Low Complexity Algorithms for Linear Recurrences

A. Bostan, F. Chyzak, B. Salvy
Algorithms Project, Inria Rocquencourt
78153 Le Chesnay (France)

T. Cluzeau
Café Project, Inria Sophia Antipolis
06902 Sophia Antipolis (France)

{Alin.Bostan,Frederic.Chyzak,Thomas.Cluzeau,Bruno.Salvy}@inria.fr

ABSTRACT

We consider two kinds of problems: the computation of polynomial and rational solutions of linear recurrences with coefficients that are polynomials with integer coefficients; indefinite and definite summation of sequences that are hypergeometric over the rational numbers. The algorithms for these tasks all involve as an intermediate quantity an integer N (dispersion or root of an indicial polynomial) that is potentially exponential in the bit size of their input. Previous algorithms have a bit complexity that is at least quadratic in N. We revisit them and propose variants that exploit the structure of solutions and avoid expanding polynomials of degree N. We give two algorithms: a probabilistic one that detects the existence or absence of nonzero polynomial and rational solutions in $\mathcal{O}(\sqrt{N}\log^2 N)$ bit operations; a deterministic one that computes a compact representation of the solution in $\mathcal{O}(N\log^3 N)$ bit operations. Similar speedups are obtained in indefinite and definite hypergeometric summation. We describe the results of an implementation.

Categories and Subject Descriptors: I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms

General Terms: Algorithms, Experimentation, Theory

Keywords: Computer algebra, polynomial and rational solutions, linear recurrences, summation, creative telescoping, complexity

1. INTRODUCTION

A central quantity for many algorithms operating on linear recurrences and their solutions is the dispersion.

Definition 1. *The* dispersion set *of two polynomials P and Q in $\mathbb{Q}[n]$ is the set of positive integer roots of the resultant $R(h) = \mathrm{Res}_n(P(n), Q(n+h))$. Its maximal element is called their* dispersion.

Thus, the dispersion is the largest integer difference between roots of P and Q. As shown by the simple example $(P, Q) = (n, n-N)$ with $N \in \mathbb{N}$, the dispersion can be exponentially large in the bit size of the input polynomials. It cannot get much worse: when the polynomials have integer coefficients whose absolute value is bounded by B, their dispersion is bounded by $4B$ [9, Fact 7.11]. This exponential size yields the dominant term in the worst-case complexity of many algorithms computing — or operating on — solutions of linear recurrences.

For instance, the computation of a Gosper form produces a polynomial whose degree N can be linear in the dispersion of its input and thus exponential in its bit size. If this polynomial is expanded it has $N+1$ coefficients; over the integers, its total bit size is $\mathcal{O}(N^2 \log N)$. This form is used in the first step of Gosper's summation algorithm and of Abramov's algorithm for computing rational solutions of linear recurrences. Thus, it makes an important contribution to the complexity of these algorithms. Once this form is computed, these algorithms search for polynomial solutions of an associated linear recurrence. This is done by linear algebra using a bound on the possible degree of solutions which is at least as large as N, leading again to a more than quadratic complexity, even when no nonzero solution exists. In turn, a parameterized variant of Gosper's algorithm forms the basis of Zeilberger's definite summation algorithm which inherits this costly behaviour. By contrast, we provide a probabilistic algorithm that detects that no nonzero rational solution of a homogeneous linear recurrence exists in $\mathcal{O}(\sqrt{N}\log^2 N)$ bit operations and a deterministic algorithm that gives a compact representation of all solutions in $\mathcal{O}(N\log^3 N)$ bit operations. All the algorithms in the present work eventually rely on the computation of polynomial solutions of linear recurrences. In a previous work [6], we dealt with the analogous problem in the linear differential case, by exploiting the linear recurrence satisfied by the coefficients of power series solutions and reducing the computation to that of *matrix factorials*. For the latter operation, there exist fast probabilistic and deterministic algorithms (see [7, 8] and the references in [6]). In the case of linear recurrences, it is not true that the coefficients of polynomial solutions satisfy a linear recurrence in general; however, it becomes true if the polynomials are expanded in a binomial basis [4, Ch. XIII, art. 5]. This is the basis of a simple quadratic algorithm [3] to compute polynomial solutions. In Section 2, we show how this conversion is performed, we recall the basic results on matrix factorials and apply them to get the announced complexities.

From there, in Section 3, we proceed in three steps: *(i)* we slightly modify the computation of the Gosper form so that

ISSAC'06 July 9–12, 2005, Genoa, Italy.

it does not expand the potentially large polynomial but instead computes a first-order, moderately-sized recurrence for it; *(ii)* we show that this first-order recurrence can be used to compute a linear recurrence satisfied by the numerators of rational solutions, in a complexity that is only logarithmic in N, both in the homogeneous and nonhomogeneous cases; *(iii)* we then compute the numerators as polynomial solutions via matrix factorials. The close relation between Abramov's and Gosper's algorithm makes it possible to transfer these results to Gosper's algorithm. Then in Section 4, we show how this machinery can be adapted to the parameterized variant needed in Zeilberger's algorithm. Finally, we describe experimental results in Section 5.

Notations and complexity measures. All along this text, $\mathcal{R}$ denotes a linear difference operator with coefficients in $\mathbb{Z}[n]$. We view it as a non-commutative polynomial in n and S_n, where S_n is the shift operator $S_n u(n) = u(n+1)$. Similarly, S_x, S_k, and S_m denote the shifts with respect to x, k, and m. To any difference operator $\mathcal{R}$ is attached a homogeneous linear recurrence equation $\mathcal{R}u = 0$. We view the solution u either as a sequence (u_n) (also denoted u_n), or as a function $u(n)$ (the cases of particular interest being polynomial and rational functions).

For our complexity analyses, the measure we us is the bit (or boolean) complexity. For this purpose, our complexity model is the multi-tape Turing machine, see for instance [15]. We use the *number of bit operations* to express time complexities in this model. We call *bit size* (or simply *size*) of an integer $a \neq 0$ the number $\lambda(a) := \lfloor \log|a| \rfloor + 1$ ($\log x$ denotes the logarithm of x in base 2). By convention, we assume that $\lambda(0) = 1$. The bit size of a matrix or vector is the sum of those of its entries. Polynomials given as input to our algorithms are stored in a dense representation; a measure of their bit size is given by the sum of the bit size of their coefficients, including the zero coefficients. Similarly, the bit size of a linear recurrence equation (LRE) is the sum of the bit sizes of its coefficients.

To simplify complexity estimates, we assume that the product of two integers of bit size d can be computed within $\mathsf{I}(d) = \mathcal{O}(d \log d \log\log d)$ bit operations using Fast Fourier Transform. To keep the notation compact, we sometimes write $\mathsf{I}(d) = \tilde{\mathcal{O}}(d)$; the tilde indicates that the factors polynomial in $\log d$ or smaller have been omitted.

For any prime number p, the bit complexities of the operations $(+, -, \times, \div)$ in the finite field $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ are in $\mathcal{O}(\mathsf{I}(\log p))$. We assume that over the rings we use, the product of two polynomials of degree at most d can be computed within $\mathcal{O}(\mathsf{M}(d))$ base ring operations (each ring operation being counted at unit cost) and that $\mathsf{M}(d) = \tilde{\mathcal{O}}(d)$. For computations in $\mathbb{F}_p[x]$, the bit complexity is bounded by multiplying the arithmetic cost estimates by the bit complexity of the basic operations in $\mathbb{F}_p$.

In all our algorithms we are interested in reducing the complexity with respect to a potentially exponential parameter x (related to a dispersion or to a root of an indicial polynomial). Thus we consider as having cost $\mathcal{O}(1)$ any operation whose complexity is polynomial in the bit size of the input recurrence or polynomials, and concentrate on the dependency of the complexity in x. In order to provide the code with an actual bound on the size of primes that need to be used so that the bound on probability of error is guaranteed, we have to perform a much more precise complexity analysis taking into account all parameters (order, degree of coefficients) (as in the proof of [6, Thm. 3]). Such a detailed analysis will appear in [5].

2. POLYNOMIAL SOLUTIONS

In symbolic summation and in the resolution of linear recurrences, all the known algorithms ultimately require polynomial solutions of linear recurrence equations.

In this section, we give algorithms for computing descriptions of the $\mathbb{Q}$-vector space of solutions of a linear recurrence operator $\mathcal{R}$ with coefficients in $\mathbb{Z}[n]$:

$$\mathcal{R}u = a_r(n)u(n+r) + \cdots + a_0(n)u(n) = 0, \quad n \geq 0. \quad (1)$$

We focus on two types of solutions of such recurrences: solutions with finite support and polynomial solutions.

In what follows, we make the hypothesis that 0 *is an ordinary point of the recurrence.* This means that the leading coefficient $a_r(n)$ does not vanish at any of the integers $0, 1, 2, \ldots$; in other words, when unwinding the recurrence, no division by zero is encountered. This condition is ensured after a generic translation $n \mapsto n+\alpha$. Under our complexity assumptions, a proper α and the corresponding translation can be computed in $\mathcal{O}(1)$ bit operations. The general case (when 0 is not ordinary) is technically more demanding but does not change the complexity estimates we give here. It will be presented in [5].

Let $\mathrm{Sol}(\mathcal{R})$ denote the vector space of solutions $u = (u_0, u_1, \ldots)$ of (1). In the case of an arbitrary $\mathcal{R}$, the dimension of $\mathrm{Sol}(\mathcal{R})$ as a $\mathbb{Q}$-vector space may be different from r (both larger or smaller). However, when 0 is an ordinary point of $\mathcal{R}$, $\mathrm{Sol}(\mathcal{R})$ has dimension exactly r and a basis is given by the sequences $u^{(j)}$, $j = 0, \ldots, r-1$, satisfying $\mathcal{R}$ and having initial conditions $(\delta_{j,i})_i$, for $0 \leq i \leq r-1$, where $\delta_{m,n}$ is the Pochhammer symbol ($\delta_{m,m} = 1$, $\delta_{m,n} = 0$ if $m \neq n$).

In §2.1, we describe the compact representation that forms the basic data structure of our algorithms. Then, in §2.2 we recall classical results that allow for the efficient computation of the Nth element of a solution of $\mathcal{R}$. In §2.3 we describe the reduction from the problem of searching for polynomial solutions to that of finding solutions with finite support. Next, we give in §2.4 algorithms to compute finitely supported and polynomial solutions of recurrences. We conclude this section by showing in §2.5 how the evaluation of a polynomial and its finite differences can be performed efficiently in the compact representation.

2.1 Compact Representation

Classically, a polynomial $u(n)$ solution of (1) is represented by its coefficients in the monomial basis $\{n^k\}$. We use an alternative data structure for $u(n)$, which is motivated by the observation that its coefficients c_k in the binomial basis $\{\binom{n}{k}\}$ obey a recurrence with polynomial coefficients.

EXAMPLE 1. *The recurrence* $(n+1)u(n+1) - (n+N+1)u(n) = 0$ *has a unique nontrivial monic polynomial solution* $u(n) = (n+1)\cdots(n+N)$. *To write down its coefficients in the monomial basis at least* $N^2 \log N$ *bits are needed. In contrast,* $u(n)$ *can be represented by the recurrence*

$$(k+1)c_{k+1} - (N-k)c_k = 0, \quad c_0 = N!$$

on the coefficients c_k *of* $u(n)$ *in the binomial basis* $\{\binom{n}{k}\}$; *the bit size of this new representation is only linear in* $N \log N$.

DEFINITION 2. *The* compact representation *of a polynomial solution of* (1) *is the data of a linear recurrence and initial conditions for its coefficients on the binomial basis, together with an upper bound on its degree.*

Our aim in this article is to demonstrate that this representation of polynomial solutions of recurrences can be carried through different algorithms from indefinite and definite hypergeometric summation and that it is beneficial from the complexity point of view. The reason why this representation deserves the name "compact" appears in §2.4 below.

2.2 High-Order Terms of Sequences

Let (u_n) be a sequence satisfying (1). The recurrence $\mathcal{R}$ can be rewritten as a first-order matrix recurrence $U_{n+1} = \mathcal{C}(n+1)U_n$, where U_n is the vector $(u_n, u_{n-1}, \ldots, u_{n-r+1})^t$ and $\mathcal{C}$ is an $r \times r$ matrix with rational function entries. The problem of computing a selected term u_N reduces to that of computing U_{r-1} and the *matrix factorial* $\mathcal{F}(N) := \mathcal{C}(N)\cdots\mathcal{C}(r)$. This makes sense since under our hypothesis the leading term of the initial recurrence does not vanish at $1, 2, \ldots, N$. The numerator and denominator of the matrix factorial can be computed efficiently, either in $\mathbb{Z}$ using a *binary splitting* algorithm, or modulo a prime p using a *baby-step/giant-step* algorithm. These algorithms are described in [6, §2.1, §3.1], see the references therein. For further use, we extract from [6] the following result.

THEOREM 1. *Let (U_i) be a sequence of vectors of rational numbers that satisfies a recurrence $U_{i+1} = C(i+1)U_i$, with $C(x)$ an $r \times r$ matrix with rational function entries in $\mathbb{Q}(x)$, $r = \mathcal{O}(1)$, C and the initial coefficients of total bit size $\mathcal{O}(1)$. Let p be a prime number such that the denominator of C does not vanish modulo p at $1, 2, \ldots, N$. Then:*

(a) $\mathcal{F}(N) := C(N)\cdots C(r)$ and U_N have bit size $\mathcal{O}(N \log N)$; their values can be computed using $\mathcal{O}\left(\mathsf{I}(N \log N) \log N\right)$ bit operations.

(b) $\mathcal{F}(N) \bmod p$ and $U_N \bmod p$ can be computed using $\mathcal{O}\left(\mathsf{M}(\sqrt{N})\mathsf{I}(\log p)\right)$ bit operations.

2.3 Expansion on the Binomial Basis

We give here an algorithm called RecToRec to perform the conversion from a recurrence with polynomial coefficients to the recurrence satisfied by the coefficients of series solutions in the binomial basis. Earlier (and perhaps slightly more complicated) algorithms have been given in [4, Chapter XIII] and [3, Section 4.2]. We start from the following two identities:

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}, \quad n\binom{n}{k} = k\binom{n}{k} + (k+1)\binom{n}{k+1}.$$

If $u(n) = \sum_{k=0}^{\infty} c_k \binom{n}{k}$, then applying these identities to rewrite $u(n+1)$ and $nu(n)$ and extracting coefficients of $\binom{n}{k}$ shows that the ring morphism $\phi : \mathbb{Q}[n, S_n] \to \mathbb{Q}[k, S_k, S_k^{-1}]$ defined by $\phi(S_n) = 1 + S_k$ and $\phi(n) = k(1 + S_k^{-1})$ sends a homogeneous LRE satisfied by $u(n)$ to another one satisfied by c_k. The image of (1) is a LRE of the form

$$(a_r(k)S_k^r + b_{r-1}(k)S_k^{r-1} + \cdots + b_{-s}(k)S_k^{-s})c_k = 0, \quad k \geq 0,$$

where the leading term is exactly that of (1) and the trailing term may involve a negative shift (by convention, $c_k = 0$ when $k < 0$). In particular, if 0 is an ordinary point for $\mathcal{R}$, so is it for $\phi(\mathcal{R})$. The resulting algorithm is as follows. Its complexity is clearly polynomial in the bit size of $\mathcal{R}$.

Algorithm RecToRec

Input: a recurrence $\mathcal{R}u = 0$, where $u(n) = \sum_k c_k \binom{n}{k}$.
Output: a recurrence $\mathcal{S}$ satisfied by the sequence (c_k), plus a set $\mathcal{E}$ of linear equations on its initial conditions.

1. Compute $\mathcal{T} = \phi(\mathcal{R})$;
2. Let $-s = \mathrm{val}_{S_k}(\mathcal{T})$ be its valuation w.r.t. S_k;
3. If $s < 0$ return $\mathcal{S} := \mathcal{T}$ and $\mathcal{E} := \emptyset$,
4. Otherwise return $\mathcal{S} := S_k^s\mathcal{T}$ and the equations $\mathcal{E} := \{(S_k^i\mathcal{T})c(0) = 0, i = 0, \ldots, s-1\}$.

EXAMPLE 2. *Starting from the recurrence*

$$u(n+1) - nu(n) = 0, \qquad n \geq 0,$$

satisfied by $u(0)$ arbitrary and $u(n) = 0$ for $n > 0$, the application of ϕ first produces $S_k + 1 - k - kS_k^{-1}$. Multiplying by S_k yields the second-order recurrence

$$c_{k+2} - kc_{k+1} - (k+1)c_k = 0, \qquad k \geq 0 \tag{2}$$

satisfied by the coefficients c_k of $u(n)$ in the basis $\{\binom{n}{k}\}$. Additionally, we obtain the linear constraint $c_1 + c_0 = 0$. This reduces the space of solutions of (2) to the one-dimensional space generated by $(-1)^k$. We thus get that $u(n) = (1-1)^n = \sum_{k\geq 0}(-1)^k\binom{n}{k}$ is 1 if $n = 0$, and 0 otherwise.

2.4 Finite Support and Polynomial Solutions

We consider here the problem of computing a basis of solutions *with finite support*, that is, whose terms beyond a certain index are all zero. The *degree* of a solution with finite support u is, by definition, the largest integer n such that $u_n \neq 0$ and $u_{n+i} = 0$, for all $i \geq 1$. A universal bound N on the degrees of all solutions with finite support of the input recurrence $\mathcal{R}u = 0$ is given by the largest positive integer root of the trailing coefficient $a_0(n)$ of $\mathcal{R}$. Note that N is generally not bounded polynomially in the bit size of $\mathcal{R}$.

Recall that $\mathrm{Sol}(\mathcal{R})$ has dimension r, with a basis formed by the sequences $u^{(j)}$, with $0 \leq j \leq r-1$, given by the initial conditions $u_i^{(j)} = \delta_{j,i}$, for $0 \leq i \leq r-1$. Thus, a finitely supported solution u is (an unknown) linear combination $\sum_{j=0}^{r-1} \lambda_j u^{(j)}$ such that the elements in the slice $u_{N+1}, \ldots, u_{N+r}$ all vanish. This yields linear constraints on the initial conditions λ_j.

To determine these constraints, it is sufficient to compute the values at indices $N+1, \ldots, N+r$ of all the elements in $\mathcal{B}$ using Thm. 1. The rank of the resulting $r \times r$ matrix gives the dimension of the vector space of solutions with finite support. Since the entries of this matrix have bit size $\mathcal{O}(N \log N)$, the desired λ_j's are determined by a kernel computation, and have also bit size $\mathcal{O}(N \log N)$. Putting together these considerations, we get the following result.

THEOREM 2. *There exists a basis $(u^{(1)}, \ldots, u^{(d)})$ $(d \leq r)$ of solutions of recurrence (1) with finite support, where each $u^{(i)}$ is uniquely specified by the data of initial conditions of bit size $\mathcal{O}(N \log N)$, with N a bound on the integer roots of a_0. The dimension d as well as the degrees of*

the $u^{(i)}$'s can be computed by a probabilistic algorithm using $\tilde{\mathcal{O}}(\mathsf{M}(\sqrt{N})\mathsf{I}(\log N))$ bit operations. The initial conditions of the $u^{(i)}$'s and their maximal degree $D \le N$ can be computed deterministically in $\mathcal{O}(\mathsf{I}(D\log D)\log D)$ bit operations.

Thm. 2 is the basis for using the name "compact representation": it shows that the compact representation has a size of the same order as the initial conditions, while the expanded polynomials have size $\mathcal{O}(N^2\log N)$.

Using the results in §2.3, Thm. 2 carries over literally to the compact representation of a basis of polynomial solutions of the recurrence $\mathcal{R}u = 0$. The corresponding statement requires a bound on the degree of polynomial solutions that is given by the roots of the *indicial polynomial*.

DEFINITION 3. *The* indicial polynomial *of $\mathcal{R}$ at infinity is the trailing coefficient of* $\mathsf{RecToRec}(\mathcal{R})$.

COROLLARY 1. *The statement of Thm. 2 holds for polynomial solutions of $\mathcal{R}u = 0$, with N the largest integer root of the indicial polynomial of $\mathcal{R}$ at infinity.*

Nonhomogeneous Equations. We now consider the equation $\mathcal{R}u(n) = f(n)$, with coefficients in $\mathbb{Z}[n]$ and right-hand side of degree $m = \mathcal{O}(1)$. Applying $\mathsf{RecToRec}$ and expanding $f(n)$ on the binomial basis, the initial problem boils down to the search of finitely supported solutions of a nonhomogeneous equation $\mathcal{S}c(k) = g(k)$, where g is a sequence with finite support, $g(i) = 0$ for $i > m$. In matrix notation, we have $U_{k+1} = C(k+1)U_k + v_{k+1}$, where U_k is the vector $(u_k, \dots, u_{k-r+1})^t$ and v_k is the vector $(g(k), 0, \dots, 0)^t$. Then the vector of initial conditions U_{r-1} satisfies the affine constraint $A(BU_{r-1} + w_{m+1}) = 0$, where $A := C(N+r)C(N+r-1)\cdots C(m)$, $B := C(m-1)C(m-2)\cdots C(r)$, $w_i := v_i + C(i)w_{i-1}$, for $1 \le i \le m+1$ and $w_0 = v_0$.

Using Thm. 1, the matrices A and B can be computed efficiently. The bit size and the computational cost of w_{m+1} is $\mathcal{O}(1)$. Thus, solving the affine system of size $\tilde{\mathcal{O}}(N)$ yields the finitely supported solutions of $\mathcal{S}c = g$ and the polynomial solutions of $\mathcal{R}u = f$ and we get the following.

COROLLARY 2. *The statement of Thm. 2 holds for polynomial solutions of the nonhomogeneous equation $\mathcal{R}u(n) = f(n)$ with N the maximum of $\deg f$ and the largest integer root of the indicial polynomial of $\mathcal{R}$ at infinity.*

2.5 Evaluation in Compact Representation

The compact representation is not only a data structure for intermediate computations. It can actually be exploited further. In particular, we now detail the evaluation at an algebraic number α of degree $\mathcal{O}(1)$ of a polynomial $u(x)$ and an iterated difference $\Delta^H(u)$ (where $\Delta = S_x - 1$ and H is potentially large). The polynomial u is given by its degree N and the recurrence

$$\sum_{i=0}^{r} a_i(k)c(k+i) = 0 \quad \text{for all } k \ge 0$$

satisfied by its coefficients c_k in the binomial basis $\{\binom{x}{k}\}$, together with initial conditions.

Since the sequence $\beta(k) := \binom{\alpha}{k}$ satisfies the first-order recurrence $(k+1)\beta(k+1) - (\alpha-k)\beta(k) = 0$, the sequence $d(k) := c(k)\binom{\alpha}{k}$ satisfies the recurrence of order r,

$$\sum_{i=0}^{r} a_i(k)d(k+i)\prod_{j=1}^{r-i+1}\frac{\alpha-(k+r-j)}{k+r-j+1} = 0 \quad \text{for all } k \ge 0.$$

Thus, its indefinite sum $D_k = \sum_{\ell=0}^{k} d(\ell)$ satisfies a similar recurrence of order $r+1$ obtained after composing with $S_k - 1$. Moreover, 0 is an ordinary point for the recurrence on D_k and initial conditions are easily deduced from those on c_k.

Now, evaluating u at α is equivalent to computing the term $D_N = \sum_{k=0}^{N} c(k)\binom{\alpha}{k}$ and this can be done by Thm. 1 in $\mathcal{O}(\mathsf{I}(N\log N)\log N)$ bit operations. Using Pascal's formula $\Delta^H\binom{x}{i} = \binom{x}{i-H}$, we deduce that $\Delta^H u(\alpha) = \sum_{k=0}^{N-H} c_{k+H}\binom{\alpha}{k}$. The recurrence satisfied by the sequence $(c_{k+H})_k$ is obtained by shifting by H the coefficients of the recurrence of (c_k). This new recurrence has bit size $\mathcal{O}(\log H)$ and initial conditions determined by binary splitting in $\mathcal{O}(\mathsf{I}(N\log N\log H)\log N\log H)$ bit operations. Here, our asymptotic bound involves the two parameters N and H, as both are potentially exponential in the input size. As above, the compact representation of the recurrence satisfied by $D_k := \sum_{\ell=0}^{k} c_{\ell+H}\binom{\alpha}{\ell}$ can be determined efficiently, as well as its Nth term, whose value is precisely $\Delta^H u(\alpha)$.

3. RATIONAL SOLUTIONS

3.1 Compact Gosper Normal Form

The classical Gosper normal form [11] (see also [13]) of a reduced rational function P/Q in $\mathbb{Q}(n)$ consists in three polynomials A, B, C in $\mathbb{Q}[n]$ such that

$$\frac{P(n)}{Q(n)} = \frac{A(n)}{B(n)}\frac{C(n+1)}{C(n)}, \tag{3}$$

with the constraints

$$\gcd(A(n), C(n)) = 1,\ \gcd(B(n), C(n+1)) = 1,$$
$$\text{and for all } h \in \mathbb{N},\ \gcd(A(n), B(n+h)) = 1. \tag{4}$$

The degree N of the polynomial $C(n)$ is potentially exponentially large. Thus, in our algorithm $\mathsf{CompactGF}$ below, we modify the usual algorithm (e.g., in [14]) slightly so that the polynomial $C(n)$ is not expanded. Similar ideas appear in [10] in the context of indefinite rational summation.

Algorithm CompactGF

Input: a couple $(P(n), Q(n))$ of polynomials.
Output: $(A(n), B(n), \{(g_i(n), h_i), i = 1, \dots, s\})$ such that $C(n) = \prod_i g_i(n-1)\cdots g_i(n-h_i)$ satisfies (3).

1. Compute $0 < h_1 < \cdots < h_s$ the positive integer roots of $\mathrm{Res}_n(P(n), Q(n+h))$;
2. $A(n) := P(n), B(n) := Q(n)$;
3. For i from 1 to s do
   a. $g_i(n) := \gcd(A(n), B(n+h_i))$;
   b. $A(n) := A(n)/g_i(n), B(n) := B(n)/g_i(n-h_i)$;
4. Return $(A, B, \{(g_i(n), h_i), i = 1, \dots, s\})$.

EXAMPLE 3. $\mathsf{CompactGF}(n, n-N) = (1, 1, \{(n, N)\})$.

Note that the input is a couple (P, Q) and not a rational function P/Q. The output of the algorithm changes if (P, Q) is replaced by (FP, FQ) for $F \in \mathbb{Q}[n]$. This property will be necessary for our treatment of rational solutions below. On

the other hand, the output A, B, and g_i's also satisfy (4) whenever P and Q have no common factor, so that the Gosper normal form of a rational function in $\mathbb{Q}(n)$ given in reduced form P/Q is obtained by CompactGF(P, Q).

As an outcome of this algorithm, the rational function $C(n)/C(n+j)$ $(j = 1, 2, \dots)$ is easily obtained as

$$\frac{C(n)}{C(n+j)} = \prod_{i=1}^{s} \frac{g_i(n+j-1-h_i)\cdots g_i(n-h_i)}{g_i(n+j-1)\cdots g_i(n)}. \qquad (5)$$

For $j = \mathcal{O}(1)$, it has "small" numerator and denominator of degrees bounded by j times those of P and Q. This equation for $j = 1$ is a homogeneous LRE that plays the rôle of a compact representation of C. The initial value $C(0)$ (and more generally $C(k)$ where $k = \mathcal{O}(1)$) has size $\mathcal{O}(N \log N)$ and can be computed by Thm. 1 within $\mathcal{O}(I(N \log N) \log N)$ bit operations. In the next sections, we use this to design "compact" variants of Abramov's and Gosper's algorithms.

PROPOSITION 1. *Algorithm CompactGF is correct. For (P, Q) with rational coefficients, it has deterministic polynomial bit complexity in the bit size of (P, Q).*

PROOF. The correctness is that of the classical algorithm since the only difference is that we do not expand C. Step 1 is dealt with by a deterministic algorithm due to Loos [12] (cf. [9, 10] for faster probabilistic algorithms). Step 3 is performed at most $\deg P \deg Q$ times, and each step is polynomial by the classical algorithms as found in [16]. □

3.2 Compact Rational Solutions

We now consider rational solutions of the LRE $\mathcal{R}u(n) = f(n)$, with f a small polynomial in $\mathbb{Q}[n]$, i.e., of bit size $\mathcal{O}(1)$.

Our starting point is the following result of Abramov [1].

LEMMA 1 (ABRAMOV). *The polynomial $C(n)$ of the Gosper form of $(a_r(n-r+1), a_0(n))$ is a multiple of the denominator of all rational solutions of $\mathcal{R}u(n) = f(n)$.*

(Note the slight mistake in the statement in Abramov's article [1]. His algorithm considers all nonnegative integer roots $0 \le h_1 < \cdots < h_s$ of the same resultant as in CompactGF, and this variant is applied to $(a_r(n-r), a_0(n))$ instead of $(a_r(n-r+1), a_0(n))$. In the case when $a_r(n-r+1)$ and $a_0(n)$ have a nontrivial gcd, factors of the denominator can be missed.) Abramov's algorithm first computes $C(n)$, then performs the change of variable $u(n) = v(n)/C(n)$, leading to

$$a_r(n)\frac{v(n+r)}{C(n+r)} + \cdots + a_0(n)\frac{v(n)}{C(n)} = f(n), \qquad (6)$$

whose polynomial solutions $v(n)$ are then sought.

In the homogeneous case ($f(n) = 0$), using (5) reduces this equation to an equation of polynomial size. This is described in HomCompactRatSols. In Step 2, the "Normalize" operation consists in expanding $C(n)/C(n+i)$ using (5) and taking the numerator of the resulting expression. Also, if necessary, we change n into $n + \alpha$ with $C(\alpha) \neq 0$, so that 0 is not a singular point in Step 3. This can be detected and changed at a cost of $\mathcal{O}(\mathsf{I}(N \log N) \log N)$ operations. In Step 4, the output is given by the compact forms of the numerators and C is given by the output of CompactGF.

In the nonhomogeneous case, reducing (6) to the same denominator would lead to an equation whose right-hand side has a potentially exponential degree. Instead, we consider the homogeneous operator $\mathcal{S} = (f(n)S_n - f(n+1))\mathcal{R}$, whose bit size is polynomial in that of $\mathcal{R}u(n) = f(n)$ and that can be treated by the algorithm above. If u_n is a rational solution of $\mathcal{S}$, then $w_n = \mathcal{R}u_n$ is a rational solution of $f(n)w_{n+1} = f(n+1)w_n$. Thus $w_n = \lambda f(n)$ for all n larger than the largest root of f and since it is rational, also for all other values of n. Thus fixing $\mathcal{R}u(k) = f(k)$ for any k such that $f(k) \neq 0$ concludes the computation. This is the basis of the following algorithm.

Algorithm HomCompactRatSols

Input: a homogeneous LRE $\mathcal{R}u(n) = 0$.
Output: a basis of its rational solutions in compact form

1. $(A, B, C) :=$ CompactGF$(a_r(n-r+1), a_0(n))$;
2. Normalize $C(n)\mathcal{R}(v(n)/C(n))$ using (5) and denote the result $\mathcal{T}v(n)$;
3. Compute a basis $\mathcal{B}$ of the polynomial solutions of $\mathcal{T}v(n) = 0$;
4. Return $\{p(n)/C(n) \mid p(n) \in \mathcal{B}\}$.

Algorithm NonhomCompactRatSols

Input: a LRE $\mathcal{R}u(n) = f(n)$, with $f \neq 0$.
Output: a particular rational solution p and a basis $(b_1, \dots, b_d)$ of rational solutions of $\mathcal{R}u$ in compact form

1. $W :=$ HomCompactRatSols$((f(n)S_n - f(n+1))\mathcal{R})$;
2. Find $k \in \mathbb{N}$ such that $f(k) \neq 0$;
3. Write $\mathcal{R}(\sum_{w\in W} \xi_w w(k)) =: \mathcal{U}(\xi)$ for an unknown $\xi = (\xi_w)_{w\in W}$ and solve $\mathcal{U}(\xi) = 0$ for a basis $(\mu^{(1)}, \dots, \mu^{(d)})$ of its solution space and $\mathcal{U}(\xi) = f(k)$ for a particular solution λ;
4. Return $p := \sum_{w\in W} \lambda_w w(n)$ and the b_i's given by $b_i := \sum_{w\in W} \mu_w^{(i)} w(n)$.

In Step 2, just iterating $k = 0, 1, \dots$ till a point where f is found to be nonzero is sufficient for our purpose. If N is a bound on the degree of the numerators and denominator computed in Step 1, then the values of the $w(k)$'s in Step 3 have size $\mathcal{O}(N \log N)$ and can be computed by binary splitting. From there, it follows that the affine equation in Step 3 has coefficients of size $\mathcal{O}(N \log N)$, which is then also a bound on the size of its solutions. These solutions can be computed in the form of a point and a basis of a vector space within $\mathcal{O}(\mathsf{I}(N \log N) \log N)$ bit operations by standard linear algebra. The same complexity is sufficient for the products of initial conditions in Step 4.

The results of this section are summarized as follows.

THEOREM 3. *Let N be the sum of the largest nonnegative integer root of the indicial polynomial of $\mathcal{R}$ at infinity and the degree of the polynomial $C(n)$ of (3) with $P(n) = a_r(n-r+1)$ and $Q(n) = a_0(n)$. The dimension of the affine*

space of rational solutions of $\mathcal{R}u(n) = f(n)$ can be computed probabilistically using $\tilde{\mathcal{O}}(\mathsf{M}(\sqrt{N})\mathsf{I}(\log N))$ bit operations. A compact representation of the solutions can be computed deterministically in $\mathcal{O}(\mathsf{I}(N \log N) \log N)$ bit operations.

PROOF. The largest integer root of the indicial polynomial of $\mathcal{R}$ at infinity is a bound on the valuations of power series solutions of $\mathcal{R}u = 0$ at infinity, including the valuation of $v(n)/C(n)$. Adding the degree of C gives the announced bound on the degree of polynomial v's. From there, the theorem follows from Cor. 1. □

3.3 A Compact Gosper Algorithm

Given a hypergeometric term $t(n)$, i.e., such that $t(n+1)/t(n) =: r(n) \in \mathbb{Q}(n)$, Gosper's algorithm [11] finds its indefinite hypergeometric sum, if it exists. Such a sum is necessarily of the form $u(n)t(n)$ for some $u(n) \in \mathbb{Q}(n)$. Thus, the problem is reduced to finding rational solutions of $u(n+1)r(n) - u(n) = 1$. This can be solved by NonhomCompactRatSols. A further optimization is present in Gosper's algorithm: if $r(n) = P(n)/Q(n)$ in reduced form, the polynomial $B(n)$ of (3) satisfies (4), so that it divides the numerator of $u(n+1)$. (This can be generalized to detect factors of numerators in LRE's). This does not affect the expression of the complexity result, which is as follows.

THEOREM 4. *Let $t(n)$ be a hypergeometric term such that $t(n+1)/t(n) =: P(n)/Q(n) \in \mathbb{Q}(n)$, with $\gcd(P, Q) = 1$. Let N be a bound on the degree of C in (3) and on the largest positive integer root of the indicial polynomial of $P(n)S_n - Q(n)$ at infinity. Then the existence of an indefinite hypergeometric sum of $t(n)$ can be determined by a probabilistic algorithm using $\tilde{\mathcal{O}}(\mathsf{M}(\sqrt{N})I(\log N))$ bit operations, a compact representation of it can be computed deterministically using $\mathcal{O}(\mathsf{I}(N \log N) \log N)$ bit operations.*

4. DEFINITE HYPERGEOMETRIC SUMS

A bivariate hypergeometric term $t(n, m)$ is such that both $t(n+1, m)/t(n, m)$ and $t(n, m+1)/t(n, m)$ belong to $\mathbb{Q}(n, m)$. Given such a term, Zeilberger's algorithm [18] computes a LRE satisfied by $T(m) = \sum_n t(n, m)$. The idea is to synthesize a *telescoping recurrence*, i.e., a rational function $Q(n, m)$ and a linear operator $P(m, S_m)$ such that

$$(S_n - 1)Q(n, m)t(n, m) = P(m, S_m)t(n, m).$$

Indeed, summing over n and granted boundary conditions known as "natural boundaries", we obtain $P(m, S_m)T(m) = 0$. If P was known, then Gosper's algorithm would find the left-hand side. This is the basis of Zeilberger's algorithm (see Figure). Termination is guaranteed only if such a LRE exists. This occurs in the so-called "proper-hypergeometric" case [17] and a general criterion has been given by Abramov [2].

Note that knowing Q permits to check the output operator P by simple rational function manipulations, which is why the rational function Q is called "certificate" in [14].

Zeilberger's algorithm is based on a refinement of Gosper's algorithm for Steps 2 and 3. It reduces the computation in Step 2 to solving a system that is linear simultaneously in the λ_i's and in another set of $N+1$ variables, where N is potentially exponential in the bit size of (E_r), see e.g. [14, §6.3]. An equivalent linear system in a small number of variables can be computed as follows.

Zeilberger's Algorithm

Input: two functions $\frac{t(n+1,m)}{t(n,m)}$ and $\frac{t(n,m+1)}{t(n,m)}$ in $\mathbb{Q}(n, m)$.
Output: a LRE $\sum_{i=0}^{r} \lambda_i(m) S_m^i(\sum_n t(n, m)) = 0$.
For $r = 0, 1, 2, \ldots$ do

1. Construct the equation (E_r)

$$u(n+1, m)\frac{t(n+1, m)}{t(n, m)} - u(n, m) = \sum_{i=0}^{r} \lambda_i(m)\frac{t(n, m+i)}{t(n, m)},$$

2. Find if there exist λ_i's in $\mathbb{Q}(m)$ so that (E_r) admits a solution $u(n, m) \in \mathbb{Q}(n, m)$;
3. If so, compute and return them; otherwise proceed to the next r.

Small Linear System

Input: the equation (E_r) from Zeilberger's algorithm.
Output: an equivalent system linear in the λ_i.

1. Compute $\mathcal{R}u(n) = f(n)$ the numerator of (E_r);
2. Compute a multiple $C(n)$ of the denominator of its rational solutions and a bound N on the degree in n of their numerators;
3. Compute $\mathcal{S}v(n)$ the numerator of $C(n)(f(n)S_n - f(n+1))\mathcal{R}(v/C)(n)$;
4. Compute $(\mathcal{T}, \mathcal{E}) :=$ RecToRec$(\mathcal{S})$; set $\mathcal{E} := \mathcal{E} \cup \{\mathcal{R}(v/C)(0) = f(0)\}$; let s be the order of $\mathcal{T}$;
5. Compute the value $(c_{N+1}, \ldots, c_{N+s}) =: V$ for a nonzero sequence solution of RecToRec$(C\mathcal{R}(v/C))$;
6. Compute the value $W := (c_{N+1}, \ldots, c_{N+s})$ for an arbitrary sequence solution of $\mathcal{T}$ obeying $\mathcal{E}$; W is of the form $W^\star + \sum_{i=0}^{r} \lambda_i W_i$, only $W^\star$ depends on the initial conditions;
7. The system $(\Sigma) := (\mu V + \sum_{i=0}^{r} \lambda_i W_i = 0)$ is simultaneously linear in the λ_i's and μ.

The important point is linearity. In Step 2, by Lemma 1, C does not depend on the λ_i's. Then, by induction, starting from $\mathcal{R}(v/C)(0) = f(0)$, the factor $f(n)$ of the leading coefficient in $\mathcal{S}$ cancels out and thus the solution $v(n)$ is linear in the λ_i. This property is then preserved by the linearity of RecToRec. The final system (Σ) has solutions if and only if (E_r) has rational solutions.

The description of Small Linear System is geared towards the use of compact representations and matrix factorials in intermediate steps. This is straightforward for Steps 1–5. In Step 6, we cannot make direct use of the factorial of the matrix associated to $\mathcal{T}$: this matrix involves the λ_i's rationally and its factorial has too large a size for our target complexity. Instead, we exploit the linearity in the λ_i's by constructing the vector W using matrix factorials for λ a vector of 0's with a 1 in ith position for $i = 0, \ldots, r$ and setting the initial condition to 0, which gives the coefficients W_i.

Our version of Zeilberger's algorithm is then as follows.

Compact Zeilberger Algorithm

Input: two functions $\frac{t(n+1,m)}{t(n,m)}$ and $\frac{t(n,m+1)}{t(n,m)}$ in $\mathbb{Q}(n,m)$.
Output: a LRE $\sum_{i=0}^{r} \lambda_i(m) S_m^i(\sum_n t(n,m)) = 0$.
For $r = 0, 1, 2, \ldots$ do

1. Take a random $m_0 \in \mathbb{Q}$ and construct (E_r) with $m = m_0$;
2. Apply Small Linear System to this equation;
3. Find if there exist nonzero solutions to this system;
4. If not, proceed to the next r;
5. Otherwise, construct (E_r), apply Small Linear System and return its solutions.

In Step 2, the whole construction can be performed by matrix factorials with integer entries, within the complexities of Thm. 1. If a rational solution exists, then the system (Σ) has for solutions the corresponding $\lambda_i(m_0)$. Thus if (Σ) does not have a nonzero solution, (E_r) does not have a rational one. This gives a fast probabilistic test. Then, in Step 5, the algorithm is used again with matrices that are polynomial in the variable m. In that case, the system (Σ) can be computed by binary splitting with $\tilde{\mathcal{O}}(\mathsf{M}(N)\log N)$ *arithmetic* operations. The final system has coefficients of degree $\mathcal{O}(N)$ with coefficients of bit size $\mathcal{O}(N \log N)$ and this is also the size of the λ_i's to be found. At the same time, we find μ which gives us a compact representation of the certificate.

An optimization is obtained by using the values of the $\lambda_i(m_0)$'s to compute the value N' of the degree of the corresponding sequence. With high probability this is the actual degree in n of the numerator of $u(n,m)$, which can be much smaller than N, thus saving a lot of computation in Step 5.

The following theorem summarizes this section.

THEOREM 5. *Let $t(n,m)$ be hypergeometric over $\mathbb{Q}$. Let N be the maximal number of variables in the linear system solved in the classical version of Zeilberger's algorithm. Then it is possible to detect probabilistically that this system does not have any nonzero solution in $\tilde{\mathcal{O}}(\mathsf{M}(\sqrt{N})\mathsf{I}(\log N))$ bit operations. If it does have a solution, it is possible to compute the corresponding λ_i's of degree $\mathcal{O}(N)$ and bit size $\mathcal{O}(N^2 \log N)$, as well as a compact representation of the certificate, in $\tilde{\mathcal{O}}(\mathsf{M}(N)\mathsf{I}(N \log N))$ bit operations.*

For the sake of comparison, a crude analysis by unrolling the triangular system of dimension $N + r + 2$ and taking into account coefficient growth leads to a $\tilde{\mathcal{O}}(N^4)$ bit complexity estimate for the classical algorithm, which can be reduced to $\tilde{\mathcal{O}}(N^3)$ by using the binomial basis.

5. EXPERIMENTAL RESULTS

5.1 Rational Solutions

	Classical		Compact	
N	R_1	R_2	R_1	R_2
2^6	0.6	0.0	0.0	0.0
2^7	3.7	0.1	0.0	0.0
2^8	37.3	0.1	0.0	0.0
2^9	429.9	0.2	0.0	0.0
2^{10}		1.0	0.0	0.0
2^{11}		6.2	0.0	0.0
2^{12}		46.4	0.0	0.0
2^{13}		362.0	0.0	0.0

Table 1: Timings for classical and compact versions of Abramov's algorithm

We consider two families of linear recurrences:

$$\begin{aligned}
&2n(N-n)(-4N-3nN+6+3n^2+8n)u(n)\\
&-(n+1)(-3nN+2N+3n^2-4n-4)(n+1-N)u(n+1)\\
&+(n+2)(-3nN-N+3n^2+2n+1)(n+2-N)u(n+2)=0,\\
&2n(n-2N)(n-N)(n^2-3nN+3n+2N^2-3N+2)u(n)\\
&-(n+1)(n+1-2N)(n+1-N)(3n^2+6n-9nN+6N^2-4N)u(n+1)\\
&+(n+2)(n+2-2N)(n+2-N)(n^2+n-3nN+2N^2)u(n+2)=0.
\end{aligned}$$

The first one (R_1) does not have any rational solution, while the second one (R_2) has $1/(n(n-2N))$ as a solution. In both cases, when N is a large integer, a large dispersion has to be considered. In Table 1, we give a comparison of the timings[1] obtained by our Maple prototype (denoted Compact) and that of the command `ratpolysols` of Maple's package `LREtools` (denoted Classical). This table illustrates the "nonexponential" character of the compact versions of the algorithms. In the first case, both output are identical (no solution). In the second case, however, we return a compact representation of the output. For instance, with $N = 2^{100}$ we get (in 0.04s) for denominator $(n(n-2^{100})(n-2^{101}))$ (in expanded form) and for numerator the recurrence

$$\begin{aligned}
&(1-k^2)c_k + (2^{100}+Ak-k^2)c_{k+1}\\
&\quad + (k^2-2k-B)c_{k+2} + (k^2-Ck+D)c_{k+3} = 0,
\end{aligned}$$

satisfied by its coefficients in the binomial basis, together with initial conditions $c_0 = -2^{100}$, $c_1 = 1$, where the coefficients $A, \ldots, D$ are 200 bit long integers.

5.2 Definite Hypergeometric Summation

We consider the following family of hypergeometric terms:

$$t(n,m) = \binom{2n+m+N}{N}\binom{2m}{2n}\binom{m}{n}.$$

For $N \in \mathbb{N}$, the sum $\sum_n t(n,m)$ satisfies a third-order homogeneous LRE. When Zeilberger's algorithm is executed on this term, the bound it has to use on the degrees of numerators of rational solutions of the equation (E_r) is $N+3(r-1)$. This plays the rôle of a "large" N and makes it possible to exhibit the complexity behaviour of the algorithms.

In Table 2, we give a comparison of the timings obtained by our prototype implementation in Maple (denoted "Compact") and those obtained by Maple's `Zeilberger` command

[1] All our tests have been run on an Intel Xeon at 3.6GHz.

N	Classical			
	$r=0$	$r=1$	$r=2$	$r=3$
16	0.1	0.2	0.3	0.6
32	0.3	0.7	1.5	3.4
64	2.9	6.8	12.	34.3
128	43.9	131.0	276.4	1202.6
256	1793.4	$>$ 2Gb		

N	Compact, random m			
	$r=0$	$r=1$	$r=2$	$r=3$
2	0.0	0.2	0.4	0.9
4	0.1	0.2	0.7	1.4
8	0.1	0.3	0.8	1.9
16	0.1	0.3	0.9	2.5
32	0.1	0.5	1.4	5.1
64	0.2	0.7	2.6	7.3
128	0.3	1.5	5.0	15.2
256	0.5	2.7	11.3	35.5
512	1.0	6.3	27.8	106.2
1024	2.2	15.7	72.7	240.1

Table 2: Timings for classical and compact versions of Zeilberger's algorithm

from the package `SumTools:-Hypergeometric` (denoted "Classical"). The indication "$>$ 2Gb" means that the computation had to be stopped after two gigabytes of memory had been exhausted. The first part of the table (Classical) suggests that the implementation does not behave well for large N: the observed behaviour is exponential instead of polynomial. Even then, it is still much better than our implementation. Indeed, we have implemented only the case with rational values of m and for small N it often takes longer for our implementation to compute the result with this value than for the classical method to find the result with a formal m. However, things change as N gets larger: the predicted behaviour is well observed. When N is multiplied by 2, the time is multiplied by slightly more than 2. Had we implemented the baby-step/giant-step version of binary splitting, the timings in the columns for random m would have been much better, since the time should be multiplied by slightly more than $\sqrt{2}$ from one line to the next. Our experiments with symbolic m show that so far, our complexity result is more of a theoretical nature: although the degrees of the coefficients of the equations grow like $\mathcal{O}(N)$, the constant in front of the $\mathcal{O}$ term is about 18 in this example, and a massive cancellation takes place in the final linear solving. The result has degrees that also grow like $\mathcal{O}(N)$, but with a much smaller constant, so that a direct resolution in $\tilde{\mathcal{O}}(N^4)$ is much faster in this range than our $\tilde{\mathcal{O}}(N^2)$.

6. REFERENCES

[1] S. A. Abramov. Rational solutions of linear difference and q-difference equations with polynomial coefficients. In A. H. M. Levelt, editor, *Symbolic and Algebraic Computation*, pages 285–289, New York, 1995. ACM Press. Proceedings of ISSAC'95, July 1995, Montreal, Canada.

[2] S. A. Abramov. Applicability of Zeilberger's algorithm to hypergeometric terms. In T. Mora, editor, *ISSAC'2002*, pages 1–7. ACM Press, July 2002. Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, July 07–10, 2002, Université de Lille, France.

[3] S. A. Abramov, M. Bronstein, and M. Petkovšek. On polynomial solutions of linear operator equations. In A. H. M. Levelt, editor, *Symbolic and Algebraic Computation*, pages 290–296, New York, 1995. ACM Press. Proceedings of ISSAC'95, July 1995, Montreal, Canada.

[4] G. Boole. *A treatise on the calculus of finite differences*. Macmillan, London, 2nd edition, 1872.

[5] A. Bostan, F. Chyzak, T. Cluzeau, and B. Salvy. Fast algorithms for polynomial and rational solutions of linear operators equations, In preparation.

[6] A. Bostan, T. Cluzeau, and B. Salvy. Fast algorithms for polynomial solutions of linear differential equations. In M. Kauers, editor, *ISSAC'05*, pages 45–52, New York, 2005. ACM Press. Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, July 2005, Beijing, China.

[7] A. Bostan, P. Gaudry, and É. Schost. Linear recurrences with polynomial coefficients and computation of the Cartier-Manin operator on hyperelliptic curves. In *International Conference on Finite Fields and Applications (Toulouse, 2003)*, volume 2948 of *Lecture Notes in Computer Science*, pages 40–58. Springer–Verlag, 2004.

[8] D. V. Chudnovsky and G. V. Chudnovsky. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited*, pages 375–472. Academic Press, Boston, MA, 1988.

[9] J. Gerhard. *Modular algorithms in symbolic summation and symbolic integration*. Number 3218 in Lecture Notes in Computer Science. Springer, 2004.

[10] J. Gerhard, M. Giesbrecht, A. Storjohann, and E. V. Zima. Shiftless decomposition and polynomial-time rational summation. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, pages 119–126, New York, 2003. ACM.

[11] R. W. Gosper. Decision procedure for indefinite hypergeometric summation. *Proceedings of the National Academy of Sciences USA*, 75(1):40–42, Jan. 1978.

[12] R. Loos. Computing rational zeros of integral polynomials by $p-$adic expansion. *SIAM Journal on Computing*, 12(2):286–293, May 1983.

[13] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. *Journal of Symbolic Computation*, 14(2-3):243–264, 1992.

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

[15] A. Schönhage, A. F. W. Grotefeld, and E. Vetter. *Fast algorithms*. Bibliographisches Institut, Mannheim, 1994. A multitape Turing machine implementation.

[16] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999.

[17] H. S. Wilf and D. Zeilberger. An algorithmic proof theory for hypergeometric (ordinary and "q") multisum/integral identities. *Inventiones Mathematicae*, 108:575–633, 1992.

[18] D. Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11:195–204, 1991.