Computing the N-th Term of a q-Holonomic Sequence

Alin Bostan

Computing the N-th Term of a q-Holonomic Sequence

Alin Bostan
Inria, France

ABSTRACT

In 1977, Strassen invented a famous baby-step / giant-step algorithm that computes the factorial $N!$ in arithmetic complexity quasi-linear in $\sqrt{N}$. In 1988, the Chudnovsky brothers generalized Strassen's algorithm to the computation of the N-th term of any holonomic sequence in the same arithmetic complexity. We design q-analogues of these algorithms. We first extend Strassen's algorithm to the computation of the q-factorial of N, then Chudnovskys' algorithm to the computation of the N-th term of any q-holonomic sequence. Both algorithms work in arithmetic complexity quasi-linear in $\sqrt{N}$. We describe various algorithmic consequences, including the acceleration of polynomial and rational solving of linear q-differential equations, and the fast evaluation of large classes of polynomials, including a family recently considered by Nogneng and Schost.

CCS CONCEPTS

• **Computing methodologies** → **Algebraic algorithms**.

KEYWORDS

Algorithms, complexity, q-factorial, q-holonomic sequences.

ACM Reference Format:
Alin Bostan. 2020. Computing the N-th Term of a q-Holonomic Sequence. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '20), July 20–23, 2020, Kalamata, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A classical question in algebraic complexity theory is: how fast can one evaluate a univariate polynomial at one point? The precise formulation of this question depends on the model of computation. We will mainly focus on the *arithmetic complexity* model, in which one counts base field operations at unit cost.

Horner's rule evaluates a polynomial P in $O(\deg(P))$ operations. Ostrowski [55] conjectured in 1954 that this is *optimal for generic polynomials* (*i.e.*, whose coefficients are algebraically independent). This optimality result was proved a few years later by Pan [57].

However, most polynomials that one might wish to evaluate have coefficients which are not algebraically independent. Paterson and Stockmeyer [58] showed that for any field $\mathbb{K}$, an arbitrary polynomial $P \in \mathbb{K}[x]$ of degree n can be evaluated using $O(\sqrt{n})$ *nonscalar* multiplications; however, their algorithm uses a linear amount of scalar multiplications, so it is not well adapted to the evaluation at points from the base field $\mathbb{K}$, since in this case the total arithmetic complexity remains linear in n.

On the other hand, for some families of polynomials, one can do much better. Typical examples are x^n and $P_n(x) := x^{n-1} + \cdots + x + 1$, which can be evaluated by repeated squaring in $O(\log n)$ operations. (Note that for $P_n(x)$ such a fast algorithm needs to perform division.) By contrast, a family $F_n(x)$ of univariate polynomials is called *hard to compute* if the complexity of the evaluation of F_n grows at least like a power in $\deg(F_n)$, whatever the algorithm used.

Paterson and Stockmeyer [58] proved the existence of polynomials in $\mathbb{K}[x]$ which are hard to compute. Specific families of hard-to-compute polynomials were first exhibited by Strassen [70]. The techniques were refined and improved by Borodin and Cook [13], Lipton [52] and Schnorr [66], who produced explicit examples of degree-n polynomials whose evaluation requires a number of operations linear in $\sqrt{n}$. Subsequently, various methods have been developed to produce similar results on *lower bounds*, *e.g.*, by Heintz and Sieveking [42] using algebraic geometry, and by Aldaz et al. [4] using a combinatorial approach. The topic is vast and very well summarized in the book by Bürgisser, Clausen and Shokrollahi [23].

In this article, we focus on *upper bounds*, that is on the design of fast algorithms for special families of polynomials, which are hard to compute, but easier to evaluate than generic polynomials. For instance, for the degree-$\binom{n}{2}$ polynomial $Q_n(x) := P_1(x) \cdots P_n(x)$, a complexity in $O(n)$ is clearly achievable. We will see in §2.1 that one can do better, and attain a cost which is almost linear in $\sqrt{n}$ (up to logarithmic factors in n). Another example is $R_n(x) := \sum_{k=0}^{n} x^{k^2}$, of degree n^2, and whose evaluation can also be performed in complexity quasi-linear in $\sqrt{n}$, as shown recently by Nogneng and Schost [54] (see §2.2). In both cases, these complexities are obtained by clever although somehow ad-hoc algorithms. The starting point of our work was the question whether these algorithms for $Q_n(x)$ and $R_n(x)$ could be treated in a unified way, which would allow to evaluate other families of polynomials in a similar complexity.

The answer to this question turns out to be positive. The key idea, very simple and natural, is to view both examples as particular cases of the following general question: given a q-holonomic sequence, that is, a sequence satisfying a linear recurrence with polynomial coefficients in q and q^n, how fast can one compute its N-th term?

In the more classical case of holonomic sequences (satisfying linear recurrences with polynomial coefficients in the index n), fast algorithms exist for the computation of the N-th term. They rely on a basic block, which is the computation of the factorial term $N!$ in arithmetic complexity quasi-linear in $\sqrt{N}$, using an algorithm due to Strassen [71]. The Chudnovsky brothers extended in [26] Strassen's algorithm to the computation of the N-th term of any holonomic sequence in arithmetic complexity quasi-linear in $\sqrt{N}$.

Our main contribution consists in transferring these results to the q-holonomic framework. It turns out that the resulting algorithms are actually simpler in the q-holonomic case than in the usual holonomic setting, essentially because multipoint evaluation on

Preprint. To appear in ISSAC '20, July 20–23, 2020, Kalamata, Greece

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

arithmetic progressions used as a subroutine in Strassen's and Chudnovskys' algorithms is replaced by multipoint evaluation on geometric progressions, which is considerably simpler [21].

A consequence of our results is that the following apparently unrelated polynomials / rational functions can be evaluated fast (note the change in notation, with the variable x denoted now by q):

- $A_n(q)$, the generating function of the number of partitions into n positive integers each occurring *at most twice* [75], *i.e.*, the coefficient of t^n in the product $\prod_{k\geq 1}(1+q^k t+q^{2k}t^2)$.
- $B_n(q) := \prod_{i=1}^{\infty}(1-q^i) \bmod q^n$; by Euler's pentagonal theorem [56, §5], $B_n(q) = 1 + \sum_{i(3i+1)<2n} (-1)^i \left(q^{\frac{i(3\,i-1)}{2}} + q^{\frac{i(3\,i+1)}{2}}\right)$.
- The number $C_n(q)$ of $2n \times 2n$ upper-triangular matrices over $\mathbb{F}_q$ (the finite field with q elements), whose square is the zero matrix; by [47], $C_n(q)$ is equal to

$$C_n(q) = \sum_j \left[\binom{2n}{n-3j} - \binom{2n}{n-3j-1}\right] \cdot q^{n^2-3j^2-j}.$$

The common feature, exploited by the new algorithm, is that the sequences $(A_n(q))_{n\geq 0}, (B_n(q))_{n\geq 0}, (C_n(q))_{n\geq 0}$ are all q-holonomic. Actually, q-holonomic sequences are ubiquitous, so the range of application of our results is quite broad. This stems from the fact that they are coefficient sequences of power series satisfying q-differential equations, or equivalently, q-shift (or, q-difference) equations. From that perspective, our topic becomes intimately connected with q-calculus. The roots of q-calculus are in works of famous mathematicians such as Rothe, Gauss and Heine. The topic gained renewed interest in the first half of the 20th century, with the work, both on the formal and analytic aspects, of Tanner, Jackson, Carmichael, Mason, Adams, Trjitzinsky, Le Caine and Hahn, to name just a few. Modern accounts of the various aspects of the theory (including historical ones) can be found in [30, 32, 48].

One of the reasons for interest in q-difference equations is that, formally, as q tends to 1, the q-derivative $\frac{f(qx)-f(x)}{(q-1)x}$ tends to $f'(x)$, thus to every differential equation corresponds a q-differential equation which goes formally to the differential equation as $q \to 1$. In nice cases, (some of) the solutions of the q-differential equation go to solutions of the associated differential equation as $q \to 1$. An early example of such a good deformation behavior is given by the basic hypergeometric equation of Heine [48, §1.10].

In computer algebra, q-holonomic sequences were considered starting from the early nineties, in the context of computer-generated proofs of identities in the seminal paper by Wilf and Zeilberger [74], notably in Section 5 ("Generalization to q-sums and q-multisums") and in Section 6.4 ("q-sums and integrals"). Creative telescoping algorithms for (proper) q-hypergeometric sequences are discussed in various references [12, 25, 61]; several implementations of those algorithms are described for instance in [45, 60, 64, 69]. Algorithms for computing polynomial, rational and q-hypergeometric solutions of q-differential equations were designed by Abramov and collaborators [1–3, 46]. These algorithms are important for several reasons. One is that they lie at the heart of the vast generalization by Chyzak [27, 28] of the Wilf and Zeilberger algorithmic theory, for the treatment of general q-holonomic (not only q-hypergeometric) symbolic summation and integration via creative telescoping. In that context, a multivariate notion of q-holonomy is needed; the foundations of the theory were laid by Zeilberger [77] and Sabbah [65] (in the language of D-modules), see also [25, § 2.5] and [37].

The simplest non-trivial holonomic sequence is $n!$, which combinatorially counts the number of permutations of n objects. If instead of direct counting, one assigns to every permutation π its number of inversions $\mathrm{inv}(\pi)$, *i.e.*, the number of pairs $1 \leq i < j \leq n$ with $\pi(i) > \pi(j)$, the refined count (by size and number of inversions) is $[n]_q! := (1+q)(1+q+q^2)\cdots(1+q+\cdots+q^{n-1})$. This is the q-analogue of $n!$, the simplest non-trivial q-holonomic sequence.

There is also a natural q-analog of the binomial coefficients, called the *Gaussian coefficients*, defined by $\binom{n}{k}_q := \frac{[n]_q!}{[k]_q![n-k]_q!}$. They have many counting interpretations, *e.g.,* they count the k-dimensional subspaces of $\mathbb{F}_q^n$ (points on Grassmannians over $\mathbb{F}_q$). There are q-analogs to (almost) everything. To select just two basic examples, the q-analog [5, Thm. 3.3] of the binomial theorem is

$$\prod_{k=1}^{n}(1+q^{k-1}x) = \sum_{k=0}^{n}\binom{n}{k}_q q^{\binom{k}{2}}x^k \tag{1}$$

and the q-version [5, Thm. 3.4] of the Chu-Vandermonde identity is

$$\sum_{k=0}^{n} q^{k^2}\binom{m}{k}_q\binom{n}{k}_q = \binom{m+n}{n}_q. \tag{2}$$

The ubiquity of q-holonomic sequences is manifest in plenty of fields: partition theory [5, 56] and other subfields of combinatorics [33, 47]; theta functions and modular forms [51, 59, 76]; special functions [48] and in particular orthogonal polynomials [49]; algebraic geometry [31], representation theory [44]; knot theory [35–37]; Galois theory [43]; number theory [29].

The main message of this article is that for any example of q-holonomic sequence occurring in those various fields, *one can compute selected coefficients faster than by a direct algorithm.*

Complexity basics. We estimate the complexities of algorithms by counting arithmetic operations $(+,-,\times,\div)$ in the base field $\mathbb{K}$ at unit cost. We use standard complexity notation, such as $\mathbf{M}(d)$ for the cost of degree-d multiplication in $\mathbb{K}[x]$ and θ for feasible exponents of matrix multiplication. The best known upper bound is $\theta < 2.3729$ [34]. Most arithmetic operations on univariate polynomials of degree d in $\mathbb{K}[x]$ can be performed in quasi-linear complexity $\tilde{O}(d)$: multiplication, shift, interpolation, gcd, resultant, *etc*[*]. A key feature of these results is the reduction to fast polynomial multiplication, which can be performed in time $\mathbf{M}(d) = O(d \log d \log\log d)$ [24, 68]. An excellent general reference for these questions is the book by von zur Gathen and Gerhard [38].

2 TWO MOTIVATING EXAMPLES

Before presenting our main results in Section 3, we describe in this section the approach and main ideas on two basic examples. Both examples concern the fast evaluation of special families of univariate polynomials. In §2.1, we consider polynomials of the form $\prod_\ell (x - q^\ell)$, and in §2.2 sparse polynomials of the form $\sum_\ell p^\ell x^{a\ell^2+b\ell}$. In both cases, we first present fast ad-hoc algorithms, then introduce equally fast alternative algorithms, which have the nice feature that they will be generalizable to a broader setting.

[*] As usual, the notation $\tilde{O}(\cdot)$ is used to hide polylogarithmic factors in the argument.

2.1 De Feo's question

Here is our first example, emerging from a question asked to the author by Luca De Feo*; this was the starting point of the article.

Let q be an element of the field $\mathbb{K}$, and consider the polynomial

$$F(x) := \prod_{i=0}^{N-1} (x - q^i) \ \in \mathbb{K}[x]. \tag{3}$$

Given another element $\alpha \in \mathbb{K}$, how fast can one evaluate $F(\alpha)$?

If $q = 0$, then $F(\alpha) = \alpha^N$ can be computed in $O(\log N)$ operations in $\mathbb{K}$, by binary powering. We assume in what follows that q is nonzero. Obviously, a direct algorithm consists in computing the successive powers $q, q^2, \ldots, q^{N-1}$ using $O(N)$ operations in $\mathbb{K}$, then computing the elements $\alpha - q, \alpha - q^2, \ldots, \alpha - q^{N-1}$ in $O(N)$ more operations in $\mathbb{K}$, and finally returning their product. The total arithmetic cost of this algorithm is $O(N)$, linear in the degree of F.

Is it possible to do better? The answer is positive, as one can use the following *baby-step / giant-step* strategy, in which, in order to simplify things, we assume that N is a perfect square**, $N = s^2$:

Algorithm 1

(1) (Baby-step) Compute the values of $q, q^2, \ldots, q^{s-1}$, and deduce the coefficients of the polynomial $G(x) := \prod_{j=0}^{s-1}(x - q^j)$.
(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and deduce the coefficients of the polynomial $H(x) := \prod_{k=0}^{s-1}(\alpha - Q^k \cdot x)$.
(3) Return the resultant $\mathrm{Res}(G, H)$.

By the basic property of resultants, the output of this algorithm is

$$\mathrm{Res}(G,H) = \prod_{j=0}^{s-1} H(q^j) = \prod_{j=0}^{s-1}\prod_{k=0}^{s-1} \left(\alpha - q^{sk+j}\right) = \prod_{i=0}^{N-1} (\alpha - q^i) = F(\alpha).$$

Using the fast subproduct tree algorithm [38, Algorithm 10.3], one can perform the baby-step (1) as well as the giant-step (2) in $O(\mathbf{M}(\sqrt{N})\log N)$ operations in $\mathbb{K}$, and by [38, Corollary 11.19] the same cost can be achieved for the resultant computation in step (3). Using fast polynomial multiplication, we conclude that $F(\alpha)$ can be computed in arithmetic complexity quasi-linear in $\sqrt{N}$.

It is possible to speed up the previous algorithm by a logarithmic factor in N using a slightly different scheme, still based on a *baby-step / giant-step* strategy, but exploiting the fact that the roots of F are in geometric progression. Again, we assume that $N = s^2$ is a perfect square. This alternative algorithm goes as follows. Note that it is very close in spirit to Pollard's algorithm [62, p. 523].

Algorithm 2

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$, and deduce the coefficients of the polynomial $P(x) := \prod_{j=0}^{s-1}(\alpha - q^j \cdot x)$.
(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and evaluate P simultaneously at $1, Q, \ldots, Q^{s-1}$.
(3) Return the product $P(1)P(Q)\cdots P(Q^{s-1})$.

Obviously, the output of this algorithm is

$$\prod_{k=0}^{s-1} P(Q^k) = \prod_{k=0}^{s-1}\prod_{j=0}^{s-1}(\alpha - q^j \cdot q^{sk}) = \prod_{i=0}^{N-1}(\alpha - q^i) = F(\alpha).$$

*Private (email) communication, 10 January, 2020.

**If N is not a perfect square, then one can compute $F(\alpha)$ as $F(\alpha) = F_1(\alpha)F_2(\alpha)$, where $F_1(\alpha) := \prod_{i=0}^{\lfloor\sqrt{N}\rfloor^2-1}(\alpha - q^i)$ is computed as in Algorithm 1, while $F_2(\alpha) := \prod_{i=\lfloor\sqrt{N}\rfloor^2}^{N-1}(\alpha - q^i)$ can be computed naively, since $N - \lfloor\sqrt{N}\rfloor^2 = O(\sqrt{N})$.

As pointed out in the remarks after the proof of [21, Lemma 1], one can compute $P(x) = P_s(x) = \prod_{j=0}^{s-1}(\alpha - q^j \cdot x)$ in step (1) without computing the subproduct tree, by using a divide-and-conquer scheme which exploits the fact that $P_{2t}(x) = P_t(x) \cdot P_t(q^t x)$ and $P_{2t+1}(x) = P_t(x) \cdot P_t(q^t x) \cdot (\alpha - q^{2t}x)$. The cost of this algorithm is $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$. As for step (2), one can use the fast chirp transform algorithms of Rabiner, Schafer and Rader [63] and of Bluestein [11]. These algorithms rely on the following observation: writing $Q^{ij} = Q^{\binom{i+j}{2}} \cdot Q^{-\binom{i}{2}} \cdot Q^{-\binom{j}{2}}$ and $P(x) = \sum_{j=0}^{s} c_j x^j$ implies that the needed values $P(Q^i) = \sum_{j=0}^{s} c_j Q^{ij}, 0 \le i < s$, are

$$P(Q^i) = Q^{-\binom{i}{2}} \cdot \sum_{j=0}^{s} c_j Q^{-\binom{j}{2}} \cdot Q^{\binom{i+j}{2}}, \ 0 \le i < s,$$

in which the sum is simply the coefficient of x^{s+i} in the product

$$\left(\sum_{j=0}^{s} c_j Q^{-\binom{j}{2}} x^{s-j}\right)\left(\sum_{\ell=0}^{2s} Q^{\binom{\ell}{2}} x^{\ell}\right).$$

This polynomial product can be computed in $2\mathbf{M}(s)$ operations (and even in $\mathbf{M}(s) + O(s)$ using the transposition principle [20, 40], since only the median coefficients $x^s, \ldots, x^{2s-1}$ are actually needed). In conclusion, step (2) can also be performed in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$, and thus $O(\mathbf{M}(\sqrt{N}))$ is the total cost of this second algorithm.

We have chosen to detail this second algorithm for several reasons: not only because it is faster by a factor $\log(N)$ compared to the first one, but more importantly because it has a simpler structure, which will be generalizable to the general q-holonomic setting.

2.2 Evaluation of some sparse polynomials

Let us now consider the sequence of sparse polynomial sums

$$v_N^{(p,a,b)}(q) = \sum_{n=0}^{N-1} p^n q^{an^2+bn},$$

where $p \in \mathbb{K}$ and $a, b \in \mathbb{Q}$ such that $2a, a+b$ are both integers. Typical examples are (truncated) modular forms [59], which are ubiquitous in number theory [76] and combinatorics [5]. For instance, the *Jacobi theta function* ϑ_3 depends on two complex variables $z \in \mathbb{C}$, and $\tau \in \mathbb{C}$ with $\Im(\tau) > 0$, and it is defined by

$$\vartheta_3(z;\tau) = \sum_{n=-\infty}^{\infty} e^{\pi i(n^2\tau + 2nz)} = 1 + 2\sum_{n=1}^{\infty} \eta^n q^{n^2},$$

where $q = e^{\pi i \tau}$ is the nome ($|q| < 1$) and $\eta = e^{2\pi i z}$. Here, $\mathbb{K} = \mathbb{C}$. Another example is the *Dedekind eta function*, appearing in Euler's famous *pentagonal theorem* [56, §5], which has a similar form

$$q^{\frac{1}{24}} \cdot \left(1 + \sum_{n=1}^{\infty} (-1)^n \left(q^{\frac{n(3n-1)}{2}} + q^{\frac{n(3n+1)}{2}}\right)\right), \quad \text{with } q = e^{2\pi i \tau}.$$

Moreover, sums of the form $v_N^{(1,a,b)}(q) = \sum_{n=0}^{N-1} q^{an^2+bn}$, over $\mathbb{K} = \mathbb{Q}$ or $\mathbb{K} = \mathbb{F}_2$, crucially occur in a recent algorithm by Tao, Crott and Helfgott [72] for the efficient construction of prime numbers in given intervals, *e.g.*, in the context of effective versions of Bertrand's postulate. Actually, (the proof of) Lemma 3.1 in [72] contains the first sublinear complexity result for the evaluation of the sum $v_N^{(p,a,b)}(q)$ at an arbitrary point q; namely, the cost is $O(N^{\theta/3})$, where $\theta \in [2,3]$ is any feasible exponent for matrix multiplication.

Subsequently, Nogneng and Schost [54] designed a faster algorithm, and lowered the cost down to $\tilde{O}(\sqrt{N})$. Our algorithm is similar in spirit to theirs, as it also relies on a *baby-step / giant-step* strategy.

Let us first recall the principle of the Nogneng-Schost algorithm [54]. Assume as before that N is a perfect square, $N = s^2$. The starting point is the remark that

$$v_N^{(p,a,b)}(q) = \sum_{n=0}^{N-1} p^n q^{an^2+bn} = \sum_{k=0}^{s-1}\sum_{j=0}^{s-1} p^{j+sk} q^{a(j+sk)^2+b(j+sk)}$$

can be written

$$\sum_{k=0}^{s-1} p^{sk} q^{as^2k^2+bsk} \cdot P(q^{2ask}), \text{ where } P(y) := \sum_{j=0}^{s-1} p^j q^{aj^2+bj} y^j.$$

Therefore, the computation of $v_N^{(p,a,b)}(q)$ can be reduced essentially to the simultaneous evaluation of the polynomial P at $s = 1+\deg(P)$ points (in geometric progression), with arithmetic cost $O(\mathbf{M}(\sqrt{N}))$.

We now describe an alternative algorithm, of similar complexity $O(\mathbf{M}(\sqrt{N}))$, with a slightly larger constant in the big-Oh estimate, but whose advantage is its potential of generality.

Let us denote by $u_n(q)$ the summand $p^n q^{an^2+bn}$. Clearly, the sequence $(u_n(q))_{n\geq 0}$ satisfies the recurrence relation

$$u_{n+1}(q) = A(q, q^n) \cdot u_n(q), \quad \text{where} \quad A(x, y) := px^{a+b}y^{2a}.$$

As an immediate consequence, the sequence with general term $v_n(q) := \sum_{k=0}^{n-1} u_k(q)$ satisfies a similar recurrence relation

$$v_{n+2}(q) - v_{n+1}(q) = A(q, q^n) \cdot (v_{n+1}(q) - v_n(q)),$$

with initial conditions $v_0(q) = 0$ and $v_1(q) = 1$. This scalar recurrence of order two is equivalent to the first-order matrix recurrence

$$\begin{bmatrix} v_{n+2} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} A(q, q^n) + 1 & -A(q, q^n) \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} v_{n+1} \\ v_n \end{bmatrix}.$$

By unrolling this matrix recurrence, we deduce that

$$\begin{bmatrix} v_{n+1} \\ v_n \end{bmatrix} = M(q^{n-1}) \begin{bmatrix} v_n \\ v_{n-1} \end{bmatrix} = M(q^{n-1}) \cdots M(q)M(1) \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

where

$$M(x) := \begin{bmatrix} pq^{a+b}x^{2a} + 1 & -pq^{a+b}x^{2a} \\ 1 & 0 \end{bmatrix},$$

hence $v_N = \begin{bmatrix} 0 & 1 \end{bmatrix} \times M(q^{N-1}) \cdots M(q)M(1) \times \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Therefore, the computation of v_N reduces to the computation of the "matrix q-factorial" $M(q^{N-1}) \cdots M(q)M(1)$, which can be performed fast by using a *baby-step / giant-step* strategy similar to the one of the second algorithm in §2.1. Again, we assume for simplicity that $N = s^2$ is a perfect square. The algorithm goes as follows.

Algorithm 3 (matrix q-factorial)

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x) \cdots M(qx)M(x)$.
(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \ldots, Q^{s-1}$.
(3) Return the product $P(Q^{s-1}) \cdots P(Q)P(1)$.

By proceeding as in Algorithm 2 in §2.1, the complexity of Algorithm 3 already is quasi-linear in $\sqrt{N}$. However, its dependence in a, b is quite high (quasi-linear in a and b). If a and b are fixed and considered as $O(1)$ this dependence is invisible, but otherwise the following variant has the same complexity with respect to N, and a much better cost with respect to a and b. It is based on the simple observation that, if $\tilde{M}(x)$ denotes the polynomial matrix

$$\tilde{M}(x) := \begin{bmatrix} prx + 1 & -prx \\ 1 & 0 \end{bmatrix}, \text{ with } r := q^{a+b}, \tag{4}$$

and if $\tilde{q} := q^{2a}$, then the following matrix q-factorials coincide:

$$M(q^{N-1}) \cdots M(q)M(1) = \tilde{M}(\tilde{q}^{N-1}) \cdots \tilde{M}(\tilde{q})\tilde{M}(1).$$

Algorithm 4 (matrix q-factorial, variant)

(0) (Precomputation) Compute $r := q^{a+b}$, $\tilde{q} := q^{2a}$, and $\tilde{M}$ in (4).
(1) (Baby-step) Compute $\tilde{q}, \tilde{q}^2, \ldots, \tilde{q}^{s-1}$; deduce the coefficients of the polynomial matrix $\tilde{P}(x) := \tilde{M}(\tilde{q}^{s-1}x) \cdots \tilde{M}(\tilde{q}x)\tilde{M}(x)$.
(2) (Giant-step) Compute $\tilde{Q} := \tilde{q}^s, \tilde{Q}^2, \ldots, \tilde{Q}^{s-1}$, and evaluate (the entries of) $\tilde{P}(x)$ simultaneously at $1, \tilde{Q}, \ldots, \tilde{Q}^{s-1}$.
(3) Return the product $\tilde{P}(\tilde{Q}^{s-1}) \cdots \tilde{P}(\tilde{Q})\tilde{P}(1)$.

Using binary powering, the cost of the additional precomputation in step (0) is only logarithmic in a and b. In exchange, the new steps (2) and (3) are performed on matrices whose degrees do not depend on a and b anymore (in the previous, unoptimized, version the degrees of the polynomial matrices were linear in a and b). The total arithmetic cost with respect to N is still quasi-linear in $\sqrt{N}$.

3 MAIN RESULTS

In this section, we generalize the algorithms from §2, and show that they apply to the general setting of q-holonomic sequences.

3.1 Preliminaries

A sequence is q-holonomic if it satisfies a nontrivial q-recurrence, that is, a linear recurrence with coefficients polynomials in q and q^n.

Definition 3.1 (q-holonomic sequence). Let $\mathbb{K}$ be a field, and $q \in \mathbb{K}$. A sequence $(u_n(q))_{n\geq 0}$ in $\mathbb{K}^{\mathbb{N}}$ is called *q-holonomic* if there exist $r \in \mathbb{N}$ and polynomials $c_0, \ldots, c_r$ in $\mathbb{K}[x, y]$, with $c_r \neq 0$, such that

$$c_r(q, q^n)u_{n+r}(q) + \cdots + c_0(q, q^n)u_n(q) = 0, \quad \text{for all } n \geq 0. \tag{5}$$

The integer r is called the *order* of the q-recurrence (5). When $r = 1$, we say that $(u_n(q))_{n\geq 0}$ is q-hypergeometric.

The most basic examples are the *q-bracket* and the *q-factorial*,

$$[n]_q := 1 + q + \cdots + q^{n-1} \quad \text{and} \quad [n]_q! := \prod_{k=1}^{n} [k]_q. \tag{6}$$

They are clearly q-holonomic, and even q-hypergeometric.

The sequences $(u_n) = (q^n)$, $(v_n) = (q^{n^2})$ and $(w_n) = (q^{\binom{n}{2}})$ are also q-hypergeometric, since they satisfy the recurrence relations

$$u_{n+1} - qu_n = 0, \quad v_{n+1} - q^{2n+1}v_n = 0, \quad w_{n+1} - q^n w_n = 0.$$

However, the sequence (q^{n^3}) is not q-holonomic [37, Ex. 2.2(b)].

Another basic example is the q-Pochhammer symbol

$$(x; q)_n := \prod_{k=0}^{n-1} (1 - xq^k) \tag{7}$$

which is also q-hypergeometric, since $(x; q)_{n+1} - (1 - xq^n)(x; q)_n = 0$. In particular, the sequence $(q; q)_n := \prod_{k=1}^{n}(1 - q^k)$, also denoted $(q)_n$, is q-hypergeometric and satisfies $(q)_{n+1} - (1 - q^{n+1})(q)_n = 0$.

As mentioned in the introduction, q-holonomic sequences show up in various contexts. As an example, in (quantum) knot theory, the ("colored") Jones function of a (framed oriented) knot (in 3-space) is a powerful knot invariant, related to the Alexander polynomial [6]; it is a q-holonomic sequence of Laurent polynomials [36]. Its recurrence equations are themselves of interest, as they are closely related to the A-polynomial of a knot, via the *AJ conjecture*, verified in some cases using massive computer algebra calculations [35].

It is well known that the class of q-holonomic sequences is closed under several operations, such as addition, multiplication, Hadamard product and monomial substitution [37, 45]. All these closure properties are effective, *i.e.*, they can be executed algorithmically on the level of q-recurrences. Several computer algebra packages are available for the manipulation of q-holonomic sequences, *e.g.*, the Mathematica packages qGeneratingFunctions [45] and HolonomicFunctions [50], and the Maple packages qsum [12], qFPS [69], qseries and QDifferenceEquations.

A simple but useful fact is that the order-r scalar q-recurrence (5) can be translated into a first-order recurrence on $r \times 1$ vectors:

$$\begin{bmatrix} u_{n+r} \\ \vdots \\ u_{n+1} \end{bmatrix} = \begin{bmatrix} -\frac{c_{r-1}}{c_r} & \cdots & -\frac{c_1}{c_r} & -\frac{c_0}{c_r} \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \times \begin{bmatrix} u_{n+r-1} \\ \vdots \\ u_n \end{bmatrix}. \quad (8)$$

In particular, the N-th term of the q-holonomic sequence (u_n) is simply expressible in terms of the *matrix q-factorial*

$$M(q^{N-1}) \cdots M(q) M(1), \quad (9)$$

where $M(q^n)$ denotes the companion matrix from equation (8).

3.2 Computation of the q-factorial

We now give the promised q-analogue of Strassen's result on the computation of $N!$ in $O(\mathbf{M}(\sqrt{N}) \log N)$ arithmetic operations. Note that Strassen's case $q = 1$ is also covered by [19, §6], where the cost $O(\mathbf{M}(\sqrt{N}))$ is reached under some invertibility assumptions.

Theorem 3.2. *Let $\mathbb{K}$ be a field, let $q \in \mathbb{K} \setminus \{1\}$ and $N \in \mathbb{N}$. The q-factorial $[N]_q!$ can be computed using $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$. The same is true for the q-Pochhammer symbol $(\alpha; q)_N$ for any $\alpha \in \mathbb{K}$.*

Proof. If $\alpha = 0$, then $(\alpha; q)_N = 1$. If $q = 0$, then $[N]_q! = 1$ and $(\alpha; q)_N = 1 - \alpha$. We can assume that $q \in \mathbb{K} \setminus \{0, 1\}$ and $\alpha \in \mathbb{K} \setminus \{0\}$. We have $[N]_q! = r^N \cdot F(q^{-1})$ and $(\alpha; q)_N = \alpha^N \cdot F(\alpha^{-1})$, where $r := q/(1-q)$ and $F(x) := \prod_{i=0}^{N-1}(x - q^i)$. Algorithm 2 can be used to compute $F(q^{-1})$ and $F(\alpha^{-1})$ in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$. The cost of computing r^N and α^N is $O(\log N)$, and thus it is negligible. □

Corollary 3.3. *Under the assumptions of Theorem 3.2 and for any $n \in \mathbb{N}$, one can compute in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$:*

- *the q-binomial coefficient $\binom{N}{n}_q$;*
- *the coefficient of x^n in the polynomial $\prod_{k=1}^{N}(1 + q^{k-1}x)$;*
- *the sum $\binom{N-n}{0}_q \binom{n}{0}_q + q\binom{N-n}{1}_q \binom{n}{1}_q + \cdots + q^{n^2}\binom{N-n}{n}_q \binom{n}{n}_q$.*

Proof. The first assertion is a direct consequence of Theorem 3.2. The second assertion is a consequence of the first one, and of (1). The third assertion is a consequence of the first one, and of (2). □

3.3 N-th term of a q-holonomic sequence

We give the promised q-analogue of Chudnovskys' result on the computation of the N-th term of an arbitrary holonomic sequence in $O(\mathbf{M}(\sqrt{N}) \log N)$ arithmetic operations. Note that Chudnovskys' case $q = 1$ is also covered by [19, §6], where the improved cost $O(\mathbf{M}(\sqrt{N}))$ is reached under additional invertibility assumptions.

Theorem 3.4. *Let $\mathbb{K}$ be a field, $q \in \mathbb{K} \setminus \{1\}$ and $N \in \mathbb{N}$. Let $(u_n(q))_{n \geq 0}$ be a q-holonomic sequence satisfying recurrence (5), and assume that $c_r(q, q^k)$ is nonzero for $k = 0, \ldots, N-1$. Then, $u_N(q)$ can be computed in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$.*

Proof. Using equation (8), it is enough to show that the matrix q-factorial $M(q^{N-1}) \cdots M(q)M(1)$ can be computed in $O(\mathbf{M}(\sqrt{N}))$, where $M(q^n)$ denotes the companion matrix from equation (8). Algorithms 3 and 4 adapt *mutatis mutandis* to this effect. □

Corollary 3.5. *Let $\mathbb{K}$ be a field, $q \in \mathbb{K}$ not a root of unity, and $N \in \mathbb{N}$. Let $e_q(x)$ be the q-exponential series*

$$e_q(x) := \sum_{n \geq 0} \frac{x^n}{[n]_q!}$$

and let $E_q^{(N)}(x) := e_q(x) \bmod x^N$ be its truncation of degree $N-1$. If $\alpha \in \mathbb{K}$, one can compute $E_q^{(N)}(\alpha)$ in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$.

Proof. Denote the summand $\frac{\alpha^n}{[n]_q!}$ by $u_n(q)$. Then $(u_n(q))_n$ is q-hypergeometric, and satisfies the recurrence $[n+1]_q u_{n+1}(q) - \alpha u_n = 0$, therefore $v_N(q) := \sum_{i=0}^{N-1} u_i(q)$ satisfies the second-order recurrence $[n+1]_q(v_{n+2}(q) - v_{n+1}(q)) - \alpha(v_{n+1}(q) - v_n(q)) = 0$. Applying Theorem 3.4 to $v_N(q)$ concludes the proof. □

Remark that the same result holds if $e_q(x)$ is replaced by any power series satisfying a q-differential equation. For instance, one can evaluate fast all truncations of Heine's q-hypergeometric series

$${}_2\phi_1([a, b], [c]; q; x) := \sum_{n \geq 0} \frac{(a; q)_n (b; q)_n}{(c; q)_n} \cdot \frac{x^n}{(q)_n}.$$

Remark that Theorem 3.4 can be adapted to the computation of *several coefficients* of a q-holonomic sequence. We omit the proof, which is similar to that of Theorem 15 in [19].

Theorem 3.6. *Under the assumptions of Theorem 3.4, let $N_1 < N_2 < \cdots < N_s = N$ be positive integers, where $s < N^{\frac{1}{2}-\varepsilon}$ for some $0 < \varepsilon < \frac{1}{2}$. Then, the terms $u_{N_1}(q), \ldots, u_{N_s}(q)$ can be computed altogether in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$.*

3.4 The case q is an integer: bit complexity

Until now, we only considered the arithmetic complexity model. We briefly discuss here the case where q is an integer (or rational) number. The arithmetic complexity model needs to be replaced by the bit-complexity model, and the matrix q-factorials from §3.1 are computed by *binary splitting* rather than by baby-steps / giant-steps.

As an illustrative example, consider the computation of the term $u_N(q) = \sum_{n=0}^{N-1} q^{n^2}$, where q is assumed to be an integer of B bits. The integer $u_N(q)$ is bounded in absolute value by Nq^{N^2}, so its bitsize is of magnitude $N^2 B$. The "naive" algorithm consisting of computing the summands q^{n^2} one after the other, before summing

them, has bit complexity $\tilde{O}(N^3B)$. This is not (quasi-)optimal with respect to the output size. Can one do better? The answer is "yes". It is sufficient to use the q-holonomic character of $u_N(q)$, and to reduce its computation to that of a q-factorial matrix (9) as in §2.2. Now the point is that, instead of using baby-steps / giant-steps, it is a better idea to use binary splitting. The complexity of this approach becomes then quasi-optimal, that is $\tilde{O}(N^2B)$, which is quasi-linear in the bitsize of the output. The following general result can be proved along the same lines.

Theorem 3.7. *Under the assumptions of Theorem 3.4, with $\mathbb{K} = \mathbb{Q}$, the term $u_N(q)$ can be computed in $\tilde{O}(N^2B)$ bit operations, where B is the bitsize of q.*

As a corollary, (truncated) solutions of q-differential equations can be evaluated using the same (quasi-linear) bit complexity. This result should be viewed as the q-analogue of the classical fact that holonomic functions can be evaluated fast using binary splitting, a 1988 result by the Chudnovsky brothers [26, §6], anticipated a decade earlier (without proof) by Schroeppel and Salamin in Item 178 of [7]; see [8, §12] for a good survey on binary splitting.

4 APPLICATIONS

4.1 Combinatorial q-holonomic sequences

As already mentioned, many q-holonomic sequences arise in combinatorics, for example in connection with the enumeration of lattice polygons, where q-analogues of the Catalan numbers $\frac{1}{n+1}\binom{2n}{n}$ occur naturally [33, 39], or in the enumeration of special families of matrices with coefficients in the finite field $\mathbb{F}_q$ [47], where sequences related to the Gaussian coefficients $\binom{n}{k}_q$ also show up.

A huge subfield of combinatorics is the theory of partitions [5], where q-holonomic sequences occur as early as in the famous Roger-Ramanujan identities [5, Ch. 7], *e.g.*,

$$1 + \sum_{n\geq 1} \frac{q^{n^2}}{(1-q)\cdots(1-q^n)} = \prod_{n\geq 0} \frac{1}{(1-q^{5n+1})(1-q^{5n+4})}$$

which translates the fact that the number of partitions of n into parts that differ by at least 2 is equal to the number of partitions of n into parts congruent to 1 or 4 modulo 5. Andrews [5, Chapter 8] laid the foundations of a theory able to capture the q-holonomy of any generating function of a so-called *linked partition ideal*.

As a consequence, a virtually infinite number of special families of polynomials coming from partitions can be evaluated fast. For instance, the family of truncated polynomials

$$F_n(x) := \prod_{k=1}^{\infty} (1-x^k)^3 \bmod x^n,$$

can be evaluated fast due to our results and to the identity [56, §6]

$$F_N(q) = \sum_{\binom{n+1}{2}<N} (-1)^n (2n+1) q^{\binom{n+1}{2}}.$$

4.2 Evaluation of q-orthogonal polynomials

In the theory of special functions, *orthogonal polynomials* play a fundamental role. There exists an extension to the q-framework of the theory, see *e.g.*, Chapter 9 in Ernst's book [32]. Amongst the most basic examples, the *discrete q-Hermite polynomials* are defined by their q-exponential generating function

$$\sum_{n\geq 0} F_{n,q}(x) \frac{t^n}{[n]_q!} = \frac{e_q(xt)}{e_q(t)e_q(-t)},$$

and therefore they satisfy the second-order linear q-recurrence

$$F_{n+1,q}(x) = xF_{n,q}(x) - (1-q^n)q^{n-1}F_{n-1,q}(x), \quad n \geq 1,$$

with initial conditions $F_{0,q}(x) = 1, F_{1,q}(x) = x$. From there, it follows that for any $\alpha \in \mathbb{K}$, the sequence $(F_{n,q}(\alpha))_{n\geq 0}$ is q-holonomic, thus the evaluation of the n-th polynomial at $x = \alpha$ can be computed fast. The same is true for the *continuous q-Hermite polynomials*, for which $2\alpha H_{n,q}(\alpha) = H_{n+1,q}(\alpha) + (1-q^n)H_{n-1,q}(\alpha)$ for $n \geq 1$, and $H_{0,q}(\alpha) = 1, H_{1,q}(\alpha) = 2\alpha$. More generally, our results in §3 imply that any family of q-orthogonal polynomials can be evaluated fast.

4.3 Polynomial and rational solutions of q-differential equations

The computation of polynomial and rational solutions of linear differential equations lies at the heart of several important algorithms, for computing hypergeometric and Liouvillian solutions, for factoring and for computing differential Galois groups [73]. Creative telescoping algorithms (of second generation) for multiple integration with parameters [28, 50] also rely on computing rational solutions, or deciding their existence. The situation is completely similar for q-differential equations: improving algorithms for polynomial and rational solutions of such equations is important in finding q-hypergeometric solutions [3], in computing q-differential Galois groups [43], and in performing q-creative telescoping [28, 49, 50].

In both differential and q-differential cases, algorithms for computing polynomial solutions proceed in two distinct phases: (i) compute a degree bound N, potentially exponentially large in the equation size; (ii) reduce the problem of computing polynomial solutions of degree at most N to linear algebra. Abramov, Bronstein and Petkovšek showed in [1] that, in step (ii), linear algebra in size N can be replaced by solving a much smaller system, of polynomial size. However, setting up this smaller system still requires linear time in N, essentially by unrolling a (q-)linear recurrence up to terms of indices close to N. For differential (and difference) equations, this step has been improved in [17, 18], by using Chudnovskys' algorithms for computing fast the N-th term of a holonomic sequence. This allows for instance to decide (non-)existence of polynomial solutions in sublinear time $\tilde{O}(\sqrt{N})$. Moreover, when polynomial solutions exist, one can represent / manipulate them in *compact form* using the recurrence and initial terms as a compact data structure.

The same improvements can be transferred to q-differential equations, in order to improve the existing algorithms [1, 2, 46]. In this case, setting up the smaller system in phase (ii) amounts to computing the N-th term of a q-holonomic sequence, and this can be done fast using our results in §3*.

*A technical subtlety is that, as pointed out in [1, §4.3], it is not obvious in the q-differential case how to guarantee the non-singularity of the q-recurrence on the coefficients of the solution. This induces potential technical complications similar to the ones for polynomial solutions of differential equations in small characteristic, which can nevertheless be overcome by adapting the approach described in [22, §3.2].

4.4 q-hypergeometric creative telescoping

In the case of differential and difference hypergeometric creative telescoping, it was demonstrated in [17] that the compact representation for polynomial solutions can be used as an efficient data structure, and can be applied to speed up the computation of Gosper forms and Zeilberger's classical summation algorithm [61, §6]. The key to these improvements lies in the fast computation of the N-th term of a holonomic sequence, together with the close relation between Gosper's algorithm and the algorithms for rational solutions.

Similarly, in the q-differential case, Koornwinder's q-Gosper algorithm [49, §5] is closely connected to Abramov's algorithm for computing rational solutions [2, §2], and this makes it possible to transfer the improvements for rational solutions to the q-Gosper algorithm. This leads in turn to improvements upon Koornwinder's algorithm for q-hypergeometric summation [49], along the same lines as in the differential and difference cases [17].

5 EXPERIMENTS

A preliminary implementation in Magma of Algorithms 1 and 2 in §2.1 delivers some encouraging timings. Of course, since these algorithms are designed to be fast in the *arithmetic model*, it is natural to make experiments over a finite field $\mathbb{K}$, or over truncations of real/complex numbers, as was done in [54] for the problem in §2.2. Recall that both Algorithms 1 and 2 compute $\prod_{i=0}^{N-1}(\alpha - q^i) \in \mathbb{K}$, given α, q in a field $\mathbb{K}$, and $N \in \mathbb{N}$. In our experiments, $\mathbb{K}$ is the finite field $\mathbb{F}_p$ with $p = 2^{30} + 3$ elements. Timings are given in Table 1. We compare the straightforward iterative algorithm (column Naive), to the fast baby-step / giant-step algorithms, one based on subproduct trees and resultants (column Algorithm 1), the other based on multipoint evaluation on geometric sequences (column Algorithm 2).

Some conclusions can be drawn by analyzing these timings:

- The theoretical complexities are perfectly reflected in practice: timings are multiplied (roughly) by 4 in column Naive, and (roughly) by 2 in columns Algorithm 1 and Algorithm 2.
- The asymptotic regime is reached from the very beginning.
- Algorithm 2 is always faster than Algorithm 1, which is itself much faster than the Naive algorithm, as expected.
- A closer look into the timings shows that for Algorithm 1, $\approx 80\%$ of the time is spent in step (3) (resultant computation), the other steps taking $\approx 10\%$ each; for Algorithm 2, step (1) takes $\approx 25\%$, step (2) takes $\approx 75\%$, and step (3) is negligible.

degree N	Naive algorithm	Algorithm 1	Algorithm 2
2^{16}	0.04	0.03	0.00
2^{18}	0.18	0.03	0.01
2^{20}	0.72	0.06	0.01
2^{22}	2.97	0.14	0.02
2^{24}	11.79	0.32	0.04
2^{26}	47.16	0.73	0.08
2^{28}	188.56	1.68	0.15
2^{30}	755.65	3.84	0.31
2^{32}	3028.25	8.65	0.64
2^{34}		19.65	1.41
2^{36}		44.42	2.96
2^{38}		101.27	6.36
2^{40}		228.58	14.99
2^{42}		515.03	29.76
2^{44}		1168.51	61.69
2^{46}		2550.28	137.30
2^{48}			297.60
2^{50}			731.63
2^{52}			1395.33
2^{54}			3355.39

Table 1 Comparative timings (in seconds) for the computation of $\prod_{i=0}^{N-1}(\alpha - q^i) \in \mathbb{F}_p$, with $p = 2^{30} + 3$ and (α, q) randomly chosen in $\mathbb{F}_p \times \mathbb{F}_p$. All algorithms were executed on the same machine, running Magma v. 2.24. For each target degree N, each execution was limited to one hour. Naive algorithm could reach degree $N = 2^{32}$, Algorithm 1 degree $N = 2^{46}$, and Algorithm 2 degree $N = 2^{54} = 8\,014\,398\,509\,481\,984$. By extrapolation, the Naive algorithm would have needed $\approx 4^{11} \times 3028.25$ sec. ≈ 400 years on the same instance, and Algorithm 2 approximately 18 hours.

6 CONCLUSION AND FUTURE WORK

We have shown that selected terms of q-holonomic sequences can be computed fast, both in theory and in practice, the key being the extension of classical algorithms in the holonomic ("$q = 1$") case. We have demonstrated through several examples that this basic algorithmic improvement has many other algorithmic implications, notably on the faster evaluation of many families of polynomials and on the acceleration of algorithms for q-differential equations.

Here are some questions that we plan to investigate in the future.

1. (Computing curvatures of q-differential equations) In the differential case, p-curvatures can be computed fast [14–16, 22]. What about the q-differential analogue? One strong motivation comes from the fact that the q-analogue [10] of Grothendieck's conjecture (relating equations over $\mathbb{Q}$ with their reductions modulo primes p) is proved [29]. This could be used to improve the computation of rational solutions.
2. (Computing points on q-curves) Counting efficiently points on (hyper-)elliptic curves leads to questions like: for $a, b \in \mathbb{Z}$, compute the coeff. of $x^{\frac{p-1}{2}}$ in $G_p(x) := (x^2 + ax + b)^{\frac{p-1}{2}}$ modulo p, for one [19] or several [41] primes p. A natural extension is to ask the same with $G_p(x)$ replaced by $\prod_{k=1}^{\frac{p-1}{2}}(q^{2k}x^2 + aq^k x + b)$. This might have applications related to Question 1, or to counting points on q-deformations [67].
3. (Computing q-deformed real numbers) Recently, Morier-Genoud and Ovsienko [53] introduced q-analogues of real numbers. How fast can one compute (truncations / evaluations of) quantized versions of numbers like e or π?
4. (Evaluating more polynomials) Is it possible to evaluate fast polynomials of the form $\sum_{n=0}^{N} x^{n^s}$, for $s \geq 3$, and many others that escape the q-holonomic class? *E.g.*, [9] presents a beautiful generalization of Algorithm 1 to the fast evaluation of isogenies between elliptic curves, by using *elliptic resultants*, with applications in isogeny-based cryptography.

Acknowledgements. I thank Luca De Feo for his initial question, who motivated this work, and for the very interesting subsequent discussions. My friendly thanks go to Lucia Di Vizio, Kilian Raschel and Sergey Yurkevich for their careful reading of the manuscript. I am indebted to the three referees for many helpful remarks. This work was supported in part by DeRerumNatura ANR-19-CE40-0018.

REFERENCES

[1] S. A. Abramov, M. Bronstein, and M. Petkovšek. On polynomial solutions of linear operator equations. In *ISSAC'95*, pages 290–296. ACM, 1995.

[2] S. A. Abramov. Rational solutions of linear difference and q-difference equations with polynomial coefficients. *Programmirovanie*, (6):3–11, 1995.

[3] S. A. Abramov, P. Paule, and M. Petkovšek. q-hypergeometric solutions of q-difference equations. *Discrete Math.*, 180(1-3):3–22, 1998.

[4] M. Aldaz, G. Matera, J. L. Montaña, and L. M. Pardo. A new method to obtain lower bounds for polynomial evaluation. *TCS*, 259(1-2):577–596, 2001.

[5] G. E. Andrews. *The theory of partitions.* Addison-Wesley, Reading, 1976.

[6] D. Bar-Natan and S. Garoufalidis. On the Melvin-Morton-Rozansky conjecture. *Invent. Math.*, 125(1):103–133, 1996.

[7] M. Beeler, R. Gosper, and R. Schroeppel. *HAKMEM.* Artificial Intelligence Memo No. 239. MIT, 1972. http://www.inwap.com/pdp10/hbaker/hakmem/algorithms.

[8] D. J. Bernstein. Fast multiplication and its applications. In *Algorithmic number theory: lattices, number fields, curves and cryptography*, *MSRIP* 44:325–384. 2008.

[9] D. J. Bernstein, L. De Feo, A. Leroux, and B. Smith. Faster computation of isogenies of large prime degree. Preprint, 2020. https://eprint.iacr.org/2020/341.

[10] J.-P. Bézivin. Les suites q-récurrentes linéaires. *Comp. Math.*, 80:285–307, 1991.

[11] L. I. Bluestein. A linear filtering approach to the computation of the discrete Fourier transform. *IEEE Trans. Electroacoustics*, AU-18:451–455, 1970.

[12] H. Böing and W. Koepf. Algorithms for q-hypergeometric summation in computer algebra. *J. Symbolic Comput.*, 28(6):777–799, 1999.

[13] A. Borodin and S. Cook. On the number of additions to compute specific polynomials. *SIAM J. Comput.*, 5(1):146–157, 1976.

[14] A. Bostan, X. Caruso, and É. Schost. A fast algorithm for computing the characteristic polynomial of the p-curvature. In *ISSAC'14*, pages 59–66. ACM, 2014.

[15] A. Bostan, X. Caruso, and É. Schost. A fast algorithm for computing the p-curvature. In *ISSAC'15*, pages 69–76. ACM, 2015.

[16] A. Bostan, X. Caruso, and É. Schost. Computation of the similarity class of the p-curvature. In *ISSAC'16*, pages 111–118. ACM, 2016.

[17] A. Bostan, F. Chyzak, T. Cluzeau, and B. Salvy. Low complexity algorithms for linear recurrences. In *ISSAC'06*, pages 31–38. ACM, 2006.

[18] A. Bostan, T. Cluzeau, and B. Salvy. Fast algorithms for polynomial solutions of linear differential equations. In *ISSAC'05*, pages 45–52. ACM, 2005.

[19] A. Bostan, P. Gaudry, and É. Schost. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM J. Comput.*, 36(6):1777–1806, 2007.

[20] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.

[21] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *J. Complexity*, 21(4):420–446, 2005.

[22] A. Bostan and É. Schost. Fast algorithms for differential equations in positive characteristic. In *ISSAC'09*, pages 47–54. ACM, 2009.

[23] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer, 1997.

[24] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.

[25] P. Cartier. Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger). *Astérisque*, (206): 41–91, 1992. S. Bourbaki.

[26] D. V. Chudnovsky and G. V. Chudnovsky. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited (Urbana-Champaign, Ill., 1987)*, pages 375–472. Academic Press, Boston, MA, 1988.

[27] F. Chyzak. Gröbner bases, symbolic summation and symbolic integration. In *Gröbner bases and applications*, volume LMS LN 251:32–60. CUP, 1998.

[28] F. Chyzak. An extension of Zeilberger's fast algorithm to general holonomic functions. *Discrete Math.*, 217(1-3):115–134, 2000.

[29] L. Di Vizio. Arithmetic theory of q-difference equations: the q-analogue of Grothendieck-Katz's conjecture on p-curvatures. *Invent. Math.*, 150:517–578, 2002.

[30] L. Di Vizio, J.-P. Ramis, J. Sauloy, and C. Zhang. Équations aux q-différences. *Gaz. Math.*, (96):20–49, 2003.

[31] T. Ekedahl and G. van der Geer. Cycle classes on the moduli of K3 surfaces in positive characteristic. *Selecta Math. (N.S.)*, 21(1):245–291, 2015.

[32] T. Ernst. *A comprehensive treatment of q-calculus.* Birkhäuser/Springer, 2012.

[33] J. Fürlinger and J. Hofbauer. q-Catalan numbers. *JCTA*, 40(2):248–264, 1985.

[34] F. Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303. ACM, 2014.

[35] S. Garoufalidis and C. Koutschan. Irreducibility of q-difference operators and the knot 7_4. *Algebr. Geom. Topol.*, 13(6):3261–3286, 2013.

[36] S. Garoufalidis and T. T. Q. Lê. The colored Jones function is q-holonomic. *Geom. Topol.*, 9:1253–1293, 2005.

[37] S. Garoufalidis and T. T. Q. Lê. A survey of q-holonomic functions. *Enseign. Math.*, 62(3-4):501–525, 2016.

[38] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* CUP, 3rd ed., 2013.

[39] I. Gessel. A noncommutative generalization and q-analog of the Lagrange inversion formula. *Trans. Amer. Math. Soc.*, 257(2):455–482, 1980.

[40] G. Hanrot, M. Quercia, and P. Zimmermann. The middle product algorithm. I. *Appl. Algebra Engrg. Comm. Comput.*, 14(6):415–438, 2004.

[41] D. Harvey. Counting points on hyperelliptic curves in average polynomial time. *Ann. of Math. (2)*, 179(2):783–803, 2014.

[42] J. Heintz and M. Sieveking. Lower bounds for polynomials with algebraic coefficients. *TCS*, 11(3):321–330, 1980.

[43] P. A. Hendriks. An algorithm for computing a standard form for second-order linear q-difference equations. *J. Pure Appl. Algebra*, 117/118:331–352, 1997.

[44] J. Hua. Counting representations of quivers over finite fields. *J. Algebra*, 226(2):1011–1033, 2000.

[45] M. Kauers and C. Koutschan. A Mathematica package for q-holonomic sequences and power series. *Ramanujan J.*, 19(2):137–150, 2009.

[46] D. E. Khmel′ nov. Improved algorithms for solving difference and q-difference equations. *Programmirovanie*, (2):70–78, 2000.

[47] A. A. Kirillov and A. Melnikov. On a remarkable sequence of polynomials. In *Algèbre non commutative, groupes quantiques et invariants (Reims, 1995)*, volume 2 of *Sémin. Congr.*, pages 35–42. Soc. Math. France, Paris, 1997.

[48] R. Koekoek, P. A. Lesky, and R. F. Swarttouw. *Hypergeometric orthogonal polynomials and their q-analogues.* Monographs in Mathematics. Springer, 2010.

[49] T. H. Koornwinder. On Zeilberger's algorithm and its q-analogue. *J. Comput. Appl. Math.*, 48(1-2):91–111, 1993.

[50] C. Koutschan. A fast approach to creative telescoping. *Math. Comput. Sci.*, 4(2-3):259–266, 2010.

[51] H. Labrande. Computing Jacobi's theta in quasi-linear time. *Math. Comp.*, 87(311):1479–1508, 2018.

[52] R. J. Lipton. Polynomials with $0-1$ coefficients that are hard to evaluate. *SIAM J. Comput.*, 7(1):61–69, 1978.

[53] S. Morier-Genoud and V. Ovsienko. On q-deformed real numbers. *Exp. Math.*, pages 1–9, 2019. To appear.

[54] D. Nogneng and É. Schost. On the evaluation of some sparse polynomials. *Math. Comp.*, 87(310):893–904, 2018.

[55] A. Ostrowski. On two problems in abstract algebra connected with Horner's rule. In *Studies in mathematics and mechanics presented to Richard von Mises*, pages 40–48. Academic Press Inc., 1954.

[56] I. Pak. Partition bijections, a survey. *Ramanujan J.*, 12(1):5–75, 2006.

[57] V. Y. Pan. Methods of computing values of polynomials. *Russian Mathematical Surveys*, 21(1):105–136, 1966.

[58] M. S. Paterson and L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2:60–66, 1973.

[59] P. Paule and S. Radu. Rogers-Ramanujan functions, modular functions, and computer algebra. In *Advances in computer algebra*, PROMS 226, 229–280, 2018.

[60] P. Paule and A. Riese. A Mathematica q-analogue of Zeilberger's algorithm based on an algebraically motivated approach to q-hypergeometric telescoping. In *Special functions, q-series and related topics*, FIC 14:179–210. AMS, 1997.

[61] M. Petkovšek, H. S. Wilf, and D. Zeilberger. $A = B$. A K Peters, 1996.

[62] J. M. Pollard. Theorems on factorization and primality testing. *Proc. Cambridge Philos. Soc.*, 76:521–528, 1974.

[63] L. R. Rabiner, R. W. Schafer, and C. M. Rader. The chirp z-transform algorithm and its application. *Bell System Tech. J.*, 48:1249–1292, 1969.

[64] A. Riese. qMultiSum—a package for proving q-hypergeometric multiple summation identities. *J. Symbolic Comput.*, 35(3):349–376, 2003.

[65] C. Sabbah. Systèmes holonomes d'équations aux q-différences. In *D-modules and microlocal geometry (Lisbon, 1990)*, pages 125–147. de Gruyter, 1993.

[66] C.-P. Schnorr. improved lower bounds on the number of multiplications / divisions which are necessary to evaluate polynomials. *TCS*, 7(3):251–261, 1978.

[67] P. Scholze. Canonical q-deformations in arithmetic geometry. *Ann. Fac. Sci. Toulouse Math. (6)*, 26(5):1163–1192, 2017.

[68] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica*, 7:395–398, 1977.

[69] T. Sprenger and W. Koepf. Algorithmic determination of q-power series for q-holonomic functions. *J. Symbolic Comput.*, 47(5):519–535, 2012.

[70] V. Strassen. Polynomials with rational coefficients which are hard to compute. *SIAM J. Comput.*, 3:128–149, 1974.

[71] V. Strassen. Einige Resultate über Berechnungskomplexität. *Jber. Deutsch. Math.-Verein.*, 78(1):1–8, 1976/77.

[72] T. Tao, E. Croot, III, and H. Helfgott. Deterministic methods to find primes. *Math. Comp.*, 81(278):1233–1246, 2012.

[73] M. van der Put and M. F. Singer. *Galois theory of linear differential equations*, volume 328 of *Grundlehren der Mathematischen Wissenschaften*. Springer, 2003.

[74] H. S. Wilf and D. Zeilberger. An algorithmic proof theory for hypergeometric (ordinary & q) multisum/integral identities. *Invent. Math.*, 108(3):575–633, 1992.

[75] K.-W. Yang. On the product $\prod_{n\geq 1}(1+q^n x+q^{2n}x^2)$. *J. Austral. Math. Soc. Ser. A*, 48(1):148–151, 1990.

[76] D. Zagier. Elliptic modular forms and their applications. In *The 1-2-3 of modular forms*, Universitext, pages 1–103. Springer, 2008.

[77] D. Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32(3):321–368, 1990.