1

Introduction

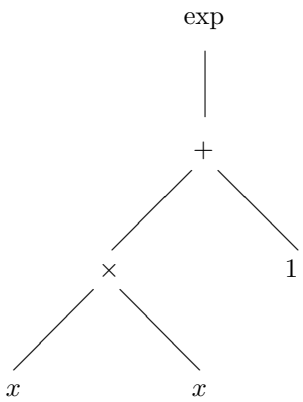
This book is about the automatic computation of asymptotic behaviour of functions of a real variable, x. This means that we shall wish to compute limits, and also to give some measure of how rapidly a given function approaches its limit. Throughout, we shall assume that our interest is in the behaviour as $x \to \infty$. This is for convenience only, since other cases can be considered by using a transformation of one of the forms $x \to \pm(x - x_0)^{-1}$. These give separate treatments of the cases when x tends to x_0 from above and below. We shall want our methods to be at least potentially implementable, and we shall, from time to time, give some consideration to actual implementations. However this book is really about algorithms and the mathematical principles that underlie them, rather than details of implementation.

Several questions present themselves:

(I). What sort of functions will be considered, and how will they be represented?
(II). How will the asymptotic behaviour be expressed?
(III). What are the key algorithmic problems which need to be overcome?
(IV). What mathematical techniques are to be employed?

This book represents an attempt to answer these questions from a particular perspective, and to report on the progress that has been achieved in various cases.

We take the view that a function is something which is built from constants and the identity function, x, by means of certain operations including arithmetic operations and others. Thus a function is given by an expression, which may be represented by a tree; for example the function $\exp(x^2 + 1)$ is represented by the tree below.



It is reasonable to ask which constants are to be allowed as leaves of the trees, and which operations are to be allowed as nodes. The obvious answer to the first question is to only allow integers. However if $\log x$ appears as a subexpression of our input function, it may be very inconvenient if $\log 2$ is not in our field of constants. We could add such constants as needed, but it is sometimes better for theoretical purposes to start from the set of real numbers as constants. Of course, this does not solve the problems as far as implementation is concerned. In fact there are substantial difficulties connected with constants in our area of study, as we shall see in the next chapter.

In addition to the arithmetic operations we may want to add real roots of algebraic equations and also consider the application of exponentials, logarithms, integrals, and solutions of more general differential equations.

As well as using trees as indicated above, it will be useful to employ a concept from differential algebra (see [45, 107] for example). A *tower* of sets of functions is a finite chain

$$\mathbb{K} = \mathcal{F}_0 \subset \mathcal{F}_1 \subset \cdots \subset \mathcal{F}_n. \tag{1.1}$$

Normally, the sets, $\mathcal{F}_i$, will be fields, or at least rings. Each $\mathcal{F}_i$, $1 \leq i \leq n$, will typically be generated from $\mathcal{F}_{i-1}$ by the addition of a single element, f_i, satisfying a differential equation over $\mathcal{F}_{i-1}$ of first order and degree. $\mathbb{K}$ will often be a field of constants. We shall frequently want to use induction on the index i, thereby reducing a problem about a function in $\mathcal{F}_i$ to one in $\mathcal{F}_{i-1}$.

One can define various classes of functions using towers (1.1), by placing restrictions on the differential equations which define the f_i. So if each f_i is a logarithm or an exponential, then the totality of functions definable using towers of function fields (1.1) is the field of *exp–log* functions. If f_i is also

allowed to be a real root of an algebraic equation over $\mathcal{F}_{i-1}$ (which we could think of as a differential equation of order zero) then Hardy's class of *L–functions* is obtained. At the next stage of generalisation, one could allow f_i to be any integral of an element in $\mathcal{F}_{i-1}$, rather than just a logarithm, while retaining the possibility of f_i being an exponential or a real root of an algebraic equation; the class of functions so defined is the class of *Liouvillian functions*. Finally if one allows the f_i to be defined by any differential equations of first order and degree, one obtains the class of *Pfaffian functions*. The theory of Hardy fields suggests that this as a natural barrier. However there are some circumstances in which one can do more.

Having looked at the form which our input might take, we now turn our attention to the output. In other words, we look at question (II). The traditional approach here has been to use asymptotic power series, and it has led to many successes. However, as we shall see in Chapter 4 such series are frequently insufficient for our purposes, and may bring other problems. We shall present three more general type of expansion that avoid these difficulties; they are *multiseries, nested expansions* and *star products*. By using any of these it is possible to express the asymptotic behaviour of arbitrary exp–log functions, Liouvillian functions, inverse functions and implicit functions.

There is one more point that needs to be made here. The traditional stance in asymptotics is to regard asymptotic series as the prime objects of study. Most of the work is carried out in terms of the series, and the functions possessing these expansions often only appear quite late in the process. By contrast, we shall regard functions as the prime objects of study. One reason for this is that the convergence and summability theories that are used with asymptotic power series are less well developed for the more general expansions mentioned above. There are also more technical reasons concerning zero equivalence. As a result of our viewpoint, we shall not generally be concerned with proving the existence of functions having a particular expansion.

As regards question (III), many of the algorithmic problems will be best described when they are encountered in later chapters. However there are two which are worth mentioning now. We have seen that we can represent our input functions by expressions using either trees or towers of the form (1.1). Then each expression corresponds to a unique function, but unfortunately the representation is not one-to-one. In practice we have to work with the expressions, rather than directly with the functions. It is therefore vital that we have some way of determining when two different expressions represent the same function. This is the problem of *zero equivalence*, which will be treated in Chapter 2.

The second problem is related to zero equivalence. Suppose that we have a tower (1.1), with an expression $f \in \mathcal{F}_n$ for which we want to determine a nested expansion, for example. By induction, we may assume that we can do this for elements of $\mathcal{F}_{n-1}$. Then if f_n is given by a simple differential equation over $\mathcal{F}_{n-1}$, say f_n is an exponential or an integral, it is not too difficult to

obtain an expansion for f_n. The trouble is that f_n may partially cancel with elements of $\mathcal{F}_{n-1}$ in the expression f. Here is an example from [97]. Let

$$f = \exp\left\{\frac{x}{x-1} + e^{-x}\right\} - \exp\left\{\frac{x}{x-1}\right\}. \tag{1.2}$$

If we proceed naively by expanding $x/(x-1)$ within the two exponentials and using the exponential series at 1, the terms will perpetually cancel out and the algorithm will fail to terminate. The problem is that e^{-x} is smaller than any constant power of x. We can recognise the existence of the cancellation by replacing e^{-x} by zero in the expression for f, obtaining an expression which is functionally equivalent to zero. (Note that we need to make explicit use of a zero-equivalence procedure here.). Then we can rewrite the right-hand side of (1.2) as an analytic function of the two 'variables', x^{-1} and e^{-x}. Since the terms not involving e^{-x} cancel We look successively at the positive powers, and discover that the coefficient of e^{-x} is $\exp\{1/(1-x^{-1})\}$, which is not equivalent to zero. Hence $f \sim e \cdot e^{-x}$. In general, we need to rewrite the given expression in terms of a finite number of sub-expressions which are all of a different 'order of growth', like x^{-1} and e^{-x}. Then we can expand in a similar way to the above. Of course, it is the rewriting that is the difficult part. In Chapter 5, we show how to do this for exp–log functions, and extend it to arbitrary Liouvillian functions. We can also uses the same ideas to handle extensions given by composition of a meromorphic solution of an algebraic differential equation with a function in the existing field.

In addition, we want to consider some cases which do not conform to the pattern of (1.1). Here there may be difficulties in obtaining the expansion of the newly-introduced object as well as in handling cancellations with existing objects. For example in Chapter 6 we look at algebraic differential equations of general order. We cannot give an algorithm to describe the asymptotics of an arbitrary solution of such equations. However we are able to obtain results for solutions that belong to some (a priori undetermined) Hardy field, and this at least has the merit of not restricting attention to a particular type of asymptotic growth in the way that searching for power-series solutions does.

Inverse functions are another case. We point out in Section 4.1 that inverse functions present formidable difficulties for asymptotic series expansions. Nested expansions on the other hand can just be inverted, [90], and multi-series can be used to give a full answer to the problem, [88]. We look in detail at inverse functions in Chapter 7.

In Chapter 8, we consider functions of several variables. This leads on to our results on implicit functions. Chapter 9 looks at functions which grow more rapidly than any iterated exponential, and related matters.

We have not yet given any answer to question (IV). Much of the theoretical underpinning for the algorithms we describe comes from the theory of Hardy fields, which is covered in Chapter 3. Beyond that, we shall need to use some differential algebra, some ideas from asymptotic series and of course many of the standard techniques of computer algebra, such as polynomial calculations,

gcds etc. In Chapter 10 we look at what can be done with oscillating functions, and here some basic ideas from measure theory come into play.

In a number of cases, for example the algebraic extensions of Chapter 5, solutions of differential equations in Chapter 6 and implicit functions in Chapter 8, our techniques do not immediately give us the existence of solutions. Instead we demonstrate that any solutions that do exist (and lie in a Hardy field in the case of solutions of differential equations of arbitrary order) must have nested expansions of one of a number of types, which we list. Of course this exactly parallels the classical situation for asymptotic series. For algebraic extensions, we are able to adapt Sturm sequences to a Hardy-field setting in order to determine the number of solutions with the given nested form. A similar technique works for implicit functions in most cases.

Before closing this chapter, we want to briefly discuss the following question: What is the natural class of functions for the analysis side of symbolic computation? Of course one cannot hope to get a single answer to such a question, since there will always be special domains for special problems, but it seems worth giving the matter some thought. For the domain of functions very much determines the kind of theory that one obtains. It is instructive to contrast the analysis of Euler, Cauchy and Weierstrass in this way.

Thus far, in what is still a young subject, much of the attention has focussed on elementary functions. Sometimes these have been augmented by certain special functions, such as the error function, and sometimes Liouvillian functions have been considered. However the basic flavour has been that of elementary functions. It is natural to try and handle these first, and some very fine work has been done. Indeed the theory of integration in finite terms can make a good claim to be the jewel in the crown of symbolic computation. Nonetheless, I believe that the concentration on elementary functions and their near relatives is ultimately too limiting. Too often the only answer that an integration package can give to a user's demand is to assert that the input is not integrable in finite terms. From one point of view this represents a mathematical triumph. The engineer may be less impressed than we might hope however! In [109], David Stoutemyer points out the need to give qualitative information about functions. In particular he argues that the presentation of a result as a complicated elementary function may be insufficient. We would add that it often cannot be done! Thus computer-algebra packages need to give information about zeros, singularities, and of course limits, of functions, rather than always looking for elementary solutions.

So what is the correct domain in which to try to do this? One can make a case that it should ultimately be the domain of solutions of algebraic differential equations, augmented by one or two other special functions such as the gamma function. I am not saying that this can be achieved now, nor anything like it, but I do believe that mostly it will be achievable, and perhaps within a reasonable time span. Surely there will always be things we do not know about

solutions of algebraic differential equations, just as there are about elementary functions, but this should not deter us.

For a view on the importance of algebraic differential equations from a different perspective, the reader is referred to [84], and the other papers of Lee Rubel on this subject.