Asymptotic Expansions of exp-log Functions

Daniel Richardson
University of Bath, U.K.
drs@maths.bath.ac.uk

Bruno Salvy
INRIA Rocquencourt, France
Bruno.Salvy@inria.fr

John Shackell
University of Kent at Canterbury, U. K.
J.R.Shackell@ukc.ac.uk

Joris Van der Hoeven
LIX, École polytechnique, France
vdhoeven@lix.polytechnique.fr

Abstract

We give an algorithm to compute asymptotic expansions of exp-log functions. This algorithm automatically computes the necessary asymptotic scale and does not suffer from problems of indefinite cancellation. In particular, an asymptotic equivalent can always be computed for a given exp-log function.

Introduction

Exp-log functions are functions obtained from a variable x and the set of rational numbers $\mathbf{Q}$ by closure under field operations and the application of exp and $\log|\cdot|$. The set of exp-log functions was studied by G. H. Hardy [6], who showed—using different terminology—that their germs at infinity form a totally ordered field. This property makes exp-log functions extremely useful for doing asymptotics.

The basic problem of effectively deciding the sign of an exp-log function at infinity (hence of computing limits) remained open for a long time. The advent of computer algebra has revived interest in this question. A first (theoretical) proof that this problem could be reduced to zero-equivalence of exp-log constants was found by B. Dahn and P. Göring, but their result does not lead to a practical algorithm. The first concrete algorithm was given by J. Shackell in [15]. D. Gruntz made a number of improvements, and also implemented the method, [5].

Since the appearance of [15], many problems of asymptotics have been treated (at least partially) from the effective point of view: functional inversion [14, 20], Liouvillian functions [16], composition [17, 20], algebraic differential equations [18]. However, implementations have not followed these recent advances. The first implementation of a non-trivial limit computation was done in [4]. A program to perform asymptotic expansions in general scales was given in [13]. A complete implementation by D. Gruntz of limit computation for exp-log functions is incorporated in MapleV.3, [5]. All these implementations rely on a heuristic zero-test for constants and functions. Other methods of zero-equivalence determination are available; see the concluding section. Also a more complete expansion program is under development [21].

We think that one of the reasons for this gap between theory and implementation is that this area is quite technical. Our aim in this paper is to describe explicitly a relatively simple algorithm for asymptotic expansions of exp-log functions, using the theoretical work referred to above. The essential idea is to make use of a special type of asymptotic scale called a *asymptotic basis* and to use *multiseries* with respect to such a basis (see Section 1). This simplifies both the description, proof and practical implementation of the algorithm.

The paper is structured as follows. In Section 1, the definitions of multiseries and asymptotic basis are given. In Section 2 we show how multiseries are handled in conjunction with an oracle for zero-equivalence, much in the same way as usual series are. In Section 3 we show how to compute an asymptotic basis suitable for expansion of a given exp-log function, and prove that multiseries can be computed for all exp-log functions. Finally, Section 4 gives a detailed example of the working of the algorithm.

1 Asymptotic scales and multiseries

In this section, we introduce the notation and the (few) definitions we shall use in this paper.

Definition 1 *An* asymptotic scale *is a finite ordered set* $\{g_1,\ldots,g_n\}$ *of positive unbounded exp-log functions such that* $\log g_i = o(\log g_{i+1})$, *for* $i=1,\ldots,n-1$.

This definition is slightly different from the classical one, where $\{x^n, n\in\mathbf{N}\}$ is a standard scale. The difference is motivated by our desire for generality as well as our effective viewpoint.

Next, we want to consider generalized series with respect to asymptotic scales. In order to express expansions of functions like

$$\frac{1}{1-x-x^e} = 1+x+x^2+x^e+x^3+2x^{1+e}+\cdots,\qquad x\to 0$$

we need to accomodate *finitely generated sets of exponents*. By this, we mean sets $S\subset\mathbf{R}$ of the form

$$S=\alpha_1\mathbf{N}+\alpha_2\mathbf{N}+\ldots+\alpha_k\mathbf{N}+\beta,$$

where the α_i's are strictly negative real numbers and $\beta\in\mathbf{R}$.

We can now define multiseries.

Definition 2 *Let g be a positive unbounded exp-log function. A* multiseries *with respect to g is a formal sum of the form*

$$f = \sum_{\alpha \in S} f_\alpha g^\alpha,$$

with finitely generated support S, the coefficients f_α being exp-log functions.

The support being finitely generated, it and its subsets are in particular well-ordered whence they have a greatest element. The element of the sum corresponding to the greatest α such that $f_\alpha \neq 0$ is called the *dominant* term of f. Similarly, it makes sense to consider the k *first* terms of a multiseries ($k \in \mathbf{N}^\star$).

Our aim is to compute asymptotic expansions in the form of multiseries with respect to an asymptotic scale $\{g_1, \ldots, g_n\}$.

Definition 3 *By a* multiseries expansion *with respect to $\{g_1, \ldots, g_n\}$, we mean a multiseries with respect to g_n, where each coefficient can recursively be expressed as a multiseries with respect to $\{g_1, \ldots, g_{n-1}\}$, each multiseries with respect to g_1 having constant coefficients. We denote the set of exp-log functions that admit such an expansion by* $\mathbf{R}[[\mathrm{g_1}; \ldots; \mathrm{g_n}]]$.

The following special asymptotic scales play an important role.

Definition 4 *An* asymptotic basis *is an asymptotic scale $\{g_1, \ldots, g_n\}$ satisfying the following conditions*

1. *$g_1 = \log_k x$ ($k \in \mathbf{N}$), where $\log_k$ denotes the logarithm iterated k times ($\log_0 x = x$);*
2. *$\log g_i \in \mathbf{R}[[\mathrm{g_1}; \ldots; \mathrm{g_{i-1}}]]$, for $2 \leq i \leq n$.*

2 Algorithmic multiseries

Given an asymptotic basis $\{g_1, \ldots, g_n\}$ we now describe the principal operations on exp-log functions in $\mathbf{R}[[\mathrm{g_1}; \ldots; \mathrm{g_n}]]$. In this section we assume that we are given an oracle for testing zero-equivalence of exp-log functions. From a computational point of view, we compute simultaneously with *explicit* exp-log functions (for instance as expression trees) and truncations of multiseries that converge to them in the sense of Poincaré expansions. When necessary, further terms of the series can be computed from the explicit representation. Recursively, an explicit form of each coefficient can be computed. This idea will be made more precise in this and the next section.

Definition 5 *A multiseries is said to be* algorithmic *when it converges to an exp-log function as an asymptotic expansion, and for any positive integer k its first k terms can be computed and are themselves algorithmic (constants being algorithmic).*

A direct consequence of this definition is that the limit at infinity of an algorithmic multiseries can be computed. The principal operations on algorithmic multiseries are as follows.

Sum The main problem here is avoiding indefinite cancellation of terms. This is done by testing whether the explicit sum is zero using the oracle. If not, then the dominant term of the sum can be obtained by computing sufficiently many terms of both summands and comparing the exponents of g_n. Once the dominant term is computed, then the oracle is used to decide whether this term is equal to the sum. Otherwise, the same process yields more and more terms. Note that if we expand without using the oracle, the algorithm may fail to terminate, since consecutive terms of identical series may be perpetually subtracted.

Product The usual product formula is used and the oracle is invoked as above to prevent indefinite cancellation.

Elementary functions Let $\Phi(z)$ be any of $\exp(z), \log(1+z), 1/(1+z)$. If f is an algorithmic multiseries tending to 0, then $\Phi(f)$ can be expressed as an algorithmic multiseries by employing the usual Taylor expansion of Φ and making use of the oracle as above. In particular, $1/f$ can be computed for any algorithmic multiseries f as follows. We first write $f = T + U$ where T is the dominant term in the g_n-expansion of f. Then we write $1/f$ as $T^{-1} \times [1/(1 + U/T)]$ and apply the same process recursively to the coefficient of g_n in T^{-1}. For the exponential and the logarithm, the case when f does not tend to 0 may require extensions of the asymptotic scale. This is treated in the next section.

It is worth mentioning that in practice, the exponents of multiseries are often integers and then fast algorithms for computing with power series can be used (e.g., Karatsuba's algorithm for product [7], Brent & Kung's algorithms for composition [1, 2],...). Besides, when the exponents are not integers, it is possible to reduce to this case by adding new variables [21].

3 Expansions of exp-log functions

Expansions are built recursively for subexpressions of the given function. The main problem is the computation of asymptotic bases. The idea is to start with an asymptotic basis consisting of just $\{x\}$, where x is the variable. The asymptotic basis is gradually extended as the computation proceeds. The only cases which may require an extension of the basis are logarithm and exponential of an algorithmic multiseries $f \in \mathbf{R}[[\mathrm{g_1}; \ldots; \mathrm{g_n}]]$ tending to infinity. During the construction, we have to ensure that $\log g_i$ is an algorithmic multiseries for $i > 1$. An asymptotic basis satisfying this property will be called *algorithmic*.

Logarithm Suppose that we wish to compute an expansion for $\log f$. By induction, we may suppose that we have obtained $f \sim A$, where $A = c g_1^{\alpha_1} \cdots g_n^{\alpha_n}$. Then letting $F = f - A$, we use

$$\log(f) = \log(A) + \log(1 + F/A).$$

The expansion of $\log(1 + F/A)$ may be performed as given in the previous section. If $\alpha_1 = 0$, then $\log A$ is computed by

$$\log A = \log c + \alpha_2 \log g_2 + \ldots + \alpha_n \log g_n,$$

which is algorithmic since the asymptotic basis is algorithmic.

Otherwise, if $\alpha_1 \neq 0$, we insert $\log g_1$ at the first place in the asymptotic basis. It is easy to see that this insertion yields an algorithmic asymptotic basis. Then computation of $\log f$ is reduced to the previous case.

Exponential Here we want to compute the exponential of a positive algorithmic multiseries f tending to infinity. The first step is to check whether f is asymptotic to the logarithm of an element of the asymptotic basis. More precisely, using recursively multiseries manipulations of the type described in this and the previous sections, we check whether $\alpha = \lim(f/\log g_i) \in \mathbf{R}^\star$ for some $i > 1$. If so, then we write

$$\exp(f) = g_i^\alpha \times \exp(f - \alpha \log g_i), \tag{1}$$

and recursively compute the exponential of $f - \alpha \log g_i$.

Otherwise, we have $\log g_i = o(f)$ and $f = o(\log g_{i+1})$ for some $i > 1$ (or $i = n$ and $\log g_n = o(f)$). In this case, the function $\exp(f)$ is inserted in the asymptotic basis between g_i and g_{i+1} (or after g_n if $i = n$). Once again, it is not difficult to verify that this yields an algorithmic asymptotic basis. Note however that examples such as $\exp(e^x/(1-x^{-1})) = \exp(e^x + e^x/x + \cdots)$ show that we cannot use $\exp(F)$ in place of $\exp(f)$ with F a finite partial sum of the multiseries for f.

We still have to prove the termination of our algorithm. The only point which might lead to infinite loops is the recursive computation of the exponential in (1). Termination is ensured by the finiteness of the asymptotic basis and the observation that the index i in (1) strictly decreases. We note that it is essential here that expansion is always performed first in g_i, where i is the largest available index. To summarize, we have proved the following.

Theorem 1 *Assume that we are given an oracle for asymptotic zero-equivalence of exp-log functions. Then for any exp-log function f, we can compute an algorithmic asymptotic basis with respect to which f can be expressed as the sum of an algorithmic multiseries.*

It should be noted that there are several possible asymptotic bases with respect to which an exp-log function can be expressed as an algorithmic multiseries. The output of our algorithm thus depends slightly on the order in which the exponentials are treated.

4 Detailed example

We now exemplify the algorithm on the function

$$f = \log\log(xe^{xe^{x}} + 1) - \exp\exp\left(\log\log x + \frac{1}{x}\right),$$

as $x \to \infty$.

Initially the asymptotic basis $B = \{g_1, \ldots, g_n\}$ is $\{x\}$. At each step we shall denote the ith element of B by g_i. The basis will grow as the computation proceeds, so that for example, g_2 may denote one function at one stage of the computation and a different one a later stage. We start with the first summand of f and its innermost subexpression e^x. The argument x of the exponential is not asymptotic to any $\log g_i$, with $i > 1$. Hence, e^x is inserted as the last element of B. The next term is $\exp(xe^x)$. The argument xe^x, whose multiseries is $g_1 g_2$, is compared to $x = \log g_2$, whose multiseries is g_1. We see that $\log g_2 = x = o(xe^x)$, and so $\exp(xe^x)$ is then inserted at the end of B. The next expression to consider is $x\exp(xe^x) + 1$ which is already in multiseries form.

We then turn to the innermost logarithm, $\log(x\exp(xe^x) + 1)$. At this stage $B = \{x, e^x, \exp(xe^x)\}$. The argument is rewritten as $g_1 g_3 + 1$. The exponent of g_1 is not zero, therefore $\log x$ is inserted at the beginning of B. We now have $B = \{\log x, x, e^x, \exp(xe^x)\}$, and

$$\log(g_2 g_4 + 1) = g_2 g_3 + g_1 + \log(1 + g_2^{-1} g_4^{-1}).$$

The logarithm on the right-hand side can be expanded as an algorithmic multiseries. The next logarithm is treated similarly. The basis need not be extended and we can write the first summand of f as

$$\begin{aligned}\log\log(g_2 g_4 + 1) = g_2 + g_1 \\ + \log\left[1 + g_2^{-1} g_3^{-1}[g_1 + \log(1 + g_2^{-1} g_4^{-1})]\right].\end{aligned}$$

We now consider the second summand of f. The expansion of $\log x$ is obviously g_1. Its logarithm necessitates inserting $\log\log x$ at the beginning of B. At this stage, $B = \{\log\log x, \log x, x, e^x, \exp(xe^x)\}$. The argument of the innermost exponential is $g_1 + g_3^{-1}$ which tends to infinity. This is found to be equivalent to the logarithm of g_2, thus we apply (1) and write

$$\exp(g_1 + g_3^{-1}) = g_2 \exp(g_3^{-1}),$$

where the argument of the new exponential tends to 0. The right-hand side of this equation is the argument of the next exponential. It is asymptotic to $g_2 = \log g_3$. Again applying (1) yields

$$\exp(\exp(g_1 + g_3^{-1})) = g_3 \exp[g_2 \exp(g_3^{-1}) - g_2].$$

The argument of the outermost exponential in the right-hand side tends to 0, thus an algorithmic multiseries is available.

At this stage, we have constructed an asymptotic basis

$$B = \{\log\log x, \log x, x, e^x, \exp(xe^x)\}$$

with respect to which we can expand

$$\begin{aligned}f = g_3 + g_2 - g_3 \exp[g_2 \exp(g_3^{-1}) - g_2] \\ + \log\left[1 + g_3^{-1} g_4^{-1}[g_2 + \log(1 + g_3^{-1} g_5^{-1})]\right]\end{aligned}$$

and its subexpressions as algorithmic multiseries.

We now give the computation of an asymptotic equivalent of f. The first step consists in computing the dominant term with respect to g_5. We illustrate the operation of the algorithm on the expansion of the subexpression $g_2 + \log(1 + g_3^{-1} g_5^{-1})$.

First, the argument $g_3^{-1} g_5^{-1}$ of the special function $\log(1 + z)$ is expanded as itself. The dominant term of the logarithm is again $g_3^{-1} g_5^{-1}$. Next, g_2 is expanded as g_2, which is then seen to be the dominant term of the sum. The rest of the computation is straightforward and yields an asymptotic expression for f of

$$g_3 + g_2 + \log[1 + g_2 g_3^{-1} g_4^{-1}] - g_3 \exp[g_2 \exp(g_3^{-1}) - g_2].$$

We then proceed with the computation of the dominant term of this expression with respect to g_4. Exactly the same type of computation leads to

$$g_3 + g_2 - g_3 \exp[g_2 \exp(g_3^{-1}) - g_2].$$

Next, we compute the dominant term of this expression with respect to g_3. Let $h = g_2 \exp(g_3^{-1}) - g_2$, so that h is the argument of the outermost exponential. The computation of the dominant term of h leads to the cancellation $g_2 - g_2$ which is recognized by the oracle, whereas the function h itself is not zero. By computing the next term of the expansion, we obtain the dominant term g_2/g_3 of h. The dominant term of $g_3 \exp(h)$ is g_3, whence a new cancellation $g_3 - g_3$. Computing the next term leads to another cancellation $g_2 - g_2$. One more term is necessary before arriving at the conclusion that the dominant term with respect to g_3 is

$$-\frac{g_2^2 + g_2}{2g_3}.$$

Thus we obtain the desired equivalence

$$f \sim -\frac{g_2^2}{2g_3} = -\frac{\log^2 x}{2x}.$$

The k first coefficients with respect to any element of the asymptotic basis can be computed in a similar way.

Conclusion

It is customary in papers about asymptotic expansions in computer algebra to assume the existence of a zero-test for certain classes of functions or constants; we have done so in this work. Here, the constants we encounter are exp-log constants, for which the zero-equivalence problem is related to Schanuel's conjecture (see [9, 10]). If Schanuel's conjecture is true, the elementary numbers are a computable algebraically closed field, and the real elementary numbers are a computable real closed field; this means that zero equivalence is decidable. This is implemented in a program which decides whether or not an elementary constant is zero; and which is guaranteed to terminate (eventually) unless it is working on a problem which includes a counterexample to Schanuel's conjecture.

The zero-equivalence problem for exp-log functions is then usually reduced to the exp-log constants case via Risch's algorithm [11] or differential algebra methods [15, 19, 8]. Special consideration of algebraic extensions based on error estimation and numerical evaluation is described in [21].

Another issue we have not mentioned is that of complexity. The algorithm as we have described it has terrible worst-case complexity. This is seen with examples of the form

$$\frac{1}{1-1/x} - \frac{1}{1-1/x} + x^{-N},$$

with N arbitrarily large (think of $N = 10^{10^{\cdot^{\cdot^{\cdot}}}}$). We refer to [21] for approaches to this problem.

Despite these theoretical problems, the algorithm as we have described it can very well be implemented in a reasonably efficient way (see [21]).

As already mentioned in the introduction, this algorithm is susceptible to many generalizations. Not only exp-log functions can be handled this way, but also more general elementary functions, or even less explicit functions. In all cases, the concepts of asymptotic basis and multiseries generalize and provide a foundation for expressing and proving algorithms. To a certain extent, differential equations can also be treated, but this is still in development. Also, the constant problems in these general contexts becomes much more difficult.

We finally observe that many different viewpoints (and vocabularies) exist on the subject of effective asymptotic expansions. Hardy fields [12] and transseries [3] provide two theoretical contexts for dealing with asymptotic expansions (the former is more analytic, the latter more algebraic). For dealing explicitly with these objects, either nested expansions [18] or multiseries can be used. It is not clear at the moment which is more efficient; this probably depends on the type of computation or problem one has in mind. An advantage of multiseries is that they are closer to the classical conception of asymptotic expansions.

References

[1] Brent, R. P., and Kung, H. T. $O((n \log n)^{3/2})$ algorithms for composition and reversion of power series. In *Analytic Computational Complexity* (1975), J. F. Traub, Ed. Proceedings of a Symposium on analytic computational complexity held by Carnegie-Mellon University.

[2] Brent, R. P., and Kung, H. T. Fast algorithms for manipulating formal power series. *Journal of the ACM 25* (1978), 581–595.

[3] Écalle, J. *Introduction aux fonctions analysables et preuve constructive de la conjecture de Dulac.* Actualités mathématiques. Hermann, Paris, 1992.

[4] Geddes, K. O., and Gonnet, G. H. A new algorithm for computing symbolic limits using hierarchical series. In *Symbolic and Algebraic Computation* (New York, 1989), P. Gianni, Ed., vol. 358 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 490–495. Proceedings ISSAC'88, Rome.

[5] Gruntz, D. *On Computing Limits in a Symbolic Manipulation System.* PhD thesis, ETH, Zürich, 1996.

[6] Hardy, G. H. *Orders of Infinity*, vol. 12 of *Cambridge Tracts in Mathematics.* Cambridge University Press, 1910.

[7] Knuth, D. E. *The Art of Computer Programming*, 2nd ed., vol. 2: Seminumerical Algorithms. Addison-Wesley, 1981.

[8] Péladan-Germa, A. Testing identities of series defined by algebraic partial differential equations. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes* (1995), G. Cohen, M. Giusti, and T. Mora, Eds., no. 948 in Lecture Notes in Computer Science, Springer-Verlag, pp. 393–407. Proceedings of the 11th International Symposium, AAECC-11, Paris, France, July 1995.

[9] Richardson, D. The elementary constant problem. In *Symbolic and Algebraic Computation* (1992), P. S. Wang, Ed., ACM Press, pp. 108–116. Proceedings of ISSAC'92, Berkeley, July 1992.

[10] Richardson, D. How to recognise zero. Research report, University of Bath, England, 1994.

[11] Risch, R. H. Algebraic properties of the elementary functions of analysis. *American Journal of Mathematics 4*, 101 (1975), 743–759.

[12] Rosenlicht, M. Hardy fields. *Journal of Mathematical Analysis and Applications 93* (1983), 297–311.

[13] Salvy, B. *Asymptotique automatique et fonctions génératrices.* Ph. D. thesis, École polytechnique, 1991.

[14] Salvy, B., and Shackell, J. Asymptotic expansions of functional inverses. In *Symbolic and Algebraic Computation* (1992), P. S. Wang, Ed., ACM Press, pp. 130–137. Proceedings of ISSAC'92, Berkeley, July 1992.

[15] Shackell, J. A differential-equations approach to functional equivalence. In *Proceedings ISSAC'89* (New York, 1989), G. Gonnet, Ed., ACM, pp. 7–10.

[16] Shackell, J. Limits of Liouvillian functions. *Proceedings of the London Mathematical Society 72* (1996), 124–156.

[17] Shackell, J. Extensions of asymptotic fields via meromorphic functions. *Journal of the London Mathematical Society 52* (1995), 356–374.

[18] Shackell, J. Rosenlicht fields. *Transactions of the American Mathematical Society 335*, 2 (1993), 579–595.

[19] Shackell, J. Zero-equivalence in function fields defined by algebraic differential equations. *Transactions of the American Mathematical Society 336*, 1 (Mar. 1993), 151–171.

[20] Van der Hoeven, J. General algorithms in asymptotics I: Gonnet and Gruntz's algorithm & II: Common operations. Tech. Rep. LIX/RR/94/10, École polytechnique, Palaiseau, France, July 1994.

[21] Van der Hoeven, J. *Asymptotique Automatique.* PhD thesis, École polytechnique, Palaiseau, France, 1996. To appear.