An Efficient Algorithm for Deriving Summation Identities from Mutual Recurrences*

BERKELEY R. CHURCHILL

Department of Mathematics, University of California, Santa Barbara
Santa Barbara, California 93117, United States of America

EDMUND A. LAMAGNA

Department of Computer Science and Statistics, University of Rhode Island, Kingston
Kingston, Rhode Island 02881, United States of America

This paper closes an algorithmic problem of summing a set of mutual recurrence relations with constant coefficients. Given an order d system of the form $A(n) = \sum_{i=1}^{d} M_i A(n-i) + G(n)$, where $A, G : \mathbb{N} \to K^m$ and $M_1, \ldots, M_d \in M_m(K)$ for some field K and natural number m, this algorithm computes the sum $\sum_{i=0}^{n} A(i)$ as a K-linear combination of $A(n), \ldots, A(n-d)$, the initial conditions and sums of the inhomogeneous term $G(n)$. The runtime of this algorithm is shown to be polynomial in m and d.

Keywords: mutual recurrence; summation; computer algebra

1. Problem Statement

An important task in computer algebra systems is evaluating indefinite sums, that is computing values $s_n = \sum_{k=0}^{n} a_k$ where the a_k is some sequence depending only on k. Today many functions can be summed, in part due to the pioneering work of several researchers [5], [7], [9]. Nonetheless, there are still countless instances where we lack algorithms to sum particular expressions, or the algorithms that exist are inefficient or produce undesirable outputs.

One area of interest is summing recurrence relations. Summing any a_k is a special case of computing the value of A_n where $A_n = A_{n-1} + a_n$ and $A_0 = a_0$. Recurrence relations arise frequently in algorithm analysis and numerical analysis of differential equations. The classical example is the Fibonacci sequence, defined as a function $F : \mathbb{N} \to \mathbb{N}$ given by $F(n) = F(n-1) + F(n-2) \quad \forall n \geq 2$ with $F(0) = 0$, $F(1) = 1$. It is well known[a] that this sequence satisfies the property $\sum_{i=0}^{n} F_i = F_{n+2} - 1$.

This identity is nice because it presents the sum in terms of the original Fibonacci symbol. An even trickier situation is a system of linear recurrences, often referred

---

[a]The proof is by induction. $F(0) = F(2) - 1$ holds true. Suppose $\sum_{i=0}^{n} F_i = F_{n+2} - 1$ and add F_{n+1} to both sides of the equation to verify the formula.

to as mutual recurrences in the literature. Consider the following example:

$A, B : \mathbb{N} \to \mathbb{Q}$ satisfy $A(n+2) - A(n+1) - A(n) - B(n) = 1$ and $-A(n) + B(n+2) - B(n+1) - B(n) = 1$ with $A(0) = B(0) = 0$ and $A(1) = B(1) = 1$. How could one write an algorithm that computes an identical expression for $\sum_{i=0}^{n} A(i)$ in terms of the symbols A and B themselves? In this paper we present an algorithm which can compute the sums of a mutual recurrence in a closed form any time the inhomogeneous term can be summed repeatedly.

2. Related Work

The primary inspiration for this work is Ravenscroft and Lamagna's work on summation of a single linear recurrence. They provide an efficient algorithm to express the sum of a homogeneous recurrence $A(n) = \sum_{i=1}^{d} m_i A(n-i)$ in terms of $A(n-d), \ldots, A(n-1)$ by using a "summing factor" to deal with recurrences that initially appear to be degenerate. They also provide a technique that handles some inhomogeneous terms [10].

Several authors study summing C-finite sequences (those determined by recurrence relations with constant coefficients). Greene and Wilf provide an algorithm to sum a general form of products of C-finite sequences [6]. Kauers and Zimmermann study determining whether summation relationships exist between different C-finite sequences [8].

Work on P-finite sequences (those determined by recurrence relations with polynomial coefficients) has also been done. Abramov and van Hoeij discuss summing P-finite sequences in terms of the original coefficients [1]. Chyzak generalizes the works of Gosper [5] and Zeilberger [13] to sum P-finite sequences that are not hypergeometric [3]. Schneider extends Karr's approach [7] to P-finite sequences as well [11].

Recently, [2] demonstrates that our problem regarding systems of mutual recurrences can be solved in the first order case, along with some other special cases. In this paper we demonstrate a reduction from the general problem to the first order case, and then provide a self-contained description of the algorithm.

3. Systems of Mutual Recurrences

Definition 3.1 (Mutual Recurrence). *Let K be a field, and $m, d \in \mathbb{Z}^+$. A system of mutual linear recurrence relations with constant coefficients on K of order d in m variables is a set of m functions $A_1(n), \ldots, A_m(n)$ mapping $\mathbb{N}$ into K satisfying*

$$\begin{pmatrix} A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \end{pmatrix} = M_1 \begin{pmatrix} A_1(n-1) \\ A_2(n-1) \\ \vdots \\ A_m(n-1) \end{pmatrix} + \cdots + M_d \begin{pmatrix} A_1(n-d) \\ A_2(n-d) \\ \vdots \\ A_m(n-d) \end{pmatrix} + \begin{pmatrix} g_1(n) \\ g_2(n) \\ \vdots \\ g_m(n) \end{pmatrix}$$

for some $M_1, \ldots, M_d \in M_m(K)$ and $g_1, \ldots, g_m$ mapping $\mathbb{N} \to K$. Typically we will refer to this as a "mutual recurrence".

We call the vector containing the $g_i(n)$ the inhomogeneous term, and this vector is typically denoted by G. If this inhomogeneous term is zero, the mutual recurrence is homogeneous. We call the values $\{A_i(j) : 1 \le i \le m, 0 \le j < d\}$ the *initial conditions* for the recurrence. The notation in this definition is used throughout the paper whenever a specific mutual recurrence is being considered.

Example 3.2. We will use the following example of a mutual recurrence to demonstrate computational procedures throughout the paper. For this example $m = 2$, so for convenience we use $A(n)$ to denote $A_1(n)$ and $B(n)$ to denote $A_2(n)$. $A(n) = 2A(n-1) + B(n-1)$, $B(n) = A(n-1) + 2B(n-2)$.

This may be written in the form stated in the definition as

$$\begin{pmatrix} A(n) \\ B(n) \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} A(n-1) \\ B(n-1) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} A(n-2) \\ B(n-2) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Here the order is $d = 2$ and, coincidentally, $m = 2$ as well. $\square$

4. Reduction to $d = 1$ Case

Summing a general mutual recurrence can be reduced to the $d = 1$ case. Let $A_1, \ldots, A_m$ be a mutual recurrence of order d. Define $B_{i,j}(n) = A_i(n + d - j)$ for $1 \le i \le m$ and $1 \le j \le d$. Then for $j > 1$, $B_{i,j}$ satisfies $B_{i,j}(n) = A_i(n + d - j) = B_{i,j-1}(n-1)$. For each i, $B_{i,1}$ may be written as a sum of $\mathrm{Span}\{B_{i',j'} : 1 \le i' \le m, 1 \le j' \le d\}$ and an inhomogeneous term. The initial conditions can be taken from the initial conditions for the original system; $B_{i,j}(0) = A_i(d - j)$. The $B_{i,j}(n)$ is a first-order system of mutual recurrences in K, and $\sum_{j=0}^{n} A_i(j) = \sum_{j=0}^{n} B_{i,d}(j)$. The representation of the new system has exactly one coefficient matrix of dimension $m \times d$. A naive implementation can derive each element in this matrix with $O(1)$ operations, so the total time for this reduction is $O(md)$. This process is identical to the well-known method of transforming linear systems of differential equations with constant coefficients into first-order systems.

Example 4.1. Following Example 3.2, define $W(n) = A(n + 1), X(n) = A(n), Y(n) = B(n+1)$ and $Z(n) = B(n)$. This gives the following system of mutual recurrences:

$$\begin{aligned} W(n) &= 2W(n-1) + Y(n-1) \\ X(n) &= W(n-1) \\ Y(n) &= W(n-1) + 2Z(n-1) \\ Z(n) &= Y(n-1). \end{aligned}$$

Solving this system provides the solutions to the original; $\sum_{i=0}^{n} A(i)$ and $\sum_{i=0}^{n} B(i)$ are exactly the same as $\sum_{i=0}^{n} X(i)$ and $\sum_{i=0}^{n} Z(i)$. $\qquad\qquad\square$

5. Summing First Order Mutual Recurrences

Consider a mutual recurrence of the form $X(n) = MX(n-1) + G(n)$, where

$$X(n) = (A_1(n) \ A_2(n) \ \cdots \ A_m(n))^t,$$

$G : N \to K$ is an inhomogeneous term and $M \in M_m(K)$. The goal is to compute $\sum_{j=0}^{n} A_i(j)$ for each $i \in \{1, \ldots, m\}$ and express the answer in terms of $\mathrm{Span}\{A_i(n-j) : 1 \le i \le m, 0 \le j \le 1\}$, the initial conditions, and an inhomogeneous term. For any function $f : \mathbb{N} \to K$ define $S(f(n)) = \sum_{i=1}^{n} f(i)$. S will be known as the summation operator. Recursively define $S_i^j(n) = S(S_i^{j-1}(n))$ and $S_i^1(n) = S_i(n) = S(A_i(n))$. This operator corresponds to the notion of summing that allowed Ravenscroft and Lamagna to symbolically sum linear recurrences [10].

It becomes convenient to factor the mutual recurrence into a matrix product as follows. In the following equation, the leftmost matrix is a block matrix, while the others are not. The following equation is identical to the definition provided earlier.

$$(I \quad -M) \begin{pmatrix} A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \\ \hline A_1(n-1) \\ A_2(n-1) \\ \vdots \\ A_m(n-1) \end{pmatrix} = G.$$

Since we will be computing S_i^j for many values of i and j, it becomes useful to expand this equation. Let k be a non-negative integer. The *k-padded representation of the mutual recurrence* is the matrix equation

$$
(0 \ 0 \ \cdots \ 0 \,|\, I \quad - M)
\begin{pmatrix}
S_1^k(n) \\
\vdots \\
S_m^k(n) \\
\hline
\vdots \\
\hline
S_1(n) \\
\vdots \\
S_m(n) \\
\hline
A_1(n) \\
\vdots \\
A_m(n) \\
\hline
A_1(n-1) \\
A_2(n-1) \\
\vdots \\
A_m(n-1)
\end{pmatrix}
= G.
$$

To simplify notation, equations of the above form are written as an augmented matrix,

$$
(0 \ 0 \ \cdots \ 0 \,|\, I \quad - M \,|\, G).
$$

The S operator now deserves some attention. Let V be a vector space over K with basis $\beta = \{A_i(n-j) : 1 \leq i \leq m, 0 \leq j \leq 1\} \cup \{S_i^j(n) : 1 \leq i \leq m, j \in \mathbb{N}\}$. Throughout our work S will only be applied to functions of the form $v(n) + i(n)$ where $v \in V$ and $i(n)$ is an inhomogeneous term that depends only on n, and never on any function of the A_i. Because S is linear, to understand S we need only understand how S acts on β; summing the inhomogeneous parts yields other inhomogeneous parts, and this may be accomplished via other methods [5], [7], [9].

From the definitions, we already know $S(S_i^j(n)) = S_i^{j+1}(n)$. The others are not hard to compute as $S(A_i(n-j)) = A_i(1-j) + A_i(2-j) + \cdots + A_i(n-j) = \sum_{k=1-j}^{0} A_i(k) + \sum_{k=1}^{n} A_i(k) - \sum_{k=n-j+1}^{n} A_i(k) = S_i(n) - \sum_{k=n-j+1}^{n} A_i(k) + \sum_{k=1-j}^{0} A_i(k)$. Therefore, applying S to a function of the form $v(n) + i(n)$ yields another function of the form $v(n) + i(n)$.

Notice that a row from an augmented matrix corresponds to an equation for the form $v(n) = i(n)$ for some $v \in V$ and inhomogeneous function i. Summing both sides of the equation and moving all inhomogeneous parts to the right-hand side gives $v'(n) = i'(n)$ for some new $v' \in V$ and inhomogeneous i. This can be re-written as a row of the matrix. This is the procedure of applying the summation operator to a row.

Example 5.1. Suppose we have this row of an augmented matrix corresponding to a first order mutual recurrence with $m = 2$,

$$(\begin{matrix} 0 & 0 & 3 & 5 \end{matrix} \mid -2 \ 6 \ 0 \ 0 \mid n+4).$$

This row corresponds to the equation

$$3S_1(n) + 5S_2(n) - 2A_1(n) + 6A_2(n) = -n - 4.$$

Applying the summation operator to each side yields

$$3S_1^2(n) + 5S_2^2(n) - 2(S_1(n) - A_1(0)) + 6(S_2(n) - A_2(0)) = -\frac{1}{2}n(n+1) - 4n,$$

and this is re-written as the augmented matrix row

$$\left(\begin{matrix} 3 & 5 & -2 & 6 \end{matrix} \ \middle| \ \begin{matrix} 0 & 0 & 0 & 0 \end{matrix} \ \middle| \ \frac{1}{2}n^2 + \frac{9}{2}n + 2A_1(0) - 6A_2(0) \right).$$

$\square$

In the following we will explicitly demonstrate the action of the summation operator on an augmented matrix. Suppose we start with

$$\left(0 \ 0 \ \cdots \ 0 \ B_j \ B_{j-1} \ \cdots \ B_1 \middle| C_1 \ C_2 \middle| G \right)$$

where $B_1, \ldots, B_j$ and C_1, C_2 are $m \times m$ matrices. In the expanded matrix equation, B_j is multiplied by the block matrix containing the $S_1^j(n), \ldots, S_m^j(n)$ as rows. A row of their product is therefore a linear combination of $S_i^j(n)$. Applying the summation operator creates an identical linear combination of $S_1^{j+1}(n), \ldots, S_m^{j+1}(n)$. The same logic applies to $B_{j-1}, \ldots, B_1$. In block matrix form, all the $B_1, \ldots, B_j$ appear to be shifted one block to the left after applying the summation operator. The result looks like

$$\left(0 \ 0 \ \cdots \ 0 \ B_j \ B_{j-1} \ \cdots \ B_1 \ * \middle| * \ * \middle| * \right).$$

To determine the block matrix represented by the leftmost asterisk, consider when $S_i(n)$ appears in the image of the S operator; it appears once for every occurrence of $A_i(n-j)$ (for any $j \in \mathbb{N}$) in the preimage. This implies $C_1 + C_2$ is the value of the leftmost asterisk.

For the kth asterisk to the right of the separator, the number of $A_i(n-k)$ in the image of S is given by the negation of the number of $A_i(n-l)$ in the preimage, where $l > k$. This result can be stated as a lemma.

Lemma 5.2. *Given an augmented block matrix of the form*

$$\left(0 \ 0 \ \cdots \ 0 \ B_j \ B_{j-1} \ \cdots \ B_1 \middle| C_1 \ C_2 \middle| G \right)$$

applying S to each row yields a new block matrix

$$\left(0\ 0\ \cdots\ 0\ B_j\ B_{j-1}\ \cdots\ B_1\ C_1+C_2 \middle| -C_2\ 0 \middle| G' \right)$$

where G' is some column matrix of inhomogeneous functions.

Starting with a k-padded representation of a mutual recurrence, we can apply the summation operator to each row up to k times. This is in addition to the standard row operations. Another permissible row operation is to duplicate a row of the left-hand matrix and the corresponding row of the inhomogeneous matrix.

The goal is to start with a k-padded representation of a recurrence, and use row operations to put the augmented matrix into the form

$$(0\ 0\ \cdots\ 0\ I\ |\ *\ *\ |\ *)$$

where $*$ denotes any matrix. Then one can back-substitute to compute identities $S_j(n) = v(n) + i(n)$ for each $1 \le j \le m$, where $v \in V$ and i is inhomogeneous.

6. Summation Algorithm

The following algorithm uses the above operations to solve this summation problem.

Algorithm 6.1.
Input: An order t mutual recurrence with s equations.
Output: Summation identities for each recurrence.

1. Apply the reduction algorithm from Section 4 to derive a first order mutual recurrence with $m = st$ equations. Let M be the coefficient matrix.
2. Compute $\mu(x)$, the minimal polynomial of M.
3. Count the number of factors of $(x-1)$ in $\mu(x)$ and call it k.
4. Compute the augmented matrix of the $k+1$-padded representation of the matrix.
5. Consider the entire augmented matrix as a single block row and duplicate it k times, so there are a total of $k+1$ block rows. Number the block rows from top to bottom starting at 1.
6. For $1 \le i \le k+1$, apply S to the $(k+2-i)^{\text{th}}$ block row i times.
7. Compute the row-reduced echelon form of the resulting matrix.
8. For $1 \le i \le m$, solve for $S_i(n)$ via backsubstitution.
9. Backsubstitute the solutions to the first order mutual recurrence into the original mutual recurrence.

Theorem 6.2. *The algorithm is correct.*

Proof.
After duplicating the block rows of the matrix, the matrix is

$$\begin{pmatrix} 0 \cdots 0 & I & -M & G \\ 0 \cdots 0 & I & -M & G \\ \vdots \ddots \vdots & \vdots & \vdots & \vdots \\ 0 \cdots 0 & I & -M & G \end{pmatrix}.$$

Using Lemma 5.2, applying the summation operator the appropriate number of times leaves a $(k+4)m \times (k+1)m$ matrix

$$\begin{pmatrix} M-I & -M & 0 & 0 & \cdots & 0 & 0 & 0 & * \\ 0 & M-I & -M & 0 & \cdots & 0 & 0 & 0 & * \\ 0 & 0 & M-I & -M & \cdots & 0 & 0 & 0 & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & M-I & M & 0 & * \end{pmatrix}.$$

All that is left to show, and the crux of the proof, is that row-reducing the square matrix on the left, Z, leaves a block identity matrix in an appropriate spot for back-substitution. Let $\mu(x)$ be any polynomial such that $\mu(M) = 0$. By the Cayley-Hamilton theorem, the characteristic polynomial of M may be taken for μ. We regard $\mu \in K[x]$ and can write $\mu(x) = \mu_k x^k + \mu_{k-1} x^{k-1} + \cdots \mu_1 x + \mu_0$ for some $\mu_i \in K$, but define for any $X \in M_m(K)$, $\mu(X) = \mu_k X^k + \cdots + \mu_1 X + \mu_0 I$. k is the greatest nonnegative integer such that $(x-1)^k$ divides $\mu(x)$ and write $\mu(x) = (x-1)^k q(x)$ for some $q(x) \in K[x]$. We define $q(X)$ for a matrix X the same way we did for μ. Notice that this allows us to perform the division algorithm with polynomials over matrices in the sense that, for any polynomial $g \in K[x]$, there exist polynomials $s, r \in K[x]$ such that $q(X) = g(X)s(X) + r(X)$, where the degree of r is less than the degree of g. This works because sums and products of a single matrix X freely commute with each other.

Label the block rows of the matrix from top to bottom starting at 1. For $1 \leq i \leq k$, multiply the row by $(M-I)^{k-i}M^{i-1}q(M)$. In the first block row, the leftmost block entry becomes $\mu(M) = (M-I)^k q(M) = 0$. For all subsequent rows $i \leq k$, the leftmost non-zero block entry equals the entry above it; namely the leftmost non-zero block in row i is $(M-I)^{k-i+1}M^{k-1}q(M)$. In row k, the rightmost block entry is $-M^k q(M)$ and the entry below it is still $M-I$. At this point, row reducing the matrix only leaves all the entries of the matrix nonzero except for these two, so

$$Z \sim \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & M^k q(M) \\ 0 & \cdots & 0 & M-I \end{pmatrix}.$$

Using the division algorithm, write $q(M) = (M-I)s(M) + \alpha M^p$ for some $\alpha \in K$, $p \in \mathbb{N}$ and $s \in K[x]$. Notice that $\alpha \neq 0$, for if it were not then $(x-1)|q(x)$ which would contradict its definition. Multiply the bottom row by $s(M)M^k$ and subtract from the row above. Divide the above row by α. This leaves the matrix

$$Z \sim \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & M^{p+k} \\ 0 & \cdots & 0 & M-I \end{pmatrix}.$$

Finally multiply the bottom row of the matrix by $(M^{p+k-1} + M^{p+k-2} + \cdots + M^1)$ and subtract from the top row. This leaves the entry $M^{p+k} - (M^{p+t-k} + M^{p+k-2} + \cdots + M^1)(M-I) = M$ in that row. Negate the bottom row and add the other row to derive I in the bottom right corner of the matrix. $\square$

Theorem 6.3. *The algorithm runtime is polynomial in m and d, not counting the time taken for summing the inhomogeneous terms. m denotes the number of recurrence relations, and d denotes the maximal order of the mutual recurrence.*

Proof. Let ω be the exponent for matrix multiplication, that is the smallest known value so that two $n \times n$ matrices can be multiplied in $O(n^\omega)$ time. Necessarily $\omega \geq 2$. As of this writing, the best known value for ω is 2.376 [4].

The reduction from the general case to the first-order case requires $O(md)$ time. The minimal polynomial of the matrix may be computed in $O((md)^\omega)$ time using an algorithm by Storjohann [12]. Counting the factors of $(x-1)$ can be done by dividing the minimal polynomial by $(x-1)$ at most md times, for a total of at most $O(m^2d^2)$ operations. Since the minimal polynomial divides the characteristic polynomial, the number of factors, k, is at most md.

Constructing the $(k+1)$-padded representation of the recurrence, duplicating rows, and summing requires $O(mk) \subset O(m^2d)$ time; there are $O(mk)$ entries in the matrix and each entry can be determined in constant time, less the time taken to sum inhomogeneous terms. Row-reducing the matrix requires $O((mk)^\omega)$ time. The back-substitution can be performed in $O(mk)$ time.

The row-reduction dominates the asymptotic time of all the other steps, so the algorithm runs asymptotically as fast as multiplying two $mk \times mk$ matrices. This is bounded by $O((m^2d)^\omega)$. $\square$

References

[1] Abramov, S.A., van Hoeij, M.: Desingularization of linear difference operators with polynomial coefficients. In: Dooley, S. (ed.) International Symposium on Symbolic and Algebraic Computation. pp. 269–275. ACM, New York, NY, USA (1999)

[2] Churchill, B.R., Lamagna, E.A.: Summing symbols in mutual recurrences. In: Fu, B., Du, D.Z. (eds.) Computing and Combinatorics, Lecture Notes in Computer Science, vol. 6842, pp. 531–542. Springer Berlin / Heidelberg (2011)

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

[4] Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. J. Symb. Comput. 9, 251–280 (March 1990)

[5] Gosper, Jr., R.W.: Decision procedure for indefinite hypergeometric summation. In: Proceedings of the National Aacademy of Sciences USA. vol. 75, pp. 40–42 (January 1978)

[6] Greene, C., Wilf, H.S.: Closed form summation of C-finite sequences. Transactions of the American Mathematical Society 359, 1161–1189 (2007)

[7] Karr, M.: Summation in finite terms. Journal of the ACM 28(2), 305–350 (April 1981)

[8] Kauers, M., Zimmermann, B.: Computing the algebraic relations of C-finite sequences and multisequences. Journal of Symbolic Computation 43(11), 787–803 (2008)

[9] Moenck, R.: On computing closed forms for summation. In: Proceedings of the MACSYMA User's Conference. pp. 225–236 (1997)

[10] Ravenscroft, Jr., R.A., Lamagna, E.A.: Summation of linear recurrence sequences. In: Milestones in Computer Algebra. pp. 125–132 (2008)

[11] Schneider, C.: A new sigma approach to multi-summation. Advances in Applied Mathematics 34(4), 740–767 (May 2005)

[12] Storjohann, A.: Deterministic computation of the frobenius form. In: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science. pp. 368–. FOCS '01, IEEE Computer Society, Washington, DC, USA (2001)

[13] Zeilberger, D.: A fast algorithm for proving terminating hypergeometric series identities. Discrete Mathematics 80(2), 207–211 (March 1990)