Algebraic Diagonals and Walks

Alin Bostan
Inria
France
Alin.Bostan@inria.fr

Louis Dumont
Inria
France
Louis.Dumont@inria.fr

Bruno Salvy
Inria, Laboratoire LIP (U. Lyon, CNRS, ENS Lyon, UCBL)
France
Bruno.Salvy@inria.fr

ABSTRACT

The diagonal of a multivariate power series \( F \) is the univariate power series \( \text{Diag} \, F \) generated by the diagonal terms of \( F \). Diagonals form an important class of power series; they occur frequently in number theory, theoretical physics and enumerative combinatorics. We study algorithmic questions related to diagonals in the case where \( F \) is the Taylor expansion of a bivariate rational function. It is classical that in this case \( \text{Diag} \, F \) is an algebraic function. We propose an algorithm that computes an annihilating polynomial for \( \text{Diag} \, F \). Generically, it is its minimal polynomial and is obtained in time quasi-linear in its size. We show that this minimal polynomial has an exponential size with respect to the degree of the input rational function. We then address the related problem of enumerating directed lattice walks. The insight given by our study leads to a new method for expanding the generating power series of bridges, excursions and meanders. We show that their first \( N \) terms can be computed in quasi-linear complexity in \( N \), without first computing a very large polynomial equation.

Categories and Subject Descriptors:
I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulations — Algebraic Algorithms

General Terms: Algorithms, Theory.

Keywords: Diagonals, walks, algorithms.

1. INTRODUCTION

Context. The diagonal of a multivariate power series with coefficients \( a_{i_1,\ldots,i_d} \) is the univariate power series with coefficients \( a_{i_1,\ldots,i_d} \). Particularly interesting is the class of diagonals of rational power series (ie, Taylor expansions of rational functions). In particular, diagonals of bivariate rational power series are always roots of nonzero bivariate polynomials (ie, they are algebraic series) [15, 17]. Since it is also classical that algebraic series are \( D \)-finite (ie, satisfy linear differential equations with polynomial coefficients), their coefficients satisfy linear recurrences and this leads to an optimal algorithm for the computation of their first terms [11, 12, 3]. In this article, we determine the degrees of these polynomials, the cost of their computation and related applications.

Previous work. The algebraicity of bivariate diagonals is classical. The same is true for the converse; also the property persists for multivariate rational series in positive characteristic [15, 24, 13]. The first occurrence we are aware of in the literature is Pólya’s article [22], which deals with a particular class of bivariate rational functions; the proof uses elementary complex analysis. Along the lines of Pólya’s approach, Furstenberg [15] gave a (sketchy) proof of the general result, over the field of complex numbers; the same argument has been enhanced later [18, 26, §6.3]. Three more different proofs exist: a purely algebraic one that works over arbitrary fields of characteristic zero [17, Th. 6.1] (see also [26, Th. 6.3.3]), one based on non-commutative power series [14, Prop. 5], and a combinatorial proof [6, §3.4.1]. Despite the richness of the topic and the fact that most proofs are constructive in essence, we were not able to find in the literature any explicit algorithm for computing a bivariate polynomial that cancels the diagonal of a general bivariate rational function.

Diagonals of rational functions appear naturally in enumerative combinatorics. In particular, the enumeration of unidimensional walks has been the subject of recent activity, see [1] and the references therein. The algebraicity of generating functions attached to such walks is classical as well, and related to that of bivariate diagonals. Beyond this structural result, several quantitative and effective results are known. Explicit formulas give the generating functions in terms of implicit algebraic functions attached to the set of allowed steps in the case of excursions [8, §4], [17], bridges and meanders [1]. Moreover, if \( a \) and \( b \) denote the upper and lower amplitudes of the allowed steps, the bound \( d_{a,b} = \binom{a+b}{a} \) on the degrees of equations for excursions has been obtained by Bousquet-Mélou, and showed to be tight for a specific family of step sets, as well as generically [7, §2.1]. From the algorithmic viewpoint, Banderier and Flajolet gave an algorithm (called the Platypus Algorithm) for computing a polynomial of degree \( d_{a,b} \) that annihilates the generating function for excursions [1, §2.3].

Contributions. We design (Section 4) the first explicit algorithm for computing a polynomial equation for the diagonal of an arbitrary bivariate rational function. We analyze its complexity and the size of its output in Theorem 14. The algorithm has two main steps. The first step is the computation of a polynomial equation for the residues of a bivariate rational function. We propose an efficient algorithm for this task, that is a polynomial-time version of Bronstein’s algorithm [9]; corresponding size and complexity bounds are given in Theorem 10. The second step is the computation of a polynomial equation for the sums of a fixed number of roots of a given polynomial. We design an additive version of the Platypus algorithm [1, §2.3] and analyze it in Theorem 12. We show in Proposition 16 that generically, the size of the minimal polynomial for the diagonal of a rational function is exponential in the degree of the input and that our algorithm computes it in quasi-optimal complexity (Theorem 14).

In the application to walks, we show how to expand to high precision the generating functions of bridges, excursions and meanders. Our main message is that pre-computing a polynomial equation for them is too costly, since that equation might have exponential size in the maximal amplitude \( d \) of the allowed steps. Our algorithms have quasi-linear complexity in the precision of the expansion, while keeping the pre-computation step in polynomial complexity in \( d \) (Theorem 18).

Structure of the paper. After a preliminary section on background
2. BACKGROUND AND NOTATION

In this section, we might be skipped at first reading. We introduce notation and technical results that will be used throughout the article.

2.1 Notation

In this article, \( \mathbb{K} \) denotes a field of characteristic 0. We denote \( \mathbb{K}[x]_n \) the set of polynomials in \( \mathbb{K}[x] \) of degree less than \( n \). Similarly, \( \mathbb{K}(x)_n \) stands for the set of rational functions in \( \mathbb{K}(x) \) with numerator and denominator in \( \mathbb{K}[x]_n \), and \( \mathbb{K}(x,y)_n \) for the set of power series in \( \mathbb{K}[x,y] \) truncated at precision \( n \).

If \( P \) is a polynomial in \( \mathbb{K}[x,y] \), then its degree with respect to \( x \) (resp. \( y \)) is denoted \( \deg_x P \) (resp. \( \deg_y P \)), and the bidegree of \( P \) is the pair \( \deg = (\deg_x, \deg_y) \). The notation \( \deg \) is used for univariate polynomials. Inequalities between bidegrees are component-wise.

The \emph{valuation} of a polynomial \( F \in \mathbb{K}[x] \) or a power series \( F \in \mathbb{K}[x] \) is its smallest exponent with nonzero coefficient. It is denoted \( \Val F \), with the convention \( \Val 0 = \infty \).

The \emph{reciprocal} of a polynomial \( P \in \mathbb{K}[x] \) is the polynomial \( \rec(P) = x^\deg P(1/x) \). If \( P = c(x - a_1) \cdots (x - a_n) \), then the \emph{newton series} \( (\mathbb{K}[x]_n) \) stands for the generating series of the Newton sums of \( P \):

\[
(\mathbb{K}[x]_n) = \sum_{n \geq 0} (a_1^n + a_2^n + \cdots + a_n^n)x^n.
\]

A squarefree decomposition of a nonzero polynomial \( Q \in A[y] \), where \( A = \mathbb{K} \) or \( \mathbb{K}[x] \), is a factorization \( Q = Q_1 \cdots Q_m \), with \( Q_i \) squarefree, the \( Q_i \)'s pairwise coprime and \( \deg(Q_i) > 0 \). The corresponding squarefree part of \( Q \) is the polynomial \( Q^* = Q_1 \cdots Q_m \). If \( Q \) is squarefree then \( Q = Q^* \).

The coefficient of \( x^d \) in a power series \( A \in \mathbb{K}[x] \) is denoted \( [x^d]A \). If \( A = \sum_{i=0}^m a_i x^i \), then \( A \mod x^d \) denotes the polynomial \( \sum_{i=0}^{d-1} a_i x^i \).

The exponential series \( \sum_{n \geq 0} a_n x^n/n! \) is denoted \( \exp(x) \).

The Hadamard product of two power series \( A \) and \( B \) is the power series \( A \odot B \) such that \( [x^d]A \odot B = [x^d][A \cdot[B] \) for all \( d \).

If \( F(x,y) = \sum_{i,j \geq 0} f_{ij} x^i y^j \) is a bivariate power series in \( \mathbb{K}[x,y] \), the \emph{diagonal} of \( F \), denoted \( \text{Diag} F \) is the univariate power series in \( \mathbb{K}[t] \) defined by \( \text{Diag} F(t) = \sum_{n \geq 0} f_{nn} t^n \).

2.2 Bivariate Power Series

In several places, we need bounds on degrees of coefficients of bivariate rational series. In most cases, these power series belong to \( \mathbb{K}(x,y)[y] \) and have a very constrained structure: there exists a polynomial \( Q \in \mathbb{K}[x] \) and an integer \( \alpha \in \mathbb{N} \) such that the power series can be written

\[
c_0 + c_1 x + \cdots + c_n x^n + \cdots \quad \text{with} \quad c_n \in \mathbb{K}[x,y], \quad \text{and} \quad \deg(c_n) \leq \alpha n,
\]

for all \( n \). We denote by \( \deg(Q) \) the set of such power series. Its main properties are summarized as follows.

Lemma 1. Let \( Q, R \in \mathbb{K}[x,y], \alpha, \beta \in \mathbb{N} \) and \( f \in \mathbb{K}[y] \).

(1) The set \( \deg(Q) \) is a subring of \( \mathbb{K}(x)[y] \).

(2) Let \( S \in \deg(Q) \) with \( S(0) = 0 \), then \( f(S) \in \deg(Q) \).

(3) The products obey

\[
\deg(Q) \cdot \deg(R) \subset \deg(\alpha + \deg R, \beta + \deg Q)(QR).
\]

Proof. For (3), if \( A = \sum a_{ij} x^i y^j / Q^d \) and \( B = \sum b_{ij} y^j / R^e \) belong respectively to \( \deg(Q) \) and \( \deg(R) \), then the nth coefficient of their product is a sum of terms of the form \( a_i(x)Q^{d+i}b_{n-i}(x)R^{e-n+i}/(QR)^n \). Therefore, the degree of the numerator is bounded by \( i(\alpha + \deg R) + (n - i)(\beta + \deg Q) \), whence (3) is proved. Property (1) is proved similarly. In Property (2), the condition on \( S(0) \) makes \( f(S) \) well-defined. The result follows from (1).

As consequences, we deduce the following two results.

Corollary 2. Let \( Q \in \mathbb{K}[x,y] \) with \( q(x) = Q(x,0) \) be such that \( q(0) \neq 0 \). Let \( Q^* \) be a squarefree part of \( Q \). Then

\[
\frac{1}{Q} \in \frac{1}{q} \deg(Q^*(x,0)).
\]

Proof. Write \( Q = q + R/q \in \deg(Q)(q) \). Then the result when \( Q \) is squarefree \( (Q = Q^*) \) follows from Part (2) of Lemma 1, with \( f = 1/(1+y) \). The general case then follows from Parts (1,3).

Proposition 3. Let \( Q \) and \( P \) be polynomials in \( \mathbb{K}[x,y] \), with \( Q(0,0) \neq 0 \), \( \deg_y Q > 0 \) and \( F = P/Q \). Then for all \( n \in \mathbb{N} \),

\[
\frac{d^n F}{d y^n} = \frac{A}{Q(Q^*)^n},
\]

with \( \deg(Q^*) = \deg(Q) - 1 \).

Proof. The Taylor expansion of \( F(x,y+t) \) has for coefficients the derivatives of \( F \). We consider it either in \( \mathbb{K}(x)[y,t] \) or \( \mathbb{K}(x,y)[t] \).

2.3 Complexity Estimates

We recall classical complexity notation and facts for later use. Let \( \mathbb{K} \) be again a field of characteristic zero. Unless otherwise specified, we estimate the cost of our algorithms by counting arithmetic operations in \( \mathbb{K} \) (denoted “ops.”) at unit cost. The soft-O notation \( \tilde{O}(\cdot) \) indicates that polylogarithmic factors are omitted in the complexity estimates.

We say that an algorithm has quasi-linear complexity if its complexity is \( \tilde{O}(d) \), where \( d \) is the maximal arithmetic size (number of coefficients in \( \mathbb{K} \) in a dense representation) of the input and of the output. In that case, the algorithm is said to be quasi-optimal.

Univariate operations. Throughout this article we will use the fact that most operations on polynomials, rational functions and power series in one variable can be performed in quasi-linear time. Standard references for these questions are the books [16] and [10]. The needed results are summarized in Fact 4 below.

4 The fact that operations can be performed in \( \tilde{O}(n) \) ops. in \( \mathbb{K} \):

(1) addition, product and differentiation of elements in \( \mathbb{K}[x]_n \), \( \mathbb{K}(x)_n \) and \( \mathbb{K}[x]_n \); integration in \( \mathbb{K}[x]_n \) and \( \mathbb{K}(x)_n \);

(2) extended gcd, squarefree decomposition and resultant in \( \mathbb{K}[x]_n \);

(3) multipoint evaluation in \( \mathbb{K}[x]_n \), \( \mathbb{K}(x)_n \) at \( O(n) \) points in \( \mathbb{K} \); interpolation in \( \mathbb{K}[x]_n \) and \( \mathbb{K}(x)_n \) from \( n \) (resp. \( 2n - 1 \)) values at pairwise distinct points in \( \mathbb{K} \);

(4) inverse, logarithm, exponential in \( \mathbb{K}[x]_n \) (when defined);

(5) conversions between \( P \in \mathbb{K}[x]_n \) and \( \deg(P) \mod x^d \in \mathbb{K}[x]_n \).

Multivariate operations. Basic operations on polynomials, rational functions and power series in several variables are hard questions from the algorithmic point of view. For instance, no general quasi-optimal algorithm is currently known for computing resultants of bivariate polynomials, even though in several important cases such algorithms are available [4]. Multiplication is the most basic non-trivial operation in this setting. The following result can be proved using Kronecker’s substitution; it is quasi-optimal for fixed number of variables \( m = O(1) \).

5 Polynomials in \( \mathbb{K}[x_1, \ldots, x_m] \) and power series in \( \mathbb{K}[x_1, \ldots, x_m] \) can be multiplied using \( \tilde{O}(2^m d_1 \cdots d_m) \) ops.
A related operation is multipoint evaluation and interpolation. The simplest case is when the evaluation points form an m-dimensional tensor product grid \( I_1 \times \ldots \times I_m \), where \( I_j \) is a set of cardinal \( d_j \).

**Fact 6** [20] Polynomials in \( \mathbb{K}[x_1, \ldots, x_m ; d_1, \ldots, d_m] \) can be evaluated and interpolated from values that they take on \( d_1 \times \ldots \times d_m \) points that form an m-dimensional tensor product grid using \( \mathcal{O}(m(\max d_1) \ldots \max d_m) \) ops.

Again, the complexity in Fact 6 is quasi-optimal for fixed \( m = O(1) \).

A general (although non-optimal) technique to deal with more involved operations on multivariable algebraic objects (eg., in \( \mathbb{K}[x, y] \)) is to use (multivariate) evaluation and interpolation on polynomials and to perform operations on the evaluated algebraic objects using Facts 4–6. To put this strategy in practice, the size of the output needs to be well controlled. We illustrate this philosophy on the example of resultant computation, based on the following easy variation of [16, Thm. 6.22].

**Fact 7** Let \( P(x, y) \) and \( Q(x, y) \) be bivariate polynomials of respective bidegrees \((d_0, d_2)\) and \((d_1, d_2)\). Then,
\[
\deg \text{Resultant}(P(x, y), Q(x, y)) \leq d_0^2 d_1^2 + d_0^2 d_2^2.
\]

**Lemma 8** Let \( P \) and \( Q \) be polynomials in \( \mathbb{K}[x_1, \ldots, x_m, y_1, \ldots, y_d] \). Then \( R = \text{Resultant}(P, Q) \) belongs to \( \mathbb{K}[x_1, \ldots, x_m, d_1, \ldots, d_m] \) where \( D_i \leq 1 + 2(d_1 - 1)(d_2 - 1) \). Moreover, the coefficients of \( R \) can be computed using \( \mathcal{O}(2^m d_1 \ldots d_md_{m+1}) \) ops. in \( \mathbb{K} \).

**Proof.** The degrees estimates follow from Fact 7. To compute \( R \), we use an evaluation-interpolation scheme: \( P \) and \( Q \) are evaluated at \( D \) points \((x_1, \ldots, x_m)\) forming an \( m \) dimensional tensor product grid; \( D \) univariate resultants in \( \mathbb{K}[y_1, \ldots, y_d] \) are computed; \( R \) is recovered by interpolation in \( \mathbb{K}[d_1, \ldots, d_m] \) ops. The second one has cost \( \mathcal{O}(Dd) \). Using the inequality \( D \leq 2^m d_1 \ldots d_md_{m+1} \) concludes the proof.

We conclude this section by recalling a complexity result for the computation of a squarefree decomposition of a bivariate polynomial.

**Fact 9** [19] A squarefree decomposition of a polynomial in \( \mathbb{K}[x, y; d_1, d_2] \) can be computed using \( \mathcal{O}(d_2^2 d_1) \) ops.

3. SPECIAL RESULTS

3.1 Polynomials for Residues

We are interested in a polynomial that vanishes at the residues of a given rational function. It is a classical result in symbolic integration that in the case of simple poles, there is a resultant formula for such a polynomial, first introduced by Rothstein [23] and Trager [27]. This was later generalized by Bronstein [9] to accommodate multiple poles as well. However, as mentioned by Bronstein, the complexity of his method grows exponentially with the multiplicity of the poles. Instead, we develop in this section an algorithm with polynomial complexity.

Let \( f \in P/Q \) be a nonzero element in \( \mathbb{K}(x) \), where \( P, Q \) are two coprime polynomials in \( \mathbb{K}[x] \). Let \( Q_1 Q_2 \cdots Q_m \) be a squarefree decomposition of \( Q \). For \( i \in \{1, \ldots, m\} \), if \( \alpha \) is a root of \( Q_i \) in an algebraic extension of \( \mathbb{K} \), then it is simple and the residue of \( f \) at \( \alpha \) is the coefficient of \( \alpha^{-1} \) in the Laurent expansion of \( f(\alpha + t) \) at \( t = 0 \). If \( V(y, t) \) is the polynomial \( (Q_1(y + t))/V_i(y, t) \), this residue is the coefficient of \( t^{-1} \) in the Taylor expansion at \( t = 0 \) of the regular rational function \( f(y + t)/V(y, t) \). This result can be used to evaluate at \( y = \alpha \). If this coefficient is denoted \( S_{i-1}(y) = A_i(y)/B_i(y) \), with polynomials \( A_i \) and \( B_i \), the residue at \( \alpha \) is a root of \( \text{Resultant}(A_i - zB_i, Q_i) \).

**Algorithm.** Squarefree Residues \((P/Q)\)

**Input.** Two polynomials \( P \) and \( Q \in \mathbb{K}[y] \).

**Output.** A polynomial in \( \mathbb{K}[z] \) canceling all the residues of \( P/Q \).

Compute \( Q_i^2 \cdots Q_m^2 \) a squarefree decomposition of \( Q \): for \( i = 1 \) to \( m \) do if \( \deg y_i = 0 \) then \( R_i \leftarrow \) else \( U_i(y) \leftarrow Q_i(y)/Q_i^2; V_i(y) \leftarrow (Q_i(y + t) - Q_i(y))/t; \)
Expand \( P(y + t)/V_i(y) \) as \( S_0 + \cdots + S_i - t^{i-1} + O(t^i) \);
Write \( S_1 \) as \( A_i(y)/B_i(y) \) with \( A_i \) and \( B_i \) coprime; \( R_i(z) \leftarrow \text{Resultant}(A_i - zB_i, Q_i) \);
return \( R_1 R_2 \cdots R_m \)

**Example 1.** Let \( d \geq 0 \) be an integer, and let \( G_d(x, y) \in \mathbb{Q}(x)[y] \) be the rational function \( y^d / (y - x)^{d+1} \). The poles have order \( d + 1 \). In this example, the algorithm can be performed by hand for arbitrary \( d \): a squarefree decomposition has \( m = d + 1 \) and \( Q_m = y - x \), the other \( Q_i \) being 1. Then \( V_m = 1 - 2y - t \) and the next step is to expand \( (y + t)^d / (1 - 2y - t)^{d+1} \).

Expanding the binomial series gives the coefficient of \( t^d \) as \( \frac{2^d}{m!} \), with
\[
A_m = \sum_{i=0}^d \binom{d+i}{i} y^i (1-2y)^{d-i}, \quad B_m = (1-2y)^{d+1}.
\]
The residues are then cancelled by \( \text{Resultant}(A_m - zB_m, Q_m) \), namely
\[
(1-4r)^{2d+1}2^d \sum_{k=0}^{\lfloor d/2 \rfloor} \frac{\binom{d}{2k}}{(2^k)^2} \left( \frac{zB_m}{2} \right)^{2k}.
\]

**Bounds.** In our applications, as in the previous example, the polynomials \( P \) and \( Q \) have coefficients that are themselves polynomials in another variable \( x \). Let \( (d_p, e_p) \), \( (d_Q, e_Q) \), \( (d_x, e_x) \) be the bidegrees in \( (x, y) \) of \( P \), \( Q \), \( Q^r \) and \( Q \), where \( Q^r = Q_1 \cdots Q_m \) is a squarefree part of \( Q \). In Algorithm 1, \( V \) has degree at most \( d_Q \) in \( x \) and total degree \( e_Q - 1 \) in \( y \). Similarly, \( P(y + t) \) has degree \( d_p \) in \( x \) and total degree \( e_p \) in \( y \). When \( e_p > 1 \), by Proposition 3, the coefficient \( S_{i-1} \) in the power series expansion of \( P(y + t)/V(y + t)/V_i(y, t) \) has denominator of bidegree bounded by \( (d_Q + jd_x, e_x - j + j(e_p - 1)) \) and numerator of bidegree bounded by \( (dp + jd_x, ep - j + j(e_p - 1)) \). Thus by Fact 7, \( \deg R_i \) is at most
\[
((i-1)d_x + \max(dp, d_Q))e_p + \min(dp, d_Q)(i-1) - i + \max(ep, 1, e_Q),
\]
while its degree in \( z \) is bounded by the number of residues \( e_p \). Summing over all \( i \) leads to the bound
\[
(e_Q - e_p) d_x + (d_Q - d_x)(e_p - 1) + e_p \max(dp, d_Q) - d_Q + d_x e_p + \max(ep, 1, e_Q).
\]
When \( e_p = 1 \), a direct computation gives the bound \( \max(dp, d_Q) + d_x e_p \).

**Theorem.** Let \( P(x, y)/Q(x, y) \in \mathbb{K}(x)[y] \). Then \( Q^r \) is a squarefree part of \( Q \) wrt \( y \). Let \( (d_x^*, d_y^*) \) be bounds on the bidegree of \( Q^r \). Then the polynomial computed by Algorithm 1 annihilates the residues of \( P/Q \), has degree in \( z \) bounded by \( d_x^* \) and degree in \( x \) bounded by \( 2d_x^*(d_y^* + 1) - 2d_y^* d_x^* \).

It can be computed in \( \mathcal{O}(m^2 d_x^* d_y^* (m^2 + d_x^* d_y^*)) \) operations in \( \mathbb{K} \).
Note that both bounds above (when $e^* > 1$ and $e^* = 1$) are upper bounded by $2d_d d_s$, independently of the multiplicities. The complexity is also bounded independently of the multiplicities by $O(d_d^2 d_s^2 d_s^3)$.

**Proof.** The bounds on the bidegree of $R = R_1 R_2 \cdots R_m$ are easily derived from the previous discussion.

By Fact 9, a squarefree decomposition of $Q$ can be computed using $O(d_d^2 d_s)$ ops. We now focus on the computations performed inside the $i$th iteration of the loop. Computing $U_i$ requires an exact division of polynomials of bidegrees at most $(d_i, d_s)$; this division can be performed by evaluation-interpolation in $O(d_i d_s)$ ops. Similarly, the trivariate polynomial $V_i$ can be computed by evaluation-interpolation wrt $(x, y)$ in time $O(d_i d_s^2)$. By the discussion preceding Theorem 10, both $A_i(x, y)$ and $B_i(x, y)$ have bidegrees at most $(d_i, E_s)$, where $d_i = d_t + d_s^2 t_s$ and $E_s = d_s + d_s^2 e_s$. They can be computed by evaluation-interpolation in $O(i d_i E_s)$ ops. Finally, the resultant $R_i(x, z)$ has bidegree at most $(d_s E_s + d_s D_s, e_s D_s)$, and since the degree in $y$ of $A_i \equiv B_i$ and $Q_i$ is at most $E_s$, it can be computed by evaluation-interpolation in $O(i (d_s E_s + d_s D_s) E_s)$ ops by Lemma 8. The total cost of the loop is thus $O(L_i)$, where

$$L_i = \sum_{j=1}^{m} (i + e_s^2)D_s E_s + d_t d_s E_s^2.$$ 

Using the (crude) bounds $D_s \leq D_m$, $E_s \leq E_m$, $\sum_{j=1}^{m} e_s^2 \leq d_s^2$ and $\sum_{j=1}^{m} d_t e_s \leq d_t^2 d_s$ shows that $L_i$ is bounded by

$$D_m E_m \sum_{j=1}^{m} (i + e_s^2) + E_m \sum_{i=1}^{m} d_t e_s \leq D_m E_m (m^2 + d_t^2) + E_m^2 d_t^2 d_s,$$

which, by using the inequalities $D_m \leq 2md_s^2$ and $E_m \leq 2md_s^2$, is shown to belong to $O(m^2 d_t^2 d_s^2 (m^2 + d_t^2))$ (using results from [2]), to be compared with the complexity bound from Theorem 10.

**Remark.** Note that one could also use Hermite reduction combined with the usual Rothstein-Trager resultant in order to compute a polynomial $R(x, z)$ that annihilates the residues. Indeed, Hermite reduction computes an auxiliary rational function that admits the same residues as the input, while only having simple poles. A close inspection of this approach provides the same bound $d_t^2$ for the degree in $y$ of $R(x, z)$, but a less tight bound for its degree in $x$, namely worse by a factor of $d_s^4$. The complexity of this alternative approach appears to be $O(d_s d_t (d_t + d_s^3))$ (using results from [2]), to be compared with the complexity bound from Theorem 10.

### 3.2 Sums of roots of a polynomial

Given a polynomial $P \in \mathbb{K}[y]$ of degree $d$, with coefficients in a field $\mathbb{K}$ of characteristic $0$, let $\alpha_1, \ldots, \alpha_d$ be its roots in the algebraic closure of $\mathbb{K}$. For any positive integer $c \leq d$, the polynomial of degree $(d^c)$ defined by

$$\Sigma^c P = \prod_{i_1 < \cdots < i_c} (y - (\alpha_{i_1} + \alpha_{i_2} + \cdots + \alpha_{i_c}))$$

has coefficients in $\mathbb{K}$. This section discusses the computation of $\Sigma^c P$ in Algorithm 2, which can be seen as an additive analogue of the Plücker algorithm of Bader and Fajloová [1].

We recall two classical formulas (see, eg, [4, §2]), the second one being valid for monic $P$ only:

$$\mathcal{N}(P) = \frac{\text{rec}(P)}{\text{rec}(cP)}, \quad \text{rec}(P) = \exp \left( \int \frac{d - \mathcal{N}(P)}{y} \, dy \right).$$

Truncating these formulas at order $d + 1$ makes $\mathcal{N}(P)$ a representation of the polynomial $P$ (up to normalization), since both conversions above can be performed quasi-optimally by Newton iteration [25, 31, 4]. The key for Algorithm 2 is the following variant of [1, §2.3].

**Proposition 11.** Let $P \in \mathbb{K}[y]$ be a polynomial of degree $d$, let $\mathcal{N}(P)$ denote the generating series of its Newton sums and let $S$ be the series $\mathcal{N}(P) \circ \exp(y)$. Let $\Psi_i$ be the polynomial in $\mathbb{K}[t_1, \ldots, t_{d-i}]$ defined by

$$\Psi_i(t_1, \ldots, t_{d-i}) = [z^c] \exp \left( \sum_{n \geq 1} (-1)^{n-1} t_n \frac{z^n}{n} \right).$$

Then the following equality holds

$$\mathcal{N}(\Sigma^c P) \circ \exp(y) = \Psi_i(S(y), S(2y), \ldots, S(cy)).$$

**Proof.** By construction, the series $S$ is

$$S(y) = \sum_{n \geq 0} (\alpha_1^n + \alpha_2^n + \cdots + \alpha_d^n) \frac{y^n}{n!} = \sum_{i=1}^{d} \exp(\alpha_i y).$$

When applied to the polynomial $\Sigma^c P$, this becomes

$$\mathcal{N}(\Sigma^c P) \circ \exp(y) = \sum_{|i| < -c} \exp((\alpha_{i_1} + \alpha_{i_2} + \cdots + \alpha_{i_c}) y)$$

$$= [z^c] \prod_{i=1}^{d} (1 + z \exp(\alpha_i y)).$$

This expression rewrites:

$$[z^c] \exp \left( \sum_{i=1}^{d} \log(1 + z \exp(\alpha_i y)) \right) = [z^c] \exp \left( \sum_{i=1}^{d} \sum_{m \geq 0} (-1)^{m-1} \frac{\exp(\alpha_i y \frac{z^m}{m})}{m} \right)$$

$$= [z^c] \exp \left( \sum_{m \geq 1} (-1)^{m-1} S_y \frac{z^m}{m} \right),$$

and the last expression equals $\Psi_i(S(y), S(2y), \ldots, S(cy))$.

**Theorem 2.** Let $P \in \mathbb{K}[x, y]_{d_t, d_s, d_c}$, let $c$ be a positive integer such that $c \leq d_t$, and let $D = (d_t^c, D)$. Let $\alpha \in \mathbb{K}[x]$ be the leading coefficient of $P$ w.r.t $y$ and let $\Sigma^c P$ be defined as in Eq. (2). Then $a^D \cdot \Sigma^c P$ is a polynomial in $\mathbb{K}[x, y]_{bidegree D}$ that cancels all sums $\alpha_1 + \cdots + \alpha_c$ of roots $\alpha_i(x)$ of $P$ with $i_1 < \cdots < i_c$. Moreover, this polynomial can be computed in $O(d_t^2 D^2)$ ops.
This result is close to optimal. Experiments suggest that for generic $P$ of bidegree $(d_x, d_y)$ the minimal polynomial of $\alpha t^i + \cdots + \alpha_s$ has bidegree $(d_x^{(d_s-1)}/d_t, d_y^{(d_s-1)}/d_t)$. In particular, our degree bound is precise in $y$, and overshoots by a factor of $d_t/c$ only in $x$. Similarly, the complexity result is quasi-optimal up to a factor of $d_t$ only.

**Proof.** The Newton series $\mathcal{N}(P)$ has the form

$$\mathcal{N}(P) = \frac{a \deg_x P + y A(x,y)}{a - y B(x,y)} xy^a \left( \sum_{n=0} \frac{y^n B(x,y)^n}{a^n} \right),$$

with $\deg_x A, \deg_y B \leq d_t$. Since both factors belong to $\mathcal{E}_d(a)$, Lemma 1 implies that $\mathcal{N}(P) \in \mathcal{E}_d(a)$. Applying this same lemma repeatedly, we get that $\Sigma P \in \mathcal{E}_{d_t}(a)$ (stability under the integration of Algorithm 2 is immediate). Since $\Sigma P$ has degree $D$ wrt $y$, we deduce that $a^D\Sigma P$ is a polynomial that satisfies the desired bound. By evaluation and interpolation at $1 + d_tD$ points, and Newton iteration for quotients of power series in $\mathbb{K}[y]_{1+d_t}$ (Fact 4), the power series $\mathcal{N}(P)$ can be computed in $O(d_tD^2)$ ops. The power series $S$ is then computed from $\mathcal{N}(P)$ in $O(d_tD^2)$ ops. To compute $F$ we use evaluation-interpolation wrt $x$ at $1 + d_tD$ points, and fast exponentials of power series (Fact 4). The cost of this step is $O(\mathcal{N}(P))$ ops. Then, $\mathcal{N}(P)$ is computed for $O(d_tD^2)$ additional ops. The last exponential is again computed by evaluation-interpolation and Newton iteration using $O(d_tD^2)$ ops. \qed

## 4. DIAGONALS

### 4.1 Algebraic equations for diagonals

The relation between diagonals of bivariate rational functions and algebraic series is classical \cite{Gosper1975,Levin1988}. We recall here the usual derivation when $\mathbb{K} = \mathbb{C}$ while setting our notation.

Let $F(x,y)$ be a rational function in $\mathbb{C}(x,y)$, whose denominator does not vanish at $(0,0)$. Then the diagonal of $F$ is a convergent power series that can be represented for small enough $t$ by a Cauchy integral

$$\operatorname{Diag} F(t) = \frac{1}{2\pi i} \oint_{F(t,y) \neq 0} \frac{1}{y} \frac{dt}{F(t,y)},$$

where the contour is for instance a circle of radius $r$ inside an annulus where $(t,y)$ remains in the domain of convergence of $F$. This is the basis of an algebraic approach to the computation of the diagonal as a sum of residues of the rational function

$$\frac{P(t,y)}{Q(t,y)} := \frac{1}{y} \frac{t}{\operatorname{Residue}(P(t,y) \div Q(t,y))},$$

with $P$ and $Q$ two coprime polynomials. For $t$ small enough, the circle can be shrunk around 0 and only the roots of $Q(t,y)$ tending to 0 when $t \to 0$ lie inside the contour \cite{Stothers1958}. These are called the small branches. Thus the diagonal is given as

$$\operatorname{Diag} F(t) = \sum_{Q(t,y) \neq 0 \land \text{Residue}(P(t,y) \div Q(t,y))} \frac{P(t,y)}{Q(t,y)},$$

where the sum is over the distinct roots of $Q$ tending to 0. We call their number the number of small branches of $Q$ and denote it by $\text{Nsmall}(Q)$.

Since the $y_i$’s are algebraic and finite in number and residues are obtained by series expansion, which entails only rational operations, it follows that the diagonal is algebraic too. Combining the algorithms of the previous section gives Algorithm 3 that produces a polynomial equation for $\operatorname{Diag} F$. The correctness of this algorithm over an arbitrary field of characteristic 0 follows from an adaptation of the arguments of Gessel and Stanley \cite[Th. 6.1]{Gessel1984}, \cite[Th. 6.3.3]{Stanley1984}.

**Example 2.** Let $d \geq 0$ be an integer, and let $F_d(x,y)$ be the rational function $1/(1 - x - y)^{d+1}$. The diagonal of $F_d$ is equal to

$$\sum_{n=0} \left( \binom{2n + d}{n} \binom{n + d}{d} \right).$$

Algorithm **AlgebraicDiagonal($A/B$)**

**Input** Two polynomials $A$ and $B \in \mathbb{K}[x,y]$, with $B(0,0) \neq 0$.

**Output** A polynomial $\Phi \in \mathbb{K}[t, \Delta]$ such that $\Phi(t, \text{Diag} A/B) = 0$.

**Write** $G \leftarrow P/Q$ with coprime polynomials $P$ and $Q$;

**R(z) as **AlgebraicResidues($P/Q$)**

$c \leftarrow \text{number of small branches of } Q$

$\Phi(t,z) \leftarrow \text{numerator(PureComposedSum}(R, c))$

**return** $\Phi(t, \Delta)$

**Algorithm 3.** Polynomial canceling the diagonal of a rational function

By the previous argument, it is an algebraic series, which is the sum of the residues of the rational function $G_d$ of Example 1 over its small branches (with $x$ replaced by $t$). In this case, the denominator is $y - t - y^2$. It has one solution tending to 0 with $t$; the other one tends to 1. Thus the diagonal is cancelled by the quadratic polynomial (1).

**Example 3.** For an integer $d > 0$, we consider the rational function

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

of bidegree $(d, d+1)$. The first step of the algorithm produces

$$G_d(t,y) = \frac{t^{d-1}}{y^d - y^d - y^{2d+1}},$$

whose denominator is irreducible with $d$ small branches. Running Algorithm 3 on this example, we obtain a polynomial $\Phi_d$ annihilating $\operatorname{Diag} F_d$, which is exponentially irreducible and whose bidegrees for $d = 1, 2, 3, 4$ are $(2, 3), (18, 10), (120, 35), (700, 126)$. From these values, it is easy to conjecture that the bidegree is given by

$$\left( \frac{d(d+1)}{d-1}, \frac{(2d+1)}{d} \right),$$

of exponential growth in the bidegree of $F_d$. In general, these bidegrees do not grow faster than in this example. In Theorem 14, we prove bounds that are barely larger than the values above.

### 4.2 Degree Bounds and Complexity

The rest of this section is devoted to the derivation of bounds on the complexity of Algorithm 3 and on the size of the polynomial it computes, which are given in Theorem 14.

**Degrees.** A bound on the bidegree of $\Phi$ will be obtained from the bounds successively given by Theorems 10 and 12.

In order to follow the impact of the change of variables in the first step, we define the diagonal degree of a polynomial $P(x,y) = \sum_{i,j} a_{i,j} x^i y^j$ as the integer $\deg_{\Phi}(P) := \sup \{ i - j \mid a_{i,j} \neq 0 \}$. We collect the properties of interest in the following.

**Lemma 13.** For any $P$ and $Q$ in $\mathbb{K}[x,y]$,

(1) $\deg_{\Phi}(P) \leq \deg_x P$;

(2) $\deg_{\Phi}(PQ) = \deg_{\Phi}(P) + \deg_{\Phi}(Q)$;

(3) there exists a polynomial $P \in \mathbb{K}[x,y]$, such that $P(x,y) = y^{-\deg_{\Phi}(P)} P_0(x,y)$, with $P_0(x,0) \neq 0$ and $\deg_{\Phi}(P) \leq \deg_x P + \deg_y P$;

(4) $\deg_{\Phi}(P^*) = (\deg_x P)^* + \deg_x (P^*)$.

**Proof.** Part (1) is immediate. The quantity $\deg_{\Phi}(P)$ is nothing else than $-\text{val}_x P(x,y)$, which makes Parts (2) and (3) clear too. From there, we get the identity $\partial_x \Phi = \partial_x \Phi$ for arbitrary $P$ and $Q$, whence $(P^*)^* = P^*$ and Part (4) is a consequence of Parts (1) and (3). \qed

Thus, starting with a rational function $F = A/B \in \mathbb{K}(x,y)$, with $(d_x, d_y)$ a bound on the bidegrees of $A$ and $B$, and $(d_x^*, d_y^*)$ a bound on
Algorithm 2 uses $\tilde{x}y$ to yield a polynomial and shifts the Newton polygon up by $d\deg$ vertices of the original Newton polygon to the vertices of the Newton polygon of $\tilde{Q}$. The $b$-bidegree of the input can be deduced from the above as

$$\deg_{\Delta} \Phi \leq \left( d_{\ast}^\ast + d_{y}^\ast \right) \left( N_{\small\text{small}}(B_{\ast}) + \deg_{\Delta}(B_{\ast}) \right).$$

(A sharper bound on the degree in $t$ can be derived as well.)

### 4.3 Optimization

Assume that the denominator of $F(x/y)$ is already partially factored as $Q(y) = \prod_{i=1}^{k} (y - \gamma_{i}(x))$, where the $\gamma_{i}$ are $k$ distinct rational branches among the $c$ small branches of $Q$. Then their corresponding (rational) residues $r_{i}$ contribute to the diagonal; therefore it is only necessary to invoke Algorithm 3 on $(\tilde{Q}, c - k)$, which produces a polynomial $\Phi$. Then the polynomial $\Phi(t, \Delta) = \Phi(t, \Delta - \sum r_{i})$ cancels the diagonal of $F$.

In particular, this optimization applies systematically for the factor $y^\alpha$ when $\alpha < 0$ (or equivalently $\alpha = 1$) in the algorithm. In this case, it yields a polynomial $\Phi$ with smaller degree than the original algorithm:

$$\deg_{\Delta} \Phi \leq \left( d_{\ast}^\ast + d_{y}^\ast \right) \left( N_{\small\text{small}}(B_{\ast}) + \deg_{\Delta}(B_{\ast}) \right).$$

### 4.4 Generic case

The bounds from Theorem 14 on the bidegree of $\Phi$ are slightly pessimistic with the variable $t$, but generally tighter wrt the variable $\Delta$, as will be proved in Proposition 16 below. We first need a lemma.

#### Lemma 15

Let $\mathbb{K}$ be a field of characteristic 0, and $P \in \mathbb{K}[y]$ be a polynomial of degree $d$, with Galois group $\Sigma_{d}$ over $\mathbb{K}$. Assume that the roots $\alpha_{1}, \ldots, \alpha_{d}$ of $P$ are algebraically independent over $\mathbb{Q}$. Then, for any $c \leq d$, the degree $d_{c}(P)$ of $P$ is irreducible in $[\mathbb{K}[y],[c]$.

**Proof.** Since $\Sigma = \alpha_{1} + \cdots + \alpha_{c}$ is a root of $\Sigma_{d}$, it suffices to prove that $\mathbb{K}(\Sigma)$ has degree $d_{c}$ over $\mathbb{K}$. The $\alpha_{i}$’s being algebraically independent, any permutation $\sigma$ in $\Sigma_{\alpha}$ of all the $\alpha_{i}$’s that leaves $\Sigma$ unchanged has to preserve $\alpha_{c+1} + \cdots + \alpha_{d}$ as well. It follows that $\mathbb{K}(\alpha_{1}, \ldots, \alpha_{d})$ has degree $d_{c}(d-c)!$ over $\mathbb{K}(\Sigma)$ and degree $d_{c}!$ over $\mathbb{K}$, so that $\mathbb{K}(\Sigma)$ has degree $d_{c}$ over $\mathbb{K}$. \(\square\)

#### Proposition 16

Let $A$ be a polynomial in $\mathbb{Q}[x,y]_{d_{x},d_{y}}$ and

$$B(x,y) = \sum_{i \leq d_{x}, j \leq d_{y}} b_{i,j} x^{i} y^{j} \in \mathbb{Q}[b_{i,j}]; x, y],$$

where the $b_{i,j}$ are indeterminates. Then the polynomial computed by Algorithm 3 with input $A/B$ is irreducible of degree $(d_{x}+d_{y})$ over $\mathbb{K} = \mathbb{Q}(b_{i,j}; x)$. \(\square\)

**Proof.** First apply the change of variables to obtain $G = P/Q$, with $Q(x,y) = \sum_{j \leq d_{x}, i = \epsilon} b_{i,j} x^{i} y^{d_{x}-i+\epsilon}$. Denote $d = d_{x} + d_{y}$. Then, the polynomial $Q(1,y)$ has the form $\sum_{j \leq d_{y}} t_{j} y^{j}$ where the $t_{j}$ are algebraically independent over $\mathbb{Q}$. Therefore, $Q(1,y)$ is Galois group $\Sigma_{d}$ over $\mathbb{Q}(t_{0}, \ldots, t_{d})$ and its roots are algebraically independent over $\mathbb{Q}$ [28, §57]. This property lifts to $Q(x,y)$ [28, §61], which thus has Galois group $\Sigma_{d}$ and algebraically independent roots, denoted $y_{1}, \ldots, y_{d}$. Now define the polynomial $R(x,y) = \prod_{j \leq d_{x}} (y - P(x,y)/\partial_{y}Q(x,y))$. Since $Q$ has simple roots, this is exactly the polynomial that is computed in Algorithm 1. The family $\{P(x,y)/\partial_{y}Q(x,y)\}$ is algebraically independent, since any algebraic relation between them would induce one for the $y_{i}$’s by clearing out denominators. In particular, the natural morphism $Gal(\mathbb{Q} / \mathbb{K}) = \Sigma_{d} \to Gal(\mathbb{R}/ \mathbb{Q})$ is injective, whence an isomorphism. (Here, $Gal(\mathbb{P} / \mathbb{K})$ denotes the Galois group of $P \in \mathbb{K}[y]$ over $\mathbb{K}$.) Since an immediate investigation of the Newton polygon of $Q$ shows that it has $d_{x}$ small branches, we conclude using Lemma 15. \(\square\)

Proposition 16 implies that for a generic rational function $A/B$ with $A \in \mathbb{K}[x,y]_{d_{x},d_{y}}$ and $B \in \mathbb{K}[x,y]_{d_{x}+1,d_{y}+1}$, the degree of $\Phi$ in $\Delta$ is $(\frac{d}{2})$. This is indeed observed on random examples.

**Example 4.** We consider a rational function $F(x,y) = 1/B(x,y)$, where $B(x,y)$ is a dense polynomial of bidegree $(d_{x},d_{y})$ chosen at random. For $d = 1,2,3,4$, algorithm AlgebraicDiagonal$(F)$ produces
irreducible outputs with bidegrees $(2, 2), (16, 6), (108, 20), (640, 70)$, that are matched by the formulas
\[
(2d^2 (2d - 2) d^{-1}, (2d) d^{-1}),
\]
so that the bound on deg_\* \( \Phi \) is tight in this case and the irreducibility of the output shows that Theorem 14 cannot be improved further.

5. WALKS

The exponential degree of the minimal polynomial of a diagonal proved in Proposition 16 concerns more generally other sums of residues, since this is the step where the exponential growth of the algebraic equations appears. This includes in particular constant terms of rational functions in \( C(x)[y] \), that can also be written as contour integrals of rational functions around the origin.

By contrast, sums of residues of a rational function always satisfy a differential equation of only polynomial size [2]. Thus, when an algebraic function appears to be connected to a sum of residues of a rational function, the use of this differential structure is much more adapted to the computation of series expansions, instead of going through a potentially large polynomial.

As an example where this phenomenon occurs naturally, we consider here the enumeration of unidimensional lattice walks, following Bandelier and Flajolet [1] and Bousquet-Mélou [7]. Our goal in this section is to study, from the algorithmic perspective, the series expansions of various generating functions (for bridges, excursions, meanders) that have been identified as algebraic [1]. One of our contributions is to point out that although algebraic series can be expanded fast [11, 12, 3], the pre-computation of a polynomial equation could have prohibitive cost. We overcome this difficulty by pre-computing differential (instead of polynomial) equations that have polynomial size only, and using them to compute series expansions to precision \( N \) for bridges, excursions and meanders in time quasi-linear in \( N \).

5.1 Preliminaries

We start with some vocabulary on lattice walks. A simple step is a vector \((1, u)\) with \(u \in \mathbb{Z}\). A step set \( \mathcal{S} \) is a finite set of simple steps. A unidimensional walk in the plane \( \mathbb{Z}^2 \) built from \( S \) is a finite sequence \((A_0, A_1, \ldots , A_n)\) of points in \( \mathbb{Z}^2 \), such that \( A_0 = (0, 0) \) and \( A_{k+1} - A_k = (1, u_k) \) with \((1, u_k) \in \mathcal{S}\). In this case \( n \) is called the length of the walk, and \( S \) is the step set of the walk. The \( y \)-coordinate of the endpoint \( A_n \), namely \( \sum_{k=1}^{n} u_k \), is called the final altitude of the walk. The characteristic polynomial of the step set \( S \) is
\[
\Phi_S(y) = \sum_{(1, u) \in \mathcal{S}} y^u.
\]

Following Bandierer and Flajolet, we consider three specific families of walks: bridges, excursions and meanders [1]. Bridges are walks with final altitude 0, meanders are walks confined to the upper half plane, and excursions are walks that are also meanders.

We define the full generating power series of walks
\[
W_S(x, y) = \sum_{n \geq 0, k \in \mathbb{Z}} w_{n,k} x^n y^k \in \mathbb{Z}[y, y^{-1}][[x]],
\]
where \( w_{n,k} \) is the number of walks with step set \( S \), of length \( n \) and final altitude \( k \). We denote by \( B_S(x) \) (resp. \( E_S(x) \), and \( M_S(x) \)) the power series \( \sum_{n \geq 0} w_{n,0} x^n \) (resp. excursions, and meanders) of length \( n \) with step set \( S \).

We omit the step set \( S \) as a subscript when there is no ambiguity. Several properties of the power series \( W, B, E \) and \( M \) are classical:

Fact 17 [1, §2.1-2.2] The power series \( W, B, E \) and \( M \) satisfy
\begin{enumerate}
\item \( W(x, y) \) is rational and \( W(x, y) = 1/(1 - x \Phi(y)) \);
\item \( B(x), E(x) \) and \( M(x) \) are algebraic;
\item \( B(x) = \left[ y^n \right] W(x, y) \);
\item \( E(x) = \exp\left( \int \left( B(x) - 1 \right) / x \, dx \right) \).
\end{enumerate}

Our main objective in what follows is to study the efficiency of computing the power series expansions of the series \( B, E \) and \( M \). In the next two sections, we first study two previously known methods, then we design a new one.

5.2 Expanding the generating power series

We denote by \( u^- \) (resp. \( u^+ \)) the largest \( u \) such that \((1, -u) \in S\) (resp. \((1, u) \in S\)) and denote by \( d \) the sum \( u^- + u^+ \). The integer \( d \) measures the vertical amplitude of \( S \); this makes \( d \) a good scale for measuring the complexity of the algorithms that will follow. We assume that both \( u^- \) and \( u^+ \) are positive, since otherwise the study of the excursions and meanders becomes trivial.

The direct method. The combinatorial definition of walks yields a recurrence relation for \( w_{n,k} \):
\[
w_{n,k} = \sum_{(1, u) \in \mathcal{S}} w_{n-1,k-u},
\]
with initial conditions \( w_{n,k} = 0 \) if \( n, k \leq 0 \) with \((n, k) \neq (0, 0)\), and \( w_{0,0} = 1 \). If \( w_{n,k} \) denotes the number of walks of length \( n \) and final altitude \( k \) that never exit the upper half plane, then \( w_{n,k} \) also satisfies recurrence [11], but with the additional initial conditions \( w_{n,k} = 0 \) for all \( k < 0 \). Then the bridges (resp. excursions, meanders) are counted by the numbers \( w_{n,0} \) (resp. \( w_{n,0} \)).

One can compute these numbers by unrolling the recurrence relation (11). Each use of the recurrence costs \( O(d) \) ops., and in the worst case one has to compute \( O(dN^2) \) terms of the sequence (for example, if the step set is \( S = \{(1, 1), \ldots , (1, d)\} \)). This leads to the computation of each of the generating series in \( O(d^4N^2) \) ops.

Using algebraic equations. Another method is suggested in [1, §2.3]. It relies on the algebraicity of \( B, E \) and \( M \) (Fact 17b). The series \( E \) and \( M \) can be expressed as products in terms of the small branches of the characteristic polynomial \( \Phi_S \) (see [1, Th. 1, Cor. 1]). From there, a polynomial equation can be obtained using the Platypus algorithm [1, §2.3], which computes a polynomial canceling the products of a fixed number of roots of a given polynomial. Given a polynomial equation \( P(z, E) = 0 \), another one for \( B \) can be deduced from the relation \( B = \frac{E^d}{E} + 1 \) as Resultant\((B - (1 + EP_k)z, P)\).

Once a polynomial equation is known for one of these three series, it can be used to compute a linear recurrence with polynomial coefficients satisfied by its coefficients [11, 12, 3]. This method produces an algorithm that computes the first \( N \) terms of \( B, E \) and \( M \) in \( O(N) \) ops. For this to be an improvement over the naive method for large \( N \), the dependence on \( d \) of the constant in the \( O(\cdot) \) should not be too large and the precomputation not too costly.

Indeed, the cost of the pre-computation of an algebraic equation is not negligible. Generically, the minimal polynomial of \( E \) has degree \( d \), which may be exponentially large with respect to \( d \) [7]. Empirically, the polynomials for \( B \) and \( M \) are similarly large.

The situation for differential equations and recurrences is different: \( B \) satisfies a differential equation of only polynomial size (see below), whereas (empirically), those for \( E \) and \( M \) have a potentially exponential size. These sizes then transfer to the corresponding recurrences and thereby to the constant in the complexity of unrolling them.

Example 5. With the step set \( S = \{(1, d), (1, 1), (1, -d)\} \) and \( d \geq 2 \), the counting series \( W_S \) equals
\[
W_S(x, y) = \frac{d^d}{y^d - x(1 + y^{d+1} + y^{2d})}.
\]

Experiments indicate that the minimal polynomial of \( B_S(x) \) has bidegree \((2d^2 - 2d - 1, d^2) \), exhibiting an exponential growth in \( d \). On the other hand, they show that \( B_S(x) \) satisfies a linear differential equation of order \( 2d - 1 \) and coefficients of degree \( d^2 + 3d - 2 \) for even \( d \), and \( d^2 + 3d - 4 \) for odd \( d \).

New Method. We now give a method that runs in quasi-linear time (with respect to \( N \)) and avoids the computation of an algebraic equation. Our method relies on the fact that periods of rational functions
Algorithm Walks($S, N$)

Input: A set $S$ of simple steps and an integer $N$
Output: $B_S$, $E_S$, and $M_S$ mod $x^{N+1}$

\[
\begin{align*}
F & \leftarrow W(x,y)/y \quad \text{[case B, E]} \quad \text{or} \quad W(x,y)/(1-y) \quad \text{[case M]} \\
D & \leftarrow \text{HermiteTelescoping}(F) \quad [2, \text{Fig. 3}] \\
R & \leftarrow \text{the recurrence of order } r \text{ associated to } D \\
I & \leftarrow \left[ y^0 W(x,y) \text{ mod } x^{r+1} \right] \quad \text{[case B, E]} \\
I & \leftarrow \left[ y^0 W(x,y)/(1-y) \text{ mod } x^{r+1} \right] \quad \text{[case M]} \\
B & \leftarrow y^0 W(x,y) \text{ mod } x^{N+1} \quad \text{(from R, I)} \\
A & \leftarrow y^0 W(x,y)/(1-y) \text{ mod } x^{N+1} \quad \text{(from R, I)} \\
E & \leftarrow \exp\left( -\int (B(x) - 1)/x \text{ dx} \right) \text{ mod } x^{N+1} \\
M & \leftarrow \exp\left( -\int A(x)/x \text{ dx} \right)/(1 - \Gamma(1)x) \text{ dx} \text{ mod } x^{N+1} \\
\end{align*}
\]
return $B_S$, $E_S$, and $M_S$

such as the one in Part (3) of Fact 17 satisfy differential equations of polynomial size in the degree of the input rational function [2]. We summarize our results in the following theorem, and then go over the proof in each case individually.

Theorem 18 Let $S$ be a finite set of simple steps and $d = u^r + u^s$. The series $B_S$ (resp. $E_S$ and $M_S$) can be expanded at order $N$ in $O(\tilde{d}^2 N)$ ops. (resp. $O(\tilde{d}^2 N)$ ops.), after a pre-computation in $O(\tilde{d}^3)$ ops.

5.3 Fast Algorithms

Bridges. To expand $B(x)$, we rely on Fact 17(3). The formula can be written $B = (1/2\pi i) \int W(x,y) dy$, the integration path being a circle inside a small annulus around the origin [1, proof of Th. 1]. Moreover, $W(x,y)/y$ is of the form $P/Q$, where $\text{bideg}(P) < (1, d)$ and $\text{bideg}(Q) < (0, 1)$. Since $P$ and $Q$ are relatively prime and $Q$ is primitive with respect to $y$, Algorithm HermiteTelescoping [2, Fig. 3] computes a telescoper for $P/Q$, which is also a differential equation satisfied by $B$, in $O(\tilde{d}^3)$ ops. The resulting differential equation has order at most $d$ and degree $O(\tilde{d}^2)$. This differential equation can be turned into a recurrence of order $O(\tilde{d}^2)$ in quasi-optimal time (see the discussion after [5, Cor. 2]). We may use it to expand $B(x)$ mod $x^N$ in $O(\tilde{d}^2 N)$ ops, once we have a way to compute the initial conditions. But this can be done using the naive algorithm described above in $O(\tilde{d}^3)$ ops. Thus, the total cost of the pre-computation is $O(\tilde{d}^3)$, as announced.

Excursions. If $B(x)$ mod $x^{N+1}$ is known, it is then possible to recover $E(x)$ mod $x^{N+1}$. Thanks to Fact 17(4), expanding $E(x)$ comes down to the computation of the exponential of a series, which can be performed using $O(N)$ ops. (Fact 4(4)).

Meanders. As in the case of excursions, the logarithmic derivative of $M(x)$ is recovered from a sum of residues by the following.

Proposition 19 The series $W$ and $M$ are related through

\[
A(x) = [y^0] \frac{y}{1-y} W(x,y), \quad M(x) = \exp\left( -\int A(x)/x \text{ dx} \right).
\]

Proof. Denote by $y_1, \ldots, y_m$ the small branches of the polynomial $y^r - x^s \Gamma(y)$. Then $M$ is given as [1, Cor. 1]:

\[
M(x) = \frac{1}{1 - x \Gamma(1)} \prod_{i=1}^m (1 - y_i).
\]