



Beating binary powering for polynomial matrices

Alin Bostan
Inria
Palaiseau, France

Vincent Neiger
Sorbonne Université, CNRS, LIP6
F-75005 Paris, France

Sergey Yurkevich
University of Vienna and Inria Saclay
Vienna, Austria

ABSTRACT

The N th power of a polynomial matrix of fixed size and degree can be computed by binary powering as fast as multiplying two polynomials of linear degree in N . When Fast Fourier Transform (FFT) is available, the resulting complexity is *softly linear* in N , i.e. linear in N with extra logarithmic factors. We show that it is possible to beat binary powering, by an algorithm whose complexity is *purely linear* in N , even in absence of FFT. The key result making this improvement possible is that the entries of the N th power of a polynomial matrix satisfy linear differential equations with polynomial coefficients whose orders and degrees are independent of N . Similar algorithms are proposed for two related problems: computing the N th term of a C-finite sequence of polynomials, and modular exponentiation to the power N for bivariate polynomials.

CCS CONCEPTS

• Computing methodologies → Algebraic algorithms.

KEYWORDS

Algebraic Algorithms; Computational Complexity; FFT; Binary Powering; C-finite Sequence; Rational Power Series; Linear Differential Equations; Creative Telescoping; Polynomial Matrices.

ACM Reference Format:

Alin Bostan, Vincent Neiger, and Sergey Yurkevich. 2023. Beating binary powering for polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation 2023 (ISSAC 2023), July 24–27, 2023, Tromsø, Norway*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3597066.3597118>

1 INTRODUCTION

A sequence $(u_n)_{n \geq 0}$ is called *C-finite* if it satisfies a linear recurrence relation whose coefficients are constant with respect to n . The famous sequence $(f_n)_{n \geq 0}$ of Fibonacci numbers, defined by the recurrence $f_{n+2} = f_{n+1} + f_n$ and the initial values $f_0 = 0, f_1 = 1$, is perhaps the most basic example of a C-finite sequence after the geometric ones $(q^n)_{n \geq 0}$. It is classical that the term f_N can be computed in $O(\log(N))$ arithmetic operations, thus as fast as q^N .

The authors thank Bruno Salvy for his remarks, and the anonymous referees for their reports. The authors are supported by the French project DE RERUM NATURA (ANR-19-CE40-0018) and by the joint French–Austrian project EAGLES (ANR-22-CE91-0007 & FWF I6130-N). The third author was supported by the ÖAW DOC fellowship P-26101.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ISSAC 2023, July 24–27, 2023, Tromsø, Norway

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0039-2/23/07...\$15.00

<https://doi.org/10.1145/3597066.3597118>

This can be achieved by *binary powering* for q^N , and in fact for f_N as well, since it is the top-right entry of C^N where C is the 2×2 companion matrix $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. This idea generalizes to any C-finite sequence $(u_n)_{n \geq 0}$: a recurrence of order $r \geq 1$ for $(u_n)_{n \geq 0}$ can be encoded, via its *companion matrix*, into an $r \times r$ matrix recurrence of order 1. Then the term u_N of the sequence appears as the first entry of the product of the vector of initial values $(u_0, \dots, u_{r-1})$ by the N th power of this $r \times r$ companion matrix [18, 31]. Then u_N can be computed in $O(\log(N))$ arithmetic operations, and in $O(N \log(N))$ bit operations if $(u_n)_{n \geq 0}$ is an integer sequence, using fast integer multiplication [21]. Here r is considered constant, i.e., $r \in O(1)$.

Fibonacci polynomials $F_n(x)$ are a natural generalization of Fibonacci numbers (see e.g. [10]). They are defined by the recurrence

$$F_{n+2}(x) = xF_{n+1}(x) + F_n(x) \quad \text{for } n \geq 0 \quad (1)$$

and the initial values $F_0(x) = 0, F_1(x) = 1$. The first few terms are $(F_n)_{n \geq 0} = (0, 1, x, x^2 + 1, x^3 + 2x, x^4 + 3x^2 + 1, \dots)$. Obviously, for all $n \geq 1$, the polynomial $F_n(x)$ is monic of degree $n - 1$ and the sum of its coefficients is $F_n(1) = f_n$.

Given $N \in \mathbb{N}$, the direct iterative algorithm for computing $F_N(x)$ has complexity $O(N^2)$. It computes, for each $n \leq N$, all the n coefficients of the intermediate polynomial $F_n(x)$; in total this amounts to $\Theta(N^2)$ coefficients. Therefore, if one wants to compute all of $(F_0, \dots, F_N)$ then this direct method is optimal with respect to the total arithmetic size of the output. However, it becomes quadratic if one is only interested in determining $F_N(x)$ alone.

To compute the polynomial $F_N(x)$ faster, one can use, as in the scalar case, the reformulation of the second-order recurrence (1) as a first-order (polynomial) matrix recurrence:

$$\begin{pmatrix} F_n & F_{n+1} \\ F_{n+1} & F_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix} \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}. \quad (2)$$

This shows that $F_n(x)$ is the top-right entry of the matrix $C(x)^n$, where $C(x)$ is the 2×2 companion matrix $C(x) = \begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix}$. One can again compute $C(x)^N$ using binary powering, whose costliest step is the multiplication of two polynomial matrices of degree about $N/2$. This yields $F_N(x)$ in complexity $O(M(N))$, where $M(N)$ denotes the cost of polynomial multiplication in degree at most N .

Using FFT-based polynomial multiplication [11], this amounts to a number of operations in the base field $\mathbb{K}$ which is quasi-linear in N . Not only does this compare favorably to the complexity $O(N^2)$ of the direct iterative algorithm, but this is even quasi-optimal (i.e., optimal up to logarithmic factors) with respect to the arithmetic size $\Theta(N)$ of the output polynomial $F_N(x)$.

In this context, the idea also generalizes to any C-finite sequence $(u_n(x))_{n \geq 0}$ of polynomials in $\mathbb{K}[x]$, which we will call *polynomial C-finite sequences*. Indeed, one can encode any recurrence of arbitrary (but independent of n) order $r \geq 1$ and coefficients in $\mathbb{K}[x]$ into a polynomial $r \times r$ matrix recurrence of order 1, and the N th term of the sequence, $u_N(x)$, can be computed as an element in the N th power of an $r \times r$ polynomial matrix multiplied by

the polynomial vector of initial values. Conversely, computing the N th power of any polynomial matrix can be reduced to computing terms in polynomial C-finite sequences (see the introduction of Section 4). Binary powering allows to solve both problems in $O(M(N))$ operations, and in $O(N^2 \log(N))$ bit operations if $\mathbb{K} = \mathbb{Q}$, considering both the recurrence order (or the matrix size) r and the recurrence degree (or the matrix degree) d as constant parameters, i.e., $r, d \in O(1)$. The main question addressed in this article is:

Can one achieve a better complexity for these tasks?

As far as scalar C-finite sequences are concerned, the algebraic complexity $O(\log(N))$ seems very difficult (if not impossible) to beat, but it is perhaps not impossible to improve the bit complexity $O(N \log(N))$ towards $O(N)$. While we do not achieve this, our results provide polynomial analogues for this type of improvement. As frequently noticed in computer algebra, polynomials are “computationally easier” to deal with than integers. In our case, philosophically, this comes from the fact that we can benefit from an additional operation on polynomials: differentiation. This possibly cryptic remark will hopefully become clear throughout Section 2. There, using Fibonacci polynomials as a test bench, we argue why it is indeed legitimate to hope for algorithms of complexity $O(N)$ for computing the N th term of a polynomial C-finite sequence.

Main result. Recall that a C-finite sequence is a sequence $(u_n)_{n \geq 0}$ of elements u_n in some ring R which satisfies a recurrence equation

$$u_{n+r} = c_{r-1}u_{n+r-1} + \dots + c_0u_n \quad \text{for all } n \geq 0, \quad (3)$$

for $c_0, \dots, c_{r-1} \in R$. In this work we consider polynomial C-finite sequences, i.e., the case $R = \mathbb{K}[x]$ for some (effective) field $\mathbb{K}$ of characteristic zero; thus $u_n = u_n(x) \in \mathbb{K}[x]$. The customary data structure for representing such a sequence consists of the polynomials $c_0(x), \dots, c_{r-1}(x)$ defining the recurrence and the r initial conditions $u_0(x), \dots, u_{r-1}(x) \in \mathbb{K}[x]$. The order of the recurrence (3) is r while its degree is the maximum of the degrees of the c_i ’s.

THEOREM 1.1. *Let $\mathbb{K}$ be an effective field of characteristic 0. Let d and r be fixed positive integers. For each of the following problems, there exists an algorithm solving it in $O(N)$ operations $(\pm, \times, \div)$ in $\mathbb{K}$:*

SEQTERM: Given a polynomial C-finite sequence $(u_n(x))_{n \geq 0}$ of order and degree at most r and d , compute the N th term $u_N(x)$.

BIVMODPOW: Given polynomials $Q(x, y)$ and $P(x, y)$ in $\mathbb{K}[x, y]$ of degrees in y and x at most r and d , with $P(x, y)$ monic in y , compute $Q(x, y)^N \bmod P(x, y)$.

POLMATPOW: Given a square polynomial matrix $M(x)$ over $\mathbb{K}[x]$ of size and degree at most r and d , compute $M(x)^N$.

Our algorithms for these problems make essential use of divisions in $\mathbb{K}$. We do not know if the complexity $O(N)$ can be achieved using only the operations $(+, -, \times)$ in $\mathbb{K}$.

Previous work. As already mentioned, the classical way of computing the N th term of a given C-finite sequence uses binary powering of the companion matrix, see e.g. [31]. Fiduccia’s algorithm [18] utilizes binary powering in a polynomial quotient ring and improves the complexity with respect to r (but not with respect to N). The fastest known algorithm [9] beats Fiduccia’s by a constant factor. In the polynomial C-finite case and assuming $r, d \in O(1)$, all these algorithms have a complexity in $O(M(N))$.

Beyond this classical approach, the previous work on the aforementioned problems consists of two distinct directions. The special case of Chebyshev polynomials of the second kind $U_n(x) = (-i)^n F_{n+1}(2ix)$ (with $F_n(x)$ the n th Fibonacci polynomial and i the imaginary unit) was considered in [27] (and later in [16]). These references present various methods for the computation of the Chebyshev polynomials (of the first and second kind) with complexity ranging from $O(N)$ to $O(N^3)$. The results in [16, 27] exploit the particular structure of these polynomials; except for possibly other families of classical orthogonal polynomials, for which explicit (hypergeometric) formulas exist, the methods in [16, 27] do not admit obvious generalizations.

An idea closely connected to a fundamental building block of our algorithms is explained in [19, Pbm. 4]. There, Flajolet and Salvy exploit the fact that, given a polynomial $P(x)$ in $\mathbb{K}[x]$, the coefficient sequence of the n th power $P(x)^n$ satisfies a linear recurrence of order independent of n , and with coefficients in $\mathbb{K}[x, n]$ of degree independent of n ; this recurrence allows them to compute (a selected coefficient of) $P(x)^N$ more efficiently than by binary powering. This idea has been applied in [7, §8] to count points on hyperelliptic curves over finite fields, with applications to cryptography. The technique also yields a general solution to SEQTERM when $r = 1$.

Outline. The following observation generalizes that in [19]: the coefficient sequence of the n th power of any algebraic function satisfies a recurrence of order and degree independent of n . From this, in Section 3, we give algorithms for SEQTERM with cost $O(N)$.

To complete the proof of Theorem 1.1, we design reductions between the three problems. Obviously $\text{POLMATPOW} \Rightarrow \text{SEQTERM}$, i.e., any algorithm for POLMATPOW with cost $O(N)$ induces one for SEQTERM with cost $O(N)$ as well. Indeed, the N th term of a polynomial C-finite sequence is equal to an entry of the product of the vector of initial values and the N th power of a companion matrix, and this polynomial vector-matrix multiplication costs $O(N)$. Conversely, it also holds that $\text{SEQTERM} \Rightarrow \text{POLMATPOW}$. One natural way to see this is to consider r^2 sequences corresponding to each entry of $M(x)^n$, with recurrence given by the characteristic polynomial of $M(x)$; see the introduction of Section 4. In Section 4.2, we give a more efficient algorithm for this reduction, based on an algorithm for $\text{SEQTERM} \Rightarrow \text{BIVMODPOW}$ described in Section 4.1.

Basics of complexity and D-finite functions. Hereafter, $\mathbb{K}$ denotes an effective field of characteristic zero. We analyze the performance of algorithms in the algebraic complexity model, meaning that arithmetic operations $(\pm, \times, \div)$ in the base field $\mathbb{K}$ are counted at unit cost. As before, $M(N)$ stands for the complexity of multiplying two polynomials in $\mathbb{K}[x]$ of degree at most N . With FFT-based multiplication $M(N) \in O(N \log(N) \log \log(N))$ [11], improved to $O(N \log(N))$ if $\mathbb{K}$ contains suitable roots of unity [14] or if $\mathbb{K}$ is a finite field [22]. A power series $f(x) \in \mathbb{K}[[x]]$ is said to be *D-finite* if it satisfies a linear differential equation (LDE) of the form

$$q_\ell(x)f^{(\ell)}(x) + \dots + q_0(x)f(x) = 0, \quad (4)$$

for some $q_0(x), \dots, q_\ell(x) \in \mathbb{K}[x]$ with $q_\ell(x) \neq 0$. Equivalently, writing $f(x) = \sum_{k \geq 0} f_k x^k$, the sequence $(f_k)_{k \geq 0}$ is *P-finite* (or, *P-recursive*), i.e., it satisfies a linear recurrence equation (LRE)

$$p_s(k)f_{k+s} + \dots + p_0(k)f_k = 0 \quad \text{for all } k \geq 0,$$

with polynomial coefficients $p_0(x), \dots, p_s(x) \in \mathbb{K}[x]$, and $p_s \neq 0$. Note that s and ℓ may differ in general, but $s \leq \ell + \max_i (\deg q_i(x))$. It also holds that $\max_i (\deg p_i(x)) \leq \ell$.

It is often useful to write (4) as $Lf(x) = 0$, where

$$L = q_\ell(x)\partial_x^\ell + \dots + q_0(x)$$

is an element in the noncommutative Weyl algebra $\mathbb{K}[x]\langle\partial_x\rangle$ of linear differential operators with multiplication governed by the Leibniz rule $\partial_x x = x\partial_x + 1$. The *order* ℓ of the differential operator L is the highest power of ∂_x occurring in L , and the *degree* of L is the highest power of x occurring in L . We recall that a *least common left multiple* (LCLM) of two differential operators $L_1, L_2 \in \mathbb{K}[x]\langle\partial_x\rangle$ is a differential operator $L \in \mathbb{K}[x]\langle\partial_x\rangle$ of minimal order such that there exist $A, B \in \mathbb{K}[x]\langle\partial_x\rangle$ with $L = AL_1 = BL_2$. LCLMs can be computed efficiently [6].

2 THE CASE OF FIBONACCI POLYNOMIALS

Before solving the first part (SEQTERM) of Theorem 1.1 in general, we propose in this section three different approaches that can be used to compute the N th Fibonacci polynomial $F_N(x)$ in complexity $O(N)$. Two of these methods have the advantage that they generalize to the case of arbitrary C-finite sequences.

The starting point of all that follows is the observation that the generating function $F(x, y) := \sum_{n \geq 0} F_n(x)y^n \in \mathbb{K}[x][[y]]$ of the sequence $(F_n(x))_{n \geq 0}$ is *rational*, and equal to $y/(1 - xy - y^2)$.

2.1 First method via a closed-form expression

By using the partial fraction decomposition

$$\frac{y}{1 - xy - y^2} = \frac{1}{\varphi_+(x) - \varphi_-(x)} \cdot \left(\frac{1}{1 - \varphi_+(x)y} - \frac{1}{1 - \varphi_-(x)y} \right)$$

where $\varphi_\pm(x) = (x \pm \sqrt{x^2 + 4})/2$ are the roots of $\varphi^2 - x\varphi - 1 = 0$, and by applying the geometric series, we get the closed-form expression

$$F_n(x) = \frac{\varphi_+(x)^n - \varphi_-(x)^n}{\varphi_+(x) - \varphi_-(x)} \quad \text{for all } n \geq 0. \quad (5)$$

Now, using the binomial formula twice, we obtain the formula

$$F_n(x) = \frac{1}{2^{n-1}} \cdot \sum_{\ell \geq 0} 4^\ell \left(\sum_{k \geq 0} \binom{n}{2k+1} \binom{k}{\ell} \right) x^{n-2\ell-1}. \quad (6)$$

The identity [20, 3.121] implies a “magic” simplification:

$$\sum_{k \geq 0} \binom{n}{2k+1} \binom{k}{\ell} = 2^{n-1-2\ell} \binom{n-\ell-1}{\ell}. \quad (7)$$

In conclusion, from (6) and (7) it follows that

$$F_n(x) = \sum_{\ell \geq 0} \binom{n-\ell-1}{\ell} x^{n-2\ell-1}. \quad (8)$$

With this expression at hand, it becomes transparent that one can compute $F_N(x)$ efficiently. Indeed, by writing $F_N(x) = \sum_{k=0}^{N-1} f_k x^k$, it follows from (8) that $(f_k)_{n \geq 0}$ satisfies the recurrence relation

$$f_{k+2} = \frac{(N+k+1)(N-k-1)}{4(k+1)(k+2)} f_k \quad \text{for all } k \geq 0. \quad (9)$$

Moreover, (8) also gives $(f_0, f_1) = (1, 0)$ for odd N and otherwise $(f_0, f_1) = (0, N/2)$. With these initial conditions, it is now clear that $F_N(x)$ can be computed in $O(N)$ by unrolling the recurrence (9).

As mentioned in the introduction, the analogue of formula (8) for the case of Chebyshev polynomials of the first kind $T_n(x)$ was already exploited in [27, §1.9]. The disadvantage of this approach is that for general polynomial C-finite sequences there is no hope for a closed-form expression like (8).

2.2 Second method via algebraic substitution

There is another method for computing $F_N(x)$ in $O(N)$, which has the advantage that it generalizes to any C-finite sequence, as we will show in Section 3.1. The crucial remark (Lemma 3.2) is that since $\varphi_\pm(x)$ is algebraic, $\varphi_\pm(x)^n$ satisfies a “small” LDE, of order and degree independent of n . The same holds for $1/(\varphi_+(x) - \varphi_-(x))$, therefore for $F_n(x)$ as well. More precisely, $\varphi_\pm(x)^n$ satisfies the LDE

$$(x^2 + 4)y''(x) + xy'(x) - n^2 y(x) = 0,$$

and $1/(\varphi_+(x) - \varphi_-(x)) = (x^2 + 4)^{-1/2}$ satisfies the LDE

$$(x^2 + 4)y'(x) + xy(x) = 0.$$

Using (5), it then follows that the polynomial $F_n(x)$ satisfies

$$(x^2 + 4)y''(x) + 3xy'(x) + (1 - n^2)y(x) = 0. \quad (10)$$

Writing $F_N(x) = \sum_{k=0}^{N-1} f_k x^k$, plugging into (10) for $n = N$ and extracting the $(k+2)$ nd coefficient, it now follows that the sequence $(f_k)_{k \geq 0}$ satisfies recurrence (9). The initial conditions f_0, f_1 are given by $F_N(x) \bmod x^2$ which can be found in complexity $O(\log(N))$ by computing the N th power of the companion matrix (2) in $\mathbb{K}[x]/(x^2)$ by binary powering and reducing mod x^2 in each step. As before, unrolling recurrence (9) with these initial terms provides a way to compute $F_N(x)$ in complexity $O(N)$.

2.3 Third method via Creative Telescoping

Writing $F(x, y) = y/(1 - xy - y^2)$ we are interested in a differential equation for the coefficient of y^N in $F(x, y)$. By Cauchy’s integral formula, we have for sufficiently small $\epsilon > 0$:

$$F_N(x) = [y^N]F(x, y) = \frac{1}{2\pi i} \oint_{|y|=\epsilon} \frac{y}{(1 - xy - y^2)y^{N+1}} dy.$$

Then the method of creative telescoping [1] can be used to find an LDE for the integral above. For example, the command

DEtools[Zeilberger](1/(1-x*y-y^2)/y^n, x, y, Dx);

in Maple immediately finds that

$$\left((x^2 + 4)\partial_x^2 + 3x\partial_x + 1 - n^2 \right) \frac{F(x, y)}{y^{n+1}} = \partial_y \left(\frac{F(x, y)}{y^n} C(x, y) \right),$$

where $C(x, y) = (n+1 - nxy - (n-1)y^2)/(1 - xy - y^2)$. By Cauchy’s integral theorem, the contour integral of the right-hand side vanishes, and (10) follows. Then one can conclude in the same way as in the previous method and compute $F_N(x)$ in complexity $O(N)$.

2.4 Comments on the three approaches

It is natural to ask ourselves what in these approaches was just luck, what was truly specific to the particular example of the Fibonacci polynomials, and what can be extended to the general case.

It is clear that the key for computing $F_N(x)$ in complexity $O(N)$ is the existence of the recurrence (9) (or equivalently the LDE (10)). Even though there is no hope for a closed-form solution in general, we shall prove that such a recurrence always exists for polynomial

C-finite sequences. We should, however, definitely be careful and avoid proving tautologic statements. Since $u_N(x)$ is a polynomial, it does satisfy the first-order LDE $u_N(x)y'(x) - u'_N(x)y(x) = 0$, but this one is trivial for our purposes. Indeed, converting this differential equation into a recurrence satisfied by the sequence of coefficients of $u_N(x)$ yields a recurrence of order $\deg(u_N)$, which is obviously useless for computing the coefficients of u_N . Rather, we would like to find an LRE/LDE whose order and degree are independent of N . This is the purpose of the next section. Specifically, in §3.1 we explain how it can be computed by algebraic substitution (generalizing §2.2) and in §3.2 we show that it can also be found via creative telescoping (generalizing §2.3).

3 POLYNOMIAL C-FINITE SEQUENCES

Recall that a polynomial C-finite sequence $(u_n(x))_{n \geq 0}$ is a sequence of polynomials $u_n(x) \in \mathbb{K}[x]$ that satisfies a recurrence

$$u_{n+r}(x) = c_{r-1}(x)u_{n+r-1}(x) + \cdots + c_0(x)u_n(x), \quad (11)$$

of some order $r \in \mathbb{N}$, with coefficients $c_0(x), \dots, c_{r-1}(x) \in \mathbb{K}[x]$. The degree of (11) is $d = \max_i(\deg c_i(x))$. The sequence $(u_n(x))_{n \geq 0}$ is defined uniquely by (11) if r initial terms $u_0(x), \dots, u_{r-1}(x)$ are prescribed. The characteristic polynomial of (11) is defined as

$$\chi(y) = y^r - c_{r-1}(x)y^{r-1} - \cdots - c_1(x)y - c_0(x) \in \mathbb{K}[x, y].$$

The generating function $U(x, y) := \sum_{n \geq 0} u_n(x)y^n$ is rational:

$$U(x, y) = \frac{v_0(x) + \cdots + v_{r-1}(x)y^{r-1}}{y^r \chi(1/y)}, \quad (12)$$

with $v_k(x) := u_k(x) - c_{r-1}(x)u_{k-1}(x) - \cdots - c_{r-k}(x)u_0(x)$.

Let $a_1(x), \dots, a_k(x) \in \mathbb{K}(x)$ be the roots of $\chi(y)$, and $m_1, \dots, m_k$ be their multiplicities. By partial fraction decomposition and geometric series, any sequence $(u_n(x))_{n \geq 0}$ satisfying (11) has the form

$$u_n(x) = q_1(n, x)a_1(x)^n + \cdots + q_k(n, x)a_k(x)^n, \quad (13)$$

where $k \leq r$ and each $q_i(n, x) \in \mathbb{K}(a_1(x), \dots, a_n(x))[n]$ is a polynomial in n of degree at most $m_i - 1$, for $i = 1, \dots, k$.

3.1 Computing $u_N(x)$ in $O(N)$

By generalizing the ideas of Section 2.2, it is not difficult to prove that the n th term of a polynomial C-finite sequence $(u_n(x))_{n \geq 0}$ satisfies an LDE whose order and degree are independent of n , and consequently, that there exists a linear recurrence relation for the coefficient sequence of $u_n(x)$ whose order (say s) and degree are again independent of n . Then, for a given $N \in \mathbb{N}$, first computing initial terms by binary powering of the companion matrix in $\mathbb{K}[x]/(x^s)$ and then unrolling this recurrence for $n = N$, we achieve a complexity $O(N)$ for the computation of $u_N(x)$.

THEOREM 3.1. *Let $(u_n(x))_{n \geq 0}$ be a polynomial C-finite sequence. Then there exists $L_n \in \mathbb{K}[n, x][\partial_x]$ with order and degree independent of n , and such that $L_n(u_n(x)) = 0$. Consequently, writing $u_n(x) = \sum_{k \geq 0} c_{n,k}x^k$, there exist, for some $s \in \mathbb{N}$ independent of n , polynomials $p_0(n, x), \dots, p_s(n, x) \in \mathbb{K}[n, x]$ of degrees independent of n , and such that the sequence $(c_{n,k})_{k \geq 0}$ satisfies the recurrence*

$$p_s(n, k)c_{n,k+s} + \cdots + p_0(n, k)c_{n,k} = 0, \quad k \geq 0. \quad (14)$$

In the theorem above it is crucial that neither the order nor the degree of L_n depend on n . Since each $u_n(x)$ is a polynomial, it is a tautology to say that it satisfies *some* LDE: one may simply take $L = \partial_x^\alpha$, where $\alpha > \deg(u_n(x))$ or $L = u_n(x)\partial_x - u'_n(x)$. However, it is a nontrivial fact that $u_n(x)$ satisfies an LDE of the form

$$p_\ell(n, x)u_n^{(\ell)}(x) + \cdots + p_0(n, x)u_n(x) = 0$$

for some $p_i(n, x) \in \mathbb{K}[n, x]$ with ℓ and $\deg_x p_i$ independent of n .

The most direct proof of Theorem 3.1 uses the explicit expression (13) for $u_n(x)$ and the following classical fact about algebraic substitution into D-finite functions. Recall that a function $a(x)$ is called *algebraic* over $\mathbb{K}(x)$ if it satisfies a nontrivial polynomial relation $P(x, a(x)) = 0$ for some $P(x, y) \in \mathbb{K}[x, y]$. Size and complexity bounds on differential equations for algebraic functions, and more generally on algebraic substitution, are given in [5, 26].

LEMMA 3.2. *Let $a(x)$ be an algebraic function over $\mathbb{K}(x)$ and let $g(x)$ be D-finite. Then $f(x) = g(a(x))$ is D-finite. In particular, $a(x)^n$ satisfies an LDE of order and degree independent of n .*

PROOF. The first part is a classical result, see for example [34, Thm. 2.7]. In the proof one shows that the vector space spanned over $\mathbb{K}(x)$ by $(f^{(i)}(x))_{i \geq 0}$ is finite-dimensional over $\mathbb{K}(x, a(x))$ which is itself finite-dimensional over $\mathbb{K}(x)$. For the second part, it is enough to set $g(x) = x^n$ which satisfies $xg'(x) = ng(x)$. $\square$

Example 3.3. Like in Section 2 let $\varphi_\pm(x) = (x \pm \sqrt{x^2 + 4})/2$ be the roots of $y(x)^2 + xy(x) - 1 = 0$. Then $\varphi_\pm(x)^n$ satisfy the LDE

$$(x^2 + 4)y''(x) + xy'(x) - n^2y(x) = 0.$$

PROOF OF THEOREM 3.1. Write $u_n(x)$ as in (13). By Lemma 3.2, each $a_i(x)^n$ satisfies an LDE of order and degree independent of n , hence the same holds for $q_i(n, x)a_i(x)^n$, and finally for $u_n(x)$. It follows that the coefficient sequence of $u_n(x)$ is P-finite with order and degree independent of n . $\square$

Since all steps in the proofs above are effective and independent of N , this leads to Algorithm 1. Its Lines 2 to 7 can be seen as “precomputations” since they do not depend on N . As already mentioned, Line 8 has complexity $O(\log(N))$ and Line 9 has complexity $O(N)$. Thus, Algorithm 1 solves SEQTERM in complexity $O(N)$, up to a potential issue during the unrolling at Line 9 of the recurrence from Line 7. Indeed, this unrolling may be impossible for some values k , namely those for which $p_s(N, k)$ vanishes. We will explain how to overcome this problem in Section 3.3.

For practical applications, however, computing the polynomials $q_i(x, n)$ in Line 4 as well as the LCLM in Line 6 is algorithmically somewhat cumbersome. Thus, generalizing the approach in Section 2.3, we now propose a variant of Algorithm 1 which replaces Lines 2 to 6 by an algorithm based on creative telescoping.

3.2 Computing L_n with Creative Telescoping

Let $U(x, y) = \sum_{n \geq 0} u_n(x)y^n \in \mathbb{K}[x][[y]]$ be the generating function (12) of $(u_n(x))_{n \geq 0}$. The sequence is C-finite, so $U(x, y)$ is a rational function. Moreover, the Cauchy integral formula implies

$$u_n(x) = \frac{1}{2\pi i} \oint_{|y|=\epsilon} \frac{U(x, y)}{y^{n+1}} dy. \quad (15)$$

Algorithm 1 SeqTermAS($(u_n)_n, N$)

Input: A polynomial C-finite sequence $(u_n(x))_{n \geq 0}$ given by (11) with initial conditions, and $N \in \mathbb{N}$.

Output: The polynomial $u_N(x)$.

- 1: $d \leftarrow \deg_x(\chi(y))$ and $\delta \leftarrow \max_{i=0, \dots, r-1}(\deg_x u_i(x))$
- 2: $\chi(y) \leftarrow$ the characteristic polynomial of $(u_n(x))_{n \geq 0}$
- 3: $a_1(x), \dots, a_k(x) \leftarrow$ the roots of $\chi(y)$
- 4: Compute minimal polynomials for $q_1(x, n), \dots, q_k(x, n) \in \mathbb{K}[a_1(x), \dots, a_k(x)][n]$ such that (13) holds.
- 5: For each i deduce an LDE $L_{i,n} \in \mathbb{K}[n, x]\langle \partial_x \rangle$ with order and degree independent of n such that $L_{i,n}(q_i(x, n)a_i(x)^n) = 0$
- 6: $L_n \leftarrow \text{LCLM}(L_{1,n}, \dots, L_{k,n}) \in \mathbb{K}[n, x]\langle \partial_x \rangle$
- 7: Compute a recurrence $p_s(n, k)c_{n,k+s} + \dots + p_0(n, k)c_{n,k} = 0$ satisfied by any solution $f_n(x) = \sum_{k \geq 0} c_{n,k}x^k$ of $L_n y = 0$
- 8: Using binary powering of the companion matrix of the initial recurrence mod x^s , compute the values $c_{N,0}, \dots, c_{N,s-1}$
- 9: Unroll the recurrence from Line 7 for $n = N$ and with initial terms from Line 8 (see also Section 3.3 for further details)
- 10: **return** $\sum_{k=0}^{Nd+\delta} c_{N,k}x^k$

A *telescoper* of $U(x, y)/y^{n+1}$ is a differential operator

$$L = p_k(x)\partial_x^k + \dots + p_0(x) \in \mathbb{K}[x]\langle \partial_x \rangle,$$

such that L applied to $U(x, y)/y^{n+1}$ is $\partial_y(C(x, y))$ for some rational function $C(x, y)$ called the *certificate*. By the Cauchy integral theorem, $\oint_{|y|=\epsilon} \partial_y(C(x, y))dy = 0$, and it follows that $Lu_n(x) = 0$, i.e., L yields a differential equation for $u_n(x)$. In this section we will prove that for $U(x, y)/y^{n+1}$ there exists a telescoper $L_n \in \mathbb{K}[n, x]\langle \partial_x \rangle$ whose order and degree do not depend on n . Our proof relies on reduction-based creative telescoping and repeatedly uses *Hermite reduction* algorithms [2, 3].

We now introduce the necessary definitions and recall the Hermite reduction method. For a more detailed introduction, a full complexity analysis, and applications of reduction-based creative telescoping to integration of bivariate rational functions, we refer to [2]. Let $\mathbb{L} = \mathbb{K}(x)$. For a polynomial $Q(y) \in \mathbb{L}[y]$, let $Q = Q_1 Q_2^2 \dots Q_k^k$ be its squarefree factorization and let $Q^* = Q_1 \dots Q_k$ denote the squarefree part of Q . We set $Q^- := Q/Q^*$. Recall that, given $P, Q \in \mathbb{L}[y]$, the *Hermite reduction* algorithm computes two polynomials $A, a \in \mathbb{L}[y]$ with $\deg_y a < \deg_y Q^*$ such that

$$\frac{P}{Q} = \partial_y \left(\frac{A}{Q^-} \right) + \frac{a}{Q^*}.$$

Given a bivariate rational function $H(x, y) = P(y)/Q(y) \in \mathbb{K}(x, y)$, one may compute the Hermite reduction (A_i, a_i) of $\partial_x^i H$ for $i = 0, 1, \dots$. Since $\deg_y a_i$ is uniformly bounded by $d^* = \deg_y Q^*$ for each i , the $d^* + 1$ functions $\{a_i(x, y) : 0 \leq i \leq d^*\}$ will be linearly dependent over $\mathbb{K}(x)$. Hence one can find $q_0(x), \dots, q_{d^*}(x) \in \mathbb{K}(x)$ not all zero, such that $\sum_{i=0}^{d^*} q_i(x) \partial_x^i = 0$. It follows then that $L = \sum_{i=0}^{d^*} q_i(x) \partial_x^i$ is a telescoper for H .

This procedure cannot be directly applied to $U(x, y)/y^{n+1}$ if n is an indeterminate. At the same time, if $n = N \in \mathbb{N}$ is fixed, it is *a priori* not obvious that $\deg_x q_i(x)$ will be independent of N . Moreover, the complexity of the algorithm will depend on N , which we want to avoid. As we will now explain, to achieve this, one should see $U(x, y)/y^{n+1}$ not as a rational function in x and y with

potentially large degree in the numerator, but as a hyperexponential function with the parameter n appearing solely as a coefficient in the logarithmic derivative. Recall that $H(x, y)$ is called *hyperexponential* if both $\partial_x H/H$ and $\partial_y H/H$ belong to $\mathbb{K}(x, y)$.

For hyperexponential functions, the Almkvist-Zeilberger algorithm [1] was the first practical method to find telescopers and certificates. Indeed, as we mentioned in Section 2.3, the command

DEtools[Zeilberger](1/(1-x*y-y^2)/y^n, x, y, Dx);

in Maple immediately finds the differential equation for the n th Fibonacci polynomial for a variable n . Note that if n is specialized to an integer N before the execution of the command above, the implemented algorithm becomes slower as N grows.

It is, however, not clear that the Almkvist-Zeilberger algorithm applied to $U(x, y)/y^{n+1}$ will always find a telescoper whose degree and order are independent of n , even though we know from Section 3.1 that an LDE with this property exists. Therefore, to have a complete algorithm based on creative telescoping, we will invoke the reduction-based method for hyperexponential functions first introduced and analyzed in [3]. Using the implementation of the latter work, the command in Maple

HermiteTelescoping(1/(1-x*y-y^2)/y^n, x, y, Dx);

also immediately finds the correct LDE for $F_n(x)$. The practical advantage for our purpose of using the reduction-based algorithm in comparison to the Almkvist-Zeilberger method is shown in Section 5 (Table 1). The theoretical advantage comes from the following lemma, which guarantees that the algorithm will find a telescoper for $U(x, y)/y^{n+1}$, and consequently an LDE for $u_n(x)$, whose order and degree do not depend on n .

LEMMA 3.4. *Let $P(y) \in \mathbb{L}[n, y]$ and $Q(y) \in \mathbb{L}[y]$ with $Q(0) \neq 0$. Set $d_n := \deg_y P(y)$, $d^* := \deg_y Q^*(y)$ and let k be the highest pure power in the square free factorization of $Q(y)$. Then there exist $B(n, y) \in \mathbb{L}(n)[y]$ and $b(n, y) \in \mathbb{L}[n, y]$ with $\deg_y b(n, y) \leq d^*$ and $\deg_n b(n, y) \leq d_n + k$ such that*

$$\frac{P(y)}{Q(y)y^{n+1}} = \partial_y \left(\frac{B(n, y)}{Q^-(y)y^n} \right) + \frac{b(n, y)}{Q^*(y)y^{n+1}}. \quad (16)$$

PROOF. We are going to prove the statement by induction on $d^- := \deg_y Q^-$. If $d^- = 0$, then $Q^* = Q$ and the Euclidean division gives $P = P_1 Q + b_1$ with $\deg_y b_1 < d^*$. Moreover,

$$\frac{P_1(y)}{y^{n+1}} = \partial_y \left(\frac{B_1(n, y)}{y^n} \right),$$

where $B_1(n, y)$ is $P_1(y)$ with the k th coefficient p_k replaced by $p_k/(k-n)$. Setting $B = B_1$ and $b = b_1$ proves the induction basis.

Now assume that $d^- > 0$ and note that

$$\partial_y \left(\frac{B(n, y)}{Q^-(y)y^n} \right) = y^{-n} \partial_y \left(\frac{B(n, y)}{Q^-(y)} \right) - y^{-n-1} n \frac{B(n, y)}{Q^-(y)},$$

so equation (16) is equivalent to

$$\frac{P(y)}{Q(y)y} = \partial_y \left(\frac{B(n, y)}{Q^-(y)} \right) - n \frac{B(n, y)}{Q^-(y)y} + \frac{b(y)}{Q^*(y)y}. \quad (17)$$

The Hermite reduction applied to $\frac{P(y)}{Q(y)y}$ yields $A(y), a(y) \in \mathbb{L}[n, y]$ with $\deg_y a(y) \leq d^*$ and $\deg_n A(y), \deg_n a(y) \leq d_n$ such that

$$\frac{P(y)}{Q(y)y} = \partial_y \left(\frac{A(y)}{Q^-(y)} \right) + \frac{a(y)}{Q^*(y)y}. \quad (18)$$

Comparing (17) and (18), we now look at

$$H(y) := \frac{a(y)}{Q^*(y)y} + n \frac{A(y)}{Q^-(y)y}.$$

The denominator of $H(y)$ is $R(y)y := \text{lcm}(Q^*, Q^-)y$. Clearly, $R^* = Q^*$ and $\deg_y R^- < d^-$. The highest pure power in the square free factorization of $R(y)$ is at most $k-1$ and the degree of the numerator in n of $H(y)$ is at most $d_n + 1$. Hence, by induction, we may write

$$\frac{a(y)}{Q^*(y)y} + n \frac{A(y)}{Q^-(y)y} = \partial_y \left(\frac{C(n, y)}{Q^-(y)} \right) - n \frac{C(n, y)}{Q^-(y)y} + \frac{c(n, y)}{Q^*(y)y}$$

with $\deg_y c(n, y) < d^*$ and $\deg_n c(n, y) \leq d_n + k$. Setting $B(n, y) = A(y) + C(n, y)$ and $b(y) = a(y) + c(n, y)$ finishes the proof. $\square$

The proof of Lemma 3.4 induces an algorithm for the computation of $B(n, y)$ and $b(n, y)$ given $P(y), Q(y) \in \mathbb{K}[x, y]$ such that (16) holds, $\deg_y b \leq \deg_y Q^*$ and also $\deg_n b$ bounded in terms of Q . It can be seen as a special case of the procedure `HERMITEREDUCTION` in [3]. The LDE for $u_n(x)$ can now be found as in Algorithm 2.

Algorithm 2 `TelescNthTerm( $U(x, y)$ )`

Input: A rational function $U(x, y) \in \mathbb{K}(x, y) \cap \mathbb{K}[[x, y]]$.

Output: A diff. operator in $\mathbb{K}[n, x]\langle \partial_x \rangle$ for $u_n(x) = [y^n]U(x, y)$.

- 1: Write $U(x, y) = P(x, y)/Q(x, y)$ and let $d^* = \deg_y Q^*(x, y)$
 - 2: For each $i = 0, \dots, d^*$ compute the polynomial $b_i(n, x, y) = b(n, y)$ as in Lemma 3.4 applied to $\partial_x^i U(x, y)/y^{n+1}$
 - 3: Find a linear relation of $\{b_i(n, x, y) : 0 \leq i \leq d^*\}$ over $\mathbb{K}(n, x)$, that is polynomials $q_0(n, x), \dots, q_{d^*}(n, x)$ not all zero with $\sum_{i=0}^{d^*} q_i(n, x) b_i(n, x, y) = 0$
 - 4: Return the differential operator $\sum_{i=0}^{d^*} q_i(n, x) \partial_x^i \in \mathbb{K}[n, x]\langle \partial_x \rangle$
-

Note that, as in the usual reduction-based creative telescoping, the linear relation at Line 3 exists because $\deg_y(b_i(n, x, y))$ is uniformly bounded by d^* . Writing $U(x, y) = P(x, y)/Q(x, y)$, the operator $L = \sum_{i=0}^{d^*} q_i(n, x) \partial_x^i$ annihilates $u_n(x) = [y^n]U(x, y)$ since

$$\begin{aligned} 2\pi i \cdot L_n u_n(x) &= L_n \oint \frac{U(x, y)}{y^{n+1}} dy = \oint L_n \frac{P(x, y)}{Q(x, y)y^{n+1}} dy \\ &= \oint \partial_y \frac{\sum_{i=1}^{d^*} q_i(n, x) B_i(n, y)}{Q^-(y)y^n} dy + \oint \frac{\sum_{i=0}^{d^*} q_i(n, x) b_i(n, x, y)}{Q^*(y)y^{n+1}} dy; \end{aligned}$$

the first integral vanishes by Cauchy's integral theorem, and the second integral vanishes by construction of the $q_i(n, x)$.

This provides a variant for Lines 2 to 6 of Algorithm 1, as described in Algorithm 3. The above-mentioned potential issue with unrolling persists; the next section deals with this problem.

3.3 The singular case

In this section we discuss the potential issue of our algorithm that can occur if the sequence for the coefficients of $u_n(x)$ cannot be unrolled due to singularities. We shall first highlight this problem and its solution by means of an example, then in the last paragraph of this section we explain the general strategy.

Consider the polynomial C-finite sequence $u_n(x)$ given by $u_{n+3}(x) - (x^2 + x + 2)u_{n+2}(x) + x(x^2 + 2x + 2)u_{n+1}(x) - 2x^3u_n(x) = 0$, for all $n \geq 0$ with initial conditions $u_0 = 3, u_1 = x^2 + x + 2, u_2 = x^4 + x^2 + 4$. The characteristic polynomial of the defining recurrence

Algorithm 3 `SeqTermCT( $(u_n)_n, N$ )`

Input: A polynomial C-finite sequence $(u_n(x))_{n \geq 0}$ given by (11) with initial conditions, and $N \in \mathbb{N}$.

Output: The polynomial $u_N(x)$.

- 1: $U(x, y) \leftarrow$ the rational generating function of $u_n(x)$ in (12)
 - 2: $L_n \leftarrow \text{TELESCNTHTERM}(U(x, y))$
 - 3: $\triangleright$ follow Lines 7 to 10 of Algorithm 1
-

is easily computed and turns out to factor completely over $\mathbb{K}[x][y]$:

$$\chi(x, y) = (y - 2)(y - x)(y - x^2).$$

With the initial conditions and after a partial fraction decomposition it follows that the generating function of $u_n(x)$ is given by

$$U(x, y) = \frac{1}{1 - 2y} + \frac{1}{1 - x^2y} + \frac{1}{1 - xy}.$$

Hence, the solution is $u_n(x) = 2^n + x^n + x^{2n}$ and can be written down in $O(\log(N))$ operations. However, as we shall explain now, the direct application of any of the methods described earlier fails.

According to Theorem 3.1, $u_n(x)$ satisfies an LDE whose degree and order are independent of n . Indeed, using creative telescoping one quickly finds an annihilator for $u_n(x) = \oint U(x, y)/y^{n+1} dx$:

$$(x^2 \partial_x^3 - 3x(n-1) \partial_x^2 + (2n-1)(n-1) \partial_x) u_n(x) = 0.$$

Converting this LDE to a recurrence for the coefficient sequence of $u_n(x) = \sum_{k \geq 0} c_{n,k} x^k$ we find

$$(2n-k)(n-k)kc_{n,k} = 0, \quad k \geq 0. \quad (19)$$

In other words, $c_{n,k} = 0$ for all $k \in \mathbb{N}$ except $k \in \{0, n, 2n\}$. In order to “unroll” this recurrence we need to know $c_{n,0}, c_{n,n}$ and $c_{n,2n}$. However, it is not immediately clear how to compute those terms for $n = N$ in $O(N)$ operations from the initial input (without using the explicit solution).

We propose the following easily generalizable solution: consider $v_n(x) = u_n(x+1)$. Then the LDE for $v_n(x)$ is given by

$$((x+1)^2 \partial_x^3 - 3(x+1)(n-1) \partial_x^2 + (2n-1)(n-1) \partial_x) v_n(x) = 0,$$

and for the coefficient sequence of $v_n(x) = \sum_{k \geq 0} d_{n,k} x^k$ we find

$$(k+1)(k+2)d_{n,k+2} - (k+1)(3N-2k-1)d_{n,k+1} + (2n-k)(n-k)d_{n,k} = 0.$$

Now the leading coefficient of the recurrence is $(k+1)(k+2) \neq 0$, so we can easily unroll it after determining the first two terms, by computing them via binary powering of the corresponding companion matrix mod x^2 . Having computed $v_N(x)$, it remains to find $u_N(x) = v_N(x-1)$. Note that expanding the polynomial results in an $O(M(N))$ algorithm. However, recall from (19) that we only need to compute $c_{N,N}$ and $c_{N,2N}$, or, in other words, the coefficients of x^N and x^{2N} in $v_N(x-1)$. For any i it holds that

$$c_{N,i} = \sum_{k \geq 0} d_{N,k} \binom{k}{i} (-1)^{k-i}, \quad (20)$$

and the sum is finite because $v_N(x)$ is a polynomial. Clearly, it can be computed in complexity $O(N)$ for any i .

Generally speaking, an issue with unrolling the recurrence for $(c_{n,k})_{k \geq 0}$ occurs if the roots of the leading polynomial are positive integers that depend on n . Indeed, roots that are nonintegral clearly do not cause any problems in the unrolling step and if a root is independent of n then we may just compute more initial terms while

the complexity of this step stays bounded by $O(\log N)$. Let S be the set of the problematic roots. Note that the size of S is independent of n since S is a subset of the roots of the leading polynomial $p_s(n, x)$ in (14) and $\deg_x p_s(n, x)$ is bounded independently of n by Theorem 3.1. Moreover, S can be nonempty only if the LDE for $u_n(x)$ is singular at 0 (that is, if $q_\ell(x) = q_\ell(n, x)$ in (4) vanishes at $x = 0$). In this case, one can always define $v_n(x) = u_n(x + c)$ for $c \in \mathbb{K}$ a nonsingular point of the LDE ($q_\ell(n, c) \neq 0$). Then the coefficients $d_{n,k}$ of $v_n(x)$ can be computed from $O(1)$ initial conditions via unrolling a recurrence. Using the formula (20) (with $-c$ instead -1) and the fact that $v_n(x)$ is a polynomial, one can compute the coefficients $c_{N,i}$ for $i \in S$. It is then possible to unroll the recurrence for $(c_{N,k})_{k \geq 0}$ and find $u_N(x)$ in complexity $O(N)$.

4 IMPACT ON POLYNOMIAL MATRIX POWER

Here is an algorithm for **POLMATPOW** using **SEQTERM**. Let $M(x)$ in $\mathbb{K}[x]^{r \times r}$ and $p_{i,j,n}(x)$ be the (i, j) entry of $M(x)^n$, for $n \geq 0$ and i and j in $\{1, \dots, r\}$. The sequence $(p_{i,j,n}(x))_{n \geq 0}$ is polynomial C -finite, with a recurrence given by the characteristic polynomial

$$\chi_M(x, y) := \det(yI_r - M(x)) = y^r - c_{r-1}(x)y^{r-1} - \dots - c_0(x).$$

That is, $p_{i,j,n+r}(x) = c_{r-1}(x)p_{i,j,n+r-1}(x) + \dots + c_0(x)p_{i,j,n}(x)$ for all $n \geq 0$. Thus, to compute $M(x)^N$, it is enough to find $\chi_M(x, y)$ (in $O(1)$, i.e. independent of N), to compute the polynomials $p_{i,j,n}(x)$ for $1 \leq i, j \leq r$ and $0 \leq n < r$ (also in $O(1)$) and to return the entries $p_{i,j,N}(x)$ of $M(x)^N$ via **SEQTERM**. As such, this approach uses r^2 calls to **SEQTERM**, with total cost $O(N)$.

This section describes an algorithm for **POLMATPOW** which uses only r such calls, through a direct reduction to **BIVMODPOW** (see Section 4.2). Our solution for **BIVMODPOW**, via r calls to **SEQTERM**, is presented in Section 4.1 and completes the proof of Theorem 1.1.

4.1 Computing bivariate modular powers

Let $\mathbb{L} = \mathbb{K}[x]$ and $P, Q \in \mathbb{L}[y]$. Assume that P , seen as a univariate polynomial in y of degree r , is monic. For $N \in \mathbb{N}$, the Euclidean division in $\mathbb{L}[y]$ ensures the existence of unique $S, R \in \mathbb{L}[y]$ such that $\deg_y(R) < r$ and $Q^N = SP + R$. The polynomial R is $Q^N \bmod P$. Assume that P and Q are fixed, and let $d := \deg_x(P)$ (which is thus in $O(1)$). Then, writing $R = \sum_{i=0}^{d-1} r_i(x)y^i$, it holds that $\deg_x r_i(x) = O(N)$. The efficient computation of R when $Q = y$, given $P(x, y)$ and N , is the first step for proving **BIVMODPOW** in Theorem 1.1.

We shall first illustrate the connection of **SEQTERM** and **BIVMODPOW** by means of an example. Let $P(x, y) = y^2 - xy - 1$ and $Q(x, y) = y$, i.e., we are looking for $F_{n-1}(x), F_n(x) \in \mathbb{L}$ such that

$$y^n = S(x, y)(y^2 - xy - 1) + yF_n(x) + F_{n-1}(x), \quad (21)$$

Algorithm 4 **BivModPow**($P(x, y), Q(x, y), N$)

Input: $P(x, y), Q(x, y) \in \mathbb{L}[y]$ with $P(x, y)$ monic in y , and $N \in \mathbb{N}$.

Output: $Q(x, y)^N \bmod P(x, y)$.

- 1: $A(x, t) \leftarrow \text{Res}_y(P(x, y), t - Q(x, y))$
- 2: $\bar{A}(x, t) \leftarrow t^r A(x, 1/t)$, where $r := \deg_t A(x, t)$
- 3: **for** $i = N - r + 1, \dots, N$ **do** $u_i(x) \leftarrow [t^i] \frac{1}{\bar{A}(x, t)}$ using **SEQTERM**
- 4: $u(x, t) \leftarrow u_{N-r+1}(x) + \dots + u_N(x)t^{r-1}$
- 5: $v(x, t) \leftarrow u(x, t)\bar{A}(x, t) \bmod t^r$; $B(x, t) \leftarrow v(1/t)t^{r-1}$
- 6: **return** $B(x, Q(x, y)) \bmod P(x, y)$

for some polynomial $S(x, y) \in \mathbb{L}[y]$. Replace y by $1/y$ in (21) and then multiply by $y^{n+1}/(1 - xy - y^2)$ to obtain

$$\frac{y}{1 - xy - y^2} = S(x, 1/y)y^{n-1} + y^n \frac{F_n(x) + yF_{n-1}(x)}{1 - xy - y^2}.$$

Now observe that $\deg_y(S(x, 1/y)y^{n-1}) \leq n-1$, hence by extracting the n th and $(n+1)$ st coefficients,

$$F_n(x) = [y^n] \frac{y}{1 - xy - y^2} \quad \text{and} \\ F_{n-1}(x) + xF_n(x) = [y^{n+1}] \frac{y}{1 - xy - y^2}.$$

We conclude that $F_k(x)$ is the k th Fibonacci polynomial, for $k = n$ and $k = n-1$. In particular, each $F_k(x)$ satisfies a linear recurrence with constant polynomials and can be found in $O(k)$ by **SEQTERM**.

This strategy, outlined on an example, generalizes in the obvious way. Explicitly, we have the following lemma (see [9, Lem. 2]).

LEMMA 4.1. *Let $P \in \mathbb{K}[x, y]$ and $r := \deg_y(P)$, with $P(x, 0) \neq 0$ and reversal $\bar{P}(x, y) := y^r P(x, \frac{1}{y})$. Write $\frac{1}{P(x, y)} = \sum_{k \geq 0} u_k(x)y^k$. Finally, let $v(x, y) = (u_{N-r+1}(x) + \dots + u_N(x)y^{r-1})\bar{P}(x, y) \bmod y^r$. Then $y^N \bmod P(x, y) = v(1/y)y^{r-1}$.*

The sequence $(u_k(x))_{k \geq 0}$ in Lemma 4.1 is C -finite because its generating function is rational. Hence, using **SEQTERM**, the $r = O(1)$ many terms $u_{N-r+1}(x), \dots, u_N(x)$ can be computed in complexity $O(N)$. It follows that the case $Q(x, y) = y$ of **BIVMODPOW** can be solved in $O(N)$ steps as well.

Finally, the computation of $Q(x, y)^N \bmod P(x, y)$ can be reduced to $y^N \bmod P(x, y)$ with a resultant precomputation (see Lemma 4.2). This leads to Algorithm 4, which solves **BivModPow** in $O(N)$.

LEMMA 4.2. *Let $P(y), Q(y) \in \mathbb{L}[y]$. Define $A(t), B(t) \in \mathbb{L}[t]$ by $A(t) = \text{Res}_y(P(y), t - Q(y))$ and $B(t) = t^N \bmod A(t)$. Then*

$$Q(y)^N \bmod P(y) = B(Q(y)) \bmod P(y).$$

PROOF. By the definition of the resultant, $A(t) = \prod_i (t - Q(a_i))$ where $a_i \in \bar{\mathbb{L}}$ are the solutions of $P(y) = 0$. Hence, $P(y)$ divides $A(Q(y))$, which, by construction, divides $B(Q(y)) - Q(y)^N$. $\square$

4.2 Computing polynomial matrix powers

Let $M \in \mathbb{K}[x]^{r \times r}$ be an $r \times r$ polynomial matrix of degree at most d . Its power M^N has degree at most $Nd \in O(N)$. Let $P(x, y)$ be the characteristic polynomial of M . Since $P(x, M) = 0$ by the Cayley-Hamilton theorem, we get $M^N = R(x, M)$ where $R(x, y) = y^N \bmod P(x, y)$. The polynomial R can be computed in $O(N)$ via **BIVMODPOW**. Then evaluating $R(x, y)$ at $y = M(x)$ has cost $O(N)$ since $\deg_x(R) \in O(N)$, $\deg_y(R) < r$ with $r \in O(1)$ and $\deg_x(M) = d \in O(1)$. Hence Algorithm 5 is correct and has complexity $O(N)$.

Algorithm 5 **PolMatPow**(M, N)

Input: matrix $M(x) \in \mathbb{K}[x]^{r \times r}$, integer $N \in \mathbb{N}$.

Output: $M(x)^N \in \mathbb{K}[x]^{r \times r}$.

- 1: $P(x, y) \leftarrow$ the characteristic polynomial of $M(x)$
- 2: $R(x, y) \leftarrow y^N \bmod P(x, y)$ ► instance of **BivModPow**
- 3: **return** $R(x, M(x))$

5 EXPERIMENTS

The main precomputation step for all our algorithms consists in starting with a rational function $U \in \mathbb{K}(x, y) \cap \mathbb{K}[[x, y]]$ and in finding a differential operator L_n that annihilates $u_n(x) = [y^n]U(x, y)$ and whose degree and order are independent of n . For this task, in practice, we may either use the method described in §3.1, or creative telescoping algorithms for hyperexponential functions. Table 1 summarizes timings for a variety of implementations.

The table reveals that, among these implementations, the fastest one for computing a telescoper of $U(x, y)/y^{n+1}$ is the reduction-based creative telescoping in Maple. More specifically, `redct` is the fastest, followed by HT. The implementation in `ore_algebra` [25] in SageMath competes best with reduction-based methods.

Table 2 gives timings of an efficient implementation of the remaining stages after precomputations: computing initial terms (IT), and unrolling (UR). We observe that IT takes negligible time compared to UR, except for extreme parameter ranges where, simultaneously, r and d are large and N is small; this is expected since these ranges correspond to cases where the order of the recurrence to be unrolled is close to N . We also see that binary powering is always slower, often by a factor more than 5, than the addition of IT and UR. The speed-up factor is summarized in Figs. 1 to 3; as expected it grows when N grows, with r and d fixed.

For large N , in most of the reported cases, performing both the precomputation and IT+UR is much faster than using binary powering. Still, this is not always true, e.g. for $r = 5$. One has to keep in mind that `redct` is not implemented in low-level Maple, and targets rational coefficients: for a more meaningful assessment of the precomputation part, it would be interesting to have an implementation of creative telescoping which is fully optimized and specialized to coefficients in a word-size prime field.

6 PERSPECTIVES

We have shown that it is possible to beat, both in theory and in practice, the basic and powerful binary powering method for computing: (i) powers of polynomial matrices, (ii) terms in polynomial C-finite sequences and (iii) modular exponentiation for bivariate polynomials. We describe below several lines of work, including possible optimizations and generalizations, left for future investigations.

More detailed complexity analysis. The most natural direction for future work is to analyze and improve the complexity of the algorithms in Theorem 1.1 with respect to the parameters r and d . For simplicity, these parameters were assumed to be $O(1)$ in this work. For the N th power of an $r \times r$ matrix $M(x)$ of degree d , binary powering has complexity $O(M(Nd)r^2 + Ndr^\omega)$, where $\omega \in [2, 3]$ is a feasible exponent of matrix multiplication over $\mathbb{K}$. With our approach, it is legitimate to target a differential equation satisfied by the entries of $M(x)^N$ of order r with coefficients in x of degree $O(dr^3)$, yielding a recurrence of order $O(dr^3)$ and coefficients in n of degree at most r . For large N , this would result in a complexity in $O(Ndr^2M(r))$. Using different LDEs, of order $O(r)$ and coefficients of degree $O(dr^2)$ could even lead to $O(NdrM(r))$.

The K th coefficient of the N th term. For some (large) integers $N, K \in \mathbb{N}$, one might be interested in computing the single coefficient $[x^K y^N]U(x, y)$ of a rational function $U \in \mathbb{K}(x, y) \cap \mathbb{K}[[x, y]]$. Equivalently it is natural to wonder: how fast can one compute the

K th coefficient of the N th term of a C-finite sequence $(u_n(x))_{n \geq 0}$? Using our method, a recurrence with initial conditions for the coefficients of $u_N(x)$ can be deduced in $O(\log(N))$ operations. Then (assuming that the recurrence is nonsingular) the K th coefficient can be found in $O(M(\sqrt{K}))$ operations by using baby-steps/giant-steps techniques [7, 12]. We expect that, at least under a genericity assumption, this problem can be solved in complexity $O(\log(N) + M(\sqrt{K}))$ which is a big improvement compared to the previous best $O(N + K)$ by Massazza and Radicioni [30].

Polynomial P-finite sequences. A somewhat related task is to study the analogous problem to `SEQTERM` for polynomial P-finite sequences, that is for $(u_n(x))_n \in \mathbb{K}[x]^{\mathbb{N}}$ satisfying

$$p_r(x, n)u_{n+r}(x) + \dots + p_0(x, n)u_n(x) = 0,$$

for $p_i(x, n) \in \mathbb{K}[x, n]$. We expect that, at least under a genericity assumption, a generalization of Lemma 3.4 (based on results in [4, 35]) should exist, implying in particular that $u_N(x)$ satisfies an LDE of order and degree independent of N . Generalizing this even further, one might study the creative telescoping problem for rational functions of the form $H(\mathbf{x}) = \frac{P(x_1, \dots, x_s)}{Q(x_1, \dots, x_s)R(x_1, \dots, x_s)^n}$. We expect that (at least generically) the minimal telescoper for $H(\mathbf{x})$ has order and degree independent of n and can be found via a Griffiths-Dwork reduction type approach, based on ideas from [8].

Connection to the Jordan–Chevalley decomposition. A different approach for computing powers of matrices uses the *Jordan–Chevalley decomposition* (also called *SN decomposition*), see e.g. [15, 17, 23, 32]. It ensures that any polynomial matrix $M \in \mathbb{K}[x]^{r \times r}$ can be written as $M = S + Z$, where $S \in \mathbb{K}(x)^{r \times r}$ is diagonalizable over $\overline{\mathbb{K}(x)}$, $Z \in \mathbb{K}(x)^{r \times r}$ is nilpotent, and $SZ = ZS$. From this decomposition it follows that $M^N = \sum_{i=0}^{r-1} \binom{N}{i} S^{N-i} Z^i$. After a change of basis, this reduces to computing a power of a diagonal matrix with algebraic functions coefficients. Using Lemma 3.2 this can be performed efficiently in $O(N)$ operations. It would be certainly interesting to compare this approach with the other methods.

A PDE approach for `SEQTERM`. There is yet another method to deduce recurrence (9). The starting point is that the generating function $F(x, y) = y/(1 - xy - y^2)$ of $F_n(x)$ satisfies the linear PDE

$$(x^2 + 4) \frac{\partial^2 F}{\partial x^2} + 3x \frac{\partial F}{\partial x} - y^2 \frac{\partial^2 F}{\partial y^2} - y \frac{\partial F}{\partial y} + F = 0, \quad (22)$$

and extracting the coefficient of $x^k y^n$ in (22) immediately gives (9). More generally, such a PDE translates into a recurrence if it is linear with polynomial coefficients in x and y and if additionally all terms of the form $x^i y^\ell \frac{\partial^k F}{\partial x^k} \frac{\partial^\ell F}{\partial y^\ell}$ have $\ell = j$. A dimension counting argument in the spirit of [29, Lem. 3] proves that such a PDE exists for *any* rational function $F(x, y)$. The existence proof is effective and amounts to linear algebra. A natural question is whether it is possible to compute such a PDE via creative telescoping (either Almkvist-Zeilberger [1] or reduction-based [2]), and how the corresponding method compares to the aforementioned ones.

Integer case in bit complexity $O(N)$. Recall the analogy between the bit complexity for finding the N th term of a C-finite sequence over $\mathbb{Z}$ and the complexity for finding the N th term of a C-finite sequence over $\mathbb{K}[x]$. Our work achieves $O(N)$ for the latter, so it is now natural to target $O(N)$ for the former, for instance for the N th Fibonacci number or simply 3^N . This remains widely open.

Table 1: Timings in seconds for creative telescoping to find a telescoper L_n of $P(x, y)/(y^{n+1}Q(x, y))$. Here $P(x, y)/Q(x, y)$ is the generating function for the sequence of the top-right entry of the powers of a randomly chosen matrix in $\mathbb{F}_p[x]^{r \times r}$ of degree d , for a 50-bit prime p , with $Q(x, y)$ the y -reversal of the characteristic polynomial of this matrix. The order of L_n is ℓ , its degree in n is d_n , and $d_x = \deg_x(L_n)$. A blank space means that the computation took more than 1000 seconds. We observe empirically that the degree in x is $dr(r+1)(2r-1)/2 - r(r-1)$ while its degree in n is $(r-1)(r+2)/2$; this is expected asymptotically by [2, Thm. 25] and Lemma 3.4, because $\deg_y Q(x, y) = r$ and $\deg_x Q(x, y) = dr$. The tested implementations are

- in Maple: redct [4]; HermiteTelescoping (HT) [8]; Zeilberger (ZB) in DEtools; creative_telescoping (c_t) [13];
- in SageMath: creative telescoping (ct) from the *ore_algebra* package [25];
- In Mathematica: FindCreativeTelescoping (FCT), CreativeTelescoping (CT), and HermiteTelescoping (HCT), see [28].

r	d	Maple				Sage ct	Mathematica			ℓ	d_n	d_x
		redct	HT	ZB	c_t		FCT	CT	HCT			
2	2	0.0	0.1	0.0	0.1	0.5	0.2	0.2	0.2	2	2	16
	4	0.0	0.0	0.0	0.1	0.6	0.4	0.4	0.3	2	2	34
	6	0.0	0.0	0.0	0.1	0.6	0.7	0.5	0.5	2	2	52
	8	0.0	0.0	0.0	0.1	0.8	1.0	0.7	0.7	2	2	70
3	1	0.0	0.2	0.0	0.5	2.0	2.0	1.3	1.3	3	5	24
	2	0.0	0.1	0.8	3.4	3.1	4.0	2.6	2.5	3	5	54
	3	0.1	0.2	0.8	9.3	5.6	10	5.7	5.4	3	5	84
	4	0.1	0.5	18	19	8.2	17	9.4	8.9	3	5	114
	5	0.2	1.1	5.1	32	12	25	14	14	3	5	144
	6	0.5	1.7	9.8	49	17	35	19	20	3	5	174
4	1	0.4	2.9	23	117	20	31	25	25	4	9	58
	2	1.7	17	410	749	45	101	96	95	4	9	128
	3	4.4	43			89	295	376	373	4	9	198
	4	12	82			172	388	752	693	4	9	268
	5	18	128			280	635			4	9	338
5	1	11	34	538		163	847	780		5	14	115
	2	64	183			515				5	14	250
	3	159	526							5	14	385
	4	345								5	14	520

Table 2: Timings in seconds, using the C++ library NTL [33] and PML [24], for computing the top-right entry of the N th power of a randomly chosen matrix in $\mathbb{F}_p[x]^{r \times r}$ of degree d , for a 50-bit prime p . The first tested method is to directly apply binary powering (BP); in the present context, the polynomial matrix multiplication of PML is based on evaluation-interpolation and 3-prime FFT. The second tested method uses Algorithm 3 and we do not count “precomputations” (already showed in Table 1), i.e. we only report timings for the two non-negligible steps that depend on N , namely Line 9 (UR, unrolling) and Line 8 (IT, initial terms) from Algorithm 1.

r	d	$N = 2^{10}$			$N = 2^{12}$			$N = 2^{14}$			$N = 2^{16}$			$N = 2^{18}$			$N = 2^{20}$			$N = 2^{22}$		
		BP	UR	IT	BP	UR	IT	BP	UR	IT	BP	UR	IT	BP	UR	IT	BP	UR	IT	BP	UR	IT
2	2	1.2e-3	5.7e-4	3.7e-5	5.3e-3	2.4e-3	4.3e-5	2.5e-2	9.7e-3	4.9e-5	1.1e-1	3.9e-2	5.5e-5	5.3e-1	1.5e-1	6.2e-5	3.3e+0	6.2e-1	6.7e-5	1.5e+1	2.5e+0	7.5e-5
	4	2.6e-3	1.3e-3	7.8e-5	1.2e-2	5.2e-3	9.4e-5	5.2e-2	2.1e-2	1.1e-4	2.4e-1	8.4e-2	1.3e-4	1.4e+0	3.4e-1	1.4e-4	7.2e+0	1.4e+0	1.6e-4	3.1e+1	5.4e+0	1.8e-4
	6	3.8e-3	2.1e-3	1.2e-4	1.7e-2	8.7e-3	1.5e-4	7.9e-2	3.5e-2	1.8e-4	3.6e-1	1.4e-1	2.1e-4	2.3e+0	5.5e-1	2.4e-4	1.0e+1	2.2e+0	2.7e-4	4.6e+1	8.9e+0	3.0e-4
	8	5.3e-3	3.1e-3	1.9e-4	2.4e-2	1.2e-2	2.4e-4	1.1e-1	5.0e-2	2.8e-4	5.3e-1	2.0e-1	3.3e-4	3.3e+0	8.0e-1	3.8e-4	1.5e+1	3.2e+0	4.3e-4	7.0e+1	1.2e+1	4.9e-4
3	1	1.4e-3	3.0e-4	1.3e-4	6.0e-3	1.3e-3	1.7e-4	2.6e-2	5.5e-3	2.1e-4	1.2e-1	2.2e-2	2.4e-4	5.8e-1	8.8e-2	2.8e-4	3.4e+0	3.5e-1	3.1e-4	1.6e+1	1.4e+0	3.5e-4
	2	2.9e-3	7.8e-4	4.0e-4	1.2e-2	3.2e-3	5.3e-4	5.6e-2	1.3e-2	6.5e-4	2.6e-1	5.2e-2	7.8e-4	1.5e+0	2.1e-1	9.1e-4	7.6e+0	8.4e-1	1.0e-3	3.4e+1	3.3e+0	1.2e-3
	3	4.3e-3	1.4e-3	7.4e-4	1.9e-2	5.8e-3	9.9e-4	8.4e-2	2.3e-2	1.2e-3	3.9e-1	9.3e-2	1.5e-3	2.2e+0	3.7e-1	1.7e-3	1.1e+1	1.5e+0	2.0e-3	4.9e+1	6.0e+0	2.2e-3
	4	6.0e-3	2.1e-3	8.0e-4	2.6e-2	8.8e-3	1.0e-3	1.2e-1	3.5e-2	1.3e-3	5.8e-1	1.4e-1	1.5e-3	3.5e+0	5.7e-1	1.8e-3	1.7e+1	2.3e+0	2.0e-3	7.1e+1	9.1e+0	2.3e-3
	5	7.4e-3	3.0e-3	1.0e-3	3.3e-2	1.2e-2	1.3e-3	1.5e-1	5.0e-2	1.7e-3	7.2e-1	2.0e-1	2.0e-3	4.3e+0	7.9e-1	2.3e-3	2.0e+1	3.2e+0	2.6e-3	8.8e+1	1.3e+1	2.9e-3
	6	9.1e-3	4.0e-3	1.2e-3	4.0e-2	1.6e-2	1.6e-3	1.8e-1	6.6e-2	1.9e-3	8.2e-1	2.7e-1	2.3e-3	5.3e+0	1.1e+0	2.7e-3	2.3e+1	4.2e+0	3.1e-3	1.1e+2	1.7e+1	3.4e-3
4	1	2.7e-3	4.2e-4	7.8e-4	1.1e-2	1.8e-3	1.1e-3	4.9e-2	7.5e-3	1.4e-3	2.2e-1	3.0e-2	1.7e-3	1.1e+0	1.2e-1	2.0e-3	6.2e+0	4.8e-1	2.3e-3	2.9e+1	1.9e+0	2.6e-3
	2	5.5e-3	1.2e-3	1.3e-3	2.4e-2	5.2e-3	1.8e-3	1.1e-1	2.1e-2	2.3e-3	4.9e-1	8.6e-2	2.8e-3	2.8e+0	3.4e-1	3.2e-3	1.4e+1	1.4e+0	3.7e-3	6.2e+1	5.5e+0	4.2e-3
	3	8.2e-3	2.4e-3	2.1e-3	3.6e-2	1.0e-2	2.9e-3	1.6e-1	4.2e-2	3.7e-3	7.3e-1	1.7e-1	4.5e-3	4.4e+0	6.7e-1	5.3e-3	2.1e+1	2.7e+0	6.1e-3	9.3e+1	1.1e+1	6.9e-3
	4	1.1e-2	4.1e-3	3.0e-3	5.0e-2	1.7e-2	4.1e-3	2.3e-1	6.9e-2	5.2e-3	1.1e+0	2.8e-1	6.4e-3	6.6e+0	1.1e+0	7.5e-3	3.1e+1	4.5e+0	8.6e-3	1.3e+2	1.8e+1	9.7e-3
	5	1.4e-2	6.0e-3	3.7e-3	6.3e-2	2.5e-2	5.2e-3	2.8e-1	1.0e-1	6.6e-3	1.3e+0	4.1e-1	8.0e-3	7.7e+0	1.6e+0	9.4e-3	3.6e+1	6.5e+0	1.1e-2	1.6e+2	2.6e+1	1.2e-2
5	1	4.4e-3	6.0e-4	1.8e-3	1.8e-2	2.7e-3	2.5e-3	8.2e-2	1.1e-2	3.3e-3	3.7e-1	4.5e-2	4.1e-3	1.7e+0	1.8e-1	4.9e-3	1.0e+1	7.3e-1	5.7e-3	4.7e+1	2.9e+0	6.5e-3
	2	9.1e-3	2.0e-3	3.5e-3	3.9e-2	9.1e-3	5.1e-3	1.8e-1	3.7e-2	6.7e-3	8.1e-1	1.5e-1	8.3e-3	4.6e+0	6.0e-1	9.9e-3	2.3e+1	2.4e+0	1.2e-2	1.0e+2	9.6e+0	1.3e-2
	3	1.3e-2	4.3e-3	5.7e-3	5.8e-2	1.9e-2	8.3e-3	2.6e-1	7.8e-2	1.1e-2	1.2e+0	3.2e-1	1.4e-2	7.1e+0	1.3e+0	1.6e-2	3.4e+1	5.1e+0	1.9e-2	1.5e+2	2.0e+1	2.2e-2
	4	1.8e-2	7.4e-3	7.8e-3	8.0e-2	3.3e-2	1.2e-2	3.8e-1	1.3e-1	1.5e-2	1.8e+0	5.4e-1	1.9e-2	1.1e+1	2.2e+0	2.3e-2	4.9e+1	8.7e+0	2.6e-2	2.1e+2	3.5e+1	3.0e-2

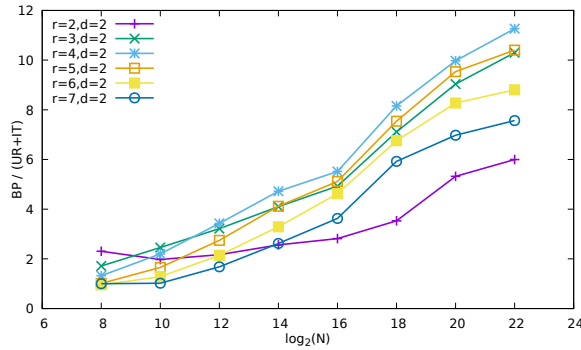


Figure 1: Speed-up versus binary powering, not counting precomputations, for $r = 2 \dots 7$, $N = 2^8, 2^{10}, \dots, 2^{22}$, and fixed $d = 2$.

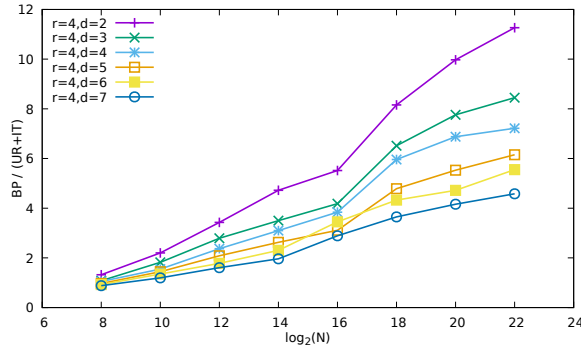


Figure 2: Speed-up versus binary powering, not counting precomputations, for $d = 2 \dots 7$, $N = 2^8, 2^{10}, \dots, 2^{22}$, and fixed $r = 4$.

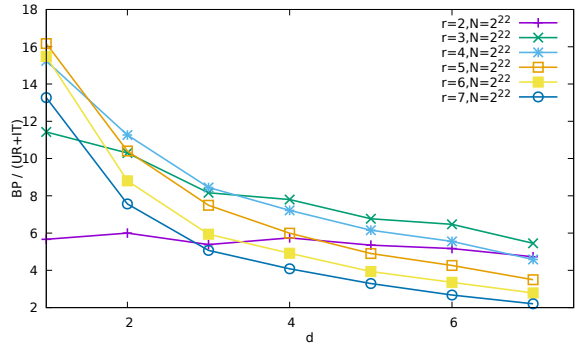


Figure 3: Speed-up versus binary powering, not counting precomputations, for $r = 2 \dots 7$, $d = 1 \dots 7$, and fixed $N = 2^{22}$.

REFERENCES

- [1] Gert Almkvist and Doron Zeilberger. 1990. The method of differentiating under the integral sign. *J. Symbolic Comput.* 10, 6 (1990), 571–591. [https://doi.org/10.1016/S0747-7171\(08\)80159-9](https://doi.org/10.1016/S0747-7171(08)80159-9)
- [2] Alin Bostan, Shaoshi Chen, Frédéric Chyzak, and Ziming Li. 2010. Complexity of creative telescoping for bivariate rational functions. In *ISSAC'10*. ACM, 203–210. <https://doi.org/10.1145/1837934.1837975>
- [3] Alin Bostan, Shaoshi Chen, Frédéric Chyzak, Ziming Li, and Guoce Xin. 2013. Hermite reduction and creative telescoping for hyperexponential functions. In *ISSAC'13*. ACM, 77–84. <https://doi.org/10.1145/2465506.2465946>
- [4] Alin Bostan, Frédéric Chyzak, Pierre Lairez, and Bruno Salvy. 2018. Generalized Hermite reduction, creative telescoping and definite integration of D-finite functions. In *ISSAC'18*. ACM, 95–102. <https://doi.org/10.1145/3208976.3208992>
- [5] Alin Bostan, Frédéric Chyzak, Grégoire Lecerc, Bruno Salvy, and Éric Schost. 2007. Differential equations for algebraic functions. In *ISSAC'07*. ACM, 25–32. <https://doi.org/10.1145/1277548.1277553>
- [6] Alin Bostan, Frédéric Chyzak, Ziming Li, and Bruno Salvy. 2012. Fast computation of common left multiples of linear ordinary differential operators. In *ISSAC'12*. ACM, 99–106. <https://doi.org/10.1145/2442829.2442847>
- [7] Alin Bostan, Pierrick Gaudry, and Éric Schost. 2007. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM J. Comput.* 36, 6 (2007), 1777–1806. <https://doi.org/10.1137/S0097539704443793>
- [8] Alin Bostan, Pierre Lairez, and Bruno Salvy. 2013. Creative telescoping for rational functions using the Griffiths-Dwork method. In *ISSAC'13*. ACM, 93–100. <https://doi.org/10.1145/2465506.2465935>
- [9] Alin Bostan and Ryuhei Mori. 2021. A simple and fast algorithm for computing the N -th term of a linearly recurrent sequence. In *SOSA'21 (Symposium on Simplicity in Algorithms)*. SIAM, 118–132. <https://doi.org/10.1137/1.9781611976496.14>
- [10] Paul F. Byrd. 1963. Expansion of analytic functions in polynomials associated with Fibonacci numbers. *Fibonacci Quart.* 1, 1 (1963), 16–29.
- [11] David G. Cantor and Erich Kaltofen. 1991. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.* 28, 7 (1991), 693–701. <https://doi.org/10.1007/BF01178683>
- [12] D. V. Chudnovsky and G. V. Chudnovsky. 1988. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited (Urbana-Champaign, Ill., 1987)*. Academic Press, Boston, MA, 375–472.
- [13] Frédéric Chyzak. 2000. An extension of Zeilberger's fast algorithm to general holonomic functions. *Discrete Math.* 217, 1-3 (2000), 115–134. [https://doi.org/10.1016/S0012-365X\(99\)00259-9](https://doi.org/10.1016/S0012-365X(99)00259-9)
- [14] James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.* 19 (1965), 297–301. <https://doi.org/10.2307/2003354>
- [15] Danielle Couty, Jean Esterle, and Rachid Zarouf. 2011. Décomposition effective de Jordan-Chevalley. *Gaz. Math.* 129 (2011), 29–49.
- [16] Sándor Czirbusz. 2012. Comparing the computation of Chebyshev polynomials in computer algebra systems. *Ann. Univ. Sci. Budapest. Sect. Comput.* 36 (2012), 23–39.
- [17] Saber N. Elyadi and William A. Harris, Jr. 1998. On the computation of A^n . *SIAM Rev.* 40, 4 (1998), 965–971. <https://doi.org/10.1137/S0036144597319235>
- [18] Charles M. Fiduccia. 1985. An efficient formula for linear recurrences. *SIAM J. Comput.* 14, 1 (1985), 106–112. <https://doi.org/10.1137/0214007>
- [19] Philippe Flajolet and Bruno Salvy. 1997. The SIGSAM challenges: symbolic asymptotics in practice. *ACM SIGSAM Bull.* 31, 4 (1997), 36–47. <https://doi.org/10.1145/274888.274890>
- [20] Henry W. Gould. 1972. *Combinatorial identities*. viii+106 pages. A standardized set of tables listing 500 binomial coefficient summations.
- [21] David Harvey and Joris van der Hoeven. 2021. Integer multiplication in time $O(n \log n)$. *Ann. of Math. (2)* 193, 2 (2021), 563–617. <https://doi.org/10.4007/annals.2021.193.2.4>
- [22] David Harvey and Joris van der Hoeven. 2022. Polynomial multiplication over finite fields in time $O(n \log n)$. *J. ACM* 69, 2 (2022), Art. 12, 40. <https://doi.org/10.1145/3505584>
- [23] Po-Fang Hsieh, Mitsuhiro Kohno, and Yasutaka Sibuya. 1996. Construction of a fundamental matrix solution at a singular point of the first kind by means of the SN decomposition of matrices. *Linear Algebra Appl.* 239 (1996), 29–76. [https://doi.org/10.1016/S0024-3795\(96\)90003-8](https://doi.org/10.1016/S0024-3795(96)90003-8)
- [24] Seung Gyu Hyun, Vincent Neiger, and Éric Schost. 2019. Implementations of Efficient Univariate Polynomial Matrix Algorithms and Application to Bivariate Resultants. In *ISSAC'19*. ACM, 235–242. <https://doi.org/10.1145/3326229.3326272>
- [25] Manuel Kauers and Marc Mezzarobba. 2019. Multivariate Ore polynomials in SageMath. *ACM Commun. Comput. Algebra* 53, 2 (2019), 57–60. <https://doi.org/10.1145/3371991.3371998>
- [26] Manuel Kauers and Gleb Pogudin. 2017. Bounds for substituting algebraic functions into D-finite functions. In *ISSAC'17*. ACM, 245–252. <https://doi.org/10.1145/3087604.3087616>
- [27] Wolfram Koepf. 1999. Efficient computation of Chebyshev polynomials in computer algebra. In *Computer Algebra Systems: A Practical Guide*. Wiley, 79–99.
- [28] Christoph Koutschan. 2010. A fast approach to creative telescoping. *Math. Comput. Sci.* 4, 2-3 (2010), 259–266. <https://doi.org/10.1007/s11786-010-0055-0>
- [29] L. Lipshitz. 1988. The diagonal of a D-finite power series is D-finite. *J. Algebra* 113, 2 (1988), 373–378. [https://doi.org/10.1016/0021-8693\(88\)90166-4](https://doi.org/10.1016/0021-8693(88)90166-4)
- [30] P. Massazza and R. Radicioni. 2005. On computing the coefficients of bivariate holonomic formal series. *Theoret. Comput. Sci.* 346, 2-3 (2005), 418–438. <https://doi.org/10.1016/j.tcs.2005.08.011>
- [31] J. C. P. Miller and D. J. Spencer Brown. 1966. An algorithm for evaluation of remote terms in a linear recurrence sequence. *Comput. J.* 9 (1966), 188–190. <https://doi.org/10.1093/comjnl/9.2.188>
- [32] Dieter Schmidt. 2000. Construction of the Jordan decomposition by means of Newton's method. *Linear Algebra Appl.* 314, 1-3 (2000), 75–89. [https://doi.org/10.1016/S0024-3795\(00\)00111-7](https://doi.org/10.1016/S0024-3795(00)00111-7)
- [33] V. Shoup. 2021. NTL: A library for doing number theory, v11.5.1. <https://libntl.org>
- [34] R. P. Stanley. 1980. Differentially finite power series. *European J. Combin.* 1, 2 (1980), 175–188. [https://doi.org/10.1016/S0195-6698\(80\)80051-5](https://doi.org/10.1016/S0195-6698(80)80051-5)
- [35] Joris van der Hoeven. 2021. Constructing reductions for creative telescoping: the general differentially finite case. *Appl. Algebra Engrg. Comm. Comput.* 32, 5 (2021), 575–602. <https://doi.org/10.1007/s00200-020-00413-3>