Factorial Polynomials in Computer Algebra Problems Related to Symbolic Summation

Eugene V. Zima

Abstract. We consider a natural succinct representation for factorial polynomials along with the set of low complexity lazy manipulation and evaluation rules. This leads to immediate improvements of the worst case running-time complexity of many basic steps in standard algorithms for indefinite and definite summation. Applications of this technique to computation of rational normal forms and anti-differences of hypergeometric terms is discussed along with a prototype Maple implementation.

Introduction

One of important intermediate representations arising in algorithms of symbolic summation is the Gosper-Petkovšek form of a rational function [1, 3] The modern term for this representation is *Polynomial Normal Form* (PNF for short [1]). One of the problems with PNF is that the degree of one of the polynomials forming the PNF can be exponential in the size of the numerator and the denominator of the input rational function. This together with the fact that the degree of PNF drives the running time complexity of summation algorithms directly influences efficiency of standard summation algorithms in computer algebra systems.

This contribution is based on very simple observation that PNF can be represented and manipulated succinctly and lazily, reducing the running time complexity of standard tasks involved in algorithms of symbolic summation. This succinct representation is natural, in a sense that it is explicit, as opposed to the idea of implicit representation described in [2].

1. Preliminaries

Let $\mathbb{K}$ be a field of characteristic zero, x – an independent variable, E – the shift operator with respect to x, i.e., $Ef(x) = f(x+1)$ for an arbitrary $f(x)$. Consider $R \in \mathbb{K}(x)$. If $z \in \mathbb{K}$ and monic polynomials $A, B, C \in \mathbb{K}[x]$ satisfy
(i) $R = z \cdot \frac{A}{B} \cdot \frac{EC}{C}$,
(ii) $A \perp E^k B$ for all $k \in \mathbb{N}$,
then (z, A, B, C) is a *polynomial normal form* (PNF) of R. If in addition,

(iii) $A \perp C$ and $B \perp EC$,
then (z, A, B, C) is a *strict* (PNF) of R (see [1] for details).

For example, for the rational function $\frac{x+1000}{x+1}$ PNF is

$$1, 1, 1, (x+999) \cdot (x+998) \cdot (x+997) \cdot \cdots \cdot (x+2) \cdot (x+1),$$

with polynomial C of degree 999.

An important notion widely used in the context of algorithmic summation is the *dispersion set* of polynomials $p(x)$ and $q(x)$, which is the set of positive integers h such that $\deg(\gcd(p(x+h), q(x))) > 0$. Another important notion is the largest element of the dispersion set known as the *dispersion*. One more piece of standard terminology required here is the notion of *shift equivalence* of polynomials: two polynomials $u(x), v(x) \in \mathbb{K}[x]$ are shift equivalent if there exists $h \in \mathbb{Z}$, such that $u(x+h) = v(x)$. Finally, following [4] define the *factorial polynomial* (a generalization of the falling factorial) for $p(x) \in \mathbb{K}[x]$ as

$$[p(x)]_k = p(x) \cdot p(x-1) \cdot \ldots \cdot p(x-k+1) \tag{1}$$

for $k > 0$ and $[p(x)]_0 = 1$. Observe, that factorial polynomials naturally appear in the last component of PNF, assuming that the dispersion of the numerator and denominator of a given rational function is nonzero.

2. Succinct representation of factorial polynomials and lazy manipulation rules

We first note, that the left hand side of (1) offers succinct (most compact) representation of the product in the right hand side for large values of k, as it requires $\Theta(\log k)$ bits to represent the polynomial $p(x) \cdot p(x-1) \cdot \ldots \cdot p(x-k+1)$ assuming the degree of $p(x)$ is fixed. The same polynomial would require $\Theta(k \log k)$ bits if represented as in [2]. For example, for $R = \frac{(x+10)^2(2x+29)}{(x+1)(2x-1)(5x+1)}$, the PNF in succinct form is

$$\left(1/5, x+10, x+1/5, [x+9]_9 \left[x + \frac{27}{2}\right]_{15}\right),$$

which is much shorter compared to the expanded representation of the degree 24 polynomial C. Factorial polynomials satisfy many obvious identities, which capture their multiplicative nature and allow manipulate them without expanding. We list only few of them for illustration purposes:

$$[p(x)]_k = [p(x)]_{k-1} \cdot p(x-k+1), \quad [p(x+1)]_k = p(x+1) \cdot [p(x)]_{k-1}, \quad \text{for } k > 0,$$
$$[p_1(x) \cdot p_2(x)]_k = [p_1(x)]_k \, [p_2(x)]_k \quad \textit{etc.}$$

Based on these it is easy to implement lazy evaluation rules, such as for example,

$$A \cdot [p(x)]_k \pm B \cdot [p(x+1)]_k = [A \cdot p(x-k+1) \pm B \cdot p(x+1)] \cdot [p(x)]_{k-1}, \tag{2}$$

which holds for arbitrary expressions A and B. Another set of rules involves computation of gcd and cancelations. For example, given natural h, k, and l:

$$\gcd([p(x)]_k, [p(x+h)]_l) = \begin{cases} 1 & \text{if } l-h \le 0, \\ [p(x)]_{\min(k,l-h)} & \text{otherwise.} \end{cases}$$

The ultimate goal of lazy manipulation rules is to avoid complete expansion of the involved factorial polynomials as much as possible.

3. Applications

The natural succinct representation together with above mentioned rules offer immediate improvement to the worst case running-time complexity of many basic steps in standard algorithms for indefinite and definite summation.

3.1. Rational Normal Forms

The notion of Rational Normal Form (RNF) was introduced in the context of hypergeometric summation (see [1] for the definition). The main steps in computation of RNF for a given rational function $R \in \mathbb{K}(x)$ involve construction of two PNFs, computation of gcd, and divisions:

```
(z,a,b,c) := PolynomialNormalForm(R,n);
(z1,a1,b1,c1) := PolynomialNormalForm(b/a,n);
g := gcd(c,c1);
return (z,b1,a1,c/g,c1/g)
```

The gain from using succinct representation is transparent from the following example: for $R = \frac{x(x+1000)}{(x+3)(x+1003)}$ the first call to PolynomialNormalForm produces $(1, x, x+1003, [x+999]_{997})$ with polynomial c of degree 997, the second call produces $(1, 1, 1, [x+1002]_{1003})$ with polynomial $c1$ of degree 1003, and $\gcd(c, c1) = [x+999]_{997}$. The RNF for R is

$$(1, 1, 1, 1, (x+1001)\,(x+1002)\,(x+1)\,(x+2)\,x\,(x+1000)),$$

with most of the terms from PNFs cancelled. Our prototype produces this result in 0.012 seconds, while standard Maple implementation requires 3.5 second on the same computer.

3.2. Gosper algorithm for summable hypergeometric terms

Recall that a nonzero expression $F(x)$ is called a hypergeometric term over $\mathbb{K}$ if there exists a rational function $r(x) \in \mathbb{K}(x)$ such that $F(x+1)/F(x) = r(x)$. Usually $r(x)$ is called the rational *certificate* of $F(x)$. The problem of indefinite hypergeometric summation (anti-differencing) is: given a hypergeometric term $F(x)$ to find a hypergeometric term $G(x)$, which satisfies the first order linear difference equation

$$(E-1)G(x) = F(x). \tag{3}$$

If found, write $\sum_x F(x) = G(x) + c$, where c is an arbitrary constant.

In [3] Gosper described a decision procedure, which starts with converting the rational certificate of the given summand to the Gosper-Petkovšek form (z, A, B, C) reducing the search for the sum to the search of a polynomial $y(x)$ solving the *key equation*:

$$A(x)y(x+1) - B(x-1)y(x) = C(x). \tag{4}$$

If $y(x)$ is found, then

$$G(x) = F(x)\frac{B(x-1)y(x)}{C(x)}. \tag{5}$$

Sometimes the polynomial $C(x)$ from (4) is called *the universal denominator*. One well-known problem with Gosper's algorithm is that the universal denominator can have pessimistically large degree (i.e., $C(x)$ and $y(x)$ can have very large degree common factor, which will cancel after substituting the solution $y(x)$ into (5)). This in turn can lead to the exponential dependency of the running time on the size of the input, even when input and output is small. Using the succinct representation of PNF along with multiplicative properties of the solution of (5) and of the factorial polynomials it is possible to remove extraneous terms in this universal denominator before solving the key equation. This leads to improvements in the running time complexity of Gosper's algorithm in case of summable non-rational hypergeometric summands with large dispersion of the rational certificate.

Implementations of all the above mentioned techniques in Maple are compared to standard Maple summation tools and show not only theoretical but also practical improvements.

References

[1] S. A. Abramov and M. Petkovšek. Rational normal forms and minimal decompositions of hypergeometric terms. *Journal of Symbolic Computation*, 33(5):521–543, 2002. Computer algebra (London, ON, 2001).

[2] A. Bostan, F. Chyzak, B. Salvy, and T. Cluzeau. Low complexity algorithms for linear recurrences. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, ISSAC '06, pages 31–38, New York, NY, USA, 2006. ACM.

[3] R. W. Gosper, Jr. Decision procedure for indefinite hypergeometric summation. *Proc. Nat. Acad. Sci. U.S.A.*, 75(1):40–42, 1978.

[4] R. Moenck. On computing closed forms for summations. In *Proceedings of the 1977 MACSYMA Users' Conference*, pages 225–236, 1977.

[5] E. V. Zima. Accelerating indefinite summation: Simple classes of summands. *Mathematics in Computer Science*, 7(4):455–472, 2013.

[6] Eugene V. Zima. Accelerating indefinite hypergeometric summation algorithms. *ACM Commun. Comput. Algebra*, 52, 3 (February 2019), pp. 96–99.

Eugene V. Zima
Physics and Computer Science dept.,
Wilfrid Laurier University,
Waterloo, Ontario, Canada
e-mail: ezima@wlu.ca