Rational Solutions of Linear Difference Equations Revisited

Sergei Abramov [*) 1], Amel Gheffar [**)], Denis Khmelnov [*) 1]

[*)] Computing Centre of the Russian Academy of Sciences
Vavilova, 40, Moscow 119991, GSP-1, Russia
sergeyabramov@mail.ru, dennis_khmelnov@mail.ru

[**)] XLIM, Université de Limoges, CNRS
123, Av. A. Thomas, 87060 Limoges cedex
f_gheffar@yahoo.fr

Abstract. *We discuss algorithms that are currently used for computing rational solutions of linear difference systems (or of scalar equations) with polynomial coefficients. A complexity analysis and a time comparison of the algorithms implemented in Maple are presented.*

1 Introduction

This paper is a summary of authors' results that have been published in [16, 7, 15, 8]. We revisit a problem of the search for rational solutions of a linear difference equation with polynomial coefficients. Rational solutions may be a building block for other types of solutions, and more general, such algorithms may be a part of various computer algebra algorithms (see [22, 9, 10, 18], etc.). Investigations of new ways to construct such solutions are quite valuable for computer algebra.

Let k be a field of characteristic 0. We consider systems of the form

$$Y(x+1) = A(x)Y(x), \tag{1}$$

$Y(x) = (Y_1(x), Y_2(x), \ldots, Y_n(x))^T$, $A(x) = (a_{ij}(x)) \in \mathrm{Mat}_n(k(x))$. It is assumed that there exists the inverse matrix $A^{-1}(x) = (\tilde{a}_{ij}(x)) \in \mathrm{Mat}_n(k(x))$. If an inhomogeneous system $Y(x+1) = A(x)Y(x) + G(x)$ is given and $A(x)$ is as in (1), $G(x) \in k(x)^n$, then by adding to $Y(x)$ an $(n+1)$-st component with value 1, one can transform the given system into a homogeneous system with an invertible matrix $B(x) \in \mathrm{Mat}_{n+1}(k(x))$ (see, e.g., [17, Sect. 2.2]). For this reason we restrict our consideration to (1). At the same time we will consider scalar equations of the form

$$y(x+n) + a_{n-1}(x)y(x+n-1) + \cdots + a_1(x)y(x+1) + a_0(x)y(x) = \varphi(x), \tag{2}$$

$\varphi(x), a_1(x), \ldots, a_{n-1}(x) \in k(x)$, $a_0(x) \in k(x) \setminus \{0\}$, and such an equation is inhomogeneous if $\varphi(x)$ is a non-zero rational function. By clearing denominators we can rewrite (2) as

$$b_n(x)y(x+n) + \cdots + b_1(x)y(x+1) + b_0(x)y(x) = \psi(x), \tag{3}$$

[1]Supported in part by the Russian Foundation for Basic Research, project no. 10-01-00249.

$\psi(x), b_1(x), \ldots, b_{n-1}(x) \in k[x]$, $b_0(x), b_n(x) \in k[x] \setminus \{0\}$. The equations (2), (3) can be represented in the operator form. For example, (3) can be written as $L(y) = \psi(x)$ where $L = b_n(x)\phi^n + \cdots + b_1(x)\phi + b_0(x)$, $\phi(y(x)) = y(x+1)$.

Currently, a few algorithms for finding rational (i.e., rational function) solutions of equations (2), (3) and systems (1) are known. The algorithms from [4, 5, 12, 16, 7] first construct a *universal denominator*, i.e., a polynomial $U(x)$ such that in the scalar case an arbitrary rational solution $y(x)$ of (2) or (3) can be represented as

$$y(x) = \frac{z(x)}{U(x)}, \tag{4}$$

where $z(x) \in k[x]$ (in other words, if (2) has a rational solution $\frac{f(x)}{g(x)}$ which is in the lowest terms then $g(x)|U(x)$). In the case of a system an arbitrary rational solution of (1) can be represented as

$$Y_i(x) = \frac{Z_i(x)}{U(x)}, \quad i = 1, 2, \ldots, n, \tag{5}$$

where $Z_1(x), Z_2(x), \ldots, Z_n(x) \in k[x]$.

The algorithm from [16, Sect. 5] is based on constructing a set of irreducible polynomials that are candidates for divisors of denominators of rational solutions, and on finding a bound for the exponent of each of these candidates in a quite simple way (the algorithm $\mathbf{A}_U$). In [7, Sect. 3.4] a version of this algorithm that finds quickly the set of all such exponents was proposed (the algorithm $\mathbf{A}'_U$).

When a universal denominator is constructed, one can substitute (4), (5) with undetermined $z(x)$ resp. $Z_i(x)$ into the initial equation resp. system to reduce the problem of searching for rational solutions to the problem of searching for polynomial solutions. After this, e.g., the algorithms from [1, 6] (the scalar case) and the corresponding algorithm from [5, 12, 19] (the case of a system) can be used.

The algorithm from [17] is applicable to the system (1) when $k = \mathbb{C}$. It finds n rational functions $R_1(x), R_2(x), \ldots, R_n(x) \in \mathbb{C}(x)$ which are called *denominator bounds* such that for any rational solution of (1) we have

$$Y_i(x) = Z_i(x)R_i(x), \quad i = 1, 2, \ldots, n, \tag{6}$$

where $Z_1(x), Z_2(x), \ldots, Z_n(x) \in \mathbb{C}[x]$ (the numerator of $R_i(x)$ is a factor of the numerator of the ith entry $Y_i(x)$ of any rational solution $Y(x)$). The substitution (6) is used instead of (4), (5). In [16, Sect. 4] a version $\mathbf{A}_B$ of this algorithm suitable for a field k of characteristic 0 was proposed, and the scalar case (2), (3) was especially considered as well.

It was shown in [7, 8] that the algorithm $\mathbf{A}'_U$ has advantages in comparison with other mentioned algorithms for reducing the problem of searching for rational solutions to the problem of searching for polynomial solutions.

The paper is organized as follows. Sect. 2 is devoted to a theoretical basis for algorithms for constructing universal denominators and denominator bounds. Sect. 3 contains descriptions (a short review) of the algorithm from [4, 5, 12, 16, 7, 17].

In Sect. 4 we give some analysis of the algorithms from [4, 5, 12, 16, 7] and show that all of them give the same universal denominator, but the algorithm $\mathbf{A}'_U$ has the lowest complexity.

A combination of one of the algorithms $\mathbf{A}'_U, \mathbf{A}_B$ with an algorithm (the same in both cases) for finding all polynomial solutions gives the algorithms $\langle \mathbf{A}'_U \rangle$, $\langle \mathbf{A}_B \rangle$ for constructing all rational solutions. In Sect. 5 we show that some natural suppositions on the used algorithm for finding all polynomial solutions allow to show that the complexity of $\langle \mathbf{A}'_U \rangle$ is less than the complexity of $\langle \mathbf{A}_B \rangle$

In Sect. 6, we discuss our implementation of the mentioned algorithms and give a time comparison of these algorithms.

In Sect. 7 some changes of the traditional scheme for finding rational solutions of scalar homogeneous equations with polynomial coefficients are proposed. In many cases these changes allow one to predict the absence of rational solutions in an early stage of the computation.

2 The Dispersion Set

Working with polynomial and rational functions over k we will write $f(x) \perp g(x)$ for $f(x), g(x) \in k[x]$ to indicate that $f(x)$ and $g(x)$ are coprime; if $F(x) \in k(x)$, then $\operatorname{den} F(x)$ is the monic polynomial from $k[x]$ such that $F(x) = \frac{f(x)}{\operatorname{den} F(x)}$ for some $f(x) \in k[x], f(x) \perp \operatorname{den} F(x)$. In this case we write $\operatorname{num} F(x)$ for $f(x)$. The set of monic irreducible polynomials of $k[x]$ will be denoted by $\operatorname{Irr}(k[x])$. If $p(x) \in \operatorname{Irr}(k[x])$, $f(x) \in k[x]$, then we define the valuation $\operatorname{val}_{p(x)} f(x)$ as the maximal $m \in \mathbb{N}$ such that $p^m(x) | f(x)$ ($\operatorname{val}_{p(x)} 0 = \infty$), and $\operatorname{val}_{p(x)} F(x) = \operatorname{val}_{p(x)}(\operatorname{num} F(x)) - \operatorname{val}_{p(x)}(\operatorname{den} F(x))$ for $F(x) \in k(x)$.

Let $A(x)$ be as in (1), then we define

$$\operatorname{den} A(x) = \operatorname{lcm}_{i=1}^{n} \operatorname{lcm}_{j=1}^{n} \operatorname{den}(a_{ij}(x)), \quad \operatorname{den} A^{-1}(x) = \operatorname{lcm}_{i=1}^{n} \operatorname{lcm}_{j=1}^{n} \operatorname{den}(\tilde{a}_{ij}(x)).$$

If $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T \in k(x)^n$ then $\operatorname{den} F(x) = \operatorname{lcm}_{i=1}^{n} \operatorname{den} F_i(x)$, and $\operatorname{val}_{p(x)} F(x) = \min_{i=1}^{n} \operatorname{val}_{p(x)} F_i(x)$. A solution $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T \in k(x)^n$ of (1) as well as a solution $F(x) \in k(x)$ of (2), (3) is a *rational* solution. If $\operatorname{den} F(x) \neq 1$ then this solution is *non-polynomial*, and *polynomial* otherwise.

If $p(x) \in \operatorname{Irr}(k[x])$, $f(x) \in k[x] \setminus \{0\}$ then we define the finite set

$$\mathcal{N}_{p(x)}(f(x)) = \{m \in \mathbb{Z} : p(x+m) | f(x)\}. \tag{7}$$

If $\mathcal{N}_{p(x)}(f(x)) = \varnothing$ then we define $\max \mathcal{N}_{p(x)}(f(x)) = -\infty$, $\min \mathcal{N}_{p(x)}(f(x)) = +\infty$.

From now on we use the notation

$$V(x) = b_n(x-n), \quad W(x) = b_0(x)$$

for equation (3), and

$$V(x) = u_1(x-1), \quad W(x) = u_0(x),$$

where $u_1(x) = \operatorname{den} A(x)$, $u_0(x) = \operatorname{den} A^{-1}(x)$, for system (1).

For $f(x), g(x) \in k[x] \setminus \{0\}$ we define their *dispersion set*:

$$\mathrm{ds}(f(x), g(x)) = \{d \in \mathbb{N} \ : \ \deg \gcd(f(x), g(x+d)) > 0\}$$

and their *dispersion*:

$$\mathrm{dis}\,(f(x), g(x)) = \max(\mathrm{ds}(f(x), g(x)) \cup \{-\infty\}).$$

The dispersion is equal to $-\infty$ iff $\deg \gcd(f(x), g(x+d)) = 0$ for all $d \in \mathbb{N}$, and belongs to $\mathbb{N}$ otherwise. The set $\mathrm{ds}(f(x), g(x))$ can be computed as the set of all integer non-negative roots of the polynomial $\mathrm{Res}_x(f(x), g(x+d)) \in k[d]$. However this set can be obtained faster if one resorts to the approach from [21] based on the full factorization of $f(x)$ and $g(x)$.

If a non-polynomial rational solution exists then the set $\mathrm{ds}(V(x), W(x))$ is not empty ([2, 5]).

3 Algorithms for Constructing Universal Denominators and Denominator Bounds

In Sect. 3.1, 3.2, 3.3, 3.4 we review algorithms for constructing universal denominators. In Sect. 3.5, 3.6 we consider an algorithm for constructing denominator bounds (versions of the algorithm from [17] given in [16]).

3.1 The Algorithm $\mathbf{A}_D$ from [4, 5]

The algorithm is as follows:

Find $\mathcal{D} = \mathrm{ds}(V(x), W(x))$. If $\mathcal{D} = \varnothing$ then terminate the algorithm with the result $U(x) = 1$ (we suppose below that $\mathcal{D} = \{d_1, d_2, \ldots, d_s\}$ and $d_1 > d_2 > \cdots > d_s$, $s \geqslant 1$). Set $U(x) = 1$ and successively for $m = 1, 2, \ldots, s$ execute the following group of assignments:
$P(x) = \gcd(V(x), W(x + d_m))$
$V(x) = V(x)/P(x)$
$W(x) = W(x)/P(x - d_m)$
$U(x) = U(x) \prod_{i=0}^{d_m} P(x - i)$.
The final value of $U(x)$ is a universal denominator for equations (2), (3) or, resp., system (1).

We will refer to this algorithm as $\mathbf{A}_D$. This algorithm is exploited in current versions of Maple [23]:
LREtools[ratpolysols], LinearFunctionSystems[UniversalDenominator].

3.2 The Algorithm from [12]

In [12] a more general problem than the search for rational solutions of system (1) was solved. However, the algorithm from [12, Prop. 3] can be used to compute a

universal denominator $u(x)$ related to (1). Using our notation (setting in addition $d = \operatorname{dis}(V(x), W(x))$) this algorithm may be represented as follows.

Consider the sequence of polynomials $\{(V_j(x), W_j(x), P_j(x))\}$ defined inductively as:

$$V_0(x) = V(x), \quad W_0(x) = W(x), \quad P_0(x) = \gcd(V(x), W(x+d)),$$

and for $j = 1, 2, \ldots, d$,

$V_j(x) = V_{j-1}(x)/P_{j-1}(x)$,

$W_j(x) = W_{j-1}(x)/P_{j-1}(x-d+j-1)$,

$P_j(x) = \gcd(V_j(x), W_j(x+d-j))$.

Then $u(x) = \prod_{j=0}^{d} \prod_{i=0}^{d-j} P_j(x-i)$.

3.3 The Algorithm $\mathbf{A}_U$ from [16]

An explicit formula for a lower bound of $\operatorname{val}_{p(x)} F(x)$ can be found in [16]: if $F(x)$ is a rational solution of equation (3) or system (1) then for any $p(x) \in \operatorname{Irr}(k[x])$ we have $\operatorname{val}_{p(x)} F(x) \geqslant -\gamma_{p(x)}$, where

$$\gamma_{p(x)} = \min \left\{ \sum_{l \in \mathbb{N}} \operatorname{val}_{p(x+l)} V(x), \ \sum_{l \in \mathbb{N}} \operatorname{val}_{p(x-l)} W(x) \right\}. \tag{8}$$

This formula was used in [16] as a base for the new algorithm $\mathbf{A}_U$ for computing a universal denominator. This algorithm can be divided into two steps. In the first step, $\mathbf{A}_U$ constructs a finite set M of irreducible polynomials that are candidates for divisors of denominators of rational solutions. At the second step, for each $p(x) \in M$ this algorithm computes the product $\prod_{p(x) \in M} p^{\gamma_{p(x)}}(x)$ which gives a universal denominator related to a given equation or system.

We define

$$M = \left\{ p(x) \in \operatorname{Irr}(k[x]) \ : \ \min \mathcal{N}_{p(x)}(W(x)) \leqslant 0, \ \ \max \mathcal{N}_{p(x)}(V(x)) \geqslant 0 \right\}.$$

For constructing this set the full factorization of polynomials $V(x), W(x)$ has to be found. Then we find the finite set $Q \subset \operatorname{Irr}(k[x])$ such that $q(x) \in Q$ iff $\min \mathcal{N}_{q(x)}(W(x)) = 0$, and $\max \mathcal{N}_{q(x)}(V(x)) \geqslant 0$. Let $Q = \{q_1(x), q_2(x), \ldots, q_s(x)\}$, $s \geqslant 1$. For each $1 \leqslant i \leqslant s$ we consider

$$M_{q_i(x)} = \{q_i(x), q_i(x+1), \ldots, q_i(x+d_i)\}, \tag{9}$$

where $d_i = \max \mathcal{N}_{q_i(x)}(V(x))$. We have $M = \bigcup_{i=1}^{s} M_{q_i(x)}$.

3.4 An Improved Version of the Algorithm $\mathbf{A}_U$ (the Algorithm $\mathbf{A}'_U$ from [7])

As it is described above the algorithm $\mathbf{A}_U$ contains two steps: the construction of the set M and the computation of $\gamma_{p(x)}$ using (8) for all $p(x) \in M$, which results in the universal denominator. Formula (8) contains the sums by $l \in \mathbb{N}$. In spite of the

fact that $\mathbb{N}$ is infinite, the sums have only finite number of summands corresponding to the irreducible factors of $V(x)$ and $W(x)$, which are equal to non-negative and non-positive shifts of $p(x)$, respectfully (the corresponding valuations are equal to the exponents of such factors in the factorization of $V(x)$ and $W(x)$). It is clear that when we compute (8) for $p(x) = q_i(x+j) \in M_{q_i(x)}$ (where $M_{q_i(x)}$ is as in (9)), the corresponding $\gamma_{q_i(x+j)}$ might be equal for many successive j. Indeed if we have computed $\gamma_{q_i(x)}$, and after that we compute $\gamma_{q_i(x+j)}$ for j from 1 to d_i, then the value can be changed only for those j for which there is an irreducible factor of $V(x)$ or $W(x)$ equal to $q_i(x+j)$ (such *critical* points can be computed in advance while constructing the set M). The consideration is a basis for the improved version of the algorithm $\mathbf{A}_U$ (more details can be found in [7, Sect. 3.4]).

We will refer to this detailed (improved) version of $\mathbf{A}_U$ as $\mathbf{A}'_U$.

3.5 The Algorithm $\mathbf{A}_B$ from [17, 16] (the case of a system)

The algorithm from [17] was modified in [16, Sect. 3, 4] in two directions: first, instead of complex numbers irreducible polynomials from $k[x]$ are considered, and second, the set of the irreducible polynomials which are used to find lowers bounds of valuations is constructed in a specific way (the modified version $\mathbf{A}_B$ uses the same set M as the algorithms $\mathbf{A}_U$, $\mathbf{A}'_U$ while the algorithm from [17] uses another set $\bar{S}$; even when $k = \mathbb{C}$ we have $M \subset \bar{S}$ and M is a proper subset of $\bar{S}$ in a large number of cases).

Let $A(x)$ be as in (1). We define $A_N(x) = A(x-1)A(x-2)\dots A(x-N)$ and $A_{-N} = A^{-1}(x)A^{-1}(x+1)\dots A^{-1}(x+N-1)$ for each positive integer N. Then $Y(x) = A_N(x)Y(x-N)$ for each solution of (1), $N = \pm 1, \pm 2, \dots$ We define $B(p(x), N, i)$ as the minimum of the valuations at $p(x)$ of the entries in the i-th row of $A_N(x)$. The algorithm $\mathbf{A}_B$ for the case of a system (1) is as follows. Computing successively matrices $A_N(x)$ for $N = 1, 2, \dots, d+1$ we find for each t such that $1 \leqslant t \leqslant s$ and $d_t \geqslant N-1$ the values $B(q_t(x+d_t-N+1), N, i)$, $i = 1, 2, \dots, n$, which give us left-hand bounds for $\mathrm{val}_{q_t(x+d_t-N+1)}Y_i(x)$, $i = 1, 2, \dots, n$. Analogously we compute successively matrices A_{-N} for $N = 1, 2, \dots, d+1$ and find for each t such that $1 \leqslant t \leqslant s$ and $d_t \geqslant N-1$ the values $B(q_t(x+N-1), -N, i)$, $i = 1, 2, \dots, n$, which give us right-hand lower bounds for $\mathrm{val}_{q_t(x+N-1)}Y_i(x)$, $i = 1, 2, \dots, n$. We have two lower bounds for each of the valuations $\mathrm{val}_{q_t(x+j)}Y_i(x)$, $i = 1, 2, \dots, n$, $t = 1, 2, \dots, s$, $j = 0, 1, \dots, d_t$, and can take the maximal one, we denote it by $\alpha_{i,j,t}$. The rational functions $R_i(x) = \prod_{\substack{1 \leqslant t \leqslant s \\ 0 \leqslant j \leqslant d_t}} q_t^{\alpha_{i,j,t}}(x+j)$, $i = 1, 2, \dots, n$, are denominator bounds.

3.6 The Algorithm $\mathbf{A}_B$ from [17, 16] (the scalar case)

In [17] the algorithm is described only for systems of the form (1). Scalar equations (2), (3) are assumed ([17, Sect. 3]) to be transformed to the system with the companion matrix $A(x)$ of the initial scalar equation. But the matrix operations are quite costly. A scalar version of the algorithm was given in [16]. We describe this version assuming that the ground field is an arbitrary field k of characteristic 0.

In the scalar case for an arbitrary non-zero integer N we can construct equations

$$y(x) = v_{N,\,n-1}(x)y(x-N) + \cdots + v_{N,\,0}(x)y(x-N-n+1) + v_{N,\,-1}(x), \qquad (10)$$

with rational function coefficients which are satisfied by all rational solutions of (2) and (3). Let $p(x) \in \mathrm{Irr}(k[x])$, $N \in \mathbb{Z} \setminus \{0\}$. We define $B(p(x), N)$ as the minimum of the valuations at $p(x)$ of the coefficients $v_{N,\,-1}(x), v_{N,\,0}(x), \ldots, v_{N,\,n-1}(x)$ in (10).

Constructing successively equations (10) for $N = 1, 2, \ldots, d+1$ we find the value $B(q_t(x+d_t-N+1), N)$ for each t such that $1 \leqslant t \leqslant s$ and $d_t \geqslant N-1$, which gives us the left-hand bound for $\mathrm{val}_{q_t(x+d_t-N+1)}y(x)$. Similarly we construct successively equations (10) for $N = -1, -2, \ldots, -d-1$ and find the value $B(q_t(x-N-1), N)$ for each t such that $1 \leqslant t \leqslant s$ and $d_t \geqslant -N-1$, which gives us right-hand lower bounds for $\mathrm{val}_{q_t(x-N-1)}y(x)$. We have two lower bounds for each of the valuations $\mathrm{val}_{q_t(x+j)}y(x)$, $t = 1, 2, \ldots, s$, $j = 0, 1, \ldots, d_t$, and can take the maximal one, we denote it by $\beta_{j,t}$. The rational function $R(x) = \prod_{\substack{1\leqslant t\leqslant s\\ 0\leqslant j\leqslant d_t}} q_t^{\beta_{j,t}}(x+j)$ is a denominator bound.

4 Complexity Comparison of $\mathbf{A}_D$ and $\mathbf{A}'_U$

Proposition 1. *([13, 7]) The universal denominators computed by the algorithm from [12] and $\mathbf{A}_D$ coincide for any given $V(x), W(x)$. Intermediate polynomials computed by $\mathbf{A}_D$ are also computed as intermediate polynomials by the algorithm from [12].*

We now give a complexity analysis of $\mathbf{A}_D$ and $\mathbf{A}'_U$.

Let $l = \max\{\deg V(x), \deg W(x)\}$, $d = \mathrm{dis}\,(V(x), W(x))$, and $T_{\gcd}(l)$ be the complexity of the gcd computation for two polynomials whose maximal degree is l. We compare the complexities $T_{\mathbf{A}_D}(l,d)$ and $T_{\mathbf{A}'_U}(l,d)$ of $\mathbf{A}_D$ and $\mathbf{A}'_U$. In this context, the complexity is the number of the field operations in k in the worst case.

Proposition 2. *([7]) If $T_{\gcd}(l)/(l\log l) \to \infty$ then the difference $T_{\mathbf{A}_D}(l,d) - T_{\mathbf{A}'_U}(l,d)$ is positive for almost all $l, d \in \mathbb{N}^+$ and*

$$T_{\mathbf{A}_D}(l,d) - T_{\mathbf{A}'_U}(l,d) = \begin{cases} \sum_{i=0}^{l} T_{\gcd}(i) + O(l\log l), & \text{if } d \geqslant l, \\ \sum_{i=l-d}^{l} T_{\gcd}(i) + O(l\log l), & \text{if } d < l. \end{cases} \qquad (11)$$

In the next proposition we use the Ω-notation which is very common in complexity theory ([20]). Unlike O-notation which is used for describing upper asymptotical bounds, the Ω-notation is used for describing lower asymptotical bounds.

Proposition 3. *([7]) Let $T_{\gcd}(l) = \Omega(l^\alpha)$, $\alpha > 1$. Then the difference $T_{\mathbf{A}_D}(l,d) - T_{\mathbf{A}'_U}(l,d)$ is positive for almost all $l, d \in \mathbb{N}^+$ and is $\Omega(S(l,d))$, where*

$$S(l,d) = \begin{cases} l^{\alpha+1}, & \text{if } d \geqslant l, \\ dl^\alpha, & \text{if } d < l. \end{cases}$$

To the authors' knowledge $T_{\gcd}(l) = \Omega(l^{\alpha})$, $\alpha > 1$, for the algorithms now in use in actual practice for gcd computations.

The fast Euclidean algorithm [14, Ch. 11] has complexity $O(l\ \log^2 l\ \log\log l)$ if Fast Fourier Transform is used to multiply polynomials. But this version of the fast Euclidean algorithm is not practical due to a big constant hidden in O. Nevertheless, if we suppose that the fast Euclidean algorithm is used and the estimate $\Omega(l\ \log^2 l\ \log\log l)$ (or, even $\Omega(l\ \log^2 l)$) is valid for the complexity of this algorithm then by Proposition 2 the difference $T_{\mathbf{A}_D}(l,d) - T_{\mathbf{A}'_U}(l,d)$ is positive (i.e., $T_{\mathbf{A}'_U}(l,d) < T_{\mathbf{A}_D}(l,d)$) for almost all $l, d \in \mathbb{N}^+$.

The reason of the appearance of the factor $\log l$ in formulas of Proposition 2 is the complexity $O(l \log l)$ of critical points sorting (see Sect. 3.4). We consider a comparison of two integer numbers as an operation in k (integer numbers are in k due to its zero characteristic). However if we take into account only arithmetic operations in k then the factor $\log l$ can be omitted. In this case we can take $\alpha \geqslant 1$ in the hypothesis of Proposition 3, and do not consider separately the fast Euclidean algorithm.

Thus the algorithm from [12] and the algorithms $\mathbf{A}_D, \mathbf{A}'_U$ produces the same universal denominators, therewith $\mathbf{A}'_U$ has the lowest complexity among them.

5 Complexity Comparison of $\langle \mathbf{A}_B \rangle$ and $\langle \mathbf{A}'_U \rangle$

The complexity of $\mathbf{A}_B$ is greater than the complexity of $\mathbf{A}_U$ ([16, Th. 2(i)]) and therefore it is greater than the the complexity of $\mathbf{A}'_U$. However the algorithm $\mathbf{A}_B$ gives quite exact lower bounds. A combination of one of the algorithms $\mathbf{A}'_U, \mathbf{A}_B$ with an algorithm (the same in both cases) for finding all polynomial solutions gives the algorithms $\langle \mathbf{A}'_U \rangle$, $\langle \mathbf{A}_B \rangle$ for constructing all rational solutions. We compare below the complexities of $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ ([8]).

Consider the scalar case. Some natural suppositions about the used algorithm for finding all polynomial solutions allow to show that the complexity of $\langle \mathbf{A}'_U \rangle$ is less than the complexity of $\langle \mathbf{A}_B \rangle$ (as in Sect. 4 the complexity is the number of the field operations in k in the worst case). For this investigation we need the notion of the height of an equation.

It is known that for any equation $L(y) = \psi(x)$ of the form (3) one can construct its *indicial equation* $I(\lambda) = 0$ at ∞ (which is an algebraic equation) and an integer number ω (which we call the *increment* of the equation) such that the degree of any polynomial solution of $L(y) = \psi(x)$ does not exceed the *height* of $L(y) = \psi(x)$:

$$h = \max\{\deg \psi - \omega,\ \tilde{\lambda}\}, \tag{12}$$

where

$$\tilde{\lambda} = \max(\{\lambda \in \mathbb{N} : I(\lambda) = 0\} \cup \{-\infty\}). \tag{13}$$

Recall that to get $I(\lambda)$ and ω one can rewrite (3) using the operator $\Delta = \phi - 1$ instead of the shift operator ϕ: $L = c_n(x)\Delta^n + \cdots + c_1(x)\Delta + c_0(x)$. Then

$$\omega = \max_{0 \leqslant j \leqslant n} (\deg c_j - j), \quad I(\lambda) = \sum_{\substack{0 \leqslant j \leqslant n \\ \deg a_j - j = \omega}} \mathrm{lc}(c_j)\lambda(\lambda-1)\ldots(\lambda-j+1).$$

We suppose that the polynomial solutions algorithm which is used by $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ computes first the height (12) and then computes polynomial solutions using the height as an upper bound for their degrees.

For an equation $L(y) = \psi(x)$ be of the form (3) we set

$l = \max\{\deg b_0(x), \deg b_1(x), \dots, \deg b_n(x)\}$,

$d = \operatorname{dis}(b_n(x-n), b_0(x))$,

$n = \operatorname{ord} L$,

$h =$ the height of the equation.

The quadruple (l, d, n, h) is the *combined size* (or just the *size* for short) of the equation $L(y) = \psi(x)$. We consider the complexities $T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h)$, $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h)$ of the algorithms $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$.

Using the algorithm $\mathbf{A}'_U$ (resp., $\mathbf{A}_B$) on the first step and clearing denominators after the corresponding substitution one gets an equation with polynomial coefficients and a polynomial right-hand side. This equation will be called the U-image (resp. B-image) of the original equation. On the final step the algorithms $\langle \mathbf{A}'_U \rangle$, $\langle \mathbf{A}_B \rangle$ find all polynomial solutions of the U- (resp. B-) image of the original equation.

The following lemma is a consequence of Remark 1 from [15]:

Lemma 1. *Let $F(x) \in k(x) \setminus \{0\}$. Let $K(z) = \chi(x)$ be the equation that we get clearing denominators after the substitution $y(x) = z(x)F(x)$ into the original equation $L(y) = \psi(x)$. Let $I(\lambda) = 0$ and $I'(\lambda) = 0$ be the indicial equations for $L(y) = \psi(x)$ and, resp. for $K(z) = \chi(x)$. Then $I'(\lambda) = I(\lambda + \deg \operatorname{num} F(x) - \deg \operatorname{den} F(x))$.*

Let the size of the original equation be (l, d, n, h). Denote by (l_U, d_U, n_U, h_U) the size of the U-image (of course, $n_U = n$). In addition denote by r_U the degree of the right-hand side of the U-image.

Lemma 2. *([8]) Let the size of an equation $L(y) = \psi(x)$ be (l, d, n, h). Then*

(i) the set M of this equation contains no more than $l(d+1)$ elements (irreducible polynomials) and

$$l_U \leqslant l(n-1), \quad h_U \leqslant hl + (d+1), \quad r_U \leqslant h + l(d+n);$$

(ii) the height of the equation

$$\begin{aligned} f(x+n+d)y(x+n) + (x+1)^l y(x+n-1) + \dots \\ \cdots + (x+1)^l y(x+1) + f(x)y(x) = (x+1)^{h+l}, \end{aligned} \tag{14}$$

where

$$f(x) = \prod_{t=1}^{l} \left(x + \frac{1}{t+1} \right),$$

is equal to h (thus the size of this equation is (l, d, n, h)), and the set M has $l(d+1)$ elements, the algorithm $\mathbf{A}'_U$ gives for this equation the universal denominator of degree $l(d+1)$, and

$$h_U = h + l(d+1), \quad l_U = l(n-1) + h, \quad r_U = h + l(d+n)$$

(by (i) these values are the maximal possible for the equations of size (l, d, n, h)).

(Lemma 1 plays a significant role in the proof of (i).)

Note that if we remove the component h from the combined size then the complexity of each of $\langle \mathbf{A}_B \rangle$, $\langle \mathbf{A}'_U \rangle$ will be equal to ∞. When the combined size as the quadruple (l, d, n, h) is fixed, the cost of the rational solutions computation is bounded.

Denote by $\mathcal{S}_{l,d,n,h}$ the set of all equations of size (l, d, n, h) and by $\mathcal{U}_{l,d,n,h}$ the subset of $\mathcal{S}_{l,d,n,h}$ which consists of the equations such that the cost of finding (or absence recognizing) polynomial solutions of the U-image of each of them is maximal among all equations from $\mathcal{S}_{l,d,n,h}$. The set $\mathcal{U}_{l,d,n,h}$ may contain more than one equation.

Proposition 4. *([8]) Let the algorithm which is used for finding polynomial solutions be such that equation (14) for any l, d, n, h is in $\mathcal{U}_{l,d,n,h}$. Then $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h) > T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h)$, and $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h) - T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h) = \Omega(dln)$.*

Our assumption that (14) belongs to $\mathcal{U}_{l,d,n,h}$ is quite reasonable. We do not specify the used algorithm for finding polynomial solutions but suppose that the algorithm uses the height of the equation as a bound for degrees of solutions. By Lemma 2 the height of the U-image of (14) is maximal and the U-image itself is maximally cumbersome among the U-images of equations from $\mathcal{S}_{l,d,n,h}$.

In the case of a system the algorithm $\langle \mathbf{A}_B \rangle$ needs to construct matrices A_N, which is even more costly than constructing equations (10).

6 Implementation and Experiments

The implementation of $\mathbf{A}_D$ is available in Maple as an internal procedure of LREtools package. We implemented $\mathbf{A}'_U$ to perform comparison experiments. Note that $\mathbf{A}'_U$ is based on the full factorization of the given polynomials $V(x)$ and $W(x)$. Our implementation uses the result of the factorization not only to construct the set M of irreducible polynomials, but also computes (8) using it. It is not the case for the implementation of $\mathbf{A}_D$ in Maple. It uses the procedure LREtools[dispersion] to compute the dispersion of polynomials which implements the algorithm [21], i.e., uses the full factorization. But the next steps of $\mathbf{A}_D$ are implemented as presented in Sect. 3.1 not exploiting the result of the factorization of the previous step.

We also performed comparison experiments for $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$. $\langle \mathbf{A}'_U \rangle$ is implemented combining our implementation of $\mathbf{A}'_U$ and Maple's LREtools[polysols] (the scalar case) and LinearFunctionalSystems[PolynomialSolution] (the case of a system) for finding all polynomial solutions. We have also implemented $\mathbf{A}_B$ (both for scalar and system cases); $\langle \mathbf{A}_B \rangle$ is implemented combining this implementation of $\mathbf{A}_B$ and Maple's LREtools[polysols] (the scalar case) and LinearFunctionalSystems[PolynomialSolution] (the case of system) for finding all polynomial solutions.

6.1 Comparison of $\mathbf{A}'_U$ and $\mathbf{A}_D$

We performed several experiments to compare $\mathbf{A}'_U$ and $\mathbf{A}_D$. The result of one of the experiment is presented below.

The both algorithms were applied to the input set:

$$V(x) = W(x) = \prod_{i=1}^{l}(x+m+i+1/i)(x-m-i+1/i)$$

for $m = 20, 100, 500, 2500$, $l = 1, 15, 30, 45, 60$. The corresponding found universal denominators are: $\prod_{i=1}^{l}\prod_{j=-m-i}^{m+i}(x-j+1/i)$. Results for the input set, in seconds:

	m=20		m=100		m=500		m=2500	
	$\mathbf{A}'_U$	$\mathbf{A}_D$	$\mathbf{A}'_U$	$\mathbf{A}_D$	$\mathbf{A}'_U$	$\mathbf{A}_D$	$\mathbf{A}'_U$	$\mathbf{A}_D$
l=1	0.016	0.015	0.000	0.000	0.000	0.016	0.031	0.031
l=15	0.078	0.375	0.109	0.422	0.172	0.531	0.578	1.032
l=30	0.359	2.890	0.407	3.063	0.531	3.484	1.266	5.344
l=45	0.860	10.641	0.796	11.547	1.516	13.234	3.078	17.656
l=60	2.406	31.187	2.719	33.484	2.657	37.125	4.766	44.797

The input set corresponds near to the worst case for both algorithms $\mathbf{A}'_U$ and $\mathbf{A}_D$, and an advantage of $\mathbf{A}'_U$ is evident.

The results of other experiments to compare $\mathbf{A}'_U$ and $\mathbf{A}_D$ are available in [7]. All the experiments confirmed the practical benefit of $\mathbf{A}'_U$.

6.2 Comparison of $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ (the scalar case)

We performed several comparison experiments to compare $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$. The result of one of the experiment is presented below.

$\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ were applied to 27 equations (14) from Lemma 2: $h = 6$ for all of them, $n = 3, 6, 9$, $l = 2, 4, 6$, $d = 5, 10, 15$. The resulting $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h) - T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h)$, in seconds:

n	l	d=5	d=10	d=15
3	2	0.546 - 0.141 = 0.405	1.438 - 0.125 = 1.313	2.796 - 0.203 = 2.593
3	4	1.359 - 0.235 = 1.124	4.188 - 0.375 = 3.813	9.594 - 0.812 = 8.782
3	6	2.703 - 0.375 = 2.328	10.172 - 0.969 = 9.203	24.937 - 1.734 = 23.203
6	2	0.813 - 0.234 = 0.579	2.015 - 0.328 = 1.687	4.625 - 0.453 = 4.172
6	4	2.313 - 0.672 = 1.641	7.515 - 1.063 = 6.452	17.235 - 2.140 = 15.095
6	6	5.094 - 1.547 = 3.547	18.484 - 3.156 = 15.328	45.656 - 6.094 = 39.562
9	2	1.047 - 0.563 = 0.484	3.062 - 0.671 = 2.391	6.610 - 1.063 = 5.547
9	4	3.687 - 1.328 = 2.359	11.063 - 2.516 = 8.547	25.484 - 4.265 = 21.219
9	6	8.281 - 3.172 = 5.109	28.453 - 6.875 = 21.578	69.672 - 13.328 = 56.344

The results correspond to Proposition 4, and moreover the difference $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h) - T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h)$ grows even faster than d for the fixed n, l.

Note that h is not in the table: $h = 6$ for all equations. Additional experiments with $h = 2, 4$ show that the results are almost independent from h – the growth of h in 3 times from 2 to 6 causes the change of $T_{\langle \mathbf{A}_B \rangle}(l, d, n, h) - T_{\langle \mathbf{A}'_U \rangle}(l, d, n, h)$ in less than 3% for fixed l, d, n).

The results of other experiments to compare $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ are available in [8]. All the experiments' results corresponded to the proposition 4.

6.3 Comparison of $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ (the case of a system)

In the experiment $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$ were applied to 3 input sets. Each set contained 20 equations of order $n = 2, 3, 4$ correspondingly. Each system had fundamental system of solutions consisting of randomly generated rational functions. Note that such inputs were more convenient for $\langle \mathbf{A}_B \rangle$, since $\mathbf{A}_B$ constructed exact bounds (without possibility to reduce) for such inputs in accordance with [17, Theorem 1].

Results for the experiment:

n	$\deg U(x)$ $\mathbf{A}'_U$	$\deg \operatorname{den} R_i(x)$ $\mathbf{A}_B$	Time $\langle \mathbf{A}'_U \rangle$	Time $\langle \mathbf{A}_B \rangle$
2	7-39	2-8	7.216	22.922
3	18-49	3-21	38.859	169.906
4	36-74	5-28	176.829	836.172

Each row in the table corresponds to the input set with the parameter n. The other columns present ranges of degrees of denominators found by $\mathbf{A}'_U$ and $\mathbf{A}_B$ for the systems in the corresponding set, as well as the total time for finding rational solutions for all systems in the set taken by $\langle \mathbf{A}'_U \rangle$ and $\langle \mathbf{A}_B \rangle$.

The time of $\langle \mathbf{A}_B \rangle$ was greater than the time of $\langle \mathbf{A}'_U \rangle$ for all the input sets in spite of the exact bounds found by $\mathbf{A}_B$ (note that $\mathbf{A}'_U$ found less exact bound as it is seen from the ranges of degrees of denominators in the table).

7 Scalar Homogeneous Equations Having no Rational Solutions

Many equations (even a "majority" of them) have no (non-zero) rational solutions. However if one uses algorithms like $\langle \mathbf{A}'_U \rangle$ or $\langle \mathbf{A}_B \rangle$ then the absence of such solutions will be recognized only in the last step of computation when U-, resp. B-image of the original equation is constructed. In [15] some changes in the scheme of these algorithms for the case of scalar homogeneous equation $L(y) = 0$ with $L = b_n(x)\phi^n + \cdots + b_1(x)\phi + b_0(x)$, $\psi(x), b_1(x), \ldots, b_{n-1}(x) \in k[x]$, $b_0(x), b_n(x) \in k[x] \setminus \{0\}$, were discussed. In any case these changes do not increase the computation cost, but allow one quite often to predict the absence of rational solutions in an early stage of computation and to stop the work. The changes are based on Lemma 1.

Proposition 5. *([15]) Let* $d = \operatorname{dis}(b_n(x - n), b_0(x)) \geqslant 0$, *and let* $\tilde{\lambda}$ *be as in (13). In this case,*

(i) if the inequality

$$\tilde{\lambda} + (d+1)\min\{\deg b_0(x),\ \deg b_n(x)\} \geqslant 0 \tag{15}$$

is not valid then $L(y) = 0$ has no rational solutions,

(ii) if $F(x)$ is an arbitrary denominator bound computed for $L(y) = 0$ and $K(z) = 0$ is the equation that we get clearing denominators after the substitution $y(x) = z(x)F(x)$ into the original equation $L(y) = 0$ (e.g., $F(x) = \frac{1}{U(x)}$ or $F(x) = R(x)$ where $U(x), R(x)$ are the result of applying $\mathbf{A}'_U$, resp. $\mathbf{A}_B$ to $L(y) = 0$) then $\tilde{\lambda} - \deg \operatorname{num} F(x) + \deg \operatorname{den} F(x)$ is an upper bound for degrees of all polynomial solutions of $K(z) = 0$.

Some simple examples. For $L = 2(x+2)\phi + (2x+3)$ the indicial equation is $2\lambda + 1 = 0$. There are no integer roots. Thus $L(y) = 0$ has no rational solutions. For $L = (x+1)(x^2+1)\phi - x(x^2-4x+1)$ the indicial equation is $\lambda + 5 = 0$. We have $\tilde{\lambda} = -5$, $d = 0$. Inequality (15) is not valid. Thus $L(y) = 0$ has no rational solutions. For $L = (x+2)\phi - x$ we have $\lambda + 2 = 0$, $\tilde{\lambda} = -2$, $d = 1$. Inequality (15) is valid. If the algorithm $\mathbf{A}'_U$ is used, then we get $R(x) = \frac{1}{x(x+1)}$. We have $-2 - 0 + 2 = 0$. Thus by Proposition 5(ii) 0 is an upper bound for degrees of polynomial solutions of the U-image, which is $(x+1)y(x+1) - xy(x) = 0$. Constants and only them are polynomial solutions.

Acknowledgments

The authors are grateful to M. Barkatou, M. van Hoeij, M. Kauers, M. Petkovšek and E. Zima for interesting discussions.

References

[1] *Abramov S.* Problems of computer algebra involved in the search for polynomial solutions of linear differential and difference equations. Moscow Univ. Comput. Math. Cybernet. **3** (1989) 63–68; transl. from Vestn. MGU. Ser. 15. Vychisl. mat. i kibernet. **3** (1989) 53–60

[2] *Abramov S.* Rational solutions of linear difference and differential equations with polynomial coefficients. USSR Comput. Math. Phys. **29** (1989) 7–12; transl. from Zh. vychisl. mat. mat. fyz. **29** (1989) 1611–1620

[3] *Abramov S.* Rational solutions of linear difference and q-difference equations with polynomial coefficients. In: ISSAC'98 Proceedings, ACM Press (1995) 303–308

[4] *Abramov S.* Rational solutions of linear difference and q-difference equations with polynomial coefficients. Programming and Comput. Software **21** (1995) 273–278; transl. from Programmirovanie **6** (1995) 3–11

[5] *Abramov S., Barkatou M.* Rational solutions of first order linear difference systems. In: ISSAC'98 Proceedings, ACM Press (1998) 124–131

[6] *Abramov S., Bronstein M., Petkovšek M.* On polynomial solutions of linear operator equations. In: ISSAC'95 Proceedings, ACM Press (1995) 290–295

[7] *Abramov S., Gheffar A., Khmelnov D.* Factorization of polynomials and gcd computations for finding universal denominators. CASC'2010 Proceedings (2010) 4–18

[8] *Abramov S., Gheffar A., Khmelnov D.* Rational solutions of linear difference equations: universal denominators and denominator bounds. Programming and Computer Software **2** 2011 (accepted)

[9] *Abramov S., van Hoeij M.* A method for the integration of solutions of Ore equations. In: ISSAC'97 Proceedings, ACM Press (1997) 172–175

[10] *Abramov S., van Hoeij M.* Integration of solutions of linear functional equations. Integral Transforms and Special Functions **8** (1999) 3–12

[11] *Barkatou M.* Rational solutions of systems of linear difference equations. J. Symbolic Computation **28** (1999) 547–567

[12] *Barkatou M.* Rational solutions of matrix difference equations: problem of equivalence and factorization. In: ISSAC'99 Proceedings, ACM Press (1999) 277–282

[13] *Chen W. Y. C., Paule P., Saad H. L.* Converging to Gosper's algorithm. Adv. in Appl. Maths. **41**(3) (2008) 351–364

[14] *von zur Gathen J., Gerhard J.* Modern Computer Algebra (Second Edition). Cambrige University Press (2003)

[15] *Gheffar A.* Linear differential, difference and q-difference homogeneous equations having no rational solutions. ACM Commun. Comput. Algebra (accepted)

[16] *Gheffar, A., Abramov, S.* Valuations of rational solutions of linear difference equations at irreducible polynomials. Adv. in Appl. Maths. (accepted)

[17] *van Hoeij M.* Rational solutions of linear difference equations. In: ISSAC'98 Proceedings, ACM Press (1998) 120–123

[18] *van Hoeij M., Levy G.* Liouvillian solutions of irreducible second order linear difference equations. In: ISSAC'2010 Proceedings, ACM Press (2010) 297–301.

[19] *Khmelnov D.E.* Search for polynomial solutions of linear functional systems by means of induced recurrences. Programming and Comput. Software **30** (2004) 61–67; transl. from Programmirovanie **2** (2004) 8–16

[20] *Knuth D.E.* Big omicron and big omega and big theta. ACM SIGACT News **8**(2) (1976) 18–23

[21] *Man Y.K., Wright F.J.* Fast polynomial dispersion computation and its application to indefinite summation. In: ISSAC'94 Proceedings, ACM Press (1994) 175–180

[22] *Petkovšek M.* Hypergeometric solutions of linear recurrences with polynomial coefficients. J. Symbolic Computation **14** (1992) 243–264

[23] Maple online help: http://www.maplesoft.com/support/help/