Fast Computation of the N-th Term of a q-Holonomic Sequence and Applications

Alin Bostan, Sergey Yurkevich

Fast Computation of the N-th Term of a q-Holonomic Sequence and Applications

Alin Bostan[a], Sergey Yurkevich[b]

[a] *Inria, Université Paris-Saclay (France)*
[b] *University of Vienna (Austria)*

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 essentially 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 $\sqrt{N}$; surprisingly, they are simpler than their analogues in the holonomic case. We provide a detailed cost analysis, in both arithmetic and bit complexity models. Moreover, 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.

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 [112] conjectured in 1954 that this is *optimal for generic polynomials*, *i.e.*, whose coefficients are algebraically independent over the prime subfield. This optimality result was proved a few years later by Pan [114].

However, most polynomials that one might wish to evaluate "in practice" have coefficients which are not algebraically independent. Paterson and Stockmeyer [116] showed, using the *baby-step/giant-step* technique, that for any field $\mathbb{K}$, an arbitrary polynomial $P \in \mathbb{K}[x]$ of degree N can be evaluated at any point in an arbitrary $\mathbb{K}$-algebra A using $O(\sqrt{N})$ *nonscalar* multiplications, *i.e.*, multiplications in A. However, their algorithm uses a linear amount of scalar multiplications, so it is not well adapted to the evaluation

Email addresses: alin.bostan@inria.fr (Alin Bostan), sergey.yurkevich@univie.ac.at (Sergey Yurkevich)

at points from the base field $\mathbb{K}$, since in this case the total arithmetic complexity, counted in terms of operations in $\mathbb{K}$, remains linear in N.

For some families of polynomials, one can do much better. Typical examples are x^N and

$$P_N(x) \coloneqq x^{N-1} + \cdots + x + 1,$$

which can be evaluated by the *square-and-multiply* technique in $O(\log N)$ operations. (Note that for $P_N(x)$ such a fast algorithm needs to perform division.) By contrast, a family $(F_n(x))_n$ of univariate polynomials is called *hard to compute* if for large enough N, 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 [115, 116] proved the existence of polynomials in $\mathbb{Q}[x]$ which are hard to compute (note that this does not follow from Pan's result [114]). However, their proof was based on a non-constructive argument. Specific families of hard-to-compute polynomials were first exhibited by Strassen [133]. For instance, he proved that for large N, the polynomial $\sum_{\ell=0}^{N} 2^{2^\ell} x^\ell$ needs at least $\sqrt{N/(3\log N)}$ operations to be evaluated. The techniques were refined and improved by Borodin and Cook [29], Lipton [104] and Schnorr [127], 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 [83] using algebraic geometry, and by Aldaz et al. [10] using a combinatorial approach. The topic is vast and very well summarized in the book by Bürgisser, Clausen and Shokrollahi [45].

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

$$Q_N(x) \coloneqq P_1(x) \cdots P_N(x),$$

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 striking example is

$$R_N(x) \coloneqq \sum_{\ell=0}^{N} x^{\ell^2},$$

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 [110] (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 [134]. The Chudnovsky brothers extended in [49] 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 in this article 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 arithmetic progressions used as a subroutine in Strassen's and Chudnovskys' algorithms is replaced by multipoint evaluation on geometric progressions, which is considerably simpler [42].

A consequence of our results is that the following apparently unrelated polynomials and 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* [141], *i.e.*, the coefficient of t^n in the product
$$\prod_{k\geq 1}(1+q^k t+q^{2k}t^2).$$
- $B_n(q) \coloneqq \prod_{i=1}^{\infty}(1-q^i) \bmod q^n$; by Euler's pentagonal theorem [113, §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 [94]; by [59], $C_n(q)$ 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-difference equations, or equivalently, q-shift (or, q-differential) 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 [125], Gauss [75] and Heine [82]. 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 [135], Jackson [87–89], Carmichael [47], Mason [106], Adams [7, 8], Trjitzinsky [137], Le Caine [102] and Hahn [77], to name just a few. Modern accounts of the various aspects of the theory (including historical ones) can be found in [57, 60, 95].

One of the reasons for interest in q-differential 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-difference 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 [82], see also [95, §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 [140], 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 [28, 48, 119]; several implementations of those algorithms are described for instance in [91, 118, 122, 132]. Algorithms for computing polynomial, rational and q-hypergeometric solutions of q-difference equations were designed by Abramov and collaborators [1, 2, 4, 92]. These algorithms are important for several reasons. One is that they lie at the heart of the vast generalization by Chyzak [50, 51] 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 [144] and Sabbah [126] (in the language of D-modules), see also [48, § 2.5] and [72].

The simplest non-trivial holonomic sequence is $(n!)_{n\geq 0}$, whose n-th term 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

$$[n]_q! \coloneqq (1+q)(1+q+q^2)\cdots(1+q+\cdots+q^{n-1}).$$

This is the q-analogue of $n!$; it is the simplest non-trivial q-holonomic sequence.

There is also a natural q-analog of the binomial coefficients, called the *Gaussian coefficients*, defined by

$$\binom{n}{k}_q \coloneqq \frac{[n]_q!}{[k]_q![n-k]_q!}.$$

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 more basic examples, the q-analog [14, Thm. 3.3] of the binomial theorem is given by

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

and the q-version [14, 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 [14, 15, 105, 113, 131, 141] and other subfields of combinatorics [16, 44, 59, 63, 93, 94, 142]; theta functions and modular forms [22, 73, 100, 101, 143]; special functions [30, 86, 95] and in particular orthogonal polynomials [97]; algebraic geometry [58], representation theory [85]; knot theory [68–72]; Galois theory [84]; number theory [6, 55, 111].

The main messages of this article are that for any example of a q-holonomic sequence occurring in those various fields, *one can compute selected coefficients faster than by a direct algorithm* and that *this fact finds a tremendous number of applications.*

Complexity basics. We estimate the arithmetic 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 currently known upper bound is $\theta < 2.3729$ [11, 64]. As usual, $O(\cdot)$ stands for the big-Oh notation and $\tilde{O}(\cdot)$ is used to hide polylogarithmic factors in the argument. 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 $\mathbf{M}(d) = O(d \log d \log\log d)$ [46, 129]. Finally, the arithmetic cost of multiplication of polynomial matrices of size n and degree d is denoted by $\mathbf{MM}(n, d)$ and we have $\mathbf{MM}(n, d) = O(n^\theta d + n^2 \mathbf{M}(d)) = \tilde{O}(n^\theta d)$ [42]. An excellent general reference for these questions is the book by von zur Gathen and Gerhard [74].

A short version of this article has appeared at the ISSAC'20 conference [31]. In the present version, we included the proofs of Theorems 6 and 8, we added a new Theorem 5 containing a detailed complexity analysis of the main algorithm (Algorithm 3) with respect to all parameters, and we displayed pseudo-code for the algorithms as well as figures visualizing their performance. We also elaborated on a task which was mentioned as future work in the previous version, namely the application of our methods to the computation of curvatures of q-difference equations, see §4.4.

The structure of the article is as follows: in Section 2 we deal with the tasks of evaluating $Q_N(x)$ and $R_N(x)$. We show that these are two instances of the same problem and provide Algorithm 3 which solves both in $O(\mathbf{M}(\sqrt{N}))$ arithmetic complexity. Section 3 is devoted to the main results; we prove there that Algorithm 3 can be used for computing terms of any q-holonomic sequence with the same cost, and provide extensions and more insight. In the same section we also consider the bit-complexity model. We identify and elaborate on several applications for our result in Section 4. In Section 5 we report on implementations of our algorithms, which deliver encouraging timings, and we finally describe future tasks and investigation fields in Section 6.

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.

2.1. Evaluation of some structured polynomials

Here is our first example, that emerged from a question asked to the first author by Luca De Feo (private email communication, 10 January 2020); this was the starting point of the article.

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

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

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 $\alpha - 1, \alpha - q, \ldots, \alpha - q^{N-1}$ in $O(N)$ more operations in $\mathbb{K}$, and finally returning their product. The total arithmetic cost of this algorithm[1] 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
$$G(x) \coloneqq \prod_{j=0}^{s-1} (x - q^j).$$
2. (Giant-step) Compute $Q \coloneqq q^s, Q^2, \ldots, Q^{s-1}$, and deduce the coefficients of the polynomial
$$H(x) \coloneqq \prod_{k=0}^{s-1} (\alpha - Q^k \cdot x).$$
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 [74, 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

[1] If $q^n = 1$ for some $n < N$, then it is enough to compute the product of $\alpha - q^i$ for $i = 0, \ldots, n-1$ and its appropriate power. The latter step can be done efficiently (in essentially $\log(N)$ operations) using binary powering. Our main interest lies therefore in $q \in \mathbb{K}$ that are not roots of unity of small order.

by [74, 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 $\sqrt{N}$.

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

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 described on page 523 of [120].

Algorithm 2

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

Obviously, the output of this algorithm is

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

As pointed out in the remarks after the proof of [42, 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 $P_{2t}(x) = P_t(q^t x) \cdot P_t(x)$ and $P_{2t+1}(x) = (\alpha - q^{2t}x) \cdot P_t(q^t x) \cdot P_t(x)$. 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 [121] and of Bluestein [27]. 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* [40, 78], 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. In fact, we do not provide a pseudo-code implementation for this algorithm, since we will do so for the more general case (Algorithm 3).

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 [117], which are ubiquitous in complex analysis [22], number theory [143] and combinatorics [14]. 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 $\eta = e^{2\pi iz}$. Here, $\mathbb{K} = \mathbb{C}$. Another example is the *Dedekind eta function*, appearing in Euler's famous *pentagonal theorem* [113, §5], which has a similar form

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

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 [136] 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 [136] 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 [110] 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 [110]. 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

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

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

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

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

$$v_{n+2}(q) - v_{n+1}(q) = A(q, q^n) \cdot (v_{n+1}(q) - v_n(q)), \tag{4}$$

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

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

where

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

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 $P(x) \coloneqq M(q^{s-1}x) \cdots M(qx)M(x)$.
(2) (Giant-step) Compute $Q \coloneqq q^s, Q^2, \ldots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \ldots, Q^{s-1}$.
(3) Return the product $P(Q^{s-1}) \cdots P(Q)P(1)$.

Clearly, this algorithm generalizes Algorithm 2 in §2.1 and, as promised, we also provide a detailed pseudo-code implementation: Step1 and Step2 & Step3:

Algorithm 3 (Step1)
Input: $s, q, M(x)$
Output: $M(q^{s-1}x)\cdots M(qx)M(x)$

$q_s \leftarrow [q, q^2, \ldots, q^{s-1}]$
$t \leftarrow s$
function $\mathcal{BS}(t)$
  if $t = 1$ **then**
    return $M(x)$
  if t is even **then**
    $p_1(x) \leftarrow \mathcal{BS}(t/2)$
    $p_2(x) \leftarrow p_1(q^{t/2}x)$ ▷ Using q_s
    return $p_2(x) \cdot p_1(x)$ ▷ Fast polynomial multiplication
  else
    $p_1(x) \leftarrow \mathcal{BS}((t-1)/2)$
    $p_2(x) \leftarrow p_1(q^{(t-1)/2}x)$ ▷ Using q_s
    $p_3(x) \leftarrow M(q^{t-1}x)$ ▷ Using q_s
    return $p_3(x) \cdot p_2(x) \cdot p_1(x)$ ▷ Fast polynomial multiplication

Algorithm 3 (Step2 & Step3)
Input: $s, Q, P(x)$
Output: $P(Q^{s-1})\cdots P(1)$

Assumptions: $Q \neq 0$, $P(x)$ polynomial matrix of size $n \times n$ and degree $d \geq s$.

$Q_d \leftarrow [Q, Q^2, \ldots, Q^{d-1}]$
$Q' \leftarrow 1/Q$
$Q'_d \leftarrow [Q^{-\binom{d}{2}}, \ldots, Q^{-\binom{1}{2}}, Q^{\binom{0}{2}}, \ldots, Q^{\binom{2d}{2}}]$ ▷ Using Q_d and Q'
$P_{s-1}, \ldots, P_0$ ▷ Empty $n \times n$ matrices
for i from 1 to n **do**
  for j from 1 to n **do**
    $p(x) \leftarrow P(x)_{i,j}$ ▷ $p(x) = c_0 + c_1 x + \cdots + c_d x^d$
    $p_1(x) \leftarrow \sum_{\ell=0}^{d} c_\ell Q^{-\binom{\ell}{2}} x^{d-\ell}$ ▷ Using Q'_d
    $p_2(x) \leftarrow \sum_{\ell=0}^{2d} Q^{\binom{\ell}{2}} x^\ell$ ▷ Using Q'_d
    $p_3(x) \leftarrow P_1(x) \cdot P_2(x)$ ▷ $p_3(x) = \sum_{\ell=0}^{3d} r_\ell x^\ell$; fast multiplication
    for k from 0 to $s-1$ **do**
      $(P_k)_{i,j} \leftarrow r_{d+k}$ ▷ $P_\ell = P(Q^\ell)$ for $\ell = 0, \ldots, s-1$
$P \leftarrow 1$
for k from 0 to $s-1$ **do**
  $P \leftarrow P_k \cdot P$
return P

By the same observations as in Algorithm 2 in §2.1, the complexity of Algorithm 3 already is quasi-linear in $\sqrt{N}$. In the next section we will discuss the complexity not only with respect to N, but to the matrix size and degree as well.

We remark that when applied to the computation of $v_N^{(p,a,b)}(q)$, the dependence in a, b

of Algorithm 3 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

$$\tilde{M}(x) := \begin{bmatrix} prx+1 & -prx \\ 1 & 0 \end{bmatrix}, \text{ with } r := q^{a+b}, \tag{5}$$

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 (5).
(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 $\sqrt{N}$.

In the next section, we will show that Algorithm 3 can be employed for the fast computation of the N-th term of *any* q-holonomic sequence. Note that the trick in Algorithm 4 relies on the fact that $M(x)$, coming from the recurrence for $v_N^{(p,a,b)}(q)$, contains only pure powers of x and q. We cannot hope for this phenomenon in general, however we advise to bear this simplification in mind for some practical purposes. In any case, we can improve on the quasi-linear cost in the degree d of the polynomial matrix $M(x)$ in Algorithm 3, obtaining a complexity of essentially $\sqrt{d}$; in essence, the idea consists in choosing $s = \sqrt{N/d}$ rather than $\sqrt{N}$, see §3.4.

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 $u_n = u_n(q)$ is q-holonomic if it satisfies a nontrivial q-recurrence, that is, a linear recurrence with coefficients given by polynomials in q and q^n.

Definition 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 $c_0(x,y), \ldots, c_r(x,y)$ in $\mathbb{K}[x,y]$, with $c_r(x,y) \neq 0$, such that*

$$c_r(q,q^n)u_{n+r}(q) + \cdots + c_0(q,q^n)u_n(q) = 0, \quad \text{for all } n \geq 0. \tag{6}$$

The integer r is called the order *of the q-recurrence* (6)*. 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*,

$$[n]_q := 1 + q + \cdots + q^{n-1} \quad \text{and} \quad [n]_q! := \prod_{k=1}^{n} [k]_q. \tag{7}$$

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

The sequences $(u_n)_{n\geq 0} = (q^n)_{n\geq 0}$, $(v_n)_{n\geq 0} = (q^{n^2})_{n\geq 0}$ and $(w_n)_{n\geq 0} = (q^{\binom{n}{2}})_{n\geq 0}$ 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 $(q^{n^3})_{n\geq 0}$ is not q-holonomic [72, Ex. 2.2(b)]. More generally, this also holds for the sequence $(q^{n^s})_{n\geq 0}$, for any $s > 2$, see [26, Th. 4.1] and also [66, Th. 1.1].

Another basic example is the q-Pochhammer symbol

$$(x;q)_n := \prod_{k=0}^{n-1} (1 - xq^k), \tag{8}$$

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 $(q)_{n+1} - (1-q^{n+1})(q)_n = 0$. In Section §2 we encountered $v_n^{(p,a,b)} = \sum_{k=0}^{n} p^k q^{ak^2+bk}$, which is q-holonomic (see Eq. (4)), but generally not q-hypergeometric.

Note that (6) reduces to a C-linear recurrence, *i.e.* a linear recurrence with *constant* coefficients, if all polynomials $c_0(x,y), \ldots, c_r(x,y)$ are constant in the variable y. For these kinds of sequences there exist quasi-optimal algorithms [41, 61, 107], therefore we assume from now on that the maximal degree d of $c_0(x,y), \ldots, c_r(x,y)$ in y is positive.

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 [19]; it is a q-holonomic sequence of Laurent polynomials [71]. Its recurrence equations are themselves of interest, as they are closely related to the A-polynomial of a knot, via the *AJ conjecture* [54, 65, 67], verified in some cases using massive computer algebra calculations [69].

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 [72, 91, 96]. 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 [91] and HolonomicFunctions [98], and the Maple packages qsum [28], qFPS [132], qseries and QDifferenceEquations.

A simple but useful fact is that the order-r scalar q-recurrence (6) can be translated into a first-order recurrence on $r \times 1$ vectors:

$$\begin{bmatrix} u_{n+r} \\ \vdots \\ u_{n+1} \end{bmatrix} = \begin{bmatrix} -\frac{c_{r-1}}{c_r} & \cdots & -\frac{c_1}{c_r} & -\frac{c_0}{c_r} \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \times \begin{bmatrix} u_{n+r-1} \\ \vdots \\ u_n \end{bmatrix}. \tag{9}$$

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

$$M(q^{N-1}) \cdots M(q) M(1), \tag{10}$$

where $M(q^n)$ denotes the companion matrix from equation (9). This observation is crucial, since it exposes the connection to the algorithms presented in the previous section.

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 [38, §6], where the cost $O(\mathbf{M}(\sqrt{N}))$ is reached under some invertibility assumptions.

Theorem 1. *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 2. *Under the assumptions of Theorem 1 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 1. 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 now offer 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 [38, §6], where the improved cost $O(\mathbf{M}(\sqrt{N}))$ is reached under additional invertibility assumptions.

Theorem 3. *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* (6)*, 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 (9), 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 (9). Algorithm 3 adapts *mutatis mutandis* to this effect. □

Remark that if q is a root of unity of order $n < N$, then the computation of $U_N(q) = M(q^{N-1}) \cdots M(q)M(1)$ can be simplified using

$$U_N(q) = M(q^k) \cdots M(1) \cdot U_n(q)^r,$$

where $r = \lfloor (N-1)/n \rfloor$ and $k = N - 1 - rn$. Algorithm 3 is used to compute $U_n(q)$ and then its r-th power is then deduced via binary powering. Finally, the product $M(q^k) \cdots M(1)$ is again computed using Algorithm 3. The total cost therefore consists of just $O(\mathbf{M}(\sqrt{n}) + \log(N))$ arithmetic operations. It follows that if, for instance, the base field $\mathbb{K}$ is the prime field $\mathbb{F}_p$, then the prime number p should be larger than N in order to exhibit the full strength of the presented algorithms.

Corollary 4. *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*

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

and let $E_q^{(N)}(x) \coloneqq e_q(x) \bmod x^N$ be its polynomial truncation of degree $N-1$. If $\alpha \in \mathbb{K}$, then 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 $(u_n(q))_{n\geq 0}$ is q-hypergeometric, and satisfies the recurrence $[n+1]_q u_{n+1}(q) - \alpha u_n = 0$, therefore $v_N(q) \coloneqq \sum_{i=0}^{N-1} u_i(q)$ 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 to $v_N(q)$ concludes the proof. □

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

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

3.4. Complexity analysis and computation of several terms

Theorem 3 established an $O(\mathbf{M}(\sqrt{N}))$ cost of the presented method for computing the N-th term of a q-holonomic sequence. We now aim at performing a detailed complexity analysis with respect to all input parameters. So we need to discuss the complexity of Algorithm 3, where we assume that $M(x) \in \mathcal{M}_n(\mathbb{K}[x]_d)$ is an $n \times n$ polynomial matrix of degree $d \geq 1$. We wish to examine the amount of field operations in $\mathbb{K}$ needed for the computation of $M(q^{N-1}) \cdots M(1)$ in terms of N, d and n. Recall that $\mathbf{MM}(n,d)$ controls the arithmetic complexity of the product in $\mathcal{M}_n(\mathbb{K}[x]_d)$ and it holds that $\mathbf{MM}(n,d) = O(n^\theta d + n^2\,\mathbf{M}(d)) = \tilde{O}(n^\theta d)$.

First, we will examine the direct application of Algorithm 3, where $s = \sqrt{N}$, to $M(x)$. It turns out that the dominating part is step (1), where we compute $P_s(x) = M(q^{s-1}x) \cdots M(x)$ using the divide-and-conquer scheme $P_{2t}(x) = P_t(q^t x) \cdot P_t(x)$ and $P_{2t+1}(x) = M(q^{2t}x) \cdot P_t(q^t x) \cdot P_t(x)$. Note that $P_t(x)$ is an $n \times n$ polynomial matrix of degree at most td and therefore the cost of this step is $O(\mathbf{MM}(n, sd)) = \tilde{O}(n^\theta d\sqrt{N})$. Step (2) is done component-wisely at each entry of $P(x)$. By the explained fast chirp transform algorithms, it essentially boils down to n^2 multiplications of two polynomials, one of degree sd and the other of degree $2sd$. The cost of the second step is therefore $O(n^2\,\mathbf{M}(sd)) = \tilde{O}(n^2 d\sqrt{N})$. The last step is the multiplication of $N/s = s$ matrices with entries in $\mathbb{K}$ and has therefore an arithmetic complexity of $O(n^\theta s) = \tilde{O}(n^\theta\sqrt{N})$.

If $d < N$ is a parameter of interest, then there is a better choice of s rather than $\sqrt{N}$. We saw that the polynomial $P_s(x)$ has degree sd and we must evaluate it at N/s points. The optimal pick for s is therefore $s = \sqrt{N/d}$, which we again can assume to be integer[2]. Then, by the same arguments as above, the costs of the three steps are $O(\mathbf{MM}(n, \sqrt{Nd})) = \tilde{O}(n^\theta\sqrt{Nd})$, $O(n^2\mathbf{M}(\sqrt{Nd})) = \tilde{O}(n^2\sqrt{Nd})$ and $O(n^\theta\sqrt{Nd})$ respectively.

Now, we address specifically the computation of the N-th term in a q-holonomic sequence. If $(u_n(q))_{n\geq 0}$ is given by a q-recurrence

$$c_r(q, q^n)u_{n+r}(q) + \cdots + c_0(q, q^n)u_n(q) = 0,$$

for $q \in \mathbb{K}$ and for polynomials $c_j(x,y) \in \mathbb{K}[x,y]$, then as observed before, we can compute $u_N(q)$ via

$$\frac{1}{c_r(q, q^{N-1}) \cdots c_r(q,q)c_r(q,1)} \cdot \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \times \tilde{M}(q^{N-1}) \cdots \tilde{M}(q)\tilde{M}(1) \times \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix},$$

where now

$$\tilde{M}(x) \coloneqq c_r(q,x) \cdot M(x) = \begin{bmatrix} -c_{r-1}(q,x) & \cdots & -c_1(q,x) & -c_0(q,x) \\ c_r(q,x) & \cdots & 0 & 0 \\ & \ddots & & \vdots \\ 0 & \cdots & c_r(q,x) & 0 \end{bmatrix}.$$

[2]Similarly as before, if $\sqrt{N/d}$ is not an integer, then we can compute $u_{N_1}(q)$ first, where $N_1 = \lfloor\sqrt{N/d}\rfloor^2 d$, and then proceed "naively". Note that $N - N_1 < 2\sqrt{Nd} - d = O(\sqrt{Nd})$.

Hence, we are interested in $\tilde{M}(q^{N-1})\cdots\tilde{M}(1)$ and $c_r(q,q^{N-1})\cdots c_r(q,1)$. If the degrees of $c_0(q,y),\ldots,c_r(q,y)$ are bounded by d, then the considerations above imply that the two q-factorials can be computed in $O(\mathbf{MM}(r,\sqrt{Nd})+r^2\mathbf{M}(\sqrt{Nd}))$ and $O(\mathbf{M}(\sqrt{Nd}))$ operations in $\mathbb{K}$, respectively. We obtain the following theorem (compare with [37, Thm. 2]).

Theorem 5. *Under the assumptions of Theorem 3, let $d \geq 1$ be the maximum of the degrees of $c_0(q,y),\ldots,c_r(q,y)$. Then, for any $N > d$, the term $u_N(q)$ can be computed in $O(r^\theta\sqrt{Nd} + r^2\,\mathbf{M}(\sqrt{Nd}))$ operations in $\mathbb{K}$.*

Theorem 3 can be adapted to the computation of *several coefficients* of a q-holonomic sequence. The proof is similar to that of Theorem 15 in [38], however simpler, because we deal with geometric progressions instead of arithmetic ones.

Theorem 6. *Under the assumptions of Theorem 5, let $N_1 < N_2 < \cdots < N_n = N$ be positive integers, where $n \leq \sqrt{N}$. Then, the terms $u_{N_1}(q),\ldots,u_{N_n}(q)$ can be computed altogether in $O(\mathbf{M}(\sqrt{N})\log N)$ operations in $\mathbb{K}$.*

Proof. As before, we assume that N is a perfect square; let $s = \sqrt{N}$. Examining the presented algorithms, we notice that on the way of computing the matrix q-factorial $U_N \coloneqq M(q^{N-1})\cdots M(q)M(1)$ we obtain the evaluated polynomials

$$P(1), P(Q), \ldots, P(Q^{s-1}),$$

where $Q \coloneqq q^s$ and $P(x) \coloneqq M(q^{s-1}x)\cdots M(qx)M(x)$. The q-factorial U_N is then found by step (3) by trivially multiplying $P(Q^{s-1})\cdots P(Q)P(1)$. Observe that while multiplying together from right to left we actually also automatically compute

$$P(Q^{j-1})\cdots P(Q)P(1) = M(q^{sj-1})\cdots M(q)M(1) = U_{sj},$$

for every $j = 1,\ldots,s$. It follows that employing Algorithm 3 and by simply taking the top right element of each U_{sj}, we find not only u_N, but actually $u_s, u_{2s}, \ldots, u_{s^2} = u_N$. This already indicates that simultaneous computation of s terms is achievable in similar complexity after some "distillation". In general, we are interested in the sequence of $u_i(q)$ at indices $i = N_1,\ldots,N_n$, hence we need to perform the following refinement step.

Let $d_0 \in \mathbb{N}$ be a positive integer with $d_0 \leq n \leq \sqrt{N}$ and assume that for some $k_1^{(0)},\ldots,k_n^{(0)}$ with $k_j^{(0)} \leq N_j < k_j^{(0)} + 2d_0$ we already know the values $U_{k_1^{(0)}},\ldots,U_{k_n^{(0)}}$. Then we can use a similar strategy as in step (1) of Algorithm 3 and deduce the polynomial matrix $P_{d_0}(x) = M(q^{d_0-1}x)\cdots M(qx)M(x)$. Compute then the values $q^{k_1^{(0)}},\ldots,q^{k_n^{(0)}}$ and evaluate $P_{d_0}(x)$ simultaneously at them. For each $j = 1,\ldots,n$ it holds that

$$P_{d_0}(q^{k_j^{(0)}})\cdot U_{k_j^{(0)}} = U_{k_j^{(0)}+d_0}.$$

We perform this multiplication for those indices j for which $k_j^{(0)} + d_0 \leq N_j < k_j^{(0)} + 2d_0$. For these j we then set $k_j^{(1)} = k_j^{(0)} + d_0$ and let $k_j^{(1)} = k_j^{(0)}$ for the other indices; moreover $d_1 \coloneqq \lceil d_0/2 \rceil$. We iterate this process at most $\ell \coloneqq \lceil \log(d_0) \rceil$ many times until $d_\ell = 1$. Then we can easily find $U_{N_1},\ldots,U_{N_n}$, from which we finally deduce $u_{N_1},\ldots,u_{N_n}$.

Each such step has a cost of at most $O(\mathbf{M}(n)) = O(\mathbf{M}(\sqrt{N}))$ base field operations. Moreover, after first employing Algorithm 3 and by the consideration above, we compute $U_1, U_s, \ldots, U_{s^2}$ in $O(\mathbf{M}(\sqrt{N}))$ base operations (Theorem 5). Hence, we may choose $d_0 = s$ and for each $j = 1, \ldots, n$ let $k_j^{(0)}$ be the largest element in $\{1, s, \ldots, (s-1)s\}$ such that $k_j^{(0)} \leq N_j$. Clearly, all conditions of the above refinement step are satisfied and we need at most $\lceil \log(s) \rceil = O(\log N)$ many such steps. The total complexity is henceforth $O(\mathbf{M}(\sqrt{N})) + O(\mathbf{M}(\sqrt{N}) \log N) = O(\mathbf{M}(\sqrt{N}) \log N)$. □

The same idea applies in the following corollary which states that if $n < \sqrt{N}/N^{\varepsilon}$ for some $\varepsilon > 0$, then we can omit the log-factor in N:

Corollary 7. *Under the assumptions of Theorem 5, let $N_1 < N_2 < \cdots < N_n = N$ be positive integers, where $n < N^{\frac{1}{2}-\varepsilon}$ for some $0 < \varepsilon < \frac{1}{2}$. Then, the terms $u_{N_1}(q), \ldots, u_{N_n}(q)$ can be computed altogether in $O(\mathbf{M}(\sqrt{N}))$ operations in $\mathbb{K}$.*

Proof. Here, we follow the exact same procedure as in the proof before. Then, regarding complexity, we use $O(\mathbf{M}(n)) = O(\mathbf{M}(N^{\frac{1}{2}-\varepsilon})) = O(\mathbf{M}(\sqrt{N})N^{-\varepsilon})$ and obtain that the same method yields a total arithmetic cost of $O(\mathbf{M}(\sqrt{N})) + O(\mathbf{M}(\sqrt{N})N^{-\varepsilon} \log N) = O(\mathbf{M}(\sqrt{N}))$. □

Remark that regarding a detailed complexity analysis for the computation of several coefficients, we have the following trade-off: either we compute at most $\sqrt{N}$ terms in the arithmetic complexity $O((r^{\theta} d\sqrt{N} + r^2 \mathbf{M}(d\sqrt{N})) \log N)$, or at most $\sqrt{N/d}$ terms, but in a better cost of $O((r^{\theta}\sqrt{Nd} + r^2 \mathbf{M}(\sqrt{Nd})) \log N)$. The proofs combine the considerations above with setting $s = \sqrt{N}$ and $s = \sqrt{N/d}$ respectively. In both cases we can get rid of the log-factor in N like in Corollary 7 by computing a factor of N^{ε} less terms.

3.5. The case q is an integer: bit complexity

Until now, we only considered the arithmetic complexity model, which is very well-suited to measure the algorithmic cost when working in algebraic structures whose basic internal operations have constant cost (such as finite fields, or floating point numbers).

Now we 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.

Recall that the most basic operations on integer numbers can be performed in quasi-optimal time, that is, in a number of bit operations which is almost linear, up to logarithmic factors, in (the maximum of) their bit size. The most basic operation is integer multiplication, for which quasi-linear time algorithms are known since the early seventies, starting with the famous paper by Schönhage and Strassen [130] who showed that two n-bit integers can be multiplied in $O(n \log n \log\log n)$ bit operations. After several successive improvements, e.g., [62, 81], we know as of 2020 that two n-bit integers can be multiplied in time $O(n \log n)$ [80]. We shall call the cost of multiplying two n-bit integers $\mathbf{M}_{\mathbb{Z}}(n)$.

In this context, the matrix q-factorials from §3.1 are computed by *binary splitting* rather than by baby-step/giant-steps. Recall that this phenomenon already occurs in the usual holonomic setting. For example, the bitsize of $u_N = N!$ is $O(N \log N)$, however both, the "naive" method of computing it using $u_n = nu_{n-1}$, or Strassen's baby-steps/giant-steps method, yield worse bit complexity. In the naive approach the problem

is that integers of unbalanced bitsize are multiplied together and hence not the full power of fast integer multiplication techniques can be employed. A more clever and very simple way is to just use the fact that

$$N! = (1 \cdots \lfloor N/2 \rfloor) \times ((\lfloor N/2 \rfloor + 1) \cdots N),$$

and that the bitsizes of both factors have magnitude $O(N/2 \log(N)) = \tilde{O}(N)$. Thus, fast integer multiplication can be used to multiply them in $\tilde{O}(N)$ bit-complexity. This idea results in the binary splitting Algorithm 5. This algorithm is very classical, and we only recall it for completeness. See [23, §12] for a good survey on this technique, and its applications.

Algorithm 5 (BinSplit)
Input: $A = [a_1, \ldots, a_N]$ list of elements from some arbitrary ring R
Output: $a_N \cdots a_1$

```
1: function F(A)
2:   if N = 1 then
3:     return A[1]
4:   return F(A[⌊N/2⌋ + 1, ..., N]) · F(A[1, ..., ⌊N/2⌋])
```

Assume that each a_i in the input of BinSplit has at most k bits and let $C(n)$ be the complexity of BinSplit if $A = [a_1, \ldots, a_n]$ is n-dimensional. It follows that

$$C(N) \leq 2C(\lceil N/2 \rceil) + \mathbf{M}_{\mathbb{Z}}(N/2 \cdot k),$$

where $\mathbf{M}_{\mathbb{Z}}(n) = \tilde{O}(n)$ is the cost of multiplication of integers with n bits. We obtain $C(N) = \tilde{O}(Nk)$. Hence, using this method, the computation of $u_N = N!$ has $\tilde{O}(N)$ bit-complexity, which is quasi-optimal. Moreover, the same idea applies to any holonomic sequence, by deducing the first order matrix recurrence and computing the matrix product using BinSplit.

Now we shall see that the q-holonomic case is similar. First, let q be a positive integer of B bits and consider the computation of the q-factorial

$$u_N(q) = (1+q)(1+q+q^2)\cdots(1+q+\cdots+q^{N-1}),$$

as an illustrative example. For each factor we have the trivial inequalities

$$q^n < 1 + q + \cdots + q^n < q^{n+1},$$

meaning that $q^{N(N-1)/2} < u_N(q) < q^{N(N+1)/2}$, so the bitsize of $u_N(q)$ is of magnitude N^2B. The "naive" algorithm of deducing $u_N(q)$ by first computing the integers q^i, then the corresponding sums and products, has $\tilde{O}(N^3B)$ binary complexity. This method is not (quasi-)optimal with respect to the output size. It is also easy to see that the presented baby-steps/giant-steps based algorithms yield bad bit-complexity as well, despite their good arithmetic cost.

Similarly, if

$$u_N(q) = \sum_{n=0}^{N-1} q^{n^2},$$

then the integer $u_N(q)$ is bounded in absolute value from above by $Nq^{(N-1)^2}$ and by $q^{(N-1)^2}$ from below, so its bitsize is again of magnitude N^2B. The "naive" algorithm consisting of computing the terms q^i one after the other before summing, has again non-optimal bit-complexity $\tilde{O}(N^3B)$.

Can one do better? The answer is "yes" and one can even achieve a complexity which is quasi-linear in the bitsize of the output. Similarly to the holonomic setting, 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 as in §2.2, which can then be handled with BinSplit. To be more precise, given an integer or rational number q of bitsize B and any q-holonomic sequence $(u_n(q))_{n\geq 0}$ defined by polynomials $c_0, \ldots, c_r \in \mathbb{Z}[x, y]$ with $c_r(q, q^n) \neq 0$ for any $n \in \mathbb{N}$, we define $M(x) \in \mathbb{Q}(x)$ by

$$\begin{bmatrix} -\frac{c_{r-1}(q,x)}{c_r(q,x)} & \cdots & -\frac{c_1(q,x)}{c_r(q,x)} & -\frac{c_0(q,x)}{c_r(q,x)} \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Then, as observed before, u_N can be read off from

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

which we aim to compute efficiently. Again, instead of using baby-steps/giant-steps, it is a better idea to use binary splitting by applying Algorithm 5 to $A = [M(1), \ldots, M(q^{N-1})]$. Note that obviously, any element in A is a matrix with rational entries of bitsize bounded by $O(NB)$. Therefore, the complexity of BinSplit does not exceed $\tilde{O}(N^2B)$ by the same argument as before, now using fast multiplication of rational numbers. These considerations prove

Theorem 8. *Under the assumptions of Theorem 3, 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-difference 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 [49, §6], anticipated a decade earlier (without proof) by Schroeppel and Salamin in Item 178 of [21].

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 [63, 76], or in the enumeration of special families of matrices with coefficients in the finite field $\mathbb{F}_q$ [93, 94, 142], where sequences related to the Gaussian coefficients $\binom{n}{k}_q$ also show up.

A huge subfield of combinatorics is the theory of partitions [14], where q-holonomic sequences occur as early as in the famous Rogers-Ramanujan identities [123, 124], see also [14, 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 [12, 13], see also [14, 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

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

can be evaluated fast due to our results and to the identity [113, §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 [60]. Amongst the most basic examples, the *discrete q-Hermite polynomials* [9, 18] 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-difference equations

The computation of polynomial and rational solutions of linear differential equations lies at the heart of several important algorithms, for computing hypergeometric, d'Alembertian and Liouvillian solutions, for factoring and for computing differential Galois groups [4, 5, 139]. Creative telescoping algorithms (of second generation) for multiple

integration with parameters [51, 98] also rely on computing rational solutions, or deciding their existence. The situation is completely similar for q-difference equations, *i.e.* equations of the form

$$Ly = a_\nu(x)y(q^\nu x) + a_{\nu-1}(x)y(q^{\nu-1}x) + \cdots + a_0(x)y(x) = 0, \tag{11}$$

with $a_j(x) \in \mathbb{Q}[x]$, for all $j = 0, \ldots, \nu$, such that $a_0(x)a_\nu(x)$ is not identically zero. Improving algorithms for polynomial and rational solutions of such equations is important for finding q-hypergeometric solutions [4], for computing q-difference Galois groups [17, 84], and for performing q-creative telescoping [51, 97, 98].

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 [35, 37], 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. Similar ideas allow to also compute rational solutions in compact form in the same complexity, see [36, Chap. 17].

The same improvements can be transferred to linear q-difference equations, in order to improve the existing algorithms [1, 2, 92]. 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-difference 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 [43, §3.2]. Similar improvements can be also transferred to systems [3, 20].

Let us finish this discussion by pointing out briefly an application of these improvements. Desingularizing a linear differential operator $L(x, \partial_x)$ consists in computing a left multiple with all apparent singularities removed. It is a central task for determining the Weyl closure of L [138]. The computation of polynomial solutions of Fourier dual operators is a basic step for performing desingularizations [52]. By duality, the order N of the desingularization corresponds to the degree of polynomial solutions of the dual $L^\star$ of L. This remark in conjunction with the fast algorithms for polynomial solutions [37], themselves based on the fast computation of matrix factorials, allows to speed up the computation of desingularizations. The situation is similar in other Ore algebras, and in particular for q-difference equations. Therefore, our algorithmic improvements in the computation of polynomial solutions of q-difference equations, themselves based on the fast computation of matrix q-factorials, have a direct impact on the acceleration of the desingularizations process for q-difference equations. It is less obvious to us whether other desingularization algorithms, such as the one from [99], could also benefit from these remarks.

4.4. Computing curvatures of q-difference equations

A natural application of the fast computation of matrix q-factorials is the computation of curvatures of q-difference equations, since in this area these objects appear quite inherently. Another strong motivation comes from the fact that the q-analogue [25] of Grothendieck's conjecture (relating equations over $\mathbb{Q}$ with their reductions modulo primes p) is proved [55, 56], while the classical differential case is widely open [90]. Still, in the latter setting algorithms have been developed allowing to compute p-curvatures fast [32–34, 43]. They allow, for example, to perform a quick heuristic, but reliable in practice, test for the existence of a basis of algebraic solutions of a linear differential operator [39].

The q-analogue of Grothendieck's conjecture investigates rational solutions of q-difference equations. Similarly to the differential setting, one naturally associates to the equation (11) the linear q-difference system $Y(qx) = A(x)Y(x)$, where

$$A(x) = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & 1 \\ -a_0/a_\nu & -a_1/a_\nu & \cdots & -a_{\nu-1}/a_\nu \end{bmatrix}.$$

Then it easily follows that if $z(x)$ solves (11), then $(z(x), z(qx), \ldots, , z(q^{\nu-1}x))^t$ is a solution of $Y(qx) = A(x)Y(x)$. Moreover, these equations are in some sense equivalent through the q-analogues of the Wronskian Lemma and the Cyclic Vector Lemma whenever q is not a root of unity or order smaller than ν; see [56] for details. If q is considered as a variable, then Di Vizio and Hardouin proved that (11) admits a full set of solutions in $\mathbb{Q}(q, x)$ if and only if for almost all natural numbers n,

$$C_n(x) \coloneqq A(q^{n-1}x) \cdots A(qx)A(x) \equiv \mathrm{Id}_\nu \mod \mathrm{GL}_\nu(R_n(x)),$$

where $R_n = \mathbb{Q}[q]/\Phi_n(q)$, with $\Phi_n(x)$ the n-th cyclotomic polynomial. The elements in the sequence $(C_n(x))_{n\geq 1}$ are known as curvatures of the q-difference system and are clearly just matrix q-factorials.

On the other hand, if $q \in \mathbb{Q}$ then it already follows from main result of [55] that (11) admits a basis of rational solutions in $\mathbb{Q}(x)$ if and only if for almost all primes p,

$$A(q^{\kappa_p-1}x) \cdots A(qx)A(x) \equiv \mathrm{Id}_\nu \mod p^\ell,$$

where $\kappa_p = \mathrm{ord}_p(q)$ and $\ell_p \in \mathbb{Z}$ such that $1 - q^{\kappa_p} = p^{\ell_p}\frac{h}{g}$, with $h, g \in \mathbb{Z}$ coprime to p.

On these types of questions there is more progress in the q-difference setting than in the classical differential one. Yet, unfortunately, the theorems above are not proven to be effective in the sense that still infinitely many conditions need to be checked in order to conclude the implication we are mostly interested in. Therefore, the computation of any finite number of curvatures only provides a heuristic for the existence of rational solutions of a q-difference equation. Moreover, the mentioned algorithms in §4.3 compute rational solutions of equations of type (11) and therefore allow to decide rigorously about the existence of such a basis. However, all these methods have a cost which is potentially exponential in the size of the input. Our goal in this section is to design a fast heuristic test for the existence of a basis of rational solutions of a q-difference equation using curvatures.

If q is a variable, we want to check whether $C_n(x) \equiv \mathrm{Id}_\nu \mod \mathrm{GL}_\nu(R_n(x))$ for many n. Clearly, after the reduction mod $\Phi_n(q)$, the polynomial $C_n(x)$ has arithmetic size n^2 over $\mathbb{Q}$ and $\tilde{O}(n^3)$ bitsize. Hence, computing $C_n(x)$ is unnecessarily costly. We propose to work over $R_{n,p} \coloneqq \mathbb{F}_p[q]/\Phi_n(q)$ for some (large) prime p; moreover, we compute $C_n(x_0)$ for some randomly chosen $x_0 \in \mathbb{F}_p$. Furthermore, in order to avoid computing general cyclotomic polynomials, we compute $C_n(x)$ only for prime numbers n. After these considerations, it is easy to see that Algorithm 3 applies and allows to deduce $C_n(x_0)$ modulo $\mathrm{GL}_\nu(R_{n,p})$ in $\tilde{O}(\sqrt{n})$ arithmetic operations in $R_{n,p}$, hence $\tilde{O}(n^{3/2})$ operations in $\mathbb{F}_p$. Finally, if we want to deduce this quantity for all primes n between 2 and some $N \in \mathbb{N}$, it is wiser to apply the accumulating remainder tree method presented in [53, 79], which allows for quasi-optimal complexity of $\tilde{O}(N^2)$ in this case.

If q is some rational number and the goal is to test whether $Y(qx) = A(x)Y(x)$ has a full set of rational solutions, one may check

$$C_{\kappa_p}(x) = A(q^{\kappa_p - 1}x) \cdots A(qx)A(x) \equiv \mathrm{Id}_\nu \mod p^\ell,$$

for many primes p. Unfortunately, finding the order κ_p in practice may be costly, therefore we shall check the weaker assumption $C_{p-1}(x) \equiv \mathrm{Id}_\nu \mod p$. Conjecturally, this equality for sufficiently many primes p is also enough to conclude on the existence of a basis of rational solutions. The presented Algorithm 3 allows to compute $C_{p-1}(x_0) \mod p$ for some $x_0 \in \mathbb{F}_p$ in $\tilde{O}(\sqrt{p})$ arithmetic cost. Finally, again, if we choose $N, x_0 \in \mathbb{N}$, then the accumulating remainder tree allows to deduce $C_{p-1}(x_0) \mod p$ for all primes p between 2 and N quasi-optimally in $\tilde{O}(N)$ bit operations.

4.5. q-hypergeometric creative telescoping

In the case of differential and difference hypergeometric creative telescoping, it was demonstrated in [35] 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 [119, §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-difference case, Koornwinder's q-Gosper algorithm [97, §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 [97], along the same lines as in the differential and difference cases [35].

5. Experiments

Algorithms 1 and 2 were implemented in Magma and Algorithm 3 in Maple. All implementations deliver 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 [110] 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 $N \in \mathbb{N}$, whereas Algorithm 3 finds $M(q^{N-1}) \cdots M(q)M(1) \in \mathbb{K}^{n \times n}$ for a

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 1 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.

given polynomial matrix $M(x) \in \mathbb{K}[x]$ of size $n \times n$ and $q \in \mathbb{K}, N \in \mathbb{N}$. In our experiments, $\mathbb{K}$ is the finite field $\mathbb{F}_p$ with $p = 2^{30} + 3$ elements.

Timings for Algorithms 1 and 2 are presented 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: as N is increased from 2^{2k} to 2^{2k+2}, timings are also 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.

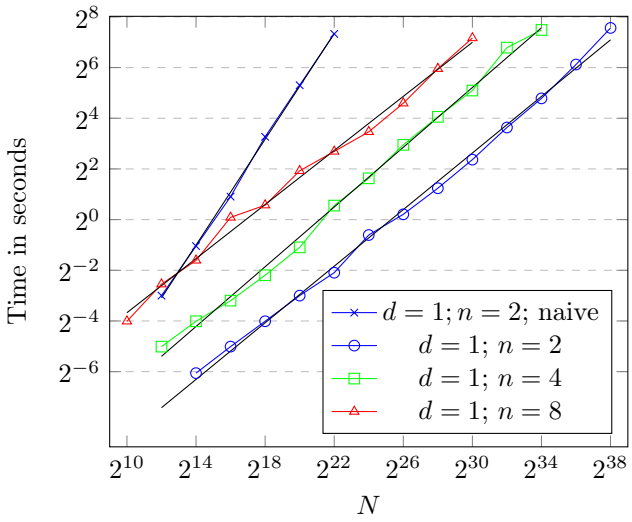

Figure 1: Timings of Algorithm 3 implemented in Maple 2020.2. We compare $d = 1$ and $n = 2, 4, 8$ for various values of N.

In order to visualize the performance of Algorithm 3, we took random polynomial matrices in $\mathcal{M}_n(\mathbb{F}_p[x])$ of degree d. Figure 1 compares the time needed to compute the Matrix q-factorial for $d = 1$ and $n = 2, 4, 8$ as N growths. The black lines represent the best linear fits to the data points and are given by the equations $y = 0.56x - 14.1, y = 0.59x - 12.5$ and $y = 0.53x - 9.0$ respectively. Note that we are plotting on a log-log scale, therefore the established complexity of $\tilde{O}(N^{1/2})$ is indicated by the coefficient of x in these linear fits which is always only slightly greater than $1/2$. In the same figure, we also show the timings for $d = 1$ and $n = 2$ of the naive algorithm given by successively computing and multiplying $M(q^i)$ together. The best linear fit almost perfectly describes this data and has a slope of $1.04 \approx 1$, in line with the linear complexity in N.

In Figure 2 we show similar timings, however now for $n = 2$ and $d = 2, 4, 8$. The linear fits to the data are now given by $y = 0.54x - 12.7, y = 0.54x - 11.9$ and $y = 0.56x - 11.5$. Again, they describe the observations very well and the coefficients of the regressions are in line with the proven complexity. We observe that, as expected, the lines have slopes of roughly $1/2$ and are closer together than in the previous figure. The naive method has a slope of $1.02 \approx 1$.

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-difference equations.

Here are some questions that should be investigated in the future.

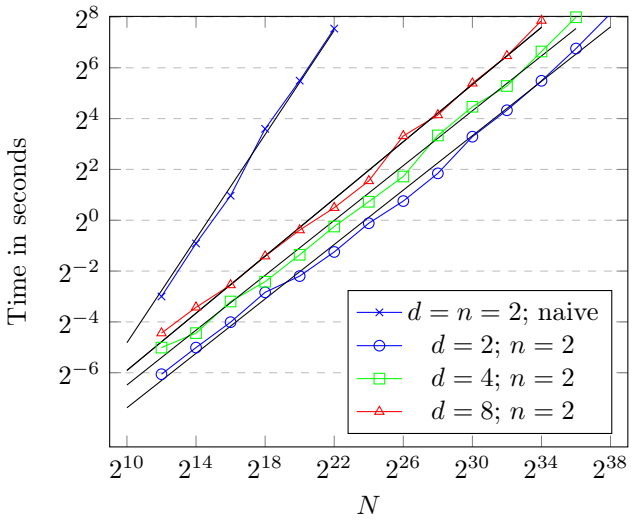

Figure 2: Timings of Algorithm 3 implemented in Maple 2020.2. We compare $n = 2$ and $d = 2, 4, 8$ for various values of N.

1. (Counting points on q-curves) Counting efficiently points on (hyper-)elliptic curves leads to questions like: for $a, b \in \mathbb{Z}$, compute the coefficient of $x^{\frac{p-1}{2}}$ in $G_p(x) := (x^3+ax+b)^{\frac{p-1}{2}}$ modulo p, for one [38] or several [79] primes p. A natural extension is to ask the same with $G_p(x)$ replaced by $\prod_{k=1}^{\frac{p-1}{2}}(q^{3k}x^3 + aq^k x + b)$. This might have applications related to §4.4, or to counting points on q-deformations [128].

2. (Computing q-deformed real numbers) Recently, Morier-Genoud and Ovsienko [108] introduced q-analogues of real numbers, see also [103, 109]. How fast can one compute (truncations or evaluations of) quantized versions of numbers like e or π?

3. (Evaluating more polynomials) Is it possible to evaluate fast polynomials of the form $\sum_{\ell=0}^{N} x^{\ell^s}$, for $s \geq 3$, and many others that escape the q-holonomic class? *E.g.*, [24] presents a beautiful generalization of Algorithm 1 to the fast evaluation of isogenies between elliptic curves, by using *elliptic resultants*, with applications to isogeny-based cryptography.

4. ("Precise" complexity of q-holonomic sequences) Can one prove non-trivial lower bounds, ideally matching the upper bounds, on examples treated in this article?

Acknowledgements. We would like to thank Luca De Feo for his initial question, who motivated this work, and for the very interesting subsequent discussions. Our warm thanks go to Lucia Di Vizio for valuable comments that led to §4.4. Moreover, we thank her and Kilian Raschel for their careful reading of the first version of this manuscript. The first author was supported in part by DeRerumNatura ANR-19-CE40-0018 and the second author by the Austrian Science Fund (FWF) P-31338.

References

[1] S. Abramov, M. Bronstein, M. Petkovšek, On polynomial solutions of linear operator equations, in: ISSAC'95, ACM Press, 1995, pp. 290–296.

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

[3] S. A. Abramov, A direct algorithm to compute rational solutions of first order linear q-difference systems, Discrete Math. 246 (1-3) (2002) 3–12.
URL https://doi.org/10.1016/S0012-365X(01)00248-5

[4] S. A. Abramov, P. Paule, M. Petkovšek, q-hypergeometric solutions of q-difference equations, Discrete Math. 180 (1-3) (1998) 3–22.
URL https://doi.org/10.1016/S0012-365X(97)00106-4

[5] S. A. Abramov, E. V. Zima, D'Alembertian solutions of inhomogeneous linear equations (differential, difference, and some other), in: ISSAC'96, ACM Press, 1996, p. 232–240.
URL https://doi.org/10.1145/236869.237080

[6] B. Adamczewski, J. P. Bell, E. Delaygue, F. Jouhet, Congruences modulo cyclotomic polynomials and algebraic independence for q-series, Sém. Lothar. Combin. 78B (2017) Art. 54, 12.

[7] C. R. Adams, On the linear ordinary q-difference equation, Ann. of Math. (2) 30 (1-4) (1928/29) 195–205.
URL https://doi.org/10.2307/1968274

[8] C. R. Adams, Linear q-difference equations, Bull. AMS 37 (6) (1931) 361–400.
URL https://doi.org/10.1090/S0002-9904-1931-05162-4

[9] W. A. Al-Salam, L. Carlitz, Some orthogonal q-polynomials, Math. Nachr. 30 (1965) 47–61.
URL https://doi.org/10.1002/mana.19650300105

[10] M. Aldaz, G. Matera, J. L. Montaña, L. M. Pardo, A new method to obtain lower bounds for polynomial evaluation, Theoret. Comput. Sci. 259 (1-2) (2001) 577–596.
URL https://doi.org/10.1016/S0304-3975(00)00149-3

[11] J. Alman, V. Vassilevska Williams, A refined laser method and faster matrix multiplication, arXiv preprint, 29 pages (2020).
URL https://arxiv.org/abs/2010.05846

[12] G. E. Andrews, Partition identities, Advances in Math. 9 (1972) 10–51.
URL https://doi.org/10.1016/0001-8708(72)90028-X

[13] G. E. Andrews, A general theory of identities of the Rogers-Ramanujan type, Bull. AMS 80 (1974) 1033–1052.
URL https://doi.org/10.1090/S0002-9904-1974-13616-5

[14] G. E. Andrews, The theory of partitions, Addison-Wesley Publishing Co., Reading,, 1976, encyclopedia of Mathematics and its Applications, Vol. 2.

[15] G. E. Andrews, The fifth and seventh order mock theta functions, Trans. Amer. Math. Soc. 293 (1) (1986) 113–134.
URL https://doi.org/10.2307/2000275

[16] G. E. Andrews, q-Catalan identities, in: The legacy of Alladi Ramakrishnan in the mathematical sciences, Springer, 2010, pp. 183–190.
URL https://doi.org/10.1007/978-1-4419-6263-8_10

[17] C. E. Arreche, Y. Zhang, Computing differential Galois groups of second-order linear q-difference equations, preprint, 2020.

[18] R. Askey, Continuous q-Hermite polynomials when $q > 1$, in: q-series and partitions, vol. 18 of IMA Vol. Math. Appl., Springer, 1989, pp. 151–158.
URL https://doi.org/10.1007/978-1-4684-0637-5_12

[19] D. Bar-Natan, S. Garoufalidis, On the Melvin-Morton-Rozansky conjecture, Invent. Math. 125 (1) (1996) 103–133.
URL https://doi.org/10.1007/s002220050070

[20] M. Barkatou, T. Cluzeau, A. El Hajj, Simple forms and rational solutions of pseudo-linear systems, in: ISSAC'19, ACM Press, 2019, pp. 26–33.
URL https://doi.org/10.1145/3326229.3326246

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

[22] R. Bellman, A brief introduction to theta functions, Athena Series: Selected Topics in Mathematics, Holt, Rinehart and Winston, 1961.
URL https://doi.org/10.1017/s0025557200044491

[23] D. J. Bernstein, Fast multiplication and its applications, in: Algorithmic number theory: lattices, number fields, curves and cryptography, vol. 44 of Math. Sci. Res. Inst. Publ., Cambridge Univ. Press, 2008, pp. 325–384.
[24] D. J. Bernstein, L. D. Feo, A. Leroux, B. Smith, Faster computation of isogenies of large prime degree, Cryptology ePrint Archive, Report 2020/341, https://eprint.iacr.org/2020/341, presented to ANTS-XIV (2020).
[25] J.-P. Bézivin, Les suites q-récurrentes linéaires, Compositio Math. 80 (3) (1991) 285–307.
URL http://www.numdam.org/item?id=CM_1991__80_3_285_0
[26] J.-P. Bézivin, Sur les équations fonctionnelles aux q-différences, Aequationes Math. 43 (2-3) (1992) 159–176.
URL https://doi.org/10.1007/BF01835698
[27] L. I. Bluestein, A linear filtering approach to the computation of the discrete Fourier transform, IEEE Trans. Electroacoustics AU-18 (1970) 451–455.
[28] H. Böing, W. Koepf, Algorithms for q-hypergeometric summation in computer algebra, J. Symbolic Comput. 28 (6) (1999) 777–799.
URL https://doi.org/10.1006/jsco.1998.0339
[29] A. Borodin, S. Cook, On the number of additions to compute specific polynomials, SIAM J. Comput. 5 (1) (1976) 146–157.
URL https://doi.org/10.1137/0205013
[30] P. B. Borwein, Padé approximants for the q-elementary functions, Constr. Approx. 4 (4) (1988) 391–402.
URL https://doi.org/10.1007/BF02075469
[31] A. Bostan, Computing the N-th Term of a q-Holonomic Sequence, in: ISSAC'20, ACM Press, 2020, pp. 46–53.
[32] A. Bostan, X. Caruso, E. Schost, A fast algorithm for computing the characteristic polynomial of the p-curvature, in: ISSAC'14, ACM Press, 2014, pp. 59–66.
URL https://doi.org/10.1145/2608628.2608650
[33] A. Bostan, X. Caruso, E. Schost, A fast algorithm for computing the p-curvature, in: ISSAC'15, ACM, New York, 2015, pp. 69–76.
[34] A. Bostan, X. Caruso, E. Schost, Computation of the similarity class of the p-curvature, in: ISSAC'16, ACM Press, 2016, pp. 111–118.
[35] A. Bostan, F. Chyzak, T. Cluzeau, B. Salvy, Low complexity algorithms for linear recurrences, in: ISSAC'06, ACM Press, 2006, pp. 31–38.
URL https://doi.org/10.1145/1145768.1145781
[36] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy, E. Schost, Algorithmes Efficaces en Calcul Formel, Palaiseau, 2017, 686 pages, in French. Printed by CreateSpace. Also available in electronic form.
URL https://hal.archives-ouvertes.fr/AECF/
[37] A. Bostan, T. Cluzeau, B. Salvy, Fast algorithms for polynomial solutions of linear differential equations, in: ISSAC'05, ACM Press, 2005, pp. 45–52.
URL https://doi.org/10.1145/1073884.1073893
[38] A. Bostan, P. Gaudry, E. Schost, Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator, SIAM J. Comput. 36 (6) (2007) 1777–1806.
URL https://doi.org/10.1137/S0097539704443793
[39] A. Bostan, M. Kauers, Automatic classification of restricted lattice walks, in: FPSAC 2009, Discrete Math. Theor. Comput. Sci. Proc., AK, Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2009, pp. 201–215.
[40] A. Bostan, G. Lecerf, É. Schost, Tellegen's principle into practice, in: Proceedings of ISSAC'03, ACM Press, 2003, pp. 37–44.
[41] A. Bostan, R. Mori, 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, 2021.
URL https://specfun.inria.fr/bostan/BoMo21.pdf
[42] A. Bostan, E. Schost, Polynomial evaluation and interpolation on special sets of points, J. Complexity 21 (4) (2005) 420–446.
URL https://doi.org/10.1016/j.jco.2004.09.009
[43] A. Bostan, E. Schost, Fast algorithms for differential equations in positive characteristic, in: ISSAC'09, ACM Press, 2009, pp. 47–54.
URL https://doi.org/10.1145/1576702.1576712
[44] M. Bousquet-Mélou, Convex polyominoes and algebraic languages, J. Phys. A 25 (7) (1992) 1935–

1944.
URL http://stacks.iop.org/0305-4470/25/1935
[45] P. Bürgisser, M. Clausen, M. A. Shokrollahi, Algebraic complexity theory, vol. 315 of Grundlehren der Mathematischen Wissenschaften, Springer, 1997.
URL https://doi.org/10.1007/978-3-662-03338-8
[46] D. G. Cantor, E. Kaltofen, On fast multiplication of polynomials over arbitrary algebras, Acta Inform. 28 (7) (1991) 693–701.
[47] R. D. Carmichael, The General Theory of Linear q-Difference Equations, Amer. J. Math. 34 (2) (1912) 147–168.
URL https://doi.org/10.2307/2369887
[48] P. Cartier, Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger), Astérisque (206) (1992) Exp. No. 746, 3, 41–91, séminaire Bourbaki, Vol. 1991/92.
[49] D. V. Chudnovsky, G. V. Chudnovsky, Approximations and complex multiplication according to Ramanujan, in: Ramanujan revisited (Urbana-Champaign, Ill., 1987), Academic Press, Boston, MA, 1988, pp. 375–472.
[50] F. Chyzak, Gröbner bases, symbolic summation and symbolic integration, in: Gröbner bases and applications (Linz, 1998), vol. 251 of London Math. Soc. Lecture Note Ser., Cambridge Univ. Press, 1998, pp. 32–60.
[51] F. Chyzak, An extension of Zeilberger's fast algorithm to general holonomic functions, Discrete Math. 217 (1-3) (2000) 115–134.
URL https://doi.org/10.1016/S0012-365X(99)00259-9
[52] F. Chyzak, P. Dumas, H. Le, J. Martin, M. M., B. Salvy, Taming apparent singularities via Ore closure, manuscript, 2016.
[53] E. Costa, R. Gerbicz, D. Harvey, A search for Wilson primes, Math. Comp. 83 (290) (2014) 3071–3091.
URL https://doi.org/10.1090/S0025-5718-2014-02800-7
[54] R. Detcherry, S. Garoufalidis, A diagrammatic approach to the AJ conjecture, Math. Ann. 378 (1-2) (2020) 447–484.
URL https://doi.org/10.1007/s00208-020-02028-y
[55] L. Di Vizio, Arithmetic theory of q-difference equations: the q-analogue of Grothendieck-Katz's conjecture on p-curvatures, Invent. Math. 150 (3) (2002) 517–578.
URL https://doi.org/10.1007/s00222-002-0241-z
[56] L. Di Vizio, C. Hardouin, Intrinsic approach to Galois theory of q-difference equations, with the preface to Part IV by Anne Granier, Mem. Amer. Math. Soc. 77 pages. To appear.
URL http://arxiv.org/abs/1002.4839
[57] L. Di Vizio, J.-P. Ramis, J. Sauloy, C. Zhang, Équations aux q-différences, Gaz. Math. (96) (2003) 20–49.
[58] T. Ekedahl, G. van der Geer, Cycle classes on the moduli of K3 surfaces in positive characteristic, Selecta Math. (N.S.) 21 (1) (2015) 245–291.
URL https://doi.org/10.1007/s00029-014-0156-8
[59] S. B. Ekhad, D. Zeilberger, The number of solutions of $X^2 = 0$ in triangular matrices over GF(q), Electron. J. Combin. 3 (1) (1996) Research Paper 2, approx. 2.
URL http://www.combinatorics.org/Volume_3/Abstracts/v3i1r2.html
[60] T. Ernst, A comprehensive treatment of q-calculus, Birkhäuser/Springer, 2012.
URL https://doi.org/10.1007/978-3-0348-0431-8
[61] C. M. Fiduccia, An efficient formula for linear recurrences, SIAM J. Comput. 14 (1) (1985) 106–112.
URL https://doi.org/10.1137/0214007
[62] M. Fürer, Faster integer multiplication, SIAM J. Comput. 39 (3) (2009) 979–1005.
URL https://doi.org/10.1137/070711761
[63] J. Fürlinger, J. Hofbauer, q-Catalan numbers, J. Combin. Theory Ser. A 40 (2) (1985) 248–264.
URL https://doi.org/10.1016/0097-3165(85)90089-5
[64] F. L. Gall, Powers of tensors and fast matrix multiplication, in: ISSAC'14, 2014, pp. 296–303.
[65] S. Garoufalidis, On the characteristic and deformation varieties of a knot, in: Proceedings of the Casson Fest, vol. 7 of Geom. Topol. Monogr., Geom. Topol. Publ., Coventry, 2004, pp. 291–309.
URL https://doi.org/10.2140/gtm.2004.7.291
[66] S. Garoufalidis, The degree of a q-holonomic sequence is a quadratic quasi-polynomial, Electron. J. Combin. 18 (2) (2011) Paper 4, 23.
URL https://doi.org/10.37236/2000

[67] S. Garoufalidis, Quantum knot invariants, Res. Math. Sci. 5 (1) (2018) Paper No. 11, 17.
URL https://doi.org/10.1007/s40687-018-0127-3
[68] S. Garoufalidis, C. Koutschan, Twisting q-holonomic sequences by complex roots of unity, in: ISSAC'12, ACM Press, 2012, pp. 179–186.
URL https://doi.org/10.1145/2442829.2442857
[69] S. Garoufalidis, C. Koutschan, Irreducibility of q-difference operators and the knot 7_4, Algebr. Geom. Topol. 13 (6) (2013) 3261–3286.
URL https://doi.org/10.2140/agt.2013.13.3261
[70] S. Garoufalidis, A. D. Lauda, T. T. Q. Lê, The colored HOMFLYPT function is q-holonomic, Duke Math. J. 167 (3) (2018) 397–447.
URL https://doi.org/10.1215/00127094-2017-0030
[71] S. Garoufalidis, T. T. Q. Lê, The colored Jones function is q-holonomic, Geom. Topol. 9 (2005) 1253–1293.
URL https://doi.org/10.2140/gt.2005.9.1253
[72] S. Garoufalidis, T. T. Q. Lê, A survey of q-holonomic functions, Enseign. Math. 62 (3-4) (2016) 501–525.
URL https://doi.org/10.4171/LEM/62-3/4-7
[73] F. G. Garvan, New fifth and seventh order mock theta function identities, Ann. Comb. 23 (3-4) (2019) 765–783.
URL https://doi.org/10.1007/s00026-019-00438-7
[74] J. von zur Gathen, J. Gerhard, Modern computer algebra, 3rd ed., Cambridge Univ. Press, 2013.
URL https://doi.org/10.1017/CBO9781139856065
[75] C. F. Gauss, Summatio quarundam serierum singularium, Opera, Vol. 2, Göttingen: Gess. d. Wiss. (1863).
[76] I. Gessel, A noncommutative generalization and q-analog of the Lagrange inversion formula, Trans. Amer. Math. Soc. 257 (2) (1980) 455–482.
URL https://doi.org/10.2307/1998307
[77] W. Hahn, Über die höheren Heineschen Reihen und eine einheitliche Theorie der sogenannten speziellen Funktionen, Math. Nachr. 3 (1950) 257–294.
URL https://doi.org/10.1002/mana.19490030502
[78] G. Hanrot, M. Quercia, P. Zimmermann, The middle product algorithm. I, Appl. Algebra Engrg. Comm. Comput. 14 (6) (2004) 415–438.
URL https://doi.org/10.1007/s00200-003-0144-2
[79] D. Harvey, Counting points on hyperelliptic curves in average polynomial time, Ann. of Math. (2) 179 (2) (2014) 783–803.
URL https://doi.org/10.4007/annals.2014.179.2.7
[80] D. Harvey, J. van der Hoeven, Integer multiplication in time $O(n \log n)$, Ann. of Math. To appear.
[81] D. Harvey, J. van der Hoeven, G. Lecerf, Even faster integer multiplication, J. Complexity 36 (2016) 1–30.
URL https://doi.org/10.1016/j.jco.2016.03.001
[82] E. Heine, Untersuchungen über die Reihe $1+\frac{(1-q^\alpha)(1-q^\beta)}{(1-q)(1-q^\gamma)}x+\frac{(1-q^\alpha)(1-q^{\alpha+1})(1-q^\beta)(1-q^{\beta+1})}{(1-q)(1-q^2)(1-q^\gamma)(1-q^{\gamma+1})}x^2+..$, J. reine angew. Math. 34 (1847) 285–328.
[83] J. Heintz, M. Sieveking, Lower bounds for polynomials with algebraic coefficients, Theoret. Comput. Sci. 11 (3) (1980) 321–330.
URL https://doi.org/10.1016/0304-3975(80)90019-5
[84] P. A. Hendriks, An algorithm for computing a standard form for second-order linear q-difference equations, J. Pure Appl. Algebra 117/118 (1997) 331–352.
URL https://doi.org/10.1016/S0022-4049(97)00017-0
[85] J. Hua, Counting representations of quivers over finite fields, J. Algebra 226 (2) (2000) 1011–1033.
URL https://doi.org/10.1006/jabr.1999.8220
[86] M. E. H. Ismail, Lectures on q-orthogonal polynomials, in: Special functions 2000: current perspective and future directions (Tempe, AZ), vol. 30 of NATO Sci. Ser. II Math. Phys. Chem., Kluwer Acad. Publ., 2001, pp. 179–219.
URL https://doi.org/10.1007/978-94-010-0818-1_8
[87] F. H. Jackson, On q-functions and a certain difference operator, Trans. Roy. Soc. Edin. 46 (11) (1909) 253–281.
URL https://doi.org/10.1017/S0080456800002751
[88] F. H. Jackson, On q-definite integrals, Quar. J. Pure Appl. Math. 41 (1910) 193–203.

[89] F. H. Jackson, q-Difference Equations, Amer. J. Math. 32 (4) (1910) 305–314.
URL https://doi.org/10.2307/2370183
[90] N. M. Katz, Algebraic solutions of differential equations (p-curvature and the Hodge filtration), Invent. Math. 18 (1972) 1–118.
URL https://doi.org/10.1007/BF01389714
[91] M. Kauers, C. Koutschan, A Mathematica package for q-holonomic sequences and power series, Ramanujan J. 19 (2) (2009) 137–150.
URL https://doi.org/10.1007/s11139-008-9132-2
[92] D. E. Khmel′ nov, Improved algorithms for solving difference and q-difference equations, Programmirovanie (2) (2000) 70–78.
URL https://doi.org/10.1007/BF02759198
[93] A. A. Kirillov, On the number of solutions of the equation $X^2 = 0$ in triangular matrices over a finite field, Funktsional. Anal. i Prilozhen. 29 (1) (1995) 82–87.
URL https://doi.org/10.1007/BF01077044
[94] A. A. Kirillov, A. Melnikov, On a remarkable sequence of polynomials, in: Algèbre non commutative, groupes quantiques et invariants (Reims, 1995), vol. 2 of Sémin. Congr., Soc. Math. France, Paris, 1997, pp. 35–42.
[95] R. Koekoek, P. A. Lesky, R. F. Swarttouw, Hypergeometric orthogonal polynomials and their q-analogues, Monographs in Mathematics, Springer, 2010.
URL https://doi.org/10.1007/978-3-642-05014-5
[96] W. Koepf, P. M. Rajković, S. D. Marinković, Properties of q-holonomic functions, J. Difference Equ. Appl. 13 (7) (2007) 621–638.
URL https://doi.org/10.1080/10236190701264925
[97] T. H. Koornwinder, On Zeilberger's algorithm and its q-analogue, J. Comput. Appl. Math. 48 (1-2) (1993) 91–111.
URL https://doi.org/10.1016/0377-0427(93)90317-5
[98] C. Koutschan, A fast approach to creative telescoping, Math. Comput. Sci. 4 (2-3) (2010) 259–266.
URL https://doi.org/10.1007/s11786-010-0055-0
[99] C. Koutschan, Y. Zhang, Desingularization in the q-Weyl algebra, Adv. in Appl. Math. 97 (2018) 80–101.
URL https://doi.org/10.1016/j.aam.2018.02.005
[100] H. Labrande, Computing Jacobi's theta in quasi-linear time, Math. Comp. 87 (311) (2018) 1479–1508.
URL https://doi.org/10.1090/mcom/3245
[101] H. Labrande, E. Thomé, Computing theta functions in quasi-linear time in genus two and above, LMS J. Comput. Math. 19 (suppl. A) (2016) 163–177.
URL https://doi.org/10.1112/S1461157016000309
[102] J. Le Caine, The linear q-difference equation of the second order, Amer. J. Math. 65 (1943) 585–600.
URL https://doi.org/10.2307/2371867
[103] B. Le Stum, A. Quirós, On quantum integers and rationals, in: Trends in number theory, vol. 649 of Contemp. Math., Amer. Math. Soc., Providence, RI, 2015, pp. 107–130.
URL https://doi.org/10.1090/conm/649/13022
[104] R. J. Lipton, Polynomials with $0-1$ coefficients that are hard to evaluate, SIAM J. Comput. 7 (1) (1978) 61–69.
URL https://doi.org/10.1137/0207004
[105] J.-C. Liu, Some finite generalizations of Euler's pentagonal number theorem, Czechoslovak Math. J. 67(142) (2) (2017) 525–531.
URL https://doi.org/10.21136/CMJ.2017.0063-16
[106] T. E. Mason, On Properties of the Solutions of Linear q-Difference Equations with Entire Function Coefficients, Amer. J. Math. 37 (4) (1915) 439–444.
URL https://doi.org/10.2307/2370216
[107] J. C. P. Miller, D. J. S. Brown, An algorithm for evaluation of remote terms in a linear recurrence sequence, Comput. J. 9 (1966) 188–190.
[108] S. Morier-Genoud, V. Ovsienko, On q-deformed real numbers, Exp. Math. (2019) 1–9. To appear.
URL https://doi.org/10.1080/10586458.2019.1671922
[109] S. Morier-Genoud, V. Ovsienko, q-deformed rationals and q-continued fractions, Forum Math. Sigma 8 (2020) Paper No. e13, 55.
URL https://doi.org/10.1017/fms.2020.9

[110] D. Nogneng, E. Schost, On the evaluation of some sparse polynomials, Math. Comp. 87 (310) (2018) 893–904.
URL https://doi.org/10.1090/mcom/3231
[111] C. F. Osgood, On the diophantine approximation of values of functions satisfying certain linear q-difference equations, J. Number Theory 3 (1971) 159–177.
URL https://doi.org/10.1016/0022-314X(71)90033-3
[112] A. Ostrowski, On two problems in abstract algebra connected with Horner's rule, in: Studies in mathematics and mechanics presented to Richard von Mises, Academic Press Inc., 1954, pp. 40–48.
[113] I. Pak, Partition bijections, a survey, Ramanujan J. 12 (1) (2006) 5–75.
URL https://doi.org/10.1007/s11139-006-9576-1
[114] V. Y. Pan, Methods of computing values of polynomials, Russian Mathematical Surveys 21 (1) (1966) 105–136.
[115] M. Paterson, L. Stockmeyer, Bounds on the evaluation time for rational polynomial, in: 12th Annual Symposium on Switching and Automata Theory (SWAT 1971), 1971, pp. 140–143.
URL https://doi.org/10.1109/SWAT.1971.5
[116] M. S. Paterson, L. J. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials, SIAM J. Comput. 2 (1973) 60–66.
URL https://doi.org/10.1137/0202007
[117] P. Paule, S. Radu, Rogers-Ramanujan functions, modular functions, and computer algebra, in: Advances in computer algebra, vol. 226 of Springer Proc. Math. Stat., Springer, Cham, 2018, pp. 229–280.
URL https://doi.org/10.1007/978-3-319-73232-9_10
[118] P. Paule, 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, vol. 14 of Fields Inst. Commun., Amer. Math. Soc., 1997, pp. 179–210.
[119] M. Petkovšek, H. S. Wilf, D. Zeilberger, $A = B$, A K Peters, 1996.
[120] J. M. Pollard, Theorems on factorization and primality testing, Proc. Cambridge Philos. Soc. 76 (1974) 521–528.
URL https://doi.org/10.1017/s0305004100049252
[121] L. R. Rabiner, R. W. Schafer, C. M. Rader, The chirp z-transform algorithm and its application, Bell System Tech. J. 48 (1969) 1249–1292.
URL https://doi.org/10.1002/j.1538-7305.1969.tb04268.x
[122] A. Riese, qMultiSum—a package for proving q-hypergeometric multiple summation identities, J. Symbolic Comput. 35 (3) (2003) 349–376.
URL https://doi.org/10.1016/S0747-7171(02)00138-4
[123] L. Rogers, S. Ramanujan, Proof of certain identities in combinatory analysis, Proc. Cambridge Philos. Soc. 19 (1919) 211–214.
[124] L. J. Rogers, Second Memoir on the Expansion of certain Infinite Products, Proc. Lond. Math. Soc. 25 (1893/94) 318–343.
URL https://doi.org/10.1112/plms/s1-25.1.318
[125] H. A. Rothe, Formulae de serierum reversione demonstratio universalis signis localibus combinatorico-analyticorum vicariis exhibita, Leipzig (1793).
[126] C. Sabbah, Systèmes holonomes d'équations aux q-différences, in: D-modules and microlocal geometry (Lisbon, 1990), de Gruyter, 1993, pp. 125–147.
[127] C.-P. Schnorr, Improved lower bounds on the number of multiplications / divisions which are necessary to evaluate polynomials, Theoret. Comput. Sci. 7 (3) (1978) 251–261.
URL https://doi.org/10.1016/0304-3975(78)90016-6
[128] P. Scholze, Canonical q-deformations in arithmetic geometry, Ann. Fac. Sci. Toulouse Math. (6) 26 (5) (2017) 1163–1192.
URL https://doi.org/10.5802/afst.1563
[129] A. Schönhage, Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2, Acta Informatica 7 (1977) 395–398.
[130] A. Schönhage, V. Strassen, Schnelle Multiplikation grosser Zahlen, Computing (Arch. Elektron. Rechnen) 7 (1971) 281–292.
URL https://doi.org/10.1007/bf02242355
[131] D. Shanks, A short proof of an identity of Euler, Proc. AMS 2 (1951) 747–749.
URL https://doi.org/10.2307/2032076
[132] T. Sprenger, W. Koepf, Algorithmic determination of q-power series for q-holonomic functions, J. Symbolic Comput. 47 (5) (2012) 519–535.

URL https://doi.org/10.1016/j.jsc.2011.12.004
[133] V. Strassen, Polynomials with rational coefficients which are hard to compute, SIAM J. Comput. 3 (1974) 128–149.
URL https://doi.org/10.1137/0203010
[134] V. Strassen, Einige Resultate über Berechnungskomplexität, Jber. Deutsch. Math.-Verein. 78 (1) (1976/77) 1–8.
[135] H. W. L. Tanner, On the Enumeration of Groups of Totitives, Proc. Lond. Math. Soc. 27 (1895/96) 329–352.
URL https://doi.org/10.1112/plms/s1-27.1.329
[136] T. Tao, E. Croot, III, H. Helfgott, Deterministic methods to find primes, Math. Comp. 81 (278) (2012) 1233–1246.
URL https://doi.org/10.1090/S0025-5718-2011-02542-1
[137] W. J. Trjitzinsky, Analytic theory of linear q-difference equations, Acta Math. 61 (1) (1933) 1–38.
URL https://doi.org/10.1007/BF02547785
[138] H. Tsai, Weyl closure of a linear differential operator, vol. 29, 2000, pp. 747–775, Symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998).
URL https://doi.org/10.1006/jsco.1999.0400
[139] M. van der Put, M. F. Singer, Galois theory of linear differential equations, vol. 328 of Grundlehren der Mathematischen Wissenschaften, Springer, 2003.
URL https://doi.org/10.1007/978-3-642-55750-7
[140] H. S. Wilf, D. Zeilberger, An algorithmic proof theory for hypergeometric (ordinary & q) multisum/integral identities, Invent. Math. 108 (3) (1992) 575–633.
URL https://doi.org/10.1007/BF02100618
[141] 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) (1990) 148–151.
[142] M. Yip, Rook placements and Jordan forms of upper-triangular nilpotent matrices, Electron. J. Combin. 25 (1) (2018) Paper 1.68, 25.
[143] D. Zagier, Elliptic modular forms and their applications, in: The 1-2-3 of modular forms, Universitext, Springer, 2008, pp. 1–103.
URL https://doi.org/10.1007/978-3-540-74119-0_1
[144] D. Zeilberger, A holonomic systems approach to special functions identities, J. Comput. Appl. Math. 32 (3) (1990) 321–368.
URL https://doi.org/10.1016/0377-0427(90)90042-X