begins at the origin and ends on the  $x$ -axis.

A *free* walk can end anywhere (not ending at any specific altitude). A *meander* is a semi-bounded free walk.

The *length* of a walk is  $n = \sum_{i=0}^s x_i$ . The *size* of a walk is  $s$ .

See Figure 1 for examples of the different types of walks.

Figure 1: Walk Examples



All of the example walks are of length 8 and are considered to have begun at the origin. They are all built from the same step set  $\mathcal{S} = \{[0, -1], [1, 0], [1, 1], [1, 2], [2, -1]\}$ . Note the vertical line in the excursion is actually 2 steps. The excursion happens to be irreducible while the rest are not.

Banderier, *et al.* [BKK<sup>+</sup>17] uses *walk/path* to reference any sequence of steps that start at the origin. The only difference in definitions is that they always count excursions and meanders as nonnegative. I will mostly consider them as nonnegative and explicitly say when I am not.

**Definition 2.** The *interior* of a walk consists of every point other than the endpoints:  $\{(x_1, y_1), \dots, (x_{s-1}, y_{s-1})\}$ . An *irreducible* walk is one in which the interior has a strictly higher altitude than the lower endpoint:  $\min\{y_1, \dots, y_{s-1}\} > \min\{y_0, y_s\}$ . For purposes of this paper and Maple package, the stationary walk<sup>2</sup> and walks that are direct steps to the right are NOT considered to be irreducible walks.<sup>3</sup>

Irreducible is also used to refer to walks that do not exactly hit the final altitude until the final step.

<sup>2</sup>A single point.

<sup>3</sup>As the interior (the edge between points) is not strictly higher than the endpoints.

### 1.3 Paper Organization

I will exclusively use  $t$  as the variable in generating functions. I also abbreviate generating function(s) as g.f.(s). This paper is organized in the following sections:

- Section 2:** Bounded: This is the section in which a computer does the best; it can give an exact g.f. solution since we have a “linear” system of equations. Computers are very good at solving these quickly and efficiently.
- Section 3:** Semi-bounded: An algebraic expression for the g.f. is no longer guaranteed. However, we can always find a polynomial in  $\mathbb{Z}[t]$  for which the g.f. is a root.<sup>4</sup>
- Section 4:** Guess-and-check: Finding a minimal polynomial is guaranteed so why not just guess? This section gives a time and memory comparison showing why that would be a poor decision.
- Section 5:** Algebraic-to-Recursive: Using the minimal polynomial of the g.f. is not always the fastest for enumerating. This section introduces another Maple package that converts the polynomial into a 1D recurrence.
- Section 6:** Unbounded: We find minimal polynomials for g.f.s of unbounded walks and show an alternative method of producing recurrences in specific cases.
- Section 7:** Asymptotics: We discover asymptotic results for several step sets to show relationships between the number of excursions, bridges, and, to a lesser degree, meanders.
- Section 8:** Applications: We use the Maple package to find some extended results and talk about probabilistic behavior. We also express minimal polynomials for excursions of small step sets.
- Section 9:** Conclusions and Future Work: A brief recap of what we can do with the **ScoringPaths** Maple package and how we can extend the work.

This paper is produced in conjunction with the Maple package **ScoringPaths**. It is downloadable from

[math.rutgers.edu/~bte14/Code/ScoringPaths/ScoringPaths.txt](http://math.rutgers.edu/~bte14/Code/ScoringPaths/ScoringPaths.txt).

I will mention various functions of the package in **bold**. All functions mentioned in this paper are included in the accompanying Maple package. Some functions have been borrowed from other packages with credit and included in **ScoringPaths** for completeness. All comparisons of time

---

<sup>4</sup>In the formal power series sense.

and memory are done with Maple 17's **CodeTools[Usage]** on Linux version 3.10.0-514.el7.x86\_64 with 8GB of RAM. All values are averages of at least 30 trials unless otherwise specified. Pay careful attention to the units in some examples.

We are considering discrete walks. As such, we can consider steps with only integral values. If the  $x$ -steps are fractional, there will likely not be any issues. If the  $y$ -steps are fractional, many functions will not work as intended. If all of the steps share a common factor in the  $y$ -value, it should be factored out leaving an equivalent problem. The bounds will need to be factored and truncated as well. Leaving the common factor may cause problems with some functions.

The g.f. may produce non-zero coefficients only every  $m^{\text{th}}$  value. It may be desirable to make the substitution  $t \rightarrow t^{1/m}$  for ease of reading. I commonly use  $B(n)$ , and occasionally  $C(n)$  as the number of walks of length  $n$ . The step set and walk restrictions will be obvious from context or given explicitly.

A few functions to create sets of random steps have been included for quick demonstrations; **RandomStepSet** produces a generic set, **RandomZeroStepSet** parses for walks that begin and end on the  $x$ -axis, **RandomSemiBoundedStepSet** parses for walks bounded below, and **RandomUnBoundedStepSet** produces a step set without any  $[0, y]$  steps.

There are several "paper" functions that automatically produce an article with information about a given step set. See the "Paper-Producing Functions" section of the **Help** function.

The package **gfun** by Salvy and Zimmermann [SZ94], a staple included with recent Maple versions, is also very useful for manipulating g.f.s. It contains many functions for translating between algebraic expressions, recurrences, and differential equations satisfied by g.f.s.

## 2 Bounded

The first case is walks that are bounded above and below. Consider an arbitrary set of steps  $\mathcal{S} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ . The goal is to find the g.f., denoted  $f_{a,b}$ , for walks with step set  $\mathcal{S}$ , starting at  $(0, 0)$ , and bounded above and below by  $a \geq 0$  and  $b \leq 0$ , respectively. All walks/paths in this section are constrained to a step set  $\mathcal{S}$  and bounded above and below.

## 2.1 Walking “Anywhere”

First assume that the walk can end anywhere between the lines  $y = a$  and  $y = b$ . A walk either never goes anywhere, +1, or it takes a step ( $t^x$ ) and continues as if it is a walk starting at a new point:  $f_{a-y,b-y}$ . The following relation accomplishes that.

$$f_{a,b} = 1 + \sum_{(x,y) \in \mathcal{S}} t^x f_{a-y,b-y}.$$

This by itself gets us nowhere. But if  $a - y < 0$  or  $b - y > 0$ , then  $f_{a-y,b-y} = 0$  since it is already starting in a prohibited region. Now write out all  $(a - b)$  equations:

$$\begin{aligned} f_{0,b-a} &= 1 + \sum_{(x,y) \in \mathcal{S}} t^x f_{0-y,b-a-y} \\ f_{1,b-a+1} &= 1 + \sum_{(x,y) \in \mathcal{S}} t^x f_{1-y,b-a+1-y} \\ &\vdots \\ f_{a-b,0} &= 1 + \sum_{(x,y) \in \mathcal{S}} t^x f_{a-b-y,0-y}. \end{aligned}$$

This is a (linear!) system of  $a - b$  equations with  $a - b$  variables, after discarding the constant 0 g.f.s, for which Maple’s solve function can easily find the solution. As a bonus, we not only have  $f_{a,b}$ , but also every  $f_{m,n}$  such that  $m - n = a - b$  (and  $m \geq 0, n \leq 0$ ).

The function **BoundedScoringPathsEqsVars** will produce all of the equations and variables for this system extremely quickly. **BoundedScoringPaths** will output only the g.f.  $f_{a,b}$ . Again, this is very fast since it is solving a linear system of only  $(a - b)$  equations. To verify the values, one can use **BoundedScoringPathsNumber**, which computes the number of walks by recursion.

Allowable steps are any as long as there is not a  $(0, -)$  AND a  $(0, +)$  such that they can sum to 0 within the bounds. Specifically, if  $(0, m)$  and  $(0, -n)$  are steps, then it is allowable (will not produce infinite values) as long as the width is small enough:  $a - b < n + m - \gcd(m, n)$ .

*Proof.* Let  $\mathcal{S} = \{[0, m], [0, -n]\}$  and assume  $m > n$ . Also assume  $\gcd(m, n) = 1$ . If  $a - b \geq n + m - 1$ , then we can always take at least one of the  $\mathcal{S}$  steps. Since there is a finite number of altitudes, then must be acollision at some point: a loop.

If  $\gcd(m, n) = g > 1$ , then reconsider the same problem after factoring  $g$  out of everything:  $a' = \lfloor \frac{a}{g} \rfloor, b' = \lfloor \frac{b}{g} \rfloor, \mathcal{S}' = \{[0, m/g], [0, -n/g]\}$ . ■

This bound only works for two 0-steps. For more than two 0-steps, the width must be even shorter. For some combinations (e.g.,  $\{[0, u - v], [0, -u], [0, u + v]\}$ ), there is no allowable width that includes all steps.<sup>5</sup>

### 2.1.1 Examples

**Example 3** (Close (American) Football Games). Consider trying to enumerate the number of American football games in which the teams are never separated by more than 1 score. On a given play,<sup>6</sup> it is possible for one team to score 2, 3, 6, 7, or 8 points. It is also possible for one team to score 6 points and the opposing team to score 1 or 2 points, though these are pretty rare occurrences. What if we want to enumerate the number of games with  $n$  scoring plays that end separated by no more than 1 score? We can use the step set

$$\mathcal{S} = \{[1, 2], [1, 3], [1, 6], [1, 7], [1, 8], [1, 5], [1, 4], \\ [1, -2], [1, -3], [1, -6], [1, -7], [1, -8], [1, -5], [1, -4]\}.$$

Each step represents 1 scoring play and how many points the home team gained relative to the away team. Since the maximum scoring play is 8 points, we make the bounds  $y = -8, 8$ . The g.f. is then found with **BoundedScoringPaths**( $\mathcal{S}, 8, -8, t$ ):

$$\frac{1 + 10t + 13t^2 - 37t^3 - 40t^4 + 28t^5 + 26t^6 - 2t^7}{1 - 4t - 59t^2 - 77t^3 + 170t^4 + 234t^5 - 92t^6 - 142t^7 - 4t^8 + 6t^9}.$$

This sequence is new in the OEIS: [A301379](#) [OEIb].

**Example 4** (Speed Enumeration - Bounded 1). Consider finding the first 1000 coefficients of the g.f. found in Example 3 above. We could use brute-force recursion in **BoundedScoringPath-Number** or take a Taylor series expansion since we have an explicit form for the g.f. from **BoundedScoringPaths**.

**Table 1:** Bounded Walk Enumeration

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Brute-Force Recursion	60.23MiB	24MiB	309.53ms	308.90ms
G.F. Construction	3.27MiB	0 bytes	30.17ms	33.50ms
Taylor Enumeration	4.86MiB	0 bytes	4.93ms	5.07ms
Total	8.13MiB	0 bytes	35.10ms	38.57ms

<sup>5</sup>Assuming that one can get to the edges of the boundary.  $(u + v) + (-u) + (u - v) + (-u) = 0$ : a loop.

<sup>6</sup>Counting untimed downs as part of the previous play.



Using the g.f. method of enumeration is about 8 times as fast and uses a much smaller amount of memory: (1/8th).

## 2.2 Returning to the $x$ -axis

The above equations are for g.f.s of walks that are able to end ANYWHERE. What if we want a g.f. for walks that must end back at the  $x$ -axis? Ayer and Zeilberger solved the bounded bridge case (but not the bounded free walk case) using different relations but the same general method [AZ08]; they also use the equivalent problem of walks between parallel lines of positive slope instead of between horizontal lines.

Let  $f_{a,b}$  now denote the g.f. for walks that begin at  $(0,0)$  and end on the  $x$ -axis. Again, we either never get moving,  $+1$ , or we take a step and then must take a path back to the  $x$ -axis. Let  $e_{a,b,c}$  denote the g.f. for paths that start at  $(0,c)$ , end on the  $x$ -axis and never touch the  $x$ -axis beforehand. Now that  $e_{a,b,c}$  is introduced, we can write the relation for  $f_{a,b}$ :

$$f_{a,b} = 1 + \left( \sum_{(x,0) \in \mathcal{S}} t^x \right) f_{a,b} + \left( \sum_{(x,y) \in \mathcal{S}; y \neq 0} t^x e_{a,b,y} \right) f_{a,b}. \quad (2.1)$$

There is also the possibility of just moving along the  $x$ -axis to start. After returning to the  $x$ -axis, we can now take any walk as before. Hence the multiplication by  $f_{a,b}$ . We now need the equations for  $e_{a,b,c}$  for  $a \geq c \geq b$ . If any step would return to the  $x$ -axis, then we are done. Otherwise, it must continue as a new  $e_{a,b,c'}$  path.

$$e_{a,b,c} = \sum_{(x,-c) \in \mathcal{S}} t^x + \sum_{(x,y) \in \mathcal{S}; y \neq -c} t^x e_{a,b,c+y}.$$

We now have a set of linear equations to solve for the  $e_{a,b,c}$ . To produce all of the equations and variables, use **EqualBoundedScoringPathsEqsVars**. Once the  $e_{a,b,c}$  system is solved, Eqn. (2.1) is linear in  $f_{a,b}$  so we (or rather a computer) will find a rational g.f. solution! Use **EqualBoundedScoringPaths** for the single solution. To verify the result, one can check using the enumeration in **EqualBoundedScoringPathsNumber**. The irreducible g.f.s can be checked with **SpecificIrreducibleBoundedScoringPathsNumber**.

### 2.2.1 Old-Time Basketball

The methods of relating generating functions were inspired by Ayer and Zeilberger's work [AZ07]. They found the following relation.

**Theorem 5.** Let  $F_w$  denote the generating function for the number of walks subject to step set  $\mathcal{S} = \{[\frac{1}{2}, 1], [\frac{1}{2}, -1], [1, 2], [1, -2]\}$ , that start at  $(0, 0)$ , end at  $(n, 0)$ , and never go below the  $x$ -axis or above the line  $y = w$ . Then  $F_w$  satisfies the following recurrence relation:

$$F_w = 1 - tF_w + 2tF_wF_{w-1} + 2t^2F_wF_{w-1}F_{w-2} - (t^3 + t^4)F_wF_{w-1}F_{w-2}F_{w-3} + t^5F_wF_{w-1}F_{w-2}F_{w-3}F_{w-4}.$$

**Proposition 6.** The initial conditions are given as follows:

$$F_0 = 1, \quad F_1 = \frac{1}{1-t}, \quad F_2 = \frac{1-t}{1-2t+3t^2},$$

$$F_3 = \frac{1-2t+3t^2}{1-3t-5t^2-2t^3+t^4}, \quad F_4 = \frac{1-3t-5t^2-2t^3+t^4}{1-4t-6t^2+2t^3}.$$

If we compute `EqualBoundedScoringPaths([1/2,1],[1/2,-1],[1,2],[1,-2],w,0,t)` for  $w = 0, \dots, 4$ , then the initial conditions match. And we can verify Theorem 5 empirically for any fixed  $w$ .

## 2.2.2 Examples

**Example 7** (Tied (American) Football Games). Consider the similar Example 3, but this time we want the teams to be tied at the end of the game. Our step set is the same. The only difference is in which method we use: `EqualBoundedScoringPaths(S,8,-8,t)`:

$$\frac{1-4t-45t^2-43t^3+98t^4+108t^5-24t^6-30t^7}{1-4t-59t^2-77t^3+170t^4+234t^5-92t^6-142t^7-4t^8+6t^9}.$$

This sequence is new in the OEIS: [A301380](#) [OEIc].

**Example 8** (Speed Enumeration - Bounded 2). Once again let us find the first 1000 coefficients of a g.f.: this time the one found in Example 7 above.

**Table 2:** Bounded Bridge Enumeration 1000 terms

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Brute-Force Recursion	55.98MiB	24MiB	278.93ms	279.03ms
G.F. Construction	10.13MiB	8MiB	102.43ms	103.20ms
Taylor Enumeration	4.85MiB	0 bytes	5.20ms	5.10ms
Total	14.98MiB	8MiB	107.63ms	108.30ms

Again, the g.f. method is faster: this time, about 2.5 times as fast and 1/4th the memory.

### 3 Semi-bounded

We now remove the restriction of bounding walks from above.<sup>7</sup> Ayyer and Zeilberger also provided relations for describing excursions that are similar to those included here [AZ08]. Duchon had previously tackled the case of excursions using much different language, but the same overall method [Duc00].

Let  $\mathcal{S}$  denote the set of steps as in the previous section. Let  $f$  now denote the g.f. for nonnegative excursions with step set  $\mathcal{S}$  that begin at  $(0,0)$  and end on the  $x$ -axis. How will we describe  $f$  in an equation? We could try to use the same method as Section 2.2 and write

$$f = 1 + \left( \sum_{(x,0) \in \mathcal{S}} t^x \right) f + \left( \sum_{(x,y) \in \mathcal{S}; y \neq 0} t^x e_y \right) f.$$

where  $e_y$  is the g.f. for walks that start at  $(0,y)$ , end at  $(n,0)$  and never hit the  $x$ -axis beforehand. However, to describe  $e_y$  as in Section 2.2 (by looking at the first step) would require writing equations for all  $e_{i>0}$  since a walk could get arbitrarily far from the  $x$ -axis before returning.<sup>8</sup> At some point, all of the  $e_i$  equations would look essentially the same and it may be possible to take limits of their form to find a solution for  $e_y$ . However, there is an easier method.

Ayyer and Zeilberger [AZ07] used a standard idea in combinatorics of “irreducible” walks to describe  $F_w$  in Section 2.2.1. We will use them as well. Let  $f_{a,b}$  denote the g.f. for nonnegative walks that start at  $(0,a)$  and end at  $(n,b)$ . Note that  $f_{0,0} = f$  is what we are typically looking for. Let  $g_{a,b}$  denote the g.f. for walks that start at  $(0,a)$ , end at  $(n,b)$ , and stay above the line  $y = \min(a,b)$  except at the respective endpoint. Note then that  $g_{a,b} = g_{a-b,0}$  (or  $= g_{0,b-a}$  if  $b > a$ ).<sup>9</sup> Then

$$f_{0,0} = 1 + \left( g_{0,0} + \sum_{(x,0) \in \mathcal{S}} t^x \right) f_{0,0}. \quad (3.1)$$

Either the walk is stationary (+1) or it returns to the  $x$ -axis ( $g_{0,0} + \sum_{(x,0) \in \mathcal{S}} t^x$ ) and then continues as if it were new (multiplication by  $f_{0,0}$ ).

$$g_{0,0} = \sum_{(x_1,y_1) \in \mathcal{S}; y_1 > 0} \sum_{(x_2,y_2) \in \mathcal{S}; y_2 < 0} t^{x_1} f_{y_1-1, -y_2-1} t^{x_2}. \quad (3.2)$$

<sup>7</sup>To look at walks solely bounded above, simply change the sign of the  $y$ -value of every step. Or note that every nonnegative excursion is in bijection with a nonpositive excursion by reversing the order of steps.

<sup>8</sup>We could instead look at the final step, but that is essentially the method described in the next paragraph.

<sup>9</sup>In this definition,  $g_{0,0}$  does not include stationary walks or walks that are solely a step directly to the right. If this is what one desires, then use  $g = g_{0,0} + 1 + \left( \sum_{(x,0) \in \mathcal{S}} t^x \right)$ .

An irreducible  $(0,0)$ -walk can be characterized by how it departs from the  $x$ -axis ( $t^{x_1}$ ) and how it returns ( $t^{x_2}$ ). There must be an intermediate walk between these two steps ( $f_{y_1-1,-y_2-1}$ ) that does not touch the  $x$ -axis: hence the shift. Now we need to describe each new  $f_{a,b}$ .

$$a > b \quad f_{a,b} = \sum_{i=0}^b g_{a-i,0} f_{0,b-i}, \quad (3.3)$$

$$a = b \quad f_{a,a} = \sum_{i=0}^{b-1} g_{a-i,0} f_{0,b-i} + f_{0,0}, \quad (3.4)$$

$$a < b \quad f_{a,b} = \sum_{i=0}^{a-1} g_{a-i,0} f_{0,b-i} + f_{0,0} g_{0,b-a}. \quad (3.5)$$

We can characterize a walk by how close it gets to the  $x$ -axis. An  $f_{a,b}$  walk can go  $i$  levels below the level of  $y = \min(a,b)$ . In this case, we have an irreducible walk down to the lowest point ( $g_{a-i,0}$ ), and then an arbitrary walk to the final level that does not go any lower ( $f_{0,b-i}$ ). If  $a = b$ , then a walk that does not go below level  $y = a$  is equivalent to  $f_{0,0}$ . If  $a < b$ , then a walk that does not go below level  $y = a$  consists of an arbitrary walk back to the same level without going lower ( $f_{0,0}$ ), followed by an irreducible walk to the final level of  $y = b$  ( $g_{0,b-a}$ ). Now we must describe the irreducible walks that have not been covered.

$$g_{a,0} = \sum_{(x,y) \in \mathcal{S}; y < 0} f_{a-1,-y-1} t^x, \quad (3.6)$$

$$g_{0,a} = \sum_{(x,y) \in \mathcal{S}; y > 0} t^x f_{y-1,a-1}. \quad (3.7)$$

An irreducible walk can be characterized by how it reaches ( $t^x$ ) the lowest point. The rest of the walk is arbitrary as long as it does not hit the  $x$ -axis: again, hence the shift.

We need to show that this iterated process does eventually terminate. This can easily be justified because the largest index of either  $g$  or  $f$  will be  $< \max_{(x,y) \in \mathcal{S}} |y|$ : the maximum step size. Thus, there is a finite number of  $g, f$  that we need to describe in order to have a closed system. Use **EqualSemiBoundedScoringPathsEqsVars** to generate these variables and their equations.

The equations are no longer linear and so Maple's built-in solve function does not work as nicely. Instead, we use the **Basis** procedure in the **Groebner** package to find

**Definition 9** (Minimal Polynomial). the (minimal) algebraic equation satisfied by the generating function:  $p$  such that  $p(f) = 0$  in terms of formal power series. We refer to  $p$  as the *minimal polynomial*. When discussing the *degree* of the minimal polynomial, we are considering the degree of  $f$  in  $p$  unless explicitly stated that we are considering  $t$ .

To produce this polynomial, use **EqualSemiBoundedScoringPaths**. To produce an “ideal” polynomial for each  $g, f$ , use **AllEqualSemiBoundedScoringPaths**.<sup>10</sup> If you want the minimal polynomial for only a single variable, use **SpecificEqualSemiBoundedScoringPaths**. The minimal polynomial is typically found when the Groebner basis receives the desired variable as the lowest order lexicographically. Occasionally the Groebner basis will not produce the minimal polynomial, but instead a product of it and another polynomial. One can recover the minimal polynomial by factoring the output and testing which factor satisfies  $p(f) = 0$  in terms of formal power series.<sup>11</sup> Every function that outputs an algebraic expression has had **FindProperRoot** appended to the end to properly parse the minimal polynomial.

**Example 10** (Semi-bounded Example). Let  $S = \{[0, -3], [1, -2], [2, 0], [3, 1]\}$ . The g.f.,  $F$ , for the number of excursions that do not go below  $y = -1$  and have step set  $S$  is found to satisfy, using **EqualSemiBoundedScoringPaths(S,-1,t,F)**,

$$t^{18}F^4 + t^{14}(t^2 - 1)F^3 + 2t^7(t^2 - 1)^2F^2 + (t^2 - 1)^5F + (t^2 - 1)^4 = 0.$$

This is actually a fairly simple answer. If we change  $[0, -3] \rightarrow [0, 3]$ , then the minimal polynomial is degree 10 in  $F$  and takes 4 lines to write.

This polynomial can then be used to discover the enumeration hidden in the coefficients of the Taylor series expansion by setting  $f_0 = 1$  and then iterating  $f_0 \rightarrow p(f_0) + f_0$  to find a fixed point solution.<sup>12</sup> Finding the coefficients in this manner requires finding  $p$ : use **EqualSemiBoundedScoringPathsCoefficients**.

Finding the minimal polynomial can be a time-consuming process. An alternative method is to iterate a fixed-point solution of a vector of all  $g, f$  and then pick out  $f_{0,0}$ . Initialize  $\{f_{a,a} = 1\}$  and every other g.f. to 0.<sup>13</sup> Now iterate all of the  $g, f$  into their respective equations, truncating to the desired coefficient. Eventually we will reach a fixed-point for the vector of solutions. To do this, use **EqualSemiBoundedScoringPathsSeries**. However, for most reasonable calculations of the coefficients ( $< 1000$  terms), finding the values via brute-force recursion (with **SpecificEqualBoundedScoringPathsNumber**) is faster than either iterating technique.

**Example 11** (Speed Enumeration - Excursions). We want to obtain the number of nonnegative

<sup>10</sup>This can be slow as it requires finding a Groebner basis for *every* variable.

<sup>11</sup>By checking a truncated version of  $f$ . In most cases  $f = 1$  is sufficient to see which factor works. In general, enumerate  $m$  terms first using the proper enumerating function and see if the corresponding polynomial is  $O(t^m)$ , signifying a root.

<sup>12</sup>This glosses over why the convergence works. It becomes an issue in the unbounded case.

<sup>13</sup>Staying stationary is only valid for general walks that begin and end at the same level.

excursions with step set  $S = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . Let  $F$  denote the corresponding g.f.; then<sup>14</sup>

$$t^4F^4 - t^2(t+1)F^3 + t(t+2)F^2 - (t+1)F + 1 = 0. \quad (3.8)$$

A truncated solution in formal power series, and the one that makes sense in terms of our problem, is

$$F = 1 + t + 3t^2 + 9t^3 + 32t^4 + 120t^5 + 473t^6 + 1925t^7 + 8034t^8 + 34188t^9 + 147787t^{10}.$$

We compare and contrast the amount of time and memory to enumerate the first 500 and 1000 coefficients of  $F$  in the following tables.

1. Set up the g.f. equations and then iterate a vector of solutions using **EqualSemiBoundedScoringPathsSeries**.
2. Solve for the minimal polynomial (Eqn. (3.8)) and iterate a single solution with **EqualSemiBoundedScoringPaths**.
3. Use Maple's **taylor** function on the minimal polynomial.<sup>15</sup>
4. Use brute-force recursion and Maple's option remember: **SpecificEqualSemiBoundedScoringPathsNumber**.

**Table 3:** 500 term Excursion Enumeration

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Vector Set-Up	16.58KiB	0 bytes	300 $\mu$ s	766 $\mu$ s
Iterating	100.04GiB	55.64MiB	4.45m	4.09m
Total	100.04GiB	55.64MiB	4.45m	4.09m
Polynomial	2.63MiB	0 bytes	19.23ms	18.63ms
Iterating	23.61GiB	20.84MiB	64.52s	58.56s
Total	23.61GiB	20.84MiB	64.54s	58.58s
Polynomial	2.63MiB	0 bytes	19.23ms	18.63ms
taylor	328.97MiB	53.85MiB	3.20s	3.18s
Total	331.60MiB	53.85MiB	3.22s	3.20s
Brute-Force Recursion	186.60MiB	328MiB	1.569s	1.472s

<sup>14</sup>Using **EqualSemiBoundedScoringPaths(S,0,t,F)**.

<sup>15</sup>Requires replacing  $F$  by  $_Z$  and using **RootOf(p)**.

Finding the polynomials takes negligible time and memory in comparison to actually enumerating the coefficients beyond the first few terms.

**Table 4:** 1000 term Excursion Enumeration

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Vector Iterating	1.31TiB	159.26MiB	63.84m	51.86m
Single Iterating	300.84GiB	15.32MiB	14.50m	11.93m
taylor	1.82GiB	489.68MiB	12.20s	11.12s
Brute-Force Recursion	0.85GiB	508MiB	7.53s	7.13s

So using recursion is the fastest, but also must reserve the most memory. The other methods use more memory in total but can recycle much of what they used. We can best all of the methods with another recursion that is faster and uses less memory; see Example 16.

Before implementing **FindProperRoot**, the Groebner basis output a degree 5 polynomial that took over 5 times as long to iterate as the degree 4 minimal polynomial. Whether iterating a single polynomial or the entire vector of solutions is faster depends on the degree of the minimal polynomial and the number of variables in the closed system. And the number of terms to enumerate. An interesting question would be to look at the time-complexity of this method of enumeration. The vector and single polynomial iteration could potentially be optimized to pick out coefficients from each monomial instead of expanding everything and then picking only relatively few terms. This would help the minimal polynomial iteration much more as the vector iteration only relies on degree 2 expressions.

### 3.1 Arbitrary Lower Limit

What if we want to consider walks that stay above an arbitrary lower bound  $y = -c$  as in Example 10? This is actually very easy. By shifting the walk to be nonnegative, we are now looking for  $f_{c,c}$ . Describe  $f_{c,c}$  using Eqn. (3.4). Again, iterating on all new  $g, f$  will eventually yield a closed system since the indices are bounded by  $\max(\max_{(x,y) \in \mathcal{S}} |y| - 1, c)$ .

This shifting technique is what the Maple package utilizes when it is given a lower limit other than 0 for semi-bounded walks. The Maple functions are always focused on obtaining g.f.s for walks that begin at the origin: other produced g.f. are purely bonus. To obtain a g.f. that

starts at  $(0, c)$ , input a lower limit of  $-c$ , and take the g.f. that corresponds to starting at the origin.

## 3.2 Meanders

What if we do not care where the walk ends, as long as it stays above  $y = -c$ ? Neither Ayer and Zeilberger [AZ08] nor Duchon [Duc00] investigated semi-bounded free walks. In this case, we can actually utilize the irreducible walks we have just created. Recall the irreducible g.f.s in Eqns. (3.6) and (3.7).

Let  $k_a$  denote the g.f. for nonnegative meanders that begin at  $(0, a)$ , restricted to step set  $\mathcal{S}$ . Then

$$k_0 = 1 + \left( g_{0,0} + \sum_{(x,0) \in \mathcal{S}} t^x \right) k_0 + \sum_{(x,y) \in \mathcal{S}; y > 0} t^x k_{y-1}.$$

The walk can be stationary (+1), it can return to the  $x$ -axis ( $g_{0,0} + \sum_{(x,0) \in \mathcal{S}} t^x$ ) and continue as a new meander ( $k_0$ ), or we take the first step ( $t^x$ ) and continue as a new meander that never returns to the  $x$ -axis ( $k_{y-1}$ ).  $g_{0,0}$ , the g.f. for irreducible walks that return to the  $x$ -axis, already has a description from our previous work. We need only describe the new  $k_i$ .<sup>16</sup>

$$k_a = \sum_{i=0}^{a-1} g_{a,i} k_0 + k_0 = \left( \sum_{i=1}^a g_{i,0} + 1 \right) k_0.$$

The meander can drop down to any lower altitude ( $g_{a,i} = g_{a-i,0}$ ) and then continue as a new meander ( $k_0$ ), never dropping further. Or the meander will never go below  $y = a$  so it is equivalent to a meander from the origin ( $k_0$ ).

And since we have already described the irreducible walks earlier, we now have a closed system that we can use to solve for  $k_0$ . To produce the entire system of equations, use **SemiBoundedScoringPathsEqsVars**. Again, we use **Groebner[Basis]** to find a minimal polynomial: **SemiBoundedScoringPaths**. To find the minimal polynomials for ALL<sup>17</sup> of the variables, use **AllSemiBoundedScoringPaths**, though this will likely take a while. For a specific  $k_a$ , use **SpecificSemiBoundedScoringPaths**. For any  $g_{a,b}, f_{a,b}$ , it is best<sup>18</sup> to use **SpecificEqualSemiBoundedScoringPaths**.

**Example 12.** For comparison, we use the same step set as in Example 10:  $\mathcal{S} = \{[0, -3], [1, -2], [2, 0], [3, 1]\}$ . The g.f.,  $K$ , for the number of meanders that do not go below  $y = -1$  and have step set  $\mathcal{S}$  is found

<sup>16</sup>If they exist.

<sup>17</sup>Including all of the irreducible and specific altitude walks from the previous section.

<sup>18</sup>And necessary. Compatibility was removed for ease of use.



to satisfy, using **SemiBoundedScoringPaths(S,-1,t,K)**,

$$\begin{aligned}
& t^{19} (t^2 + t + 1) K^4 + t^{12} (4t^6 + t^4 + 3t^3 + 6t^2 + t - 3) K^3 \\
& + 3t^6 (2t^9 - 2t^8 + t^7 + 4t^6 + 2t^5 - 2t^4 - t^3 + 3t^2 - 1) K^2 \\
& + (4t^{12} - 8t^{11} + 7t^{10} + 11t^9 - 7t^8 - 3t^7 + 7t^6 + 6t^5 - 6t^4 - 2t^3 + 4t^2 - 1) K \\
& + t^9 - 3t^8 + 4t^7 + 2t^6 - 6t^5 + 4t^4 + 3t^3 - 3t^2 + 1 = 0.
\end{aligned}$$

Sadly, the coefficients do not factor as nicely as in Example 10.

Now that we have a method of describing the g.f. for meanders, let us compare how fast it is for enumeration.

**Example 13** (Speed Enumeration - Meanders). We use the simpler step set  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$  and let  $K$  be the g.f. for the number of nonnegative meanders with step set  $\mathcal{S}$ . Then<sup>19</sup>

$$t^2(5t - 1)^2 K^4 + t(5t - 1)^2 K^3 + 3t(5t - 1) K^2 + (5t - 1) K + 1 = 0. \quad (3.9)$$

A truncated solution is

$$K = 1 + 3t + 12t^2 + 51t^3 + 226t^4 + 1025t^5 + 4724t^6 + 22022t^7 + 103550t^8.$$

We repeat our analysis of differing methods of enumeration from Example 11.

1. Iterating a vector of solutions using **SemiBoundedScoringPathsSeries**.
2. Iterating a fixed point solution after solving for the polynomial (Eqn. (3.9)) with **SemiBoundedScoringPaths**.<sup>20</sup>
3. Using Maple's **taylor** function on the minimal polynomial.<sup>21</sup>
4. Enumerating using brute-force recursion and Maple's option **remember**: **SpecificSemiBoundedScoringPathsNumber**.

---

<sup>19</sup>Using **SemiBoundedScoringPaths(S,0,t,K)**.

<sup>20</sup>All of this can be accomplished with the one function **SemiBoundedScoringPathsCoefficients**.

<sup>21</sup>Requires replacing  $K$  by  $\_Z$  and using **RootOf(p)**.

Table 5: 1000 term Meander Enumeration

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Vector Set-Up	21.07KiB	0 bytes	233 $\mu$ s	233 $\mu$ s
Vector Iterating	2.18TiB	15.29MiB	102.64m	84.48m
Total	2.18TiB	15.29MiB	102.64m	84.48m
Polynomial	6.37MiB	0 bytes	43.00ms	43.03ms
Single Iterating	304.99GiB	30.58MiB	13.88m	11.64m
Total	305.00GiB	30.58MiB	13.88m	11.64m
Polynomial	6.37MiB	0 bytes	43.00ms	43.03ms
taylor	1.85GiB	490MiB	11.33	10.62s
Total	1.86GiB	490MiB	11.37s	10.66s
Brute-Force Recursion	1.08GiB	0.52GiB	7.93s	7.465s

Again, there is an even faster recursive formula. See Example 17.

## 4 Guess-and-Check Method

Zeilberger provided a guess-and-check method and Maple package, **WID** [EZ15], for finding algebraic expressions. In the semi-bounded case where all steps have  $x$ -step 1, Phillippe Duchon guaranteed that the results are algebraic. Thus, the guess-and-check method WILL work, eventually, if you set the search parameter high enough. For semi-bounded cases with differing  $x$ -steps and unbounded cases, there is no guarantee that guessing will eventually work [MR09]. Though you may get lucky and produce an algebraic equation that has the minimal polynomial as a root.

Another advantage of this paper's method over guess-and-check is that this method is typically much much faster. Setting up the equations takes a set amount of time that is linear in the maximum step size,  $|\mathcal{S}|$ , and the lower limit. The time sink comes in finding a Groebner basis, but this is still typically faster.

**Empir** will take a list of the first few coefficients of the g.f.  $F$  and attempt to find an algebraic equation that  $F$  satisfies by guessing the degree and trying to solve for the coefficients.

**EmpirF** will do this faster<sup>22</sup> by utilizing the **gfun** package. Both **Empir** and **EmpirF** require enumerating the first few terms.

**Example 14** (Excursion Time Comparison). Consider looking for an algebraic equation for the g.f. for nonnegative excursions with step set  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . It takes<sup>23</sup>

**Table 6:** Finding Excursion Minimal Polynomial  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$

Method	Memory Used	Memory Allocation	CPU Time	Real Time
<b>ESBSP</b>	2.63MiB	0 bytes	19.23ms	18.63ms
<b>Empir</b>	91.61MiB	5.60MiB	734ms	735ms
<b>EmpirF</b>	4.33MiB	0 bytes	33.87ms	35.90ms

**EqualSemiBoundedScoringPaths (ESBSP)** takes about 3% of the time and memory that **Empir** requires and 60% of the time and memory as **EmpirF**.

**Example 15** (Meander Time Comparison). Now try to find an algebraic equation satisfied by the g.f. for nonnegative meanders with step set  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ .<sup>24</sup>

**Table 7:** Finding Meander Minimal Polynomial  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$

Method	Memory Used	Memory Allocation	CPU Time	Real Time
<b>S BSP</b>	6.37MiB	0 bytes	43.00ms	43.03ms
<b>Empir</b>	91.68MiB	5.45MiB	749ms	770ms
<b>EmpirF</b>	4.74MiB	0 bytes	36.53ms	38.63ms

**S BSP=SemiBoundedScoringPaths.**

As expected, this package is still much faster than **Empir**. Interestingly, **EmpirF** appears to be as fast, if not a little faster. **EmpirF** is still handicapped in its range of applications and so the slight speed-up is sacrificed for versatility.

In fact, adding  $[1, 3]$  to the step set already makes **EmpirF** fail for the preset search bound. The minimal polynomial has degree and order 10 in that case: easily found by **SemiBoundedScoringPaths** in 1/4s.

<sup>22</sup>Almost always.

<sup>23</sup>The actual minimal polynomial is shown in Example 11.

<sup>24</sup>The minimal polynomial is shown in Example 13.

The other bonus is that this paper's method allows for any size  $x$ -step. **W2D** could potentially be used to solve this problem once it has been restricted to look for the coefficient of  $x^n \cdot y^0$ . However, this would make the already slow guessing method even slower.

## 5 Algebraic to Recursive

### 5.1 Conversion

There is a classical method for deducing from the algebraic function satisfying the g.f. a linear recurrence with polynomial coefficients satisfied by the coefficients of the g.f. in question. See Chapter 6 of "The Concrete Tetrahedron" by Kauers and Paule [KP11].

The method is implemented in the Maple package `gfun` [SZ94] and also in Zeilberger's Maple package **SCHUTZENBERGER**, that is used here. The **SCHUTZENBERGER** package also contains **EmpirF** for obtaining the minimal polynomial, but we now have a much better method of producing the minimal polynomial. To convert an algebraic formula to a recurrence formula, use **algtorec**. Let  $B(n)$  denote the number of walks of length  $n$ .

Interestingly, sometimes a larger (non-minimal) polynomial produces a better (lower-order) recurrence.

**Example 16** (Better Excursion Recursion). Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . Let  $F$  denote the g.f. for nonnegative excursions.<sup>25</sup> Then its coefficients satisfy

$$\begin{aligned}
0 = & 3125(n+1)(n+2)(n+3)(n+4)B(n) \\
& - 250(n+4)(n+3)(n+2)(27n+122)B(n+1) \\
& + 25(n+4)(n+3)(107n^2+1457n+4316)B(n+2) \\
& + 10(n+4)(304n^3+3233n^2+9864n+6513)B(n+3) \\
& - (2821n^4+56794n^3+425771n^2+1407974n-1731540)B(n+4) \\
& + 2(n+7)(413n^3+6986n^2+39356n+73830)B(n+5) \\
& - (n+8)(n+7)(99n^2+1241n+3900)B(n+6) \\
& + 2(2n+15)(n+9)(n+8)(n+7)B(n+7).
\end{aligned} \tag{5.1}$$

---

<sup>25</sup>The minimal polynomial is given in Example 11.

The following are the time and memory requirements for various stages of enumerating. We need to create the minimal polynomial with **EqualSemiBoundedScoringPaths**, convert it to an improved recursive formula, Eqn. (5.1), for the coefficients of  $F$  with **algtorec**, and then enumerate with **SeqFromRec**.

**Table 8:** Enumerating Excursions More Efficiently

Method	Memory Used	Memory Allocation	CPU Time	Real Time
<b>ESBSP</b>	2.63MiB	0 bytes	19.23ms	18.63ms
<b>algtorec</b>	35.84MiB	3.12MiB	267.93ms	262.40ms
<b>SeqFromRec</b>	123.05MiB	24MiB	375.27ms	375.37ms
Total	161.52MiB	27.12MiB	662.43ms	656.40ms

**ESBSP=EqualSemiBoundedScoringPaths**

As expected, this streamlined recurrence is much faster and less memory-intensive than the basic recurrence in Example 11. There is an up-front cost for creating the improved recurrence, but if the goal is to enumerate enough terms, it can be worth it. In this case, enough is less than 500.

Before **FindProperRoot** was implemented to automatically parse the minimal polynomial, we converted a larger polynomial into a sixth-order recurrence in about 2.4 seconds.

**Example 17** (Better Meander Recursion). Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$  and  $K$  denote the g.f. for nonnegative meanders.<sup>26</sup> The coefficients now satisfy

$$\begin{aligned}
0 = & 625(n+1)(n+2)(n+3)(n+4)B(n) \\
& - 250(5n+21)(n+4)(n+3)(n+2)B(n+1) \\
& + 50(n+4)(n+3)(7n^2+95n+270)B(n+2) \\
& + 20(n+4)(32n^3+367n^2+1365n+1620)B(n+3) \\
& - (n+5)(463n^3+6691n^2+32442n+52704)B(n+4) \\
& + 2(n+5)(n+6)(53n^2+593n+1674)B(n+5) \\
& - 4(n+5)(2n+13)(n+7)(n+6)B(n+6).
\end{aligned}$$

Oddly it is a lower order recurrence despite the slightly higher complexity of describing meanders. The following are the time and memory requirements for various stages of enumerating.

<sup>26</sup>The minimal polynomial is given in Example 13.

**Table 9:** Enumerating Meanders More Efficiently

Method	Memory Used	Memory Allocation	CPU Time	Real Time
<b>SBSP</b>	6.37MiB	0 bytes	43.00ms	43.03ms
<b>algtorec</b>	110.77MiB	29.39MiB	659.73ms	633.10ms
<b>SeqFromRec</b>	89.58MiB	4MiB	284.57ms	257.23ms
Total	206.72MiB	33.39MiB	987.30ms	933.36ms

SBSP=SemiBoundedScoringPaths

This improved recurrence is about 8 times as fast as the basic recurrence in Example 13. It is not quite as much of a savings as the case of excursions, but it is still extremely good.

## 5.2 Searching

There is an alternative to converting the minimal polynomial into a linear recurrence. Since we know that this will be possible, we could simply guess at the form of the recurrence and use a suitable number of starting values to determine the coefficients. **Findrec** will accomplish this guessing method. The problem is that we do not know an upper bound for the order and degree. Setting the bound extremely high (or searching until a recurrence is found) would suffice. However, this is not ideal.

**Example 18.** Let  $\mathcal{S} = \{[1, 2], [1, -3]\}$ . The minimal polynomial is given later in Section 8.3.2. Let  $B(n)$  denote the number of nonnegative excursions of length  $5n$ .  $B(n)$  was found to satisfy a 4<sup>th</sup> order, degree 11 recurrence relation using **Findrec** and a 7<sup>th</sup> order, degree 9 recurrence relation using **algtorec**.<sup>27</sup> Both relations<sup>28</sup> are fairly large and so are not included here.<sup>29</sup> We compare and contrast the methods in the table below.

**Table 10:** Alternative Recurrence Step Set  $\{[1, 2], [1, -3]\}$

Method	Degree	Order	Memory Used	Allocated	CPU Time	Real Time
<b>algtorec</b>	9	7	165.84GiB	2.46GiB	46.99h	12.06h
<b>Findrec</b>	11	4	0.50GiB	388.01MiB	4.47s	4.31s

<sup>27</sup>2 trials.

<sup>28</sup>The **Findrec** recurrence matches with Andrew Lohr's calculation as expected since they are computing the same result.

<sup>29</sup>They are available at [math.rutgers.edu/~bte14/Articles/ScoringPaths/23Recurrence.txt](http://math.rutgers.edu/~bte14/Articles/ScoringPaths/23Recurrence.txt).

Simply guessing at the form appears to be the *MUCH* better choice here.

**Example 19.** Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . See Example 11 for the minimal polynomial. Let  $B(n)$  denote the number of nonnegative excursions of length  $n$ . Then

$$\begin{aligned}
0 = & 2(n+4)(2n+11)(n+7)(n+6)B(n+5) \\
& - (n+6)(43n^3 + 597n^2 + 2738n + 4142)B(n+4) \\
& + (124n^4 + 2110n^3 + 13305n^2 + 36815n + 37686)B(n+3) \\
& - 5(n+3)(2n^3 - 22n^2 - 305n - 726)B(n+2) \\
& - 25(n+3)(n+2)(8n^2 + 72n + 159)B(n+1) \\
& + 125(n+5)(n+3)(n+2)(n+1)B(n).
\end{aligned}$$

This was found with **Findrec** while **algtoREC** found

$$\begin{aligned}
0 = & 2(2n+15)(n+9)(n+8)(n+7)B(n+7) \\
& - (n+8)(n+7)(99n^2 + 1241n + 3900)B(n+6) \\
& + 2(n+7)(413n^3 + 6986n^2 + 39356n + 73830)B(n+5) \\
& - (2821n^4 + 56794n^3 + 425771n^2 + 1407974n + 1731540)B(n+4) \\
& + 10(n+4)(304n^3 + 3233n^2 + 9864n + 6513)B(n+3) \\
& + 25(n+4)(n+3)(107n^2 + 1457n + 4316)B(n+2) \\
& - 250(n+4)(n+3)(n+2)(27n+122)B(n+1) \\
& + 3125(n+1)(n+2)(n+3)(n+4)B(n).
\end{aligned}$$

The comparison in methods is given below.

**Table 11:** Alternative Recurrence Step Set  $\{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$

Method	Degree	Order	Memory Used	Allocated	CPU Time	Real Time
<b>algtoREC</b>	4	7	38.32MiB	42.61MiB	355.60ms	354.44ms
<b>Findrec</b>	4	5	62.38MiB	28.00MiB	618.07ms	630.87ms

Conversion is actually more efficient, though not ideal, in this case. Though the difference is not as extreme as that shown in Table ??.

With smaller step sizes, conversion appears to be “better”. This is likely due to the intensive task of converting higher degree polynomials, while guessing avoids that hurdle. For small

enough examples, guessing is a larger search space than conversion.

Because of the way **Findrec** is programmed, it is guaranteed that a found recurrence has order less than or equal to that of **algtopec**. Depending on the goal when enumerating, lower degree (save computation) or lower order (save memory) is optimal.

One can use **algtopec**'s order and degree as a proper bound for use in **Findrec**. But instead of directly converting the minimal polynomial, it may be possible to find a bound on the order and degree of a recurrence based on the degrees of the g.f. and  $t$  in the minimal polynomial. We may also be able to derive an upper bound based on the number of variables in the closed system we created.

I was unable to prove any useful bounds, and these examples are evidence against bounding the order and degree directly by the degrees of the g.f. and  $t$ . All three examples have minimal polynomials with smaller degree (and total degree) than the order of the found recurrences. In fact, we know that the sum of the degree and order of any recurrence has to be greater than the total degree of the minimal polynomial in these cases since we searched everything lower.

The first example showed that **Findrec** can be an *EXTREME* improvement over **algtopec** and the second example showed when **algtopec** can just edge out **Findrec**. The following example demonstrates that **algtopec** can be the significantly better choice.

**Example 20.** Let  $\mathcal{S} = \{[1, -1], [3, -1], [1, 0], [3, 0], [2, 1], [1, 2], [2, 2]\}$ . The minimal polynomial, letting  $K$  denote the g.f. of meanders with step set  $\mathcal{S}$ , is

$$1 + (t^3 + 5t^2 + 4t - 1)K + t(4t + 3)(2t^3 + 2t^2 + 3t - 1)K^2 + t(t + 1)(2t^3 + 2t^2 + 3t - 1)K^3.$$

The recurrences found by conversions and searching are much too large to include here. For their information, see

[math.rutgers.edu/~bte14/Articles/ScoringPaths/RecurrenceOutputFile](http://math.rutgers.edu/~bte14/Articles/ScoringPaths/RecurrenceOutputFile).



**Table 12:** Alternative Recurrence Step Set  $\{[1, -1], [3, -1], [1, 0], [3, 0], [2, 1], [1, 2], [2, 2]\}$

Method	Degree	Order	Memory Used	Allocated	CPU Time	Real Time
Polynomial			2.55MiB	24.00MiB	20.80ms	24.50ms
<b>algtorec</b>	3	31	73.92MiB	11.39MiB	491.03ms	475.13ms
Total			76.47MiB	35.39MiB	511.83ms	499.63ms
161 terms			24.11MiB	0 bytes	190.00ms	193.00ms
<b>Findrec</b>	5	15	2.46GiB	504MiB	23.30s	21.20s
Total			2.48GiB	504MiB	23.49s	21.39s

Conversion is surprisingly 43 times faster and has roughly  $1/32^{nd}$  of the memory requirements!

We chose to enumerate 161 terms of the sequence because that was sufficient, once we knew the degree and order of the converted recurrence, to guarantee **Findrec** would encounter a solution. Typically, we do not use both **Findrec** and **algtorec**; we should simply find a set large bound of numbers. But enumerating was a small portion of **Findrec**'s time so it does not matter too much to this example.

We have used meanders instead of excursions for our  $3^{rd}$  example, but they are similar enough to compare. The important facets of the problem are the degree of the minimal polynomial and the number of steps.

It would appear that increasing the number of steps does not affect the runtime of **algtorec** as much as the runtime of **Findrec**. The runtime of **algtorec** is highly dependent on the degree of the minimal polynomial.<sup>30</sup> The degree of the minimal polynomial is generally correlated with the number of variables in the system, though not necessarily equal or bounded one way or the other. Example 11 has 7 variables in the closed system and a minimal polynomial of degree only 4. Section 8.3 shows that a step set of  $\{[1, 3], [1, -5]\}$  for nonnegative excursions yields a closed system with 18 variables yet the minimal polynomial has degree 56!

The most important aspects of the step set to **algtorec** runtime are then the maximum and minimum<sup>31</sup>  $y$ -steps since these dictate how many other walks we must consider.

**Findrec** is much weaker with larger step sets. Inherently, the recurrences will become more

<sup>30</sup>The degree of  $t$  is much less relevant.

<sup>31</sup>Most negative.

complicated, which means **Findrec** must consider many more terms in its guesses. The worst case (most number of terms to address) will occur when the order and degree of our search are the same.

Since **algtoec** works better with minimal polynomials of lower degree, and finding the minimal polynomial is generally very fast (see Example 13), I have chosen to have programs such as **PaperSemiBounded** and **BookEqualSemiBounded** use **Findrec** when the minimal polynomial has degree  $\geq 8$  and **algtoec** otherwise. The cutoff was arbitrarily chosen based on a few random examples. A rigorous cutoff would be useful for more widespread implementation.

## 6 Unbounded

We remove the lower bound in this section and transition from analyzing excursions to analyzing bridges. I was unable to find any current literature that analyzes the unbounded case with generating functions; thus, I am led to believe that this section is novel work. Again, all walks are assumed to be constructed from a step set  $\mathcal{S}$ .

### 6.1 Walking “Anywhere”

Enumerating walks that go anywhere *can* be a trivial task. First off, the  $y$ -values do not matter except for describing multiple steps with the same  $x$ -step. If all of the steps have  $x$ -value  $m > 0$ , then the number of walks of length  $m \cdot n$  is simply  $|\mathcal{S}|^n$ . If we have steps with varying  $x$ -values, then the problem becomes finding combinations of the  $x$ -values that sum to  $n$ . In general, the g.f. will be

$$\frac{1}{1 - \sum_{(x,y) \in \mathcal{S}} t^x}.$$

This g.f. is produced by **UnBoundedScoringPaths**. The terms are generated by (using Taylor series) **UnBoundedScoringPathsCoefficients** or (via recursion) **UnBoundedScoringPathsNumber**. Using Maple’s Taylor series expansion is typically faster, but both are extremely quick at enumerating the first 1000 terms.<sup>32</sup>

---

<sup>32</sup>By utilizing Maple’s **option remember** in the recursive case.

## 6.2 Returning to the $x$ -axis

A lot of the work we have done in the semi-bounded case will prove useful here. We cannot use the exact same method as in semi-bounded Section 3. Suppose we tried describing a walk with a negative change in altitude. The first and last steps could both be positive. Then we would need to describe a walk that has a larger negative change in altitude. And so on.

We try a slightly different method. Let  $G$  denote the g.f. for walks with step set  $\mathcal{S}$  that begin at  $(0,0)$  and end on the  $x$ -axis: a bridge. We choose to introduce another g.f., which we denote  $h_i$ , for the number of walks from  $(0,i)$  to  $(n,0)$  that do not touch the  $x$ -axis beforehand, with the exception of  $h_0$  starting on the  $x$ -axis.<sup>33</sup> Recall the definitions of  $g_{a,b}$  and  $f_{a,b}$ . The first equation is similar to  $f_{0,0}$ :

$$G = 1 + \left( h_0 + \sum_{(x,0) \in \mathcal{S}} t^x \right) G.$$

A walk can be stationary, or it returns to the  $x$ -axis after some number of steps, at which point it can take another  $G$ -walk.

$$h_0 = 2g_{0,0} + \sum_{(x,y) \in \mathcal{S}; y \leq -2} \sum_{i=1}^{-y-1} g_{0,i} t^x h_{i+y} + \sum_{(x,y) \in \mathcal{S}; y \geq 2} \sum_{i=1}^{y-1} g_{y-i,0} t^x h_i.$$

An  $h_0$ -walk can be purely positive  $(g_{0,0})$  or purely negative  $(g_{0,0})$ .<sup>34</sup> Note that walks below the  $x$ -axis are in bijection with walks above the  $x$ -axis by reversing the order of steps.<sup>35</sup> We can then use  $g_{0,-a} = g_{a,0}$  and  $f_{-a,-b} = f_{b,a}$ . It is also possible for the  $h$ -walk to cross the  $x$ -axis without touching it. This would involve an irreducible walk  $(g_{0,i})$ , a step across the  $x$ -axis ( $t^x$ ), and another walk  $(h_{i+y})$  to return to the  $x$ -axis. The previous sentence described crossing the  $x$ -axis from above to below. We could instead cross from below to above. This would consist of an irreducible walk  $(g_{0,i-y} = g_{y-i,0})$ , a step across the  $x$ -axis ( $t^x$ ), and another walk  $(h_i)$  to return to the  $x$ -axis.

We already have equations to describe the irreducible walks. We now need to describe the

---

<sup>33</sup> $h_0$  does NOT include stationary walks nor walks that are steps directly to the right. For that, one will want to use  $h = h_0 + 1 + \sum_{(x,0) \in \mathcal{S}} t^x$ .

<sup>34</sup>This is also partially why irreducible walks do not include steps to the right. If  $g_{0,0}$  did, then I would have to write  $h_0 = 2g_{0,0} - \sum_{(x,0) \in \mathcal{S}} t^x$ , which is less elegant.

<sup>35</sup>Equivalent to rotating the walk  $180^\circ$ .

other  $h$ -walks.

$$\begin{aligned}
 j > 0 \quad h_j &= g_{j,0} + \sum_{(x,y) \in \mathcal{S}; y \leq -2} \sum_{i=1}^{-y-1} f_{j-1,i-1} t^x h_{i+y}, \\
 j < 0 \quad h_j &= g_{0,-j} + \sum_{(x,y) \in \mathcal{S}; y \geq 2} \sum_{i=1}^{y-1} f_{y-i-1,-j-1} t^x h_i.
 \end{aligned}$$

An  $h_{j>0}$ -walk can either be a walk “directly” to the  $x$ -axis ( $g_{j,0}$ ) or it consists of an arbitrary walk that does not touch the  $x$ -axis ( $f_{j-1,i-1}$ ), followed by a step across ( $t^x$ ), and another walk ( $h_{i+y}$ ) to return to the  $x$ -axis. Similarly an  $h_{j<0}$ -walk can either be a walk “directly” to the  $x$ -axis ( $g_{j,0} = g_{0,-j}$ ) or it consists of an arbitrary walk that does not touch the  $x$ -axis ( $f_{j+1,i-y+1} = f_{y-i-1,-j-1}$ ), followed by a step across ( $t^x$ ), and another walk ( $h_i$ ) to return to the  $x$ -axis.

A brief lapse in concentration allowed me to write the alternative (and much simpler) relation:

$$h_0 = 2g_{0,0} + \sum_{(x,y) \in \mathcal{S}; y \neq 0} t^x h_y.$$

The reason for shunning this “simpler” description is that the  $h_y$  may be irreducible walks directly back to the  $x$ -axis, and thus double-count a walk from  $g_{0,0}$ .

The system of equations will be closed since the index of  $h_i$ -walks is bounded by  $\max_{(x,y) \in \mathcal{S}} |y| - 1$ . All of the equations and variables are produced in **EqualUnBoundedScoringPathsEqsVars**. Again, we use the **Basis** procedure in the **Groebner** package to find a polynomial  $p$  such that  $p(G) = 0$  in terms of formal power series. To produce this polynomial, use **EqualUnBoundedScoringPaths**. To produce the minimal polynomial for  $h_j$ , use **SpecificUnBoundedScoringPaths**.

We can try to produce the coefficients of the Taylor series using the iteration  $G_0 \rightarrow p(G_0) + G_0$ . However, because of convergence issues, this will typically not work. It is also potentially not ideal because it requires finding a Groebner basis, which can be time-consuming. However, iterating a vector of solutions à la the semi-bounded Section 3 does work. To check these values, use **SpecificUnBoundedScoringPathsNumber** to verify  $G$  and **SpecificIrreducibleUnBoundedScoringPathsNumber** to verify  $h_j$ .

### 6.3 Alternative Method

There is an alternative method for enumerating unbounded walks in the specific case that all of the steps have  $x$ -step 1 [AZ90]. The functions themselves are taken from the Maple package **EKHAD** by Zeilberger. I will start with an example.

**Example 21.** Consider the function  $G(t, n) = (t^{-2} + t^{-1} + t + t^2)^n$ . This represents a possible step set of  $\mathcal{S} = \{[1, -2], [1, -1], [1, 1], [1, 2]\}$ . A specific monomial in the expansion of  $G(t, n)$  comes from all the ways of picking a power of  $t$  from each term; this is equivalent to picking which step to take and in which order.

1.  $G(t, 0) = 1$ . There is only 1 walk: the stationary walk.
2.  $G(t, 1) = t^{-2} + t^{-1} + t + t^2$ . There are 4 possible steps each leading to a different final altitude.
3.  $G(t, 2) = t^{-4} + 2t^{-3} + t^{-2} + 2t^{-1} + 4 + 2t + t^2 + 2t^3 + t^4$ . There are 4 ways to return to the  $x$ -axis after 2 steps.

$G(t, n)$  enumerates all the ways to change altitude by  $c$  in the coefficient of the  $t^c$  monomial.

In general, for a step set  $\mathcal{S}$ , let  $G(t, n) = \left(\sum_{(x,y) \in \mathcal{S}} t^y\right)^n$ . The procedure **AZd(A, t, n, N)** from package **EKHAD** will give a recurrence for the (contour around 0) integral of  $A$  (with respect to  $t$ ) under hypergeometric assumptions, i.e., **AZd** returns the residue. And if we represent  $A$  as a power series, that means we extract the  $t^{-1}$  term. So to find a recurrence for the number of walks that return to the  $x$ -axis, we will use  $A = G(t, n)/t$ .

**Example 22.** The input **AZd** $((t^{-2} + t^{-1} + t + t^2)^n, t, n, N)$  yields as output:

$$18(2n+1)(5n+8)(n+1) + (n+1)(35n^2 + 91n + 54)N - 2(n+2)(2n+3)(5n+3)N^2.$$

This translates as, if we let  $B(n)$  denote the number of walks that return to the  $x$ -axis after  $n$  steps,

$$\begin{aligned} 18(2n+1)(5n+8)(n+1)B(n) + (n+1)(35n^2 + 91n + 54)B(n+1) \\ - 2(n+2)(2n+3)(5n+3)B(n+2) = 0. \end{aligned} \quad (6.1)$$

We can find a recursion for *any* change in altitude, not just bridges. If we are interested in walks that change in elevation by  $c$ , then we should use  $A = G(t, n)/t^{c+1}$ .

## 6.4 Selected Step Sets

The following examples are all produced by the one call:

**EqualUnBoundedScoringPaths( $S,t,G$ ).**

**Example 23** (Old-Time Basketball). We will follow up the previous example by using this paper's g.f. method of dynamical programming with the same step set. Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 1], [1, 2]\}$ . Let  $G$  denote the g.f. for the number of unbounded bridges subject to step set  $\mathcal{S}$ . Then  $G$  satisfies

$$(9t + 4)(4t - 1)^2 G^4 - 2(3t - 2)(4t - 1)G^2 + t = 0.$$

And the coefficients satisfy (using **algtoREC**)

$$\begin{aligned} 108(n+1)(2n+1)B(n) + (78n^2 + 246n + 216)B(n+1) \\ - (n+2)(17n-9)B(n+2) - 2(n+3)(2n+5)B(n+3) = 0. \end{aligned}$$

Interestingly, we have obtained a different recurrence than what was found with the **AZd** method: Eqn. (6.1). Both recurrences are correct. Typically, **AZd** will produce a lower order recurrence but with higher degree polynomial coefficients.

The simple alteration of the step set to  $\mathcal{S}' = \{[2, -2], [1, -1], [1, 1], [2, 2]\}$  makes the alternative method in Section 6.3 ineffectual. However, the method presented in this paper is not bothered in the slightest. Let  $G'$  denote the g.f. for the number of unbounded bridges subject to step set  $\mathcal{S}'$ . Then  $G'$  satisfies

$$(8t^2 + 5) (4t^4 - 8t^2 + 1)^2 G'^4 + 2 (4t^2 - 3) (4t^4 - 8t^2 + 1) G'^2 + 1 = 0.$$

Now we only have the option of conversion or guessing to find a recurrence for the coefficients of  $G'$ . Using **algtoREC** took only 69ms and produced:

$$\begin{aligned} 32(n+3)(n+2)B(n) + 4(5n^2 + 110n + 356)B(n+2) - 8(15n^2 + 160n + 439)B(n+4) \\ - (59n + 438)(n+6)B(n+6) + 10(n+8)(n+6)B(n+8) = 0. \end{aligned}$$

The reason to have  $x$ -step as all 1s versus having  $x$ -step equal to  $y$ -step is that the choice changes what  $G$  counts. If all  $x$ -steps are 1,  $G$  will count the number of ways to be tied after  $n$  total baskets. If  $x$ -step equals  $|y|$ -step, then  $G'$  counts the number of ways to be tied at a score of  $\frac{n}{2}$  to  $\frac{n}{2}$ . Note that this interpretation means  $G'$  has coefficient 0 for all odd powers of  $t$ ; teams cannot be tied after an odd number of points have been scored. We could make the substitution  $t \rightarrow \sqrt{t}$  in  $G'$  and then  $G'$  would count the number of ways to be tied at a score of  $n - n$ .

**Example 24** (Current Basketball). Let  $\mathcal{S} = \{[1, -3], [1, -2], [1, -1], [1, 1], [1, 2], [1, 3]\}$ . Let  $G$  denote the g.f. for the number of unbounded bridges subject to step set  $\mathcal{S}$ . Then  $G$  satisfies

$$\begin{aligned} & \left(8t^2 - 68t - 27\right)^2 (6t - 1)^4 (2t + 1)^4 G^8 \\ & + 4 \left(68t^2 + 10t - 9\right) \left(8t^2 - 68t - 27\right) (6t - 1)^3 (2t + 1)^3 G^6 \\ & + 2 \left(9120t^4 + 3744t^3 - 1264t^2 - 212t + 135\right) (6t - 1)^2 (2t + 1)^2 G^4 \\ & + 4 \left(1216t^4 + 832t^3 + 4t^2 - 46t + 7\right) (6t - 1) (2t + 1) G^2 \\ & + \left(16t^2 + 8t - 1\right)^2 = 0. \end{aligned}$$

**EqualUnBoundedScoringPaths( $\mathcal{S}, t, G$ )** originally gave a much larger polynomial that  $G$  satisfies. After **FindProperRoot** was added to parse the output, the smaller polynomial above was the result. The coefficients satisfy (using **algtoec**)

$$\begin{aligned} & 36864 (n + 1) (n + 2) (n + 3) B(n) \\ & - 3072 (n + 3) (n + 2) (97n + 142) B(n + 1) \\ & - 64 (n + 3) \left(4031n^2 + 17601n + 19504\right) B(n + 2) \\ & - 16 \left(1684n^3 + 13491n^2 + 31178n + 15240\right) B(n + 3) \\ & + 24 \left(663n^3 + 7222n^2 + 28628n + 41563\right) B(n + 4) \\ & + 4 \left(467n^3 + 7011n^2 + 35842n + 62490\right) B(n + 5) \\ & - 2 (n + 6) \left(115n^2 + 1080n + 2273\right) B(n + 6) \\ & - 3 (3n + 20) (3n + 19) (n + 7) B(n + 7) = 0. \end{aligned}$$

or, using **AZd**,

$$\begin{aligned}
& 96 (n + 1) (n + 2) (n + 3) \left( 2058n^3 + 20335n^2 + 66857n + 73300 \right) B(n) \\
& - 8 (n + 2) (n + 3) \left( 201684n^4 + 2295356n^3 \right. \\
& \quad \left. + 9540055n^2 + 16998380n + 10742400 \right) B(n + 1) \\
& - 4 (n + 3) \left( 310758n^5 + 4313617n^4 + 23611469n^3 \right. \\
& \quad \left. + 63712598n^2 + 84804508n + 44608800 \right) B(n + 2) \\
& - 2 (n + 3) \left( 41160n^5 + 591920n^4 + 3361281n^3 \right. \\
& \quad \left. + 9453847n^2 + 13262292n + 7512000 \right) B(n + 3) \\
& + 3 (3n + 11) (3n + 10) (n + 4) \left( 2058n^3 + 14161n^2 + 32361n + 24720 \right) B(n + 4) = 0.
\end{aligned}$$

Let  $S' = \{[3, -3], [2, -2], [1, -1], [1, 1], [2, 2], [3, 3]\}$ . Let  $G'$  denote the g.f. for the number of unbounded bridges subject to step set  $S'$ . Then  $G'$  satisfies

$$\begin{aligned}
& \left( 108t^6 - 99t^4 - 52t^2 - 44 \right)^2 \left( 4t^6 + 4t^4 + 8t^2 - 1 \right)^4 G'^8 \\
& + 4 \left( 36t^6 + 29t^4 + 24t^2 - 20 \right) \left( 108t^6 - 99t^4 - 52t^2 - 44 \right) \left( 4t^6 + 4t^4 + 8t^2 - 1 \right)^3 G'^6 \\
& + 2 \left( 2160t^{12} + 4248t^{10} + 4347t^8 + 992t^6 - 16t^4 - 1184t^2 + 976 \right) \left( 4t^6 + 4t^4 + 8t^2 - 1 \right)^2 G'^4 \\
& + 4 \left( 112t^{12} + 288t^{10} + 665t^8 + 412t^6 + 552t^4 - 112t^2 + 96 \right) \left( 4t^6 + 4t^4 + 8t^2 - 1 \right) G'^2 \\
& + \left( 4t^6 + 3t^4 + 12t^2 + 4 \right)^2 = 0.
\end{aligned}$$

Again, **EqualUnBoundedScoringPaths**( $S', t, G'$ ) originally gave a more complicated polynomial before **FindProperRoot** was added. We cannot use **AZd** for this case. However, we can still obtain a recurrence using **algtoec**. The conversion only took about 13 seconds but produced a recurrence of order 66 and degree 3 and as such is not included here.

## 6.5 Time Comparison

To reiterate the benefits of this paper and Maple package over guessing for the polynomial, the following examples illustrate the time savings.



**Example 25** (Finding a Polynomial). Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1]\}$ . Let  $G$  be the g.f. for bridges subject to step set  $\mathcal{S}$ . Then

$$(16t^3 + 8t^2 + 11t - 4)G^3 + (3 - 2t)G + 1 = 0.$$

This takes

**Table 13:** Finding Minimal Polynomial  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1]\}$

Method	Memory Used	Memory Allocation	CPU Time	Real Time
<b>EUSP</b>	2.10MiB	0 bytes	18.27ms	21.53ms
<b>Empir</b>	34.94MiB	0.73MiB	294.93ms	295.63ms
<b>EmpirF</b>	1.44MiB	0 bytes	14.27ms	16.23ms

In this case, **EmpirF** appears to be (25%) faster than **EqualUnBoundedScoringPaths (EUSP)** but the time is so small the difference is likely to go unnoticed. The next example shows why the **ScoringPaths** package is superior to using **EmpirF**.

**Example 26** (Finding a Polynomial 2). Let  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . Let  $G$  be the g.f. for bridges subject to step set  $\mathcal{S}$ . Then

$$(5t + 4)(5t - 1)^2(t - 1)^2G^4 + 2(t - 1)(5t - 2)(5t - 1)G^2 + t = 0.$$

This takes

**Table 14:** Finding Minimal Polynomial  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$

Method	Memory Used	Memory Allocation Change	CPU Time	Real Time
<b>EUSP</b>	7.70MiB	0 bytes	58.33ms	61.43ms
<b>Empir</b>	377.91MiB	31.29MiB	2.89s	2.75s
<b>EmpirF</b>	Failed			

**EmpirF** happened to fail for this  $\mathcal{S}$  while **EqualUnBoundedScoringPaths (EUSP)** only tripled in time and memory usage. The only difference between this example and Example 25 is the second step with positive  $y$ -value.

Eventually guessing should work, but we have no way of knowing what the upper bound of search space is before blindly investigating. Expressing the g.f. equations is a set amount of time that is guaranteed to produce the solution. The time complexity to find the minimal polynomial with Groebner bases is an interesting question we leave to the reader.

We have many different ways to enumerate the number of bridges with step set  $\mathcal{S}$ . We will compare their speed and memory usage here.

**Example 27** (Speed Enumeration - Unbounded). As with most previous examples, we use the benchmark step set  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1], [1, 2]\}$ . The following are time and memory requirements for enumerating 1000 terms of the sequence of bridges with step set  $\mathcal{S}$ . The different methods are

1. Iterating the single polynomial found with **EqualUnBoundedScoringPaths** does not work due to convergence issues beyond the scope of this paper.
2. **taylor** cannot be used. It gives the error: “does not have series solution”. **taylor** tends to work only if the minimal polynomial contains a  $G^1$  term. See Example 25.
3. Iterating a vector solution with **EqualUnBoundedScoringPathsSeries**.
4. Brute force recursion using **SpecificUnBoundedScoringPathsNumber**.
5. Converting minimal polynomial to recurrence:
  - (a) Obtain the polynomial with **EqualUnBoundedScoringPaths**.
  - (b) Convert to recurrence with **algtorec**. The coefficients satisfy

$$\begin{aligned}
 0 = & 125(n+1)(n+2)B(n) - 25(n+2)(n-1)B(n+1) \\
 & - 5(21n^2 + 89n + 96)B(n+2) + (n^2 - 43n - 140)B(n+3) \\
 & + 2(n+4)(n+7)B(n+4).
 \end{aligned}$$

- (c) Enumerate using **SeqFromRec**.
6. Using **AZd** and then enumerating with **SeqFromRec**.

$$\begin{aligned}
 0 = & 25(3n+8)(n+2)(n+1)B(n) - 5(3n+5)(2n+5)(n+2)B(n+1) \\
 & - (3n+8)(19n^2 + 76n + 75)B(n+2) + 2(3n+5)(2n+5)(n+3)B(n+3).
 \end{aligned}$$

**Table 15:** 1000 term Unbounded Enumeration

Method	Memory Used	Memory Allocation	CPU Time	Real Time
Vector Set-Up	20.47KiB	0 bytes	200 $\mu$ s	200 $\mu$ s
Vector Iterating	2.79TiB	30.54MiB	2.27h	110.54m
Polynomial	7.76MiB	0 bytes	59.40ms	63.03ms
Conversion	5.97MiB	8KiB	49.03ms	51.97ms
Enumeration	20.64MiB	2.16MiB	84.87ms	81.50ms
Total	34.37MiB	2.17MiB	193.30ms	196.50ms
<b>AZd</b>	2.45MiB	0 bytes	25.77ms	30.53ms
<b>SeqFromRec</b>	19.34MiB	3.59MiB	74.40ms	74.43ms
Total	21.79MiB	3.59MiB	100.17ms	104.96ms
Brute-Force Recursion	1.41GiB	4.29MiB	12.69s	11.91s

The alternate method is the fastest in this case. It is about half of the time as converting, but the actual enumeration after the preliminary set-up is very similar; the memory usage is similarly low. However, **AZd** is only appropriate for specific cases with constant  $x$ -step. The key take-away is that enumerating by combining **ScoringPaths** and **algtovec** is much faster and leaner than brute-force recursion and applicable in a wide-range of cases.

If we change the step set back to  $\mathcal{S} = \{[1, -2], [1, -1], [1, 0], [1, 1]\}$ , then we can use **taylor** on the minimal polynomial, as well as iterating the single solution (after an appropriate change of iteration). The difference is that Example 25 had a  $G^1$  term in its minimal polynomial while Example 26 did not. This allows one to write  $G$  as copies of itself, which is what the iterative method is effectively accomplishing.

## 7 Asymptotics

For determining asymptotic behavior, I avoided analyzing the bounded cases as those resulted in explicit rational solutions, which can already be handled very easily. The asymptotic behavior of unbounded general walks is simply finding the number of combinations of  $x$ -steps to yield a length of  $n$ ; the actual walk altitude does not matter.

For semi-bounded and unbounded cases, once we have recurrences for the coefficients, we can derive asymptotic expressions. Because of the nature of these quantities, the number of walks of length  $n$  will always follow  $b^n$  for some base  $b$  [FS09]. Wimp and Zeilberger [WZ85] created a method for automatically determining asymptotics for a linear recurrence, including the constant of the leading term! Their package **AsyRec** provides **Asy**, which attempts to determine the exponential power with which a recurrence grows. **AsyC**, along with suitable starting values, will also give the correct constant.

One task we can use this for is finding what proportion of walks are nonnegative. Let  $\mathcal{S} = \{[1/2, -1], [1/2, 1]\}$ .<sup>36</sup> Let  $F$  denote the g.f. for nonnegative excursions, and  $B(n)$  denote the number of nonnegative excursions of length  $n$ . Then<sup>37</sup>

$$\begin{aligned} 0 &= 1 - F + tF^2, \\ 0 &= (4n + 2)B(n) - (n + 2)B(n + 1), \\ B(n) &\sim \frac{4^n}{\sqrt{\pi n^{3/2}}} \cdot \left(1 - \frac{9}{8n} + \frac{145}{128n^2} - \frac{1155}{1024n^3} + \frac{36939}{32768n^4} - \frac{295911}{262144n^5}\right) \end{aligned}$$

Because the minimal polynomial is simple, we could use the Lagrange Inversion Formula to compute exact values of  $B(n)$ .<sup>38</sup> But for most step sets with  $|\mathcal{S}| > 2$ , this will not work, so the LIF is not considered here. Let  $G$  denote the g.f. for bridges and  $C(n)$  the number of bridges of length  $n$ . Then<sup>39</sup>

$$\begin{aligned} 0 &= 1 + (4t - 1)G^2, \\ 0 &= (4n + 2)C(n) - (n + 1)C(n + 1), \\ C(n) &\sim \frac{4^n}{\sqrt{\pi}\sqrt{n}} \cdot \left(1 - \frac{1}{8n} + \frac{1}{128n^2} + \frac{5}{1024n^3} - \frac{21}{32768n^4} - \frac{399}{262144n^5}\right). \end{aligned}$$

Then the proportion of binary bridges that are Dyck paths is asymptotically

$$\frac{B(n)}{C(n)} \sim \frac{1}{n}.$$

This matches with the known exact proportion of  $\frac{1}{n+1}$ .

The above example was not exactly revolutionary, but the method allows for “quick” analysis of walks with any step set.

---

<sup>36</sup> $x$ -step is  $1/2$  so that excursions and bridges do not all have even length, producing a bunch of extraneous 0s.

<sup>37</sup>All of this can be produced by the one command **EqualSemiBoundedPaper(S,20,true,true)**.

<sup>38</sup>This is what Duchon used for several simple cases [Duc00].

<sup>39</sup>This can be produced by **UnBoundedPaper(S,20,true,true)**.

Dyck paths have been studied quite extensively. The typical result is for walks with step set  $\{[1,0],[0,1]\}$  from  $(0,0)$  to  $(n,n)$  that stay below the line  $y = x$ . This is equivalent to enumerating nonnegative excursions with step set  $\{[1,-1],[1,1]\}$ . Changing the slope of the upper bound to rational  $\frac{a}{b}$  is equivalent to using a step set  $\{[1,-a],[1,b]\}$ ; this is called Rational Catalan Combinatorics. For a general conversion from walks below a line with slope  $m$ , reflect the walk across the line through the origin with slope  $\frac{m}{2}$ . This produces an equivalent nonnegative walk in the right hand plane. Andrew Lohr studied the field of paths below rational slope with the goal of obtaining asymptotic constants [LZ17]. For his results, go to <http://sites.math.rutgers.edu/~ajl213/DrZ/RSP.pdf>.

Lohr states a result by Duchon in 2000 [Duc00] that for any slope  $\frac{a}{b}$ , the number of paths below a line of that slope is asymptotically  $\Theta\left(\frac{1}{n}\binom{(a+b)n}{an}\right)$ . Are we able to say anything about step sets of size  $> 2$ ? Intuition would say the growth rate should be larger, but how much larger?

First we need to tackle the scenario if we have steps with  $x$ -step 0. Recall we can only have steps directly up OR down. The maximum number of up-steps we could have in an excursion or bridge of length  $n$  is

$$n \frac{-\min\{\frac{y}{x} : (x,y) \in \mathcal{S}, x \neq 0\}}{\min\{y : (0,y) \in \mathcal{S}\}}.$$

Replace the mins with maxs for down-steps.<sup>40</sup> We can only take so many steps in one direction before we have to start returning in order to make it to the  $x$ -axis before length  $n$ . Let  $c$  denote the fraction above ( $c = 0$  for step sets without vertical steps). The “worst case” scenario<sup>41</sup> is that the remaining steps all have  $x$ -step 1. Then we have  $\binom{n+cn}{n}$  ways to place the vertical steps. Let  $S$  denote the size of  $\mathcal{S}$  with the 0-steps removed. Then the number of walks is upper bounded (usually very poorly) by

$$\binom{n(1+c)}{n} \cdot S^n \approx \sqrt{\frac{1+c}{2\pi cn}} [S(1+c)(1+1/c)^c]^n.$$

So the number of every type of walk is  $O\left(\frac{b^n}{\sqrt{n}}\right)$  for some  $b$ .<sup>42</sup> The bound may seem to contradict the Dyck path example, but remember that we cut the steps in half to avoid extraneous 0s. A good goal would be to improve this bound to something more meaningful.

---

<sup>40</sup>We technically do not need the  $x \neq 0$  since we assumed there were no steps directly down.

<sup>41</sup>Largest number of walks.

<sup>42</sup>Which matches with the earlier mentioned asymptotic.

**Example 28.** Let  $\mathcal{S} = \{[1, -1], [1, 0], [1, 2]\}$ . Then, assuming  $F, B, G, C$  are as before,

$$0 = 1 + (t - 1)F + t^3F^3,$$

$$0 = 31(n + 1)(n + 2)B(n) - 6(2n + 5)(n + 2)B(n + 1) \\ + 2(6n^2 + 36n + 53)B(n + 2) - 2(2n + 9)(n + 3)B(n + 3),$$

$$B(n) \sim 0.8001188640 \cdot \frac{2.889881575^n}{n^{3/2}} \cdot \left(1 - \frac{1.7475722}{n} - \frac{2.6532889}{n^2} + \frac{4.0131981}{n^3}\right),$$

and

$$0 = 1 - 3(t - 1)G + (31t^3 - 12t^2 + 12t - 4)G^3,$$

$$0 = 31(n + 1)(n + 2)C(n) - 6(n + 2)(2n + 3)C(n + 1) \\ + 2(6n^2 + 24n + 23)C(n + 2) - 2(n + 3)(2n + 3)C(n + 3),$$

$$C(n) \sim 0.3488331868 \cdot \frac{2.889881575^n}{\sqrt{n}} \cdot \left(1 - \frac{0.24757219}{n} - \frac{0.03549572}{n^2} + \frac{0.046925761}{n^3}\right).$$

The exponential bound is simply  $|\mathcal{S}|^n = 3^n$ : fairly close to the actual asymptotic base. And the proportion of excursions to bridges is

$$\frac{B(n)}{C(n)} \sim \frac{2.293700526}{n} :$$

also just a constant times  $\frac{1}{n}$ .

Finding the asymptotic behavior does not work in every case, e.g., meanders with step set  $\{[0, -1], [1, -1], [2, -1], [2, 0], [2, 1]\}$ . If the asymptotic does not match with empirical data, then **AsyC** will notify the user of this.

Is the ratio of nonnegative excursions to bridges always some constant times  $\frac{1}{n}$ ? The following example is more experimental evidence for this conjecture.

**Example 29.** Let  $\mathcal{S} = \{[1, -2], [3, 0], [0, 1], [2, 1], [2, -2]\}$  and  $F, B, G, C$  as before. Then

$$0 = 1 + (t - 1)(t^2 + t + 1)F + t(1 + t)(t^2 + 1)^2F^3,$$

$$B(n) \sim \frac{4}{15} \cdot \frac{7.898354145^n}{n^{3/2}},$$

$$0 = 1 - 3(t - 1)(t^2 + t + 1)G + (4t^9 + 15t^6 + 27t^5 + 54t^4 + 66t^3 + 27t^2 + 27t - 4)G^3,$$

$$C(n) \sim \frac{15\sqrt{3}}{58} \cdot \frac{7.898354145^n}{\sqrt{n}}.$$

The actual recurrences obtained from conversion (**algtoec**) are not listed here due to size. They are degree 2 order 18 and degree 2 order 16, respectively. The base of the exponential is exactly

the same for  $B$  and  $C$ ; they are the same root of the same polynomial.<sup>43</sup> And once again,

$$\frac{B(n)}{C(n)} \sim \frac{232\sqrt{3}}{675} \cdot \frac{1}{n}.$$

To compute the ratio for any step set, use the shorthand **RatioOfWalks(S)**.

We can also try to reason the ratio heuristically. Suppose we start with an excursion,  $E$ . We can reorder the steps cyclically (or in any order) and still have a bridge. Consider the set of such walks; there are size of  $E$  such walks. The reason for reordering cyclically would be to maintain the “unique” excursion in the set, while the rest are bridges. Actually,  $E$  is only unique if it was an irreducible excursion but potentially there is not much overlap.

Similarly, from a bridge  $B$  we could reorder the steps cyclically and we must obtain at least 1 excursion: starting from the step after the minimum altitude of  $B$ . We actually have the same number of excursions in this cyclic set as points of minimum altitude.

We have a relation between excursions and bridges that appears to be roughly linear. Some issues may be that an excursion of length  $n$  does not necessarily have size  $n$ . We still have a linear relationship between the two bounded by  $\max\{x, (x, y) \in \mathcal{S}\}$ . And even steps with  $x$ -step 0 do not mess this up too horribly because we are considering excursions and bridges; the walk returns to the  $x$ -axis so can only go so far away before it must start returning since the return rate  $\frac{y}{x}$  is finite (we do not have steps directly up AND down). This again leads to a linear number of  $x$ -step 0 allowed steps.

## 7.1 Discriminant

Another interesting result from analyzing asymptotic behavior is what the base of the exponent appears to be. For all three of our examples, the base can be found by taking the reciprocal of the smallest modulus of the roots of the coefficient of the leading term in the minimal polynomial of

---

<sup>43</sup>The polynomial in question is  $4x^9 - 27x^8 - 27x^7 - 66x^6 - 54x^5 - 27x^4 - 15x^3 - 4$  and the root has index=1 in Maple notation.

the bridges g.f., i.e.,

$$\begin{aligned} (\min\{|z| : 4z - 1 = 0\})^{-1} &= 4, \\ \left(\min\{|z| : 31z^3 - 12z^2 + 12z - 4 = 0\}\right)^{-1} &= \frac{3}{2^{2/3}} + 1 \approx 2.89, \\ \left(\min\{|z| : 4z^9 + 15z^6 + 27z^5 + 54z^4 + 66z^3 + 27z^2 + 27z - 4 = 0\}\right)^{-1} &= 7.90. \end{aligned}$$

This is the first estimate of the asymptotics from the inverse of the radius of convergence. Actually, the polynomial whose root modulus we need can be taken to be the discriminant of the minimal polynomial, which in the step sets given is the same for excursions and bridges.

In general, one method that appears to find the base  $b$  for the exponential asymptotic behavior is

1. Find the discriminant of the minimal polynomial.
2. Take the smallest positive real root.
3. Take the reciprocal of that root.

This will miss the possible sub-exponential factors of  $n^{-3/2}$ , etc. The shorthand for this computation is implemented as **AsymptoticBase**. A more detailed analysis of singularities and the associated asymptotics is available in *Analytic Combinatorics* by Flajolet and Sedgewick [FS09].

## 7.2 Meanders

We have yet to analyze the asymptotic behavior of meanders. Let  $K$  denote the g.f. and  $D(n)$  the number of nonnegative meanders of length  $n$ . For the case of Dyck paths with a step set of  $\{[1, 1], [1, -1]\}$  (since we can have odd length walks now), meanders satisfy

$$\begin{aligned} 1 + (2t - 1)K + t(2t - 1)K^2 &= 0, \\ 4(n + 1)D(n) + 2D(n + 1) - (n + 3)D(n + 2) &= 0, \\ D(n) &\sim \frac{0.797}{\sqrt{n}} 2^n. \end{aligned}$$

This says there are roughly 2.283 nonnegative meanders for each bridge of the same length. All of the asymptotic behavior in this section was found using the conversion to recurrence and extracting the asymptotics from there.



It turns out that meanders can follow very different asymptotic behavior depending on the step set.

**Example 30.** For  $\mathcal{S} = \{[1, -1], [1, 0], [1, 2]\}$ ,

$$\begin{aligned}
0 &= 1 + (4t - 1)K + 3t(3t - 1)K^2 + t(3t - 1)^2K^3, \\
0 &= 93(n + 1)(n + 2)(n + 3)D(n) - 2(80n + 293)(n + 3)(n + 2)D(n + 1) \\
&\quad + (n + 3)(115n^2 + 793n + 1318)D(n + 2) \\
&\quad - 4(18n^3 + 210n^2 + 820n + 1071)D(n + 3) \\
&\quad + 4(n + 4)(7n^2 + 65n + 152)D(n + 4) - 2(n + 5)(n + 4)(2n + 11)D(n + 5), \\
D(n) &\sim \frac{3 - \sqrt{5}}{2}3^n.
\end{aligned}$$

The base of the exponent happens to match the inverse of the smallest modulus of the roots of the discriminant of the minimal polynomial.

So meanders can, by virtue of their endpoint flexibility, greatly outnumber excursions and bridges.

Finally, meanders do not always follow  $|\mathcal{S}|^n$ . Let  $\mathcal{S} = \{[1, -1], [1, 0], [1, 1], [2, 2]\}$ . Then

$$\begin{aligned}
0 &= 1 + (3t^2 + 3t - 1)K + t(3t + 1)(t^2 + 3t - 1)K^2 + t^2(t^2 + 3t - 1)^2K^3, \\
D(n) &\sim 0.307 \left( \frac{3 + \sqrt{13}}{2} \right)^n \approx 0.307 \cdot 3.303^n.
\end{aligned}$$

The recurrence found was 11<sup>th</sup> order and 3<sup>rd</sup> degree so is not included here.<sup>44</sup> It was produced in less than one second using **algto**. And  $\mathcal{S} = \{[1, -2], [2, -1], [1, 0], [1, 2], [2, 1]\}$ , yields relations of

$$\begin{aligned}
0 &= 1 + (2t^2 + 3t - 1)K + t(t + 2)(2t^2 + 3t - 1)K^2 \\
&\quad + t(2t^2 + 3t - 1)^2K^3 + (2t^2 + 3t - 1)^2t^2K^4, \\
D(n) &\sim \frac{7\sqrt{2}}{13\sqrt{n}} \left( \frac{3 + \sqrt{17}}{2} \right)^n \approx \frac{0.7615}{\sqrt{n}} \cdot 3.562^n.
\end{aligned}$$

The recurrence for this step set was 25<sup>th</sup> order and of 4<sup>th</sup> degree.

---

<sup>44</sup>The 0.307 is actually a root of  $169x^4 - 1014x^3 + 507x^2 - 78x + 4$ . Maple's identify command was used on experimental data.

The behavior of meanders appears to be a lot harder to peg down than the anticipated  $\frac{b^n}{\sqrt{n}}$  for bridges and  $\frac{b^n}{n^{3/2}}$  for excursions. Though it appears that meanders are always at least as large as bridges asymptotically.

## 8 Applications

### 8.1 Combining Solutions

After obtaining all of these g.f.s, we can produce much more. Sports are always a subject of interest for a good portion of the population. Sports statistics are an integral part of many a fan base. So a question of interest beyond the number of ways to be tied, may be the number of ways to win by at least  $X$ . The task at hand then is how to describe a walk such as that. We have spent the majority of this paper breaking down walks into smaller components. We can now use those components to build other types of walks. The important step is making sure that we count all of our walks, and do not double count any walks.

Suppose we want to win by  $\geq 2$  and never trail by more than 3.  $f_{3,5}$  will count walks that drop by no more than 3, and we will have a 2 point lead at the end. Then any walk that stays above that line ( $k_0$ ) will produce what we want. So does  $f_{3,5} \cdot k_0$  count what we want? Not necessarily.  $f_{3,5}$  ensures that at some point we are exactly 2 points ahead of where we began. But depending on the step set, we may skip over this lead and never actually hit the altitude 2 steps higher than our beginning. In addition,  $f_{3,5}$  may finish with a step up and then down, while  $k_0$  could start that way; effectively tracing the same walk in “different” ways. Thus,  $f_{3,5} \cdot k_0$  double counts some walks and misses others.

There is at least one way, though not as elegant, of describing these walks. Let  $L$  denote the g.f. of interest. Then  $L = k_3 - f_{3,0} - f_{3,1} - f_{3,2} - f_{3,3} - f_{3,4}$ . We count all meanders that never drop more than 3 points, and then remove those that change in altitude by exactly  $-3, -2, -1, 0, 1$ .

**Example 31.** Let  $\mathcal{S} = \{[0, -1], [1, 0], [1, 1], [1, 2]\}$ . Then  $L$  satisfies

$$\begin{aligned}
0 = t & \left( 3t^{12} - 63t^{11} + 555t^{10} - 2673t^9 + 7671t^8 - 13371t^7 \right. \\
& \left. + 13745t^6 - 7554t^5 + 1615t^4 + 179t^3 - 138t^2 + 21t - 1 \right) \\
& + \left( 9t^{14} - 216t^{13} + 2286t^{12} - 13986t^{11} + 54558t^{10} - 141402t^9 + 246687t^8 \right. \\
& \left. - 288270t^7 + 221709t^6 - 109548t^5 + 33981t^4 - 6448t^3 + 715t^2 - 42t + 1 \right) L \\
& - 3t^3 \left( 3t^7 - 36t^6 + 153t^5 - 261t^4 + 126t^3 + 50t^2 - 26t + 2 \right) L^2 + 9t^6 L^3,
\end{aligned}$$

and has truncated expansion

$$\begin{aligned}
L = t + 21t^2 + 305t^3 + 4064t^4 + 52431t^5 + 666657t^6 \\
+ 8420130t^7 + 106070229t^8 + 1335635352t^9 + 16832212452t^{10}.
\end{aligned}$$

The minimal polynomial was derived by finding a Groebner basis. We had to describe all of  $\{k_3, f_{3,0}, f_{3,1}, f_{3,2}, f_{3,3}, f_{3,4}\}$  as well, which included a chain of describing further walks. But those methods are already set and have been shown to be closed.

## 8.2 Weighted Walks

One extension that would be fairly easy to implement is adding a weight to each step;  $[x, y]$  has an associated weight  $w$ . Then, for example, we would write<sup>45</sup>

$$f_{a,b} = 1 + \sum_{(x,y,w) \in \mathcal{S}} w \cdot t^x f_{a-y, b-y}.$$

Simply multiply by the weight whenever we take a certain step. Now we can accomplish more with the weights in place. If  $\sum_{(x,y,w) \in \mathcal{S}} w = 1$  (and all  $x = 1$ ), then  $w$  represents the probability of taking a specific step. And then  $f_{a,b}$  is the g.f. for the probability that a given walk of length  $n$  maintains altitude  $a \geq y \geq b$ . If we want to know the probability of a bounded walk being a bounded bridge, use weights to describe  $f'_{a,b}$ , the g.f. for probability of a general walk being a bounded bridge, and divide its coefficients by the appropriate coefficient of  $f_{a,b}$ , the probability of a general walk being bounded. This is an explicit description of using  $\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}$ . If all of the weights are the same, then simply enumerate each one to get the probabilities; it is much easier.

<sup>45</sup>Taken from general walks in Bounded Section 2.

**Example 32.** Let us try finding the probability of a general walk with step set  $\mathcal{S} = \{[1, 2, 1/3], [1, -1, 1/6], [1, -2, 1/2]\}$  being bounded above by  $y = 3$  and below by  $y = -2$ . Then the g.f. is explicitly

$$3 \frac{3888 + 3888t - 756t^2 - 972t^3 - 12t^4 - t^5}{11664 - 7776t^2 - 432t^3 + 1296t^4 + t^6},$$

which has Taylor expansion

$$1 + t + \frac{17}{36}t^2 + \frac{49}{108}t^3 + \frac{77}{324}t^4 + \frac{811}{3888}t^5 + \frac{53}{432}t^6 + \frac{3407}{34992}t^7 + \frac{26483}{419904}t^8 + \frac{58247}{1259712}t^9 + O(t^{10}).$$

We need the steps to all have the same  $x$ -value, otherwise we aren't representing the probability that a given walk of length  $n$  has some property. We would somehow be combining the probability that a walk has length  $n$  and the probability that it satisfies our desired property(ies) in a way I cannot currently describe.

### 8.3 2-step Examples

With all of the tools at our disposal, let's now use them to produce more information about many sequences.

#### 8.3.1 Bridges with $\mathcal{S} = \{[1, 1], [1, -k]\}$

It is well known<sup>46</sup> that the g.f. for nonnegative excursions, denoted  $f$ , satisfies

$$f = 1 + t^{k+1}f^{k+1}.$$

We would like to show something about bridges with this step set. The g.f. for the first  $k = 0, \dots, 7$  step sets have minimal polynomials

$$\begin{aligned} &(t-1)G + 1, \\ &(4t^2-1)G^2 + 1, \\ &(27t^3-4)G^3 + 3G + 1, \\ &(256t^4-27)G^4 + 18G^2 + 8G + 1, \\ &(3125t^5-256)G^5 + 160G^3 + 80G^2 + 15G + 1, \\ &(46656t^6-3125)G^6 + 1875G^4 + 1000G^3 + 225G^2 + 24G + 1, \\ &(823543t^7-46656)G^7 + 27216G^5 + 15120G^4 + 3780G^3 + 504G^2 + 35G + 1. \end{aligned}$$

---

<sup>46</sup>Try proving it for yourself. Or prove it for any finite  $k$  with this package.

The general form appears to have leading term  $([(k+1)t]^{k+1} - k^k)G^{k+1}$ . The  $G^1$  term is fairly easy to see have coefficient  $(k+1)(k-1)$ . The  $G^2$  term, after some manipulations, has coefficients that follow  $\frac{k^2}{2}(k+1)(k-2)$ . The computer could only obtain up to  $k = 9$  before memory requirements became too large ( $> 3GB$  allocated). From that limited data, I found that the  $G^3$  term has coefficient that follows  $\frac{k^3}{6}(k+1)(k-1)(k-3)$ . Does this generalize? And if so, how?

These results are not groundbreaking as we can already enumerate the number of bridges of length  $(a+b)n$  with step set  $\{[1, a], [1, -b]\}$  with the simple  $\binom{(a+b)n}{an}$ . They do allow a different view of the walks by considering how they can be built from copies of themselves.

### 8.3.2 Duchon Numbers

The Duchon numbers<sup>47</sup> are one of the cases that Lohr analyzed for asymptotic behavior [LZ17]. We can derive more information for the g.f., which we will denote  $f$ , of the Duchon numbers. The sequence is defined as the number of paths of length  $5n$  from  $(0,0)$  to the line  $y = 2x/3$  with unit North and East steps that stay below the line or touch it. It is equivalent to enumerating excursions with step set  $\{[1/5, 2], [1/5, -3]\}$ . To find the minimal polynomial for its g.f., all we have to do is type **EqualSemiBoundedScoringPaths( $S,0,t,f$ )** and hit return.

$$0 = 1 - f + 2tf^5 - tf^6 + tf^7 + t^2f^{10}.$$

We can also produce the minimal polynomial for the related g.f. of irreducible walks (those that only touch at the endpoints). We almost accomplish this with the call **SpecificEqualSemiBoundedScoringPaths( $S,0,t,f$ )**. But to match OEIS [A293946](#) [OEId], we must allow the stationary walk. Simply substitute  $g[0,0] = g - 1$  and we are done.

$$0 = g^{10} - 19g^9 + 162g^8 - 816g^7 + 2688g^6 - (2t + 6048)g^5 + (19t + 9408)g^4 - (73t + 9984)g^3 + (142t + 6912)g^2 - (140t + 2816)g + t^2 + 56t + 512.$$

### 8.3.3 Excursions with $S = \{[1, 2], [1, -k]\}$

We have some information for this family in the Duchon numbers:  $k = 3$ . But what do they look like in general? We will assume  $\gcd(2, k) = 1$  otherwise we could reduce the steps for an equiv-

---

<sup>47</sup>OEIS [A060941](#) [OEIa].

alent problem. The first few g.f. have minimal polynomial (we have made the transformation  $t \rightarrow t^{1/(2+k)}$  for ease of reading)

$$\begin{aligned}
\text{A001764: } k = 1 & \quad f^3t - f + 1, \\
\text{A060941: } k = 3 & \quad f^{10}t^2 + f^7t - f^6t + 2f^5t - f + 1, \\
\text{A300386: } k = 5 & \quad f^{21}t^3 + 2f^{16}t^2 - f^{15}t^2 + 3f^{14}t^2 \\
& \quad + f^{11}t - f^{10}t + 2f^9t - 2f^8t + 3f^7t - f + 1, \\
\text{A300387: } k = 7 & \quad f^{36}t^4 + 3f^{29}t^3 - f^{28}t^3 + 4f^{27}t^3 + 3f^{22}t^2 - 2f^{21}t^2 + 6f^{20}t^2 \\
& \quad - 3f^{19}t^2 + 6f^{18}t^2 + f^{15}t - f^{14}t + 2f^{13}t - 2f^{12}t \\
& \quad + 3f^{11}t - 3f^{10}t + 4f^9t - f + 1, \\
\text{A300388: } k = 9 & \quad f^{55}t^5 + 4f^{46}t^4 - f^{45}t^4 + 5f^{44}t^4 + 6f^{37}t^3 - 3f^{36}t^3 + 12f^{35}t^3 \\
& \quad - 4f^{34}t^3 + 10f^{33}t^3 + 4f^{28}t^2 - 3f^{27}t^2 + 9f^{26}t^2 - 6f^{25}t^2 \\
& \quad + 12f^{24}t^2 - 6f^{23}t^2 + 10f^{22}t^2 + f^{19}t - f^{18}t + 2f^{17}t \\
& \quad - 2f^{16}t + 3f^{15}t - 3f^{14}t + 4f^{13}t - 4f^{12}t + 5f^{11}t - f + 1, \\
\text{A300389: } k = 11 & \quad f^{78}t^6 + 5f^{67}t^5 - f^{66}t^5 + 6f^{65}t^5 + 10f^{56}t^4 - 4f^{55}t^4 + 20f^{54}t^4 \\
& \quad - 5f^{53}t^4 + 15f^{52}t^4 + 10f^{45}t^3 - 6f^{44}t^3 + 24f^{43}t^3 - 12f^{42}t^3 \\
& \quad + 30f^{41}t^3 - 10f^{40}t^3 + 20f^{39}t^3 + 5f^{34}t^2 - 4f^{33}t^2 + 12f^{32}t^2 \\
& \quad - 9f^{31}t^2 + 18f^{30}t^2 - 12f^{29}t^2 + 20f^{28}t^2 - 10f^{27}t^2 \\
& \quad + 15f^{26}t^2 + f^{23}t - f^{22}t + 2f^{21}t - 2f^{20}t + 3f^{19}t - 3f^{18}t \\
& \quad + 4f^{17}t - 4f^{16}t + 5f^{15}t - 5f^{14}t + 6f^{13}t - f + 1.
\end{aligned}$$

The degree is simply  $\frac{1}{2}(k+1)(k+2)$ . If one looks closer they may recognize that the degrees then decrease at a consistent rate. The degree drops by  $k$  and then by 1 twice to create a “group” of 3 terms. The degree then drops again by  $k-2$  and then by 1 to create a group of 5 terms. It can be seen that each polynomial follows this 1, 3, 5, 7, ... pattern, with the power of  $t$  decreasing by 1 in each successive group. There is always a  $-f + 1$  included.

We have empirically shown the general form of the minimal polynomial, but were unable to describe what the coefficients themselves are.

[A001764](#) [OEIe] and [A060941](#) [OEIa] are sequences currently in the OEIS. [A300386](#) [OEIf], [A300387](#) [OEIg], [A300388](#) [OEIh], and [A300389](#) [OEIi] are new and have been recently submitted and accepted.

### 8.3.4 Excursions with $\mathcal{S} = \{[1,3], [1, -k]\}$

Let us push our computers further. What does this family look like? We cannot derive a lot of information empirically as the  $\{[1,3], [1, -5]\}$  case already takes 28 seconds to run. The case  $k = 7$  ran for over one day and had used 3GiB of allocated memory before it was terminated. Again, we made the transformation  $t \rightarrow t^{1/(3+k)}$  for compact reading.

$$\underline{A002293}: \quad k = 1 \quad f^4 t - f + 1,$$

$$\underline{A060941}: \quad k = 2 \quad f^{10} t^2 + f^7 t - f^6 t + 2f^5 t - f + 1,$$

$$\begin{aligned} \underline{A300390}: \quad k = 4 \quad & f^{35} t^5 - f^{31} t^4 + f^{30} t^4 - f^{29} t^4 + 5f^{28} t^4 - f^{25} t^3 + f^{24} t^3 + 3f^{23} t^3 \\ & - 4f^{22} t^3 + 10f^{21} t^3 + f^{19} t^2 - f^{18} t^2 + 5f^{17} t^2 + 3f^{16} t^2 - 6f^{15} t^2 \\ & + 10f^{14} t^2 + f^{13} t - f^{12} t + 3f^{10} t + f^9 t - 4f^8 t + 5f^7 t - f + 1, \end{aligned}$$

$$\begin{aligned} \underline{A300391}: \quad k = 5 \quad & f^{56} t^7 - 2f^{51} t^6 + f^{50} t^6 - f^{49} t^6 + 7f^{48} t^6 + f^{46} t^5 - f^{45} t^5 - 3f^{43} t^5 \\ & + 5f^{42} t^5 - 6f^{41} t^5 + 21f^{40} t^5 - 3f^{37} t^4 - 3f^{36} t^4 + 8f^{35} t^4 + 10f^{34} t^4 \\ & - 15f^{33} t^4 + 35f^{32} t^4 - 2f^{31} t^3 + 2f^{30} t^3 - 9f^{28} t^3 + 22f^{27} t^3 \\ & + 10f^{26} t^3 - 20f^{25} t^3 + 35f^{24} t^3 + 3f^{22} t^2 + 5f^{21} t^2 - 9f^{20} t^2 \\ & + 18f^{19} t^2 + 5f^{18} t^2 - 15f^{17} t^2 + f^{16} (21t + 1) t - f^{15} t + 3f^{13} t \\ & - 3f^{12} t + 5f^{11} t + f^{10} t - 6f^9 t + 7f^8 t - f + 1. \end{aligned}$$

[A002293](#) [OEIj] and [A060941](#) [OEIa] are already in the OEIS while [A300390](#) [OEIk] and [A300391](#) [OEIl] are new. There appears to be some pattern again in the degree of  $f$ , though it is not discernible at first with so few datum. The degree happens to follow  $\binom{3+k}{3}$ .

**Conjecture 33.** Let  $\mathcal{S} = \{[1, a], [1, -b]\}$  with  $\gcd(a, b) = 1$ . Let  $f$  denote the g.f. for excursions (or meanders or bridges) with step set  $\mathcal{S}$ . Then the minimal polynomial of  $f$  has degree  $\binom{a+b}{a}$ .

The above conjecture is supported by all of the examples in this Section 8.3, as well as several other quick tests of accuracy. Though what does this mean in terms of deconstructing a walk with step set  $\mathcal{S}$ ?

This is somewhat intuitive for the unbounded case since there are  $\binom{a+b}{a}$  ways to have a bridge of smallest length:  $a + b$ . And we have provided ample evidence that the asymptotic behavior is only off by  $\frac{c}{n}$  for these excursions.

## 9 Conclusion and Future Work

To analyze walks, we began by examining the first step (Bounded Section 2), the last step (Semi-bounded Section 3), and finally a middle step that crosses the  $x$ -axis (Unbounded Section 6).

The way we dissected the g.f. equations is not unique. You could describe them in slightly different ways that may be more optimal. However, the minimal polynomial is called that for a reason; there is no “better” way to describe the g.f. except for an exact solution in special cases. It is important to make sure you do not double count or miss any walks in your descriptions. Define your various types of walks in a very particular manner.

For bounded cases we produced the new OEIS sequences [A301379](#) [OEIb] and [A301380](#) [OEIc] as well as [A301381](#) [OEIm] and [A300998](#) [OEIn], which are not included in this thesis, though they were produced in conjunction. Enumerating some semi-bounded and unbounded examples may be much faster by actually solving the polynomial for the g.f. in cases where  $\deg(p) \leq 4$  or  $p$  has “nice” roots.

One interesting note that appears in the semi-bounded case is that, no matter what we have chosen as our step set, the minimal polynomial has had terms  $-F + 1$ . This seems to indicate that there is always a way to write semi-bounded excursions and meanders as some combination (and deductions) of copies of ONLY ITSELF. Though it may be that the self-description is extremely complicated seemingly without (but it must be there) any combinatorial interpretation: see Example 10. We cannot necessarily do that with unbounded cases because there is not always a  $-F$  term.

Another trick we can use the minimal polynomial for is a bijective proof. All of the walks counted by terms with positive coefficients are also equally counted by those terms with negative coefficients. The trouble with this is that the two sides are very artificial and rarely something of interest by themselves.

In Section 7, we tried looking at the asymptotics of excursions and bridges. One note that came up was the relationship between the number of excursions and bridges. It is known that the number of Dyck paths is  $\frac{1}{n+1}$  of the total number of bridges. With a different looking step set, we



still obtained a  $\frac{c}{n}$  asymptotic relationship (for a constant  $c$ ). Is this always the case? We provided some heuristics and examples in support of the relationship but no found definitive proof. The relationship to meanders of the same step set appears to be much harder to state exactly.

A little further in Section 8.3, we examined many 2-step cases. I contributed the new sequences [A300386](#) [OEIf], [A300387](#) [OEIg], [A300388](#) [OEIh], [A300389](#) [OEIi], [A300390](#) [OEIk], and [A300391](#) [OEIl] to the OEIS. These are equivalent to walks that stay below certain lines of rational slope. A further question: how does one translate walks with general step sets bounded by lines into bridges, excursions, or meanders? What about bounded by something other than straight lines?

An extension that would be fairly easy, though laborious to implement, would be generalizing from 2D walks to 3D walks. Or to  $n$ -dimensional walks.

Beyond simply enumerating walks, we might want to know more about them: how many peaks or valleys do they contain? what is the area beneath the curve? how many times does the walk hit its maximal/minimal altitude? We can try to answer these questions by adding in a catalytic variable to count this new measurement. The generating function relations are very similar, but the g.f.s themselves are now functions of 2 (or more) variables. This can lead to systems that are not closed (under current descriptions). However, the system can still be iterated to enumerate terms.

Ayyer and Zeilberger analyzed how many times a bounded bridge or excursion hits its boundaries [AZ08] in the bounded and semi-bounded cases. The extension to measuring the area under the curve has been started. The bounded case still yields explicit solutions but the semi-bounded and unbounded cases are now left as a system of equations rather than a minimal polynomial.

Thank you for reading this paper. I hope you have enjoyed it and can make use of this package.

## 10 Acknowledgements

I would like to thank Doron Zeilberger for his direction in this paper. And thank you to Michael Saks for his discussion on the methods as well as insights into analysis of the asymptotics. This research was funded by a SMART Scholarship: USD/R&E (The Under Secretary of Defense-Research and Engineering), National Defense Education Program (NDEP) / BA-1, Basic Research.

Thank you for reading this paper. I hope you have enjoyed it and can make use of this package.

## References

- [AZ90] Gert Almkvist and Doron Zeilberger, *The method of differentiating under the integral sign*, Journal of Symbolic Computation **10** (1990), 571–591.
- [AZ07] Arvind Ayyer and Doron Zeilberger, *The Number of [Old-Time] Basketball games with Final Score  $n:n$  where the Home Team was never losing but also never ahead by more than  $w$  Points*, The Electronic Journal of Combinatorics **14.1** (2007), R19.
- [AZ08] ———, *Two dimensional directed lattice walks with boundaries*, Tapas in experimental mathematics, vol. 457, Contemporary Mathematics, Amer. Math. Soc., Providence, RI, 2008, pp. 1–19.
- [BKK<sup>+</sup>17] Cyril Banderier, Christian Krattenthaler, Alan Krinik, Dmitry Kruchinin, Vladimir Kruchinin, David Tuan Nguyen, and Michael Wallner, *Explicit formulas for enumeration of lattice paths: basketball and the kernel method*, arXiv:1609.06473v2 (2017).
- [BORW05] R. Brak, A.L. Owczarek, A. Rechnitzer, and S.G. Whittington, *A directed walk model of a long chain polymer in a slit with attractive walls*, J. Phys. A **38** (2005), 4309–4325.
- [DR71] E. A. DiMarzio and R. J. Rubin, *Adsorption of a Chain Polymer between Two Plates*, J. Chem. Phys. **55** (1971), 4318–4336.
- [Duc00] Philippe Duchon, *On the enumeration and generation of generalized Dyck words*, Discrete Mathematics **225** (2000), 121–135.
- [EZ15] Shalosh B. Ekhad and Doron Zeilberger, *The Method(!) of "Guess and Check"*, arXiv:1502.04377v1 (2015).

- [FS09] Philippe Flajolet and Robert Sedgewick, *Analytic Combinatorics*, Cambridge University Press, June 2009.
- [KP11] Manuel Kauers and Peter Paule, *The Concrete Tetrahedron*, Springer, 2011.
- [LZ17] Andrew Lohr and Doron Zeilberger, *Asymptotics of Rational Catalan Combinatorics*, RSP, April 2017.
- [MR09] Marni Mishna and Andrew Rechnitzer, *Two non-holonomic lattice walks in the quarter plane*, *Theor. Computer Science* **410** (2009), 3616–3630.
- [Nei] Neil J. A. Sloane, *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/>.
- [OEIa] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A060941>.
- [OEIb] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A301379>.
- [OEIc] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A301380>.
- [OEId] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A293946>.
- [OEIe] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A001764>.
- [OEIf] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300386>.
- [OEIg] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300387>.
- [OEIh] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300388>.
- [OEIi] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300389>.
- [OEIj] *OEIS Foundation Inc. (2018), The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A002293>.

- [OEIk] OEIS Foundation Inc. (2018), *The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300390>.
- [OEII] OEIS Foundation Inc. (2018), *The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300391>.
- [OEIm] OEIS Foundation Inc. (2018), *The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A301381>.
- [OEIn] OEIS Foundation Inc. (2018), *The On-Line Encyclopedia of Integer Sequences*, <http://oeis.org/A300998>.
- [SZ94] Bruno Salvy and Paul Zimmermann, *Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable*, ACM Transactions on Mathematical Software (TOMS) **20** (1994), no. 2, 163–177.
- [vR00] E. J. Janse van Rensburg, *The statistical mechanics of interacting walks, polygons, animals and vesicles*, vol. 18, Oxford University Press, Oxford, 2000.
- [WZ85] Jet Wimp and Doron Zeilberger, *Resurrecting the asymptotics of linear recurrences*, Journal of mathematical analysis and applications **111** (1985), no. 1, 162–176.