Summation in Finite Terms using Sage

Burçin Eröcal*
Research Institute for Symbolic Computation, Linz, Austria

The summation analogue of the Risch integration algorithm developed by Karr [4, 5] uses towers of difference fields to model nested indefinite sums and products, as the Risch algorithm uses towers of differential fields to model the so called *elementary functions* [2]. The algorithmic machinery developed by Karr, and later generalized and extended in [1, 13, 7, 6], allows one to find solutions of first order difference equations over such towers of difference fields, in turn simplifying expressions involving sums and products.

We present an implementation of this machinery in the open source computer algebra system Sage [14]. Due to the nature of open source software, this allows direct experimentation with the algorithms and structures involved while taking advantage of the state of the art primitives provided by Sage. Even though these methods are used behind the scenes in the summation package Sigma [10] and they were previously implemented in [3], this is the first open source implementation.

1 Mathematical Preliminaries

We call a field F equipped with an automorphism σ a *difference field*. In order to model a sum or a product in a given expression, we extend a difference field with a new indeterminate t, where the action of σ on t is determined by the shift behavior of the given sum or product. For example, take $\mathbb{Q}(n)$ with $\sigma(n) = n + 1$. In order to model $\sum_{i=1}^{n} \frac{1}{i}$, we add a new indeterminate t such that $\sigma(t) = t + \frac{1}{n+1}$. Similarly, in order to model 2^n, we take $\sigma(t) = 2t$.

The summation analogues of the elementary extensions in the Risch algorithm [2] are called Π and Σ extensions by Karr. Intuitively, Π extensions correspond to product extensions and Σ extensions to sum expressions. Definitions and structure theorems about these can be found in [5]. In the following discussion we'll mainly use the definition of a $\Pi\Sigma$-field or extension from [4, Definition 10].

2 Design Issues

Sage provides an algebraic type hierarchy similar to that of Magma. In order to add the required structures to work with $\Pi\Sigma$-fields we have extended this hierarchy with a new construct `PiSigmaExtension`, which takes 3 arguments: the base field, and the coefficients α and β where $\sigma(t) = \alpha t + \beta$.

```
sage: from karr.pi_sigma_field import PiSigmaExtension
sage: F.<n> = PiSigmaExtension(QQ, 1, 1)
sage: sigma = F.sigma(); sigma
Ring endomorphism of PiSigmaField with constant field Rational Field and tower:
[('n', 1, 1)]
  Defn: n |--> n + 1
sage: sigma(n)
n + 1
sage: E.<b> = PiSigmaExtension(F, 2, 0)
sage: sigma = E.sigma(); sigma
Ring endomorphism of PiSigmaField with constant field Rational Field and tower:
[('n', 1, 1),
 ('b', 2, 0)]
  Defn: b |--> 2*b
        n |--> n + 1
```

*email: burcin@erocal.org

Each algebraic structure in Sage is represented by a corresponding *parent* and an *element* class. Recent additions to the Sage library also allow automatic creation of these classes based on *category* definitions, inspired by those of Aldor or MuPAD. Our initial implementation uses the former method to define new parent and element classes inheriting from `FractionField` and `FractionFieldElement` respectively.

As a result of the object oriented design of Sage, the class defining elements of ΠΣ-fields contains methods to compute properties such as *specification of the equivalence* [4, Definition 13], Π and Σ-regularity [4, Definition 16-17].

```
sage: (n+1).spec(n+3)
2
sage: n.spec(n^2) is None
True
sage: t = n*b+1
sage: t.spec( (sigma^3)(t) )
3
sage: (n+1).pi_regularity(n^2 + 3*n + 2)
2
sage: (n*b).pi_regularity(2*n^2*b^2 + 2*n*b^2)
2
sage: (n+1).sigma_regularity(n^2 + 4*n + 4)
3
# ._sfactorial() computes the sigma factorial
sage: (n*b).sigma_regularity((n*b)._sfactorial(4))
4
```

For a difference field F, σ, let $H(F) = \left\{ \frac{\sigma(g)}{g} \,\middle|\, g \in F^* \right\}$. Given $f_1, \ldots, f_k \in F$, we can compute the set of $(n_1, \ldots, n_k) \in \mathbb{Z}^k$ such that $f_1^{n_1} f_2^{n_2} \cdots f_k^{n_k} \in H(F)$ using the algorithm described in [4, Theorem 8].

```
sage: from karr.orbit import homogeneous_group_exponents
sage: f1 = (n+1)*(n+2)
sage: f2 = (n+2)*(n+3)
sage: f3 = n*(n+1)
sage: homogeneous_group_exponents(f1, f2, f3, parent=F)
[ 1  0 -1]
[ 0  1 -1]
```

We can also compute denominator bounds [1, 9] and degree bounds [11, 4] which allow us to use the algorithm described in [12] to solve first order linear difference equations and find telescopers over ΠΣ-fields. More precisely, given a ΠΣ-field (F, σ) with $\mathrm{Const}_\sigma(F) = K$ and $a_0, a_1, f_1, \ldots, f_n \in F$, $c_1, \ldots, c_n \in K$, we can find $g \in F$ and $c_1, \ldots, c_n \in K$ satisfying $a_1\sigma(g) + a_0 g = c_1 f_1 + \cdots + c_n f_n$. Note that the solutions form a vector space $V \subset K^n \times F$.

In the following example, we consider the sum $\sum_{i=1}^{n-1} H_i^2$ where H_i denotes the harmonic sum $\sum_{i=1}^{n} \frac{1}{i}$ and find an equivalent expression containing only single sums. By extending the previously defined difference field `F` $= (\mathbb{Q}(n), \sigma)$ with an element h satisfying $\sigma(h) = h + \frac{1}{n+1}$, we construct the ΠΣ-field $\mathbb{Q}(n, h)$ containing our summand h^2 .

```
sage: H.<h> = PiSigmaExtension(F, 1, 1/(n+1))
sage: h, n = H.gens()
```

We call the `solve_plde()` function, to find $g \in H$ such that $\sigma(g) - g = h^2$. The first argument is the field we work in, `H` in this case. The second argument is a tuple with the coefficients (a_0, a_1). In this case we have $(a_1, a_0) = (1, -1)$, which we convert to be elements of H using the Python command `map(H, (1, -1))`. The last argument is a list containing the f_i's, where we have only the summand h^2.

```
sage: from karr.plde import solve_plde
sage: solve_plde(H, map(H, (1, -1)), [h^2])
(
[1]   [h^2*n - 2*h*n - h + 2*n]
[0], [                      1]
)
```

The answer has two matrices, in $K^{2\times 1}$ and $F^{2\times 1}$ respectively. A row of the first matrix contains the c_i component of a solution, while the corresponding row in the second matrix gives g. We have $c_1 = 1, g = h^2n - 2hn - h + 2n$ in the first row and the trivial solution $c_1 = 0, g = 1$ in the second row. From the first row we can deduce that our sum $\sum_{i=1}^{n-1} H_i^2$ is equal to $g(n)-g(1) = H_n^2*n-2H_n*n-H_n+2n$.

Next example reproduces the computations in Example 2.3 of [12], which were used in the proof of the identity

$$\sum_{k=0}^{n}(1-3(n-2k)H_k)\binom{n}{k}^3 = (-1)^n$$

appearing in [8].

```
sage: K.<n> = FractionField(QQ['n'])
sage: F.<k> = PiSigmaExtension(K, 1, 1)
sage: E.<b> = PiSigmaExtension(F, ((n-k)/(k+1))^3, 0)
sage: H.<h> = PiSigmaExtension(E, 1, 1/(k+1))
sage: f = (b*(1 + h*(-6*k + 3*n)), \
        (b*(1 + n)^3*(1 + h*(3 - 6*k + 3*n))) / (1 - k + n)^3, \
        (b*(1 + n)^3*(2 + n)^3*(1 + h*(6 - 6*k + 3*n))) / \
                (2 + k^2 + k*(-3 -2*n) + 3*n + n^2)^3)
sage: C, g = solve_plde(H, map(H, (1, -1)), f)
sage: C.row(0)
(6*n + 6, 12*n + 18, 6*n + 12)
sage: g.row(0).factor()
(6) * (k - n - 2)^-3 * (k - n - 1)^-3 * (n + 1) * b * k^2 * \
        (3*h*k^5 - 15*h*k^4*n - 24*h*k^4 + 27*h*k^3*n^2 + 90*h*k^3*n + \
        72*h*k^3 - 24*h*k^2*n^3 - 120*h*k^2*n^2 - 195*h*k^2*n - 102*h*k^2 + \
        12*h*k*n^4 + 78*h*k*n^3 + 186*h*k*n^2 + 192*h*k*n + 72*h*k - 2*k^4 + \
        12*k^3*n + 18*k^3 - 27*k^2*n^2 - 84*k^2*n - 63*k^2 + 28*k*n^3 + \
        134*k*n^2 + 208*k*n + 104*k - 12*n^4 - 78*n^3 - 186*n^2 - 192*n - 72)
```

Note that this is an application of the creative telescoping method and `f` contains 3 components, $S(n)$, $S(n+1)$ and $S(n+2)$ where $S(n) = \sum_{k=0}^{n}(1-3(n-2k)H_k)\binom{n}{k}^3$.

3 Future work

This implementation will provide a basis for a fully algorithmic treatment of algebraic extensions that are required to handle expressions such as $(-1)^n$ which frequently occur in combinatorial identities. We also plan to support extensions of the Karr algorithm such as [13].

References

[1] Manuel Bronstein. On solutions of linear ordinary difference equations in their coefficient field. *Journal of Symbolic Computation*, 29(6):841 – 877, 2000.

[2] Manuel Bronstein. *Symbolic integration. I*, volume 1 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2005. Transcendental functions, With a foreword by B. F. Caviness.

[3] Johannes Oswald Gaertner. Summation in finite terms - presentation and implementation of M. Karr's algorithm. Master's thesis, Research Institute for Symbolic Computation, JKU, Linz, May 1986.

[4] Michael Karr. Summation in finite terms. *J. ACM*, 28(2):305–350, 1981.

[5] Michael Karr. Theory of summation in finite terms. *Journal of Symbolic Computation*, 1(3):303 – 315, 1985.

[6] Manuel Kauers and Carsten Schneider. Application of unspecified sequences in symbolic summation. In *ISSAC 2006*, pages 177–183. ACM, New York, 2006.

[7] Manuel Kauers and Carsten Schneider. Symbolic summation with radical expressions. In *ISSAC 2007*, pages 219–226. ACM, New York, 2007.

[8] P. Paule and C. Schneider. Computer proofs of a new family of harmonic number identities. *Adv. in Appl. Math.*, 31(2):359–378, 2003.

[9] C. Schneider. A collection of denominator bounds to solve parameterized linear difference equations in ΠΣ-extensions. *An. Univ. Timisoara Ser. Mat.-Inform.*, 42(2):163–179, 2004.

[10] C. Schneider. The summation package Sigma: Underlying principles and a rhombus tiling application. *Discrete Math. Theor. Comput. Sci.*, 6(2):365–386, 2004.

[11] Carsten Schneider. Degree bounds to find polynomial solutions of parameterized linear difference equations in ΠΣ-fields. *Appl. Algebra Engrg. Comm. Comput.*, 16(1):1–32, 2005.

[12] Carsten Schneider. Solving parameterized linear difference equations in terms of indefinite nested sums and products. *J. Difference Equ. Appl.*, 11(9):799–821, 2005.

[13] Carsten Schneider. A refined difference field theory for symbolic summation. *J. Symbolic Comput.*, 43(9):611–644, 2008.

[14] W. A. Stein et al. *Sage Mathematics Software*. The Sage Development Team. `http://www.sagemath.org`.