Computing the N-th term of a q-holonomic sequence[1,2]

Sergey Yurkevich

Inria and University of Vienna

Friday 26th November, 2021

[1] Joint work with Alin Bostan, arxiv.org/abs/2012.08656

[2] Slides available at homepage.univie.ac.at/sergey.yurkevich/data/Nthqhol_slides.pdf

Problem statement

Given a sequence $(u_n)_{n\geq 0}$ and $N \in \mathbb{N}$, we want to compute u_N as fast as possible.

Problem statement

Given a sequence $(u_n)_{n\geq 0}$ and $N \in \mathbb{N}$, we want to compute u_N as fast as possible.

- u_n lie in some field $\mathbb{K}$.

Problem statement

Given a sequence $(u_n)_{n\geq 0}$ and $N \in \mathbb{N}$, we want to compute u_N as fast as possible.

- u_n lie in some field $\mathbb{K}$.
- The sequence is given by some *recurrence relation* and initial conditions.

Problem statement

Given a sequence $(u_n)_{n\geq 0}$ and $N \in \mathbb{N}$, we want to compute u_N as fast as possible.

- u_n lie in some field $\mathbb{K}$.
- The sequence is given by some *recurrence relation* and initial conditions.
- By "fast" we mean with as few *arithmetic operations* in $\mathbb{K}$ as possible.

Problem statement

Given a sequence $(u_n)_{n\geq 0}$ and $N \in \mathbb{N}$, we want to compute u_N as fast as possible.

- u_n lie in some field $\mathbb{K}$.
- The sequence is given by some *recurrence relation* and initial conditions.
- By "fast" we mean with as few *arithmetic operations* in $\mathbb{K}$ as possible.
- Tremendous number of applications:
  - Algebraic complexity theory (e.g., evaluation of polynomials [Strassen, 1977])
  - Computations on real numbers (e.g., constants approximation [Chudnovsky^2, 1987])
  - Algorithmic number theory (e.g., Wilson primes search [Costa,Gerbicz,Harvey, 2014])
  - Effective algebraic geometry (e.g., counting points on curves [Harvey, 2014])
  - etc.

Holonomic (aka P-recursive) sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called *holonomic* if it satisfies a linear recurrence relation with polynomial coefficients:

$$c_r(n)u_{n+r} + \cdots + c_0(n)u_n = 0 \qquad n \geq 0.$$

Holonomic (aka P-recursive) sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called *holonomic* if it satisfies a linear recurrence relation with polynomial coefficients:

$$c_r(n)u_{n+r} + \cdots + c_0(n)u_n = 0 \qquad n \geq 0.$$

- Examples:
  - $u_n = q^n$ satisfies $u_{n+1} - qu_n = 0$;
  - $u_n = n!$ satisfies $u_{n+1} - (n+1)u_n = 0$;
  - $u_n = \sum_{k=0}^{n} \binom{n}{k}^2 \binom{n+k}{k}$ satisfies $(n+2)^2 u_{n+2} - (11n^2 + 33n + 25)u_{n+1} - (n+1)^2 u_n = 0$.

Holonomic (aka P-recursive) sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called *holonomic* if it satisfies a linear recurrence relation with polynomial coefficients:

$$c_r(n)u_{n+r} + \cdots + c_0(n)u_n = 0 \qquad n \geq 0.$$

- Examples:
  - $u_n = q^n$ satisfies $u_{n+1} - qu_n = 0$;
  - $u_n = n!$ satisfies $u_{n+1} - (n+1)u_n = 0$;
  - $u_n = \sum_{k=0}^{n} \binom{n}{k}^2 \binom{n+k}{k}$ satisfies $(n+2)^2 u_{n+2} - (11n^2+33n+25)u_{n+1} - (n+1)^2 u_n = 0$.
- Given $N \in \mathbb{N}$, one can compute u_N in $\tilde{O}(\sqrt{N})$ arithmetic operations [Strassen, 1977], [Chudnovsky^2, 1988]. Naive: $O(N)$

q-holonomic sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called q-*holonomic* if for some $q \in \mathbb{K}$ it satisfies a linear q-recurrence relation with polynomial coefficients:

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

q-holonomic sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called *q-holonomic* if for some $q \in \mathbb{K}$ it satisfies a linear q-recurrence relation with polynomial coefficients:

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

- Examples:
  - $u_n = q^n$ satisfies $u_{n+1} - qu_n = 0$;
  - $u_n = [n]_q! = (1+q)\cdots(1+q+\cdots+q^{n-1})$ satisfies $(q-1)u_{n+1} - (q^{n+1}-1)u_n = 0$;
  - $u_n = \sum_{k=0}^{n} 2^{k^2}$ satisfies $u_{n+2} - (2^{2n+1}+1)u_{n+1} + 2^{2n-1}u_n = 0$.

q-holonomic sequences

- A sequence $(u_n)_{n\geq 0} \in \mathbb{K}$ is called *q-holonomic* if for some $q \in \mathbb{K}$ it satisfies a linear q-recurrence relation with polynomial coefficients:

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

- Examples:
  - $u_n = q^n$ satisfies $u_{n+1} - qu_n = 0$;
  - $u_n = [n]_q! = (1+q)\cdots(1+q+\cdots+q^{n-1})$ satisfies $(q-1)u_{n+1} - (q^{n+1}-1)u_n = 0$;
  - $u_n = \sum_{k=0}^{n} 2^{k^2}$ satisfies $u_{n+2} - (2^{2n+1}+1)u_{n+1} + 2^{2n-1}u_n = 0$.
- Given $N \in \mathbb{N}$, one can compute u_N in $\tilde{O}(\sqrt{N})$ arithmetic operations [Bostan, Y., 2020]. Naive: $O(N)$

(Arithmetic) complexity basics

- Arithmetic complexity means we count base operations $(+, -, \times, \div)$ in $\mathbb{K}$ at unit cost. Hence, in practice $\mathbb{K}$ is a finite field.

(Arithmetic) complexity basics

- Arithmetic complexity means we count base operations $(+, -, \times, \div)$ in $\mathbb{K}$ at unit cost. Hence, in practice $\mathbb{K}$ is a finite field.
- $O(\cdot)$ stands for the big-Oh notation and $\tilde{O}(\cdot)$ is used to hide polylogarithmic factors in the argument.

(Arithmetic) complexity basics

- Arithmetic complexity means we count base operations $(+, -, \times, \div)$ in $\mathbb{K}$ at unit cost. Hence, in practice $\mathbb{K}$ is a finite field.
- $O(\cdot)$ stands for the big-Oh notation and $\tilde{O}(\cdot)$ is used to hide polylogarithmic factors in the argument.
- $\mathbf{M}(d)$ is the cost of multiplication of two polynomials in $\mathbb{K}[x]$ of degree d. It is known that $\mathbf{M}(d) = O(d \log d \log\log d) = \tilde{O}(d)$. (Using FFT) Naive: $O(d^2)$
- Given $P(x) \in \mathbb{K}[x]$ of degree d, one can evaluate $P(x)$ at $q, q^2, \ldots, q^d \in \mathbb{K}$ simultaneously in complexity $O(\mathbf{M}(d))$. (Using Bluestein's trick) Naive: $O(d^2)$
- Two matrices in $\mathbb{K}^{n\times n}$ can be multiplied in complexity $O(n^\omega)$, where the best current bound is $\omega < 2.3729$. Naive: $O(n^3)$

Main theorem

Theorem (Bostan, Y., 2020)

Let $q \in \mathbb{K} \setminus \{1\}$ and $N \in \mathbb{N}$. Let $(u_n)_{n \geq 0}$ be a q-holonomic sequence satisfying the recurrence

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

and assume that $c_r(q, q^k)$ is nonzero for $k = 0, \ldots, N$. Then, u_N can be computed in $O(\mathbf{M}(\sqrt{N})) = \tilde{O}(\sqrt{N})$ operations in $\mathbb{K}$. *Naive: $O(N)$*

Main theorem

Theorem (Bostan, Y., 2020)

Let $q \in \mathbb{K} \setminus \{1\}$ and $N \in \mathbb{N}$. Let $(u_n)_{n \geq 0}$ be a q-holonomic sequence satisfying the recurrence

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

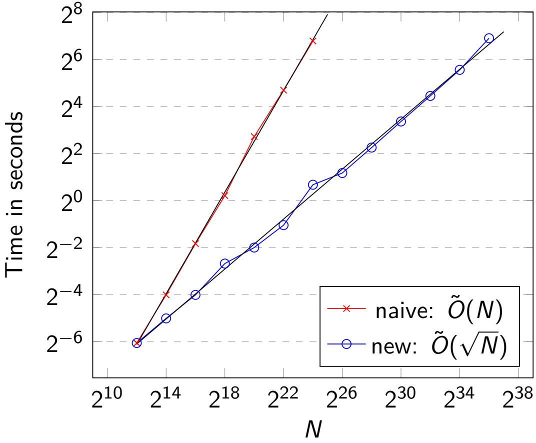
and assume that $c_r(q, q^k)$ is nonzero for $k = 0, \ldots, N$. Then, u_N can be computed in $O(\mathbf{M}(\sqrt{N})) = \tilde{O}(\sqrt{N})$ operations in $\mathbb{K}$. Naive: $O(N)$

Theorem (Bostan, Y. 2020)

Under the assumptions of the theorem above, 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 can be computed in $O(r^\omega \sqrt{Nd} + r^2\, \mathbf{M}(\sqrt{Nd}))$ operations in $\mathbb{K}$.

Timings

Computing the N-th term of $u_n = \sum_{k=0}^{n} q^{k^2} \in \mathbb{F}_p$, where $p = 2^{50} + 55$ is prime and $q \in \mathbb{F}_p$ randomly chosen.



An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.

An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.
- Generically, Horner's rule needs $O(\deg P)$ operations.

An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.
- Generically, Horner's rule needs $O(\deg P)$ operations.
- Our results imply that one can do better for large families of polynomials.

An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.
- Generically, Horner's rule needs $O(\deg P)$ operations.
- Our results imply that one can do better for large families of polynomials.
- [Nogneng, Schost, 2018]: The truncated Jacobi theta function

$$\vartheta_N(x) := 1 + x + x^4 + x^9 + \cdots + x^{N^2}$$

can be evaluated at $q \in \mathbb{K}$ in $\tilde{O}(\sqrt{N})$ arithmetic operations.

An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.
- Generically, Horner's rule needs $O(\deg P)$ operations.
- Our results imply that one can do better for large families of polynomials.
- [Nogneng, Schost, 2018]: The truncated Jacobi theta function

$$\vartheta_N(x) := 1 + x + x^4 + x^9 + \cdots + x^{N^2}$$

  can be evaluated at $q \in \mathbb{K}$ in $\tilde{O}(\sqrt{N})$ arithmetic operations.
- Also follows from our result: $\vartheta_N(q) = u_N$, where $u_n = \sum_{k=0}^{n} q^{k^2}$ is q-holonomic.

An application: evaluation of polynomials

- Task: Given a polynomial $P(x) \in \mathbb{K}[x]$ and $q \in \mathbb{K}$, deduce $P(q) \in \mathbb{K}$ fast.
- Generically, Horner's rule needs $O(\deg P)$ operations.
- Our results imply that one can do better for large families of polynomials.
- [Nogneng, Schost, 2018]: The truncated Jacobi theta function

$$\vartheta_N(x) := 1 + x + x^4 + x^9 + \cdots + x^{N^2}$$

  can be evaluated at $q \in \mathbb{K}$ in $\tilde{O}(\sqrt{N})$ arithmetic operations.
- Also follows from our result: $\vartheta_N(q) = u_N$, where $u_n = \sum_{k=0}^{n} q^{k^2}$ is q-holonomic.
- Same complexity and reasoning for $\prod_{i=0}^{N}(x - a^i)$, or q-Hermite polynomials, or $\prod_{i=1}^{\infty}(1 - x^i)^3 \bmod x^n$, etc.

Idea of the proof

Note that

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

can be translated into a first-order matrix-vector recurrence

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

Idea of the proof

Note that

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

can be translated into a first-order matrix-vector recurrence

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

Hence, u_N can be easily expressed in terms of the matrix q-factorial

$$M(q^{N-1}) \cdots M(q)M(1) \in \mathbb{K}^{r \times r}.$$

Idea of the proof

Note that

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

can be translated into a first-order matrix-vector recurrence

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

Hence, u_N can be easily expressed in terms of the matrix q-factorial

$$M(q^{N-1}) \cdots M(q)M(1) \in \mathbb{K}^{r \times r}.$$

$\Rightarrow$ New problem: Given $M(x) \in \mathbb{K}[x]^{r \times r}$, compute $M(q^{N-1}) \cdots M(q)M(1)$ fast.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r \times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r\times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r \times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $N = s^2$

(1) (Baby-step) Compute $q, q^2, \dots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x) \cdots M(qx)M(x)$.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r\times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $\quad N = s^2$

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x)\cdots M(qx)M(x)$. Divide-and-Conquer $\Rightarrow O(\mathbf{M}(s))$

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r\times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $N = s^2$

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x)\cdots M(qx)M(x)$. Divide-and-Conquer $\Rightarrow O(\mathbf{M}(s))$

(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \ldots, Q^{s-1}$.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r\times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $N = s^2$

(1) (Baby-step) Compute $q, q^2, \dots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x)\cdots M(qx)M(x)$. Divide-and-Conquer $\Rightarrow O(\mathbf{M}(s))$

(2) (Giant-step) Compute $Q := q^s, Q^2, \dots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \dots, Q^{s-1}$. Bluestein's trick $\Rightarrow O(\mathbf{M}(s))$

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r\times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $N = s^2$

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x)\cdots M(qx)M(x)$. Divide-and-Conquer $\Rightarrow O(\mathbf{M}(s))$

(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \ldots, Q^{s-1}$. Bluestein's trick $\Rightarrow O(\mathbf{M}(s))$

(3) Return the product $P(Q^{s-1})\cdots P(Q)P(1)$.

Matrix q-factorial with baby-step/giant-step

Task: Given $M(x) \in \mathbb{K}[x]^{r \times r}$ and $N \in \mathbb{N}$, compute

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

in $O(\mathbf{M}(\sqrt{N}))$ arithmetic operations.

Note: The naive algorithm has $O(N)$ complexity. Assume that $N = s^2$ for $s \in \mathbb{N}$.

Main algorithm (matrix q-factorial) $N = s^2$

(1) (Baby-step) Compute $q, q^2, \ldots, q^{s-1}$; deduce the coefficients of the polynomial matrix $P(x) := M(q^{s-1}x) \cdots M(qx)M(x)$. Divide-and-Conquer $\Rightarrow O(\mathbf{M}(s))$

(2) (Giant-step) Compute $Q := q^s, Q^2, \ldots, Q^{s-1}$, and evaluate (the entries of) $P(x)$ simultaneously at $1, Q, \ldots, Q^{s-1}$. Bluestein's trick $\Rightarrow O(\mathbf{M}(s))$

(3) Return the product $P(Q^{s-1}) \cdots P(Q)P(1)$. $O(s)$

Main takeaways

- The fast computation of the N-th term in a sequence has important consequences and many applications.
- Given a q-holonomic sequence, we can compute its N-th term faster than naively: $O(\mathbf{M}(\sqrt{N})) = \tilde{O}(\sqrt{N})$ instead of $O(N)$.

$\mathbb{K} = \mathbb{Q}$: Bit complexity

- If q is an integer, the arithmetic complexity model is replaced by the bit-complexity model.
- $\mathbf{M}_{\mathbb{Z}}(n)$ denotes the cost of multiplication of two integers of bitsize n.
- It is now known that $\mathbf{M}_{\mathbb{Z}}(n) = O(n \log n) = \tilde{O}(n)$ [Harvey, van der Hoeven].
- Let B be the bitsize of q and $(u_n)_{n \geq 0}$ q-holonomic. Naively, u_N can be computed in $\tilde{O}(N^3 B)$. We can do better (using binary splitting):

Theorem (Bostan, Y. 2020)

Let $q \in \mathbb{Q} \setminus \{1\}$ and $N \in \mathbb{N}$. Let $(u_n)_{n \geq 0}$ be a q-holonomic sequence satisfying the recurrence

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

and assume that $c_r(q, q^k)$ is nonzero for $k = 0, \ldots, N$. The term u_N can be computed in $\tilde{O}(N^2 B)$ bit operations, where B is the bitsize of q.

Computation of several terms

Theorem (Bostan, Y. 2020)

Under the assumptions of the main theorem, let $N_1 < N_2 < \cdots < N_s = N$ be positive integers, where $s \leq \sqrt{N}$. Then, the terms $u_{N_1}, \ldots, u_{N_s}$ can be computed altogether in $O(\mathbf{M}(\sqrt{N}) \log N)$ operations in $\mathbb{K}$.