Certified polynomial approximations for D-finite functions

Thomas GREGOIRE

August 26, 2012

Abstract

This paper describes the work carried out during a two-month internship in the AriC team at the Laboratoire de l'Informatique du Parallélisme, under the supervision of Marc Mezzarobba and Nicolas Brisebarre. The topic is the systematic generation of polynomial approximations, together with proofs of correctness, for the class of univariate functions satisfying linear ordinary differential equations with polynomial coefficients.

Contents

1 Introduction

1.1 The Dynamic Dictionary of Mathematical Functions

This work lies within the framework of dictionaries of mathematical reference data, more particularly of mathematical functions. Probably the most famous example is the *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* [1]. The idea is to provide, in an accessible way, a huge amount of information of all kinds, prepared by specialists and carefully proofread: definitions, identities, approximations, plots, tables of values and so on. Such an encyclopedia is of great interest. For example, the *Handbook of Mathematical Functions* is one of the most cited and most widely distributed scientific books of all times.

When the need of an update of the *Handbook of Mathematical Functions* arose, the National Institute of Standards and Technology launched an online project, the *Digital Library of Mathematical Functions* (`http://dlmf.nist.gov/`). Like for its precursor, the idea was still to gather a lot of information, but presented in a different manner: it is a freely available web site[1], accessible to everyone, including non-experts. Moreover DLMF makes use of the advantages proposed by the World Wide Web. It takes the form of a hypertext document, which makes the search of a given paragraph far easier. Another important advantage, in contrast with written media, is that the web offers a great extensibility — no need to buy a more recent edition of the dictionary after revision — and interactivity.

The Dynamic Dictionary of Mathematical Functions (DDMF) [3] is a research project which makes it one step further, by taking advantage of the development of *Computer Algebra Systems* (CAS). It is also an online encyclopedia of mathematical functions, but contrary to the aforementioned dictionaries, all its pages are *automatically* generated (Figure 1). It is based on DynaMoW [5], an OCaml library allowing to generate mathematical web sites using content generated by a CAS — in the case of DDMF, Maple. Thus, information is computed directly from the function data structure in the CAS, then laid out, taking care of the particularities of each function, and finally displayed to the user.

DDMF is targeted for a broad audience. It provides great interactivity: the user can for example change the precision of the evaluator, the number of terms displayed in the Taylor expansion or the bounds of the plot domain. Hence, anyone can get information they are looking for, even if it has not been computed by the authors or by anyone before, the dictionary evolving according to their needs.

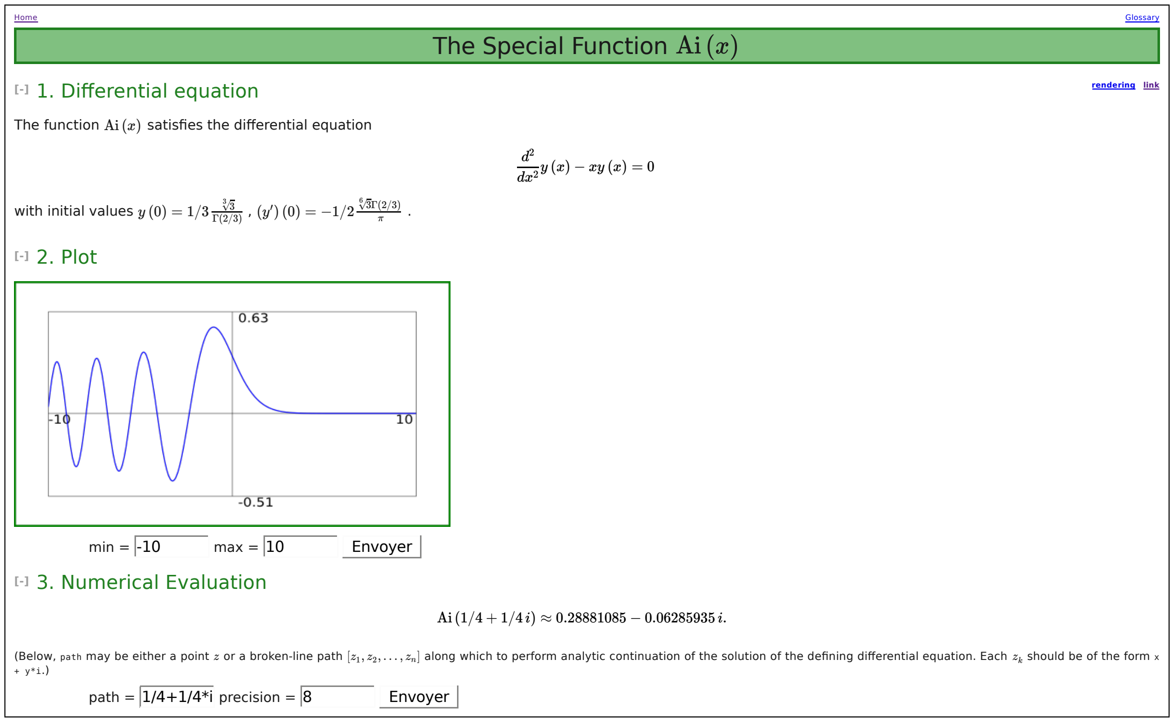
In such a framework, a major issue with automatic content generation is *reliability*. One of the amazing goals of DDMF is to turn this issue into a feature by providing the user not only information about the way content was generated, but also to provide detailed *proofs of correctness*, allowing him to understand and check step-by-step the calculations. Moreover, this is a way to make DDMF more accessible to the non-expert user than the output of a CAS.

This work is a step in this direction, into making this encyclopedia both *accessible* and *reliable*.

1.2 The general framework of D-finite functions

To design an encyclopedia like DDMF, one needs a common *data structure* for functions. Of course we have to carefully choose the set of functions we will be working on: the aim is not only to represent functions, but also to do calculations on them, that is, to *manipulate* those functions. Let us present the class of functions we will be working with, and detail some of the reasons motivating this choice.

[1]Yet printed versions are still available.


Home Glossary

The Special Function $\mathrm{Ai}\,(x)$

[-] 1. Differential equation rendering link

The function $\mathrm{Ai}\,(x)$ satisfies the differential equation

$$\frac{d^2}{dx^2} y\,(x) - xy\,(x) = 0$$

with initial values $y\,(0) = 1/3\,\frac{\sqrt[3]{3}}{\Gamma(2/3)}$, $(y')\,(0) = -1/2\,\frac{\sqrt[6]{3}\Gamma(2/3)}{\pi}$.

[-] 2. Plot

min = -10 max = 10 Envoyer

[-] 3. Numerical Evaluation

$$\mathrm{Ai}\,(1/4 + 1/4\,i) \approx 0.28881085 - 0.06285935\,i.$$

(Below, `path` may be either a point z or a broken-line path $[z_1, z_2, \ldots, z_n]$ along which to perform analytic continuation of the solution of the defining differential equation. Each z_k should be of the form `x + y*i`.)

path = 1/4+1/4*i precision = 8 Envoyer

Figure 1: The Dynamic Dictionary of Mathematical Functions

Definition 1. *Let $\mathbb{K}$ be a subfield[2] of $\mathbb{C}$. A function $f : \mathbb{K} \to \mathbb{K}$ is **D-finite** if it satisfies a linear homogeneous differential equation with polynomial coefficients:*

$$P_r f^{(r)} + \cdots + P_1 f' + P_0 f = 0,$$

with $P_i \in \mathbb{K}[X]$ for $i = 0 \ldots r$ and $P_r \neq 0$.

The first interesting property of this class is that D-finite functions can be represented by a particular differential equation, that is, an order and a finite number of polynomials, together with (a finite number of) initial values. Therefore they can be represented in the memory of a computer[3]. Of course, this is essential for computer algebra.

Now, how large is this class? Algebraic functions[4] are D-finite [4], as well as trigonometric functions like $\sin(z)$ or $\cos(z)$. A lot of special functions[5], like $\mathrm{Ai}(z)$ or erf(z) are D-finite too. There are of course some exceptions. A simple counterexample is the tangent function. Nevertheless, about 60% of the functions referenced in the *Handbook of Mathematical Functions* are solution of linear differential equations, which represents an important part of the functions used everyday by scientists and engineers.

D-finite functions also play an important role in algebraic and symbolic computation, for example for automatic proofs of identities (see for example [4]). D-finite functions are interesting from the data structure

[2]For the implementation of the algorithms, one needs to work with a *computable field*, *e.g.*, a field whose elements may be injected into the set of binary numbers in such a way that the functions corresponding to the field operations are computable on a Turing machine. For example, rational or algebraic numbers.

[3]Remember that even the set of maps $\mathbb{Q} \to \mathbb{Q}$ is uncountable and thus impossible to represent on a computer.

[4]A function f is *algebraic* if there exists a bivariate polynomial P with rational coefficients such that $P(f(x), x) = 0$.

[5]Special functions are particular functions which occur often enough in mathematics, physics or engineering that they get a name. There is obviously no general formal definition [2].

point of view too, the set of D-finite functions being stable by numerous operations. This is another good reason to use them in DDMF. What follows is just an example of the kind of results which can be obtained. Another way to discover the opportunities opened by D-finite functions is the DDMF, which actually showcases the algorithmics associated to this field of research.

Theorem 1.

- ▷ *The sum of two D-finite functions is D-finite.*
- ▷ *The product of two D-finite functions is D-finite.*
- ▷ *Any algebraic function is D-finite.*

All the proofs are actually *algorithmic*, in the sense that, for example, we have an efficient algorithm to compute a differential equation satisfied by an algebraic function, together with initial values. For a proof of those results, see [4]. For a detailed bibliography concerning D-finite functions, see [15].

1.3 Certified evaluation

Once the class of functions to study is chosen, one wants to design algorithms to manipulate those functions. Evaluation is probably one of the operations which come to mind first when speaking about mathematical functions. Our topic here is *certified* evaluation, meaning that we aim both at *reliability* and *efficiency.*

In everyday applications, the user can afford to ask the computer for a result, wait for its answer, then trust the algorithms and implementation and consider the result as correct. It is not always the case. In critical systems (*e.g.*, nuclear industry or aeronautics) it is not sufficient and users need more confidence in the result. This can be achieved by providing *proofs of correctness* and, in the most delicate situations, by *formal certification* in a proof assistant.

In the context of function evaluation, there are already tools galore to evaluate functions, with different levels of certification. To mention only a few of them, CRlibm [6] for finite-precision, MPFR [17] for arbitrary-precision or MPFI [16] for interval arithmetic. In the context of D-finite functions, we have NumGfun [14].

In this work, we are interested in *polynomial approximation.* The idea is simply to replace the functions by simpler objects, *polynomials*, while controlling the error. One of the reasons is that we have fast evaluation algorithms for polynomials (*e.g.* Horner's scheme). Thus, one can use them to evaluate functions[6].

In DDMF, as outlined before, we need to make the user *confident* in our results. The aim is a little bit different from the case of libraries like MPFR. We are not as much interested in the *existence* of those proofs as in their *readability.* If he wishes so, the user should be able to *understand* what computations have been done to get the result and check them, step-by-step.

1.4 Aims of this internship

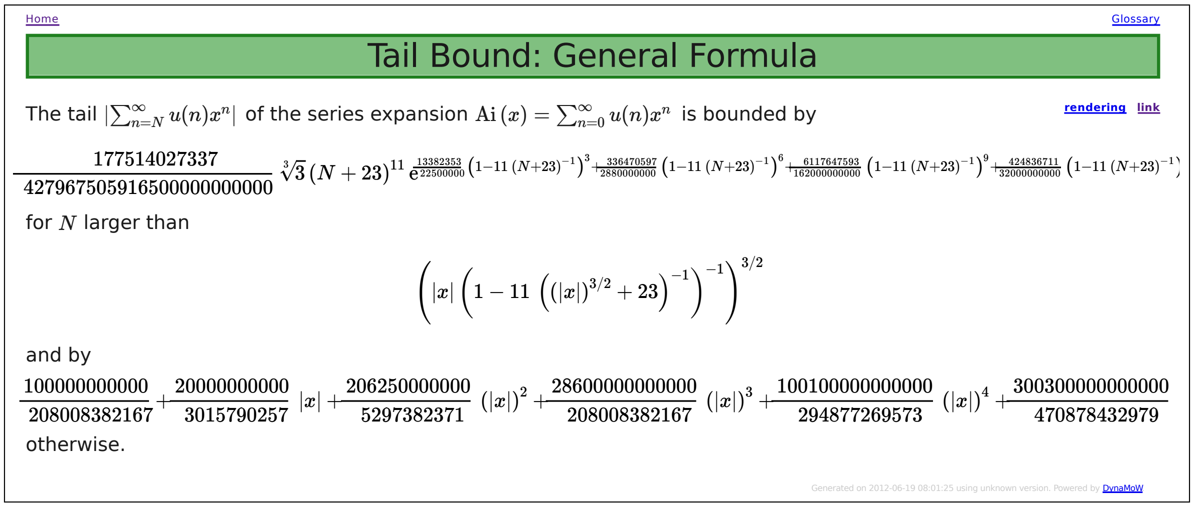
Before this work, there was a polynomial approximation module in DDMF. It was a by-product of the algorithms described in [15] computing tight bounds on the Taylor coefficients of D-finite functions. In this previous polynomial approximation module, the aim was asymptotic tightness (*i.e.* when the degree tends to infinity), not simplicity. This resulted in a robust system, able to deal with high precisions and high degrees. This system is far worse for polynomials of small degree. Moreover, the drawback is that it

[6]This is not the only application. Polynomial approximation can also be used, for example, to establish bounds on a function.

uses general formulas, so that we have to trust the computer for the whole calculation — a big numerical application. The result is illustrated in Figure 2.

One of the aims of this internship was to reimplement a polynomial approximation module and to find alternative proofs and/or simplify the existing ones in order to generate simpler, easy-to-follow proofs. That is, to design a complementary module in the above-mentioned tightness-efficiency/simplicity trade-off.

Finally, concerning proofs of correctness, there is a project, launched by Frédéric Chyzak and Assia Mahboubi on the "coqification" of DDMF, that is, on the generation of Coq proofs for some of the results stated on the web site. A long-term goal could be to prove a polynomial approximation module similar to ours in Coq. All the algorithms in this work have been designed while keeping this aim in mind. In particular, it enforced limitations on the tools we could use and the way we could do the computations.


Home Glossary

Tail Bound: General Formula

The tail $|\sum_{n=N}^{\infty} u(n)x^n|$ of the series expansion $\mathrm{Ai}\,(x) = \sum_{n=0}^{\infty} u(n)x^n$ is bounded by rendering link

$$\frac{177514027337}{42796750591650000000000} \sqrt[3]{3}\,(N+23)^{11}\, e^{\frac{13382353}{22500000}\left(1-11\,(N+23)^{-1}\right)^3+\frac{336470597}{2880000000}\left(1-11\,(N+23)^{-1}\right)^6+\frac{6117647593}{162000000000}\left(1-11\,(N+23)^{-1}\right)^9+\frac{424836711}{32000000000}\left(1-11\,(N+23)^{-1}\right.}$$

for N larger than

$$\left(|x|\left(1-11\left((|x|)^{3/2}+23\right)^{-1}\right)^{-1}\right)^{3/2}$$

and by

$$\frac{100000000000}{208008382167}+\frac{20000000000}{3015790257}\,|x|+\frac{206250000000}{5297382371}\,(|x|)^2+\frac{28600000000000}{208008382167}\,(|x|)^3+\frac{100100000000000}{294877269573}\,(|x|)^4+\frac{300300000000000}{470878432979}$$

otherwise.

Generated on 2012-06-19 08:01:25 using unknown version. Powered by DynaMoW

Figure 2: Details for a given polynomial approximation before this work.

2 Polynomial approximations via the Cauchy-Kovalevskaya method

To obtain polynomial approximations for D-finite functions, we used a classical tool, the Cauchy-Kovalevskaya method. It has already been used for symbolic-numeric computations on D-finite functions in [15], to obtain tight bounds on Taylor coefficients of D-finite functions. This technique was already used before in [18]. This section briefly recapitulates this method and sketches its application to the problem of polynomial approximation for D-finite functions. Our work has been to simplify and implement the different steps of this process, as detailed in the next section.

2.1 Polynomial approximations

Firstly let us state the polynomial approximation problem in a formal way.

Problem 1. *Given a D-finite function f defined by a differential equation*

$$a^{[r]}f^{(r)} + \cdots + a^{[0]}f = 0,$$

with $a^{[r]}(0) \neq 0$, $a^{[i]} \in \mathbb{K}[X]$ for $i = 0 \ldots r$, given initial values

$$f^{(0)}(0), \ldots, f^{(r-1)}(0),$$

and given[7] $\eta > 0$ *and* $\epsilon > 0$, *compute a polynomial function* P *such that*

$$|f(z) - P(z)| < \epsilon,$$

whenever $|z| \leq \eta$, $z \in \mathbb{K}$.

We study here the special case where P is obtained by truncating the Taylor series expansion of f. The aim is to find a correct (and hopefully as small as possible) truncation order according to the parameters η and ϵ.

2.2 Majorant series and equations

Let us recall some basic definitions.

Definition 2. *Let* f *be an analytic function defined on a neighborhood of* 0. *Write its Taylor expansion at* 0 *as*

$$f(z) = \sum_{k=0}^{+\infty} a_k z^k.$$

Then a series $g(z) = \sum_{k\geq 0} b_k z^k$ *is a* ***majorant series*** *of* f *if*

$$\forall k \in \mathbb{N}, \quad |a_k| \leq b_k.$$

This relation is written as follows:

$$f(z) \lhd g(z).$$

Suppose that $f(z)$ and $g(z)$ are two analytic functions such that $f(z) \lhd g(z)$. Then

$$\begin{aligned} \left| f(z) - \sum_{k=0}^{n} a_k z^k \right| &= \left| \sum_{k=n+1}^{+\infty} a_k z^k \right| \\ &\leq \sum_{k=n+1}^{+\infty} |a_k||z|^k \\ &\leq \sum_{k=n+1}^{+\infty} b_k |z|^k. \end{aligned}$$

Hence, if we find a majorant series $g(z)$ whose remainder has a “simple” expression, it will be easy to find an integer n such that

$$\left| f(z) - \sum_{k=0}^{n} a_k z^k \right| \leq \varepsilon,$$

whenever $|z| \leq \eta$, for any $\varepsilon > 0$ and $\eta > 0$ such that $f(z)$ is analytic on the disk $|z| \leq \eta$.

The *Cauchy-Kovalevskaya method* rests on the following proposition, which gives a way to compute majorant series for solutions of ordinary differential equations.

Lemma 1. *Let* u *and* v *be analytic solutions of the following differential equations*

$$\begin{aligned} u^{(r)} &= a^{[r-1]} u^{(r-1)} + \cdots + a^{[0]} u, \\ v^{(r)} &= b^{[r-1]} v^{(r-1)} + \cdots + b^{[0]} v, \end{aligned}$$

[7]In this paper, we do not deal with singularities. Hence η has to be less than the minimum modulus of a singularity of f.

for some meromorphic functions $\left(a^{[i]}\right)_{i=0}^{r-1}$ *and* $\left(b^{[i]}\right)_{i=0}^{r-1}$ *satisfying*

$$a^{[i]} \lhd b^{[i]}$$

for $i = 0 \ldots r-1$. *Suppose moreover that none of those functions has a singularity at* 0 *and that*

$$\left|u^{(i)}(0)\right| \leq v^{(i)}(0)$$

for $i = 0 \ldots r-1$. *Then* $u \lhd v$.

We will prove a special case of this lemma in Section 3.2 below. The proof of the version stated here is similar and may be found in [10].

According to this lemma, the problem of finding majorant series for a D-finite function boils down to the search of majorant series for the coefficients of a differential equation it satisfies, *i.e.*, for *rational functions*.

2.3 Application: Taylor approximation for D-finite functions

From now on we can sketch the application of the Cauchy-Kovalevskaya method to Problem 1. Consider a function f, satisfying the differential equation

$$f^{(r)} = a^{[r-1]} f^{(r-1)} + \cdots + a^{[0]} f,$$

where the $\left(a^{[i]}\right)_{i=0}^{r-1}$ are rational functions.

We will suppose that f admits at least one singularity. If it is not the case, for example if f is the exponential function, the reader can easily check that everything which follows is still correct if one replaces the value $\tilde{\alpha}$ defined below by any constant $\tilde{\alpha}$ such that $\frac{1}{\tilde{\alpha}} > \eta$, η being defined in Problem 1. In this case, our bounds are really crude. We explain in Section 3.4 how to remedy this problem.

A general result[8] about analytic functions is that the exponential behaviour of coefficients $(u_n)_{n\in\mathbb{N}}$ of the Taylor expansion of f at 0 is given by $(\tilde{\alpha}^n)_{n\in\mathbb{N}}$, where $\frac{1}{\tilde{\alpha}}$ is the minimum modulus of a singularity of f. More precisely, in the framework of non-entire D-finite functions, there exists a sub-exponential factor $\phi(n)$ such that

$$|u_n| \leq \tilde{\alpha}^n \phi(n),$$

for any integer n. Moreover $\tilde{\alpha}$ is the smallest positive number for which such a bound exists [15].

This invites to look for majorant series of f in the form[9]

$$f(z) \lhd \frac{A}{(1-\tilde{\alpha}z)^{\lambda}}, \tag{1}$$

for some constants A and λ.

To find such a majorant series, we proceed as follows [13]:

[8] This is a consequence of the *Cauchy-Hadamard formula* [9], which states that for a formal power series $\sum_{n\geq 0} a_k z^k$, the *radius of convergence* R satisfies the equality

$$\frac{1}{R} = \limsup_{n\to+\infty} \sqrt[n]{|a_n|}.$$

[9] At this point, the reader may wonder why we introduced a new parameter λ. In fact, as we will see in Section 3.1, $\tilde{\alpha}$ is not easy to compute, and we will replace it by an *approximation* $\alpha > \tilde{\alpha}$. Nevertheless, majorant series of the form $\frac{A}{1-\alpha z}$ (that it, with an approximation of $\tilde{\alpha}$) does not always exist. This why we introduced this "multiplicity" λ.

1. For every i, find majorant series for $a^{[i]}$ in the form
$$a^{[i]} \lhd \frac{M^{[i]}}{(1-\tilde{\alpha}z)^{r-i}}.$$

2. Check that the right-hand side of (1) is a solution of
$$y^{(r)} = \sum_{i=0}^{r-1} \frac{M^{[i]}}{(1-\tilde{\alpha}z)^{r-i}} y^{(i)},$$
if λ is[10] the (unique) positive root of
$$\tilde{\alpha}^r \lambda^{\uparrow r} = \sum_{i=0}^{r-1} M^{[i]} \tilde{\alpha}^i \lambda^{\uparrow i},$$
where $\lambda^{\uparrow i} = \lambda(\lambda+1)\dots(\lambda+i-1)$ denotes the rising factorial.

3. Let $v(z) = (1-\tilde{\alpha}z)^{-\lambda}$. According to Lemma 1, if we take
$$A = \max_i \left(\frac{f^{(i)}(0)}{v^{(i)}(0)} \right),$$
then
$$f(z) \lhd \frac{A}{(1-\tilde{\alpha}z)^{\lambda}}.$$

Note that this is just one choice among many possible classes of majorant series for D-finite functions. This one has been chosen for its simplicity and because it was easy to use to generate simple proofs for DDMF.

To conclude this section, and before diving into more details, let us outline the fact that our system is essentially independent of the underlying arithmetic. This flexibility enables, depending on the situation, to implement our algorithms in exact (rational) arithmetic, as well as finite-precision, arbitrary-precision or interval arithmetic. To simplify, we will elude questions of complexity, the choice of a computation model depending on the arithmetic used in the implementation. In DDMF (see Section 3.5) we have worked in arbitrary-precision arithmetic.

3 Contributions

In the previous section, we described a general method to find majorant series for D-finite functions. During this internship, we focused on the simplification of all the steps of this process. The aims were twofold. On the one hand it simplified the implementation of the polynomial approximation module. On the other hand it allowed to automatically generate proofs of correctness for polynomial approximations which are easy to follow for a human reader. In particular the main difference between the bounds derived in this work and those described in [15] lies in the trade-off between tightness and simplicity. We focused on simplicity and on the readability of the proofs.

[10]In fact, λ can we replaced by any greater value. In particular, depending on the underlying arithmetic, one could prefer to replace λ by an integer.

3.1 Lower bounds on moduli of singularities

First of all, for a given D-finite function f it is difficult to compute $\tilde{\alpha}$, defined as before as the reciprocal of the minimum modulus of a singularity of f. Those singularities are necessarily singularities of the differential equation satisfied by f, that is, poles of the coefficients, but the converse is false. Nevertheless, we observe that in the previous algorithm we can replace $\tilde{\alpha}$ by any value α such that $\tilde{\alpha} \le \alpha$, because

$$\frac{A}{(1-\tilde{\alpha} z)^{\lambda}} \lhd \frac{A}{(1-\alpha z)^{\lambda}}.$$

In this section, we describe an algorithm to compute such a lower bound on $\tilde{\alpha}$. Note that in our application, we want the majorant series to converge on a disk $D : |z| \le \eta$. Thus, the problem can be stated as follows.

Problem 2. *Given some rational functions $(a_i)_{i=0}^{r-1}$ and a positive real value $\eta < \frac{1}{\tilde{\alpha}}$, compute a value α such that, for every pole z_p of the functions, one has:*

$$\eta < \frac{1}{\alpha} < \frac{1}{|z_p|}.$$

It is straightforward that this problem can be solved if we can solve the following one.

Problem 3. *Given a polynomial*

$$P = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_0,$$

with $a_0 a_n \ne 0$, one would like to design an algorithm which computes

$$\rho = \min_{P(z)=0} |z|$$

by successive refinement. More precisely, for any given $0 < r < \rho$, one wants to compute a value R such that

$$r < R \le \rho.$$

This section is devoted to the resolution of Problem 3. We first express rough upper and lower bounds on ρ in terms of the coefficients $(a_k)_{k=0}^{n}$ of our polynomial P. Next we use the Dandelin-Graeffe iterative method [7] to refine this bound and make it as tight as needed.

A two-sided bound. For now, we shall prove the following technical result.

Lemma 2. *With the notations of Problem 3, let*

$$R = \frac{1}{2} \min_{\substack{1 \le k \le n \\ a_k \ne 0}} \left| \frac{a_0}{a_k} \right|^{1/k}.$$

Then

$$R < \rho \le 2nR. \tag{2}$$

This bound is a variant of the one which appears in [11], exercise 4.6.2-20. A lot of similar results on the localization of the roots of polynomials can be found in [9].

Firstly, let z be a complex number such that $|z| \le R$. We are going to prove that $P(z) \ne 0$. Consider the following trivial equality:

$$a_0 = \sum_{k=0}^{n} a_k z^k - \sum_{k=1}^{n} a_k z^k = P(z) - \sum_{k=1}^{n} a_k z^k.$$

By the triangular inequality, we have

$$|a_0| \leq |P(z)| + \sum_{k=1}^{n} |a_k| R^k. \tag{3}$$

Now let us look more carefully at the definition of R. We have

$$R^k \leq \frac{1}{2^k} \left| \frac{a_0}{a_k} \right|,$$

for every $1 \leq k \leq n$ such that $a_k \neq 0$. Hence, for every $1 \leq k \leq n$, we have

$$|a_k| R^k \leq \frac{|a_0|}{2^k}. \tag{4}$$

Next, if we inject (4) into (3), we get

$$|P(z)| \geq |a_0| \left(1 - \sum_{k=1}^{n} \frac{1}{2^k} \right) > 0,$$

(remember that $a_0 \neq 0$) so that $P(z) \neq 0$. Thus we have proved that P could not cancel on the disk $|z| \leq R$ so we necessarily have $\rho > R$.

Now let us focus on the second inequality, namely $\frac{\rho}{2n} \leq R$. To prove this one, we introduce the reciprocal polynomial

$$Q = X^n P(1/X) = a_0 X^n + a_1 X^{n-1} + \dots a_{n-1} X + a_n.$$

Because $a_0 a_n \neq 0$, its roots are exactly the reciprocals of the roots of P. In particular, for every root z of Q, we have

$$|z| \leq \rho^{-1}.$$

Thus, if we write Vieta's formulas for the polynomial Q:

$$(-1)^k \frac{a_k}{a_0} = \sum_{1 \leq i_1 < i_2 < \dots < i_k \leq n} z_{i_1} z_{i_2} \dots z_{i_k},$$

where $z_1, z_2, \dots, z_n$ are the n roots of Q, then the following inequality holds:

$$\left| \frac{a_k}{a_0} \right| \leq \binom{n}{k} \rho^{-k}$$

(there are $\binom{n}{k}$ terms in the sum).

Since

$$\binom{n}{k} = \frac{n(n-1)\dots(n-k+1)}{k!} \leq n^k,$$

we can write

$$\rho^k \leq n^k \left| \frac{a_0}{a_k} \right|,$$

whenever $a_k \neq 0$. In particular, we get

$$\rho \leq 2nR,$$

which finishes the proof of (2).

Finally, we can rewrite the bound (2) in a more convenient form:

$$\frac{\rho}{2n} \leq R < \rho. \tag{5}$$

Algorithm 1: Graeffe transform

```
    Input: A polynomial $P = \sum_{k=0}^{n} a_k X^k$.
    Output: Its Graeffe transform $G(P)$.

1 $q = \lfloor n/2 \rfloor$;
2 $P_e = \sum_{i=0}^{q} a_{2i} X^i$;
3 $P_o = \sum_{i=0}^{n-q-1} a_{2i+1} X^i$;
4 return $P_e^2 - X P_o^2$;
```

The Dendelin-Graeffe method. Consider a polynomial

$$P = \lambda(X - \alpha_1)(X - \alpha_2)\ldots(X - \alpha_n).$$

Its *Graeffe transform* $G(P)$ is defined by:

$$G(P) = \mu(X - \alpha_1^2)(X - \alpha_2^2)\ldots(X - \alpha_n^2). \tag{6}$$

Since we are only interested in the roots of P and $G(P)$, the exact values of the constants λ and μ are irrelevant and we shall ignore them.

The interesting property is that $G(P)$ can be computed efficiently, even if the roots of P are unknown. Indeed, let us write

$$P = P_e(X^2) + X P_o(X^2).$$

Then[11]

$$G(P) \propto P_e(X)^2 - X P_o(X)^2.$$

This follows directly from the fact that

$$G(P)(X^2) \propto P(X)P(-X).$$

This leads to a simple yet efficient algorithm to compute the Graeffe transform of a polynomial, Algorithm 1.

As an application, consider our original problem and define a sequence of polynomials $(P^{[i]})_{i \geq 0}$ as follows:

$$\begin{array}{rcl} P^{[0]} & = & P, \\ P^{[i+1]} & = & G(P^{[i]}), \end{array}$$

for every integer $i \geq 0$. If we define as before:

$$\rho_i = \min_{P^{[i]}(z)=0} |z|,$$

then we have immediately from (6)

$$\rho_i = \rho_0^{2^i}.$$

Now let

$$R_i = \frac{1}{2} \min_{\substack{1 \leq k \leq n \\ a_k \neq 0}} \left| \frac{a_0^{[i]}}{a_k^{[i]}} \right|^{1/k},$$

[11]As indicated above, there is an irrelevant multiplicative factor. We write $P \propto Q$ to say that there exists a constant λ such that $P = \lambda Q$.

where for every integer i,

$$P^{[i]} = a_n^{[i]} X^n + \cdots + a_1^{[i]} X + a_0^{[i]}.$$

According to the bound (5), we have

$$\rho_0 \left(\frac{1}{2n}\right)^{2^{-i}} \leq R_i^{2^{-i}} < \rho_0,$$

so that

$$\lim_{i \to \infty} R_i^{2^{-i}} = \rho_0^-.$$

This leads to an algorithm solving Problem 3, namely Algorithm 2. It will terminate whenever

$$r < \rho_0 \left(\frac{1}{2n}\right)^{2^{-i}},$$

i.e., when

$$i > \log_2 \frac{\log 2n}{\log \frac{\rho_0}{r}}.$$

Let us define R as in the bound (5). Then according to this bound, Algorithm 2 terminates after at most

$$1 + \left\lceil \log_2 \frac{\log 2n}{\log 2n \frac{R}{r}} \right\rceil$$

iterations.

Let us summarize the results we have obtained:

Theorem 2. *Algorithm 2 solves Problem 3. It is correct and, given a polynomial $P = \sum_{k=0}^{n} a_k z^k$, it terminates after at most*

$$1 + \left\lceil \log_2 \frac{\log 2n}{\log 2n \frac{R}{r}} \right\rceil$$

applications of Graeffe's transform, where

$$R = \frac{1}{2} \min_{\substack{1 \leq k \leq n \\ a_k \neq 0}} \left| \frac{a_0}{a_k} \right|^{1/k}.$$

3.2 Majorant series for rational functions

As outlined before, Lemma 1 reduces the problem of finding majorant series for a D-finite function to the one of finding majorant series for rational functions. There are several ways to find majorant series for this class of functions, see for example [15]. We designed a different, elementary method which avoids partial fraction decompositions. The price of this simplicity is obviously that our bounds are less tight than those of the algorithms cited above.

Let $f(z) = \frac{P(z)}{Q(z)}$ be a rational function. Without loss of generality, suppose that $Q(0) = 1$. We are given two positive numbers m and α. The aim is to find a constant M such that

$$f(z) \lhd \frac{M}{(1 - \alpha z)^m}.$$

Algorithm 2: Lower bound improvement

```
Input: A polynomial $P$ of degree $n$, a positive real number $r < \rho$, where $\rho = \min_{P(z)=0} |z|$.
Output: A positive real number $R$ such that $r < R \le \rho$.

1 $Q = P$;
2 $R = \frac{1}{2} \min \left\{ \left| \frac{Q_0}{Q_k} \right|^{1/k} : 1 \le k \le n,\ Q_k \neq 0 \right\}$ ;        // $Q = \sum_{i=0}^{n} Q_i X^i$
3 $p = 1$;
4 while $R \le r$ do
5     $p = p/2$;
6     $Q = \texttt{Graeffe}(Q)$ ;        // Algorithm 1
7     $R = \left( \frac{1}{2} \min \left\{ \left| \frac{Q_0}{Q_k} \right|^{1/k} : 1 \le k \le n,\ Q_k \neq 0 \right\} \right)^p$ ;        // $Q = \sum_{i=0}^{n} Q_i X^i$
8 end
9 return $R$;
```

Let us recall the Taylor expansion of the right-hand side for future reference[12]:

$$\frac{M}{(1-\alpha z)^m} = \sum_{k=0}^{+\infty} M \binom{k+m-1}{k} \alpha^k z^k. \tag{7}$$

We will find constants M_0 and M_1 such that

$$P(z) \quad \triangleleft \quad \frac{M_0}{(1-\alpha z)^{m/2}} \quad \text{and} \tag{8}$$

$$\frac{1}{Q(z)} \quad \triangleleft \quad \frac{M_1}{(1-\alpha z)^{m/2}}. \tag{9}$$

Then it is easy to see that the constant $M = M_0 M_1$ is a solution to our initial problem.

Majorant series for polynomials. Finding a constant M_0 to satisfy (8) is easy. Just write

$$P(z) = \sum_{k=0}^{n} a_k z^k$$

and take

$$M_0 = \max_{0 \le k \le n} \frac{|a_k|}{\binom{k+m/2-1}{k} \alpha^k},$$

according to the Taylor expansion (7).

Majorant series for reciprocals of polynomials. To find a constant M_1 satisfying (9), let us write

$$g(z) = \frac{1}{Q(z)} = \sum_{k=0}^{+\infty} u_k z^k$$

[12]Note that here, m is not necessarily an integer. Recall that for a complex number α and a non-negative integer n, we define

$$\binom{\alpha}{n} = \frac{\alpha(\alpha-1)\dots(\alpha-n+1)}{n!}.$$

and

$$Q(z) = \prod_{i=1}^{d}(1-\alpha_i z).$$

On the one hand, by definition of $\tilde{\alpha}$, we have

$$|\alpha_i| \leq \tilde{\alpha},$$

for $i = 1 \ldots d$. On the other hand, we have formally

$$\begin{aligned} \frac{g'(z)}{g(z)} &= \sum_{i=1}^{d} \frac{\alpha_i}{1-\alpha_i z} \\ &= \sum_{i=1}^{d}\sum_{k=0}^{+\infty} \alpha_i^{k+1} z^k \\ &= \frac{1}{z}\sum_{k=1}^{+\infty} S_k z^k, \end{aligned}$$

where $S_k = \sum_{i=1}^{d} \alpha_i^k$. We will reformulate it in terms of Taylor coefficients. Let us recall the following facts about generating series:

Lemma 3. *Let $A(z) = \sum_{n\geq 0} a_n z^n$ and $B(z) = \sum_{n\geq 0} b_n z^n$. Then*

$$zA'(z) = \sum_{n\geq 0} n a_n z^n$$

and

$$A(z)B(z) = \sum_{n\geq 0}\left(\sum_{k=0}^{n} a_k b_{n-k}\right) z^n.$$

Hence, the equation

$$zg'(z) = g(z)\sum_{k=1}^{+\infty} S_k z^k$$

becomes

$$nu_n = \sum_{k=0}^{n} u_k S_{n-k}. \tag{10}$$

Now remember we want to find a value M_1 such that (9) holds. According to (7), in terms of Taylor coefficients it is equivalent to finding a value M_1 such that

$$|u_k| \leq M_1 \binom{k+m/2-1}{k} \alpha^k, \tag{11}$$

for every integer k.

Let n be an integer. Suppose that (11) holds for every $k < n$. Then according to the recurrence formula

(10), we have

$$\begin{aligned}
|u_n| &= \left|\frac{1}{n}\sum_{k=0}^{n} u_k S_{n-k}\right| \\
&\leq \frac{1}{n}\sum_{k=0}^{n} |u_k||S_{n-k}| \\
&\leq \frac{1}{n}\sum_{k=0}^{n}\left(M_1\binom{k+m/2-1}{k}\alpha^k\right)\left(d\tilde{\alpha}^{n-k}\right) \\
&\leq M_1\binom{n+m/2-1}{n}\alpha^n\frac{d}{n}\sum_{k=0}^{n}\left(\frac{\tilde{\alpha}}{\alpha}\right)^{n-k} \\
&\leq M_1\binom{n+m/2-1}{n}\alpha^n\frac{d}{n}\frac{1}{1-\frac{\tilde{\alpha}}{\alpha}}.
\end{aligned}$$

Hence if n_0 is any integer satisfying

$$\frac{d}{n_0}\frac{1}{1-\frac{\tilde{\alpha}}{\alpha}} \leq 1,$$

then by induction

$$M_1 = \max_{0\leq k\leq n_0}\frac{|u_k|}{\binom{k+m/2-1}{k}\alpha^k}$$

is a solution to our problem.

3.3 Truncation order estimation

At this point, we are able to find majorant series for any D-finite function f in the form

$$f(z) \triangleleft \frac{A}{(1-\alpha z)^{\lambda}}. \qquad (12)$$

In this section, we show how to use this result to find polynomial approximations of f.

Let $\epsilon > 0$ and $\eta > 0$ such that the Taylor series $\sum_{k\geq 0} a_k z^k$ of f at 0 converges on the disk $D : |z| \leq \eta, z \in \mathbb{C}$. Remember that we are looking for an integer n such that

$$\left|f(z) - \sum_{k=0}^{n} a_k z^k\right| \leq \epsilon, \qquad (13)$$

for any z in D.

Now, for $z \in D$, we have:

$$\begin{aligned}
\left|f(z) - \sum_{k=0}^{n} a_k z^k\right| &= \left|\sum_{k=n+1}^{+\infty} a_k z^k\right| \\
&\leq \sum_{k=n+1}^{+\infty} |a_k||z|^k \\
&\leq \sum_{k=n+1}^{+\infty} |a_k|\eta^k.
\end{aligned}$$

According to (12), we can replace $|a_k|$ by the general coefficient of the Taylor expansion of $\phi(z) = \frac{A}{(1-\alpha z)^\lambda}$. *Cauchy's estimate* [9] states that this coefficient is bounded by $\frac{M}{r^k}$, where[13] $r = \frac{1}{2}\left(\eta + \frac{1}{\alpha}\right)$ and[14] $M = \max_{\theta\in[0,2\pi]} |\phi(re^{i\theta})|$. Thus,

$$\begin{aligned}\left|f(z) - \sum_{k=0}^{n} a_k z^k\right| &\leq \sum_{k=n+1}^{+\infty} \frac{M}{r^k}\eta^k \\ &\leq M\left(\frac{\eta}{r}\right)^{n+1} \frac{1}{1-\frac{\eta}{r}}.\end{aligned}$$

Hence to satisfy (13), it suffices to take

$$n = \log_{\frac{\eta}{r}}\left(\left(1-\frac{\eta}{r}\right)\frac{\epsilon}{M}\right) - 1$$

and one gets an ϵ-approximation polynomial on the disk D.

3.4 Economization of power series

The first version of our implementation was producing really high-degree polynomial approximations — which was expected, considering the crudeness of our bounds. On a suggestion of Marc Mezzarobba, we used an heuristic trick to improve the truncation order returned by our algorithms, known as *economization of power series* [12]. The idea is the following: imagine the user asks for an ϵ-approximation of a given D-finite function f. Firstly, we use our algorithms to compute an $\epsilon/2$-approximation of this function f, that is we get a polynomial $P = \sum_{k=0}^{n} a_k z^k$ such that

$$|f(z) - P(z)| < \epsilon/2,$$

on a disk $|z| \leq \eta$ selected by the user. Then we compute the quantity

$$d = \min\left\{0 \leq k \leq n : \sum_{i=k+1}^{n} |a_i|\eta^i < \epsilon/2\right\}.$$

Thus, if we take $Q = \sum_{k=0}^{d} a_k z^k$, by triangular inequality we have

$$\begin{aligned}|f(z) - Q(z)| &\leq |f(z) - P(z) + P(z) - Q(z)| \\ &\leq |f(z) - P(z)| + |P(z) - Q(z)| \\ &< \epsilon/2 + \left|\sum_{k=d+1}^{n} a_k z^k\right| \\ &\leq \epsilon/2 + \sum_{k=d+1}^{n} |a_k||z|^k \\ &\leq \epsilon/2 + \sum_{k=d+1}^{n} |a_k|\eta^k \\ &\leq \epsilon/2 + \epsilon/2 \\ &\leq \epsilon.\end{aligned}$$

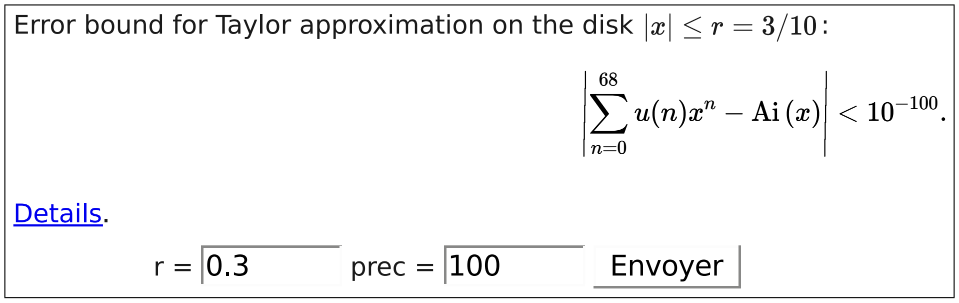
So when such a d exists, we obtain a polynomial approximation of degree smaller than the one returned by our algorithm for an $\epsilon/2$-approximation. This heuristic trick does very well in practice ; for all the functions presented in the DDMF, d exists and is much smaller than the order returned by our algorithm when computing an ϵ-approximation, leading to a division of the degree of output polynomials by a factor of 10.

[13] Any r such that $\eta < r < \frac{1}{\alpha}$ would do the job.

[14] It is straightforward to prove that $M = \frac{1}{|1-\alpha r|^\lambda}$, so that M can easily be computed.

3.5 Implementation

We have implemented those results in a polynomial approximation module and integrated it into DDMF. Figure 3 shows the interface - a new paragraph in the main page. Every polynomial approximation can be proved on demand. Proofs are fully instantiated and deal with special cases. The beginning of such a proof is illustrated in Figures 4 and 5.



Error bound for Taylor approximation on the disk $|x| \leq r = 3/10$:

$$\left| \sum_{n=0}^{68} u(n) x^n - \mathrm{Ai}\,(x) \right| < 10^{-100}.$$

Details.

r = 0.3 prec = 100 Envoyer

Figure 3: New paragraph in the main page of the DDMF.

4 Future work

The work on D-finite functions is far from being over. In this last section, we give two directions for further research.

4.1 Code generation

We have implemented a small code generator in DDMF, based on the work previously described. It outputs C code using the MPC library [8] to evaluate any D-finite function on a disk centered at 0, up to any given precision. Such an evaluation code is illustrated in Listing 1. It evaluates, using Horner's scheme, a numerical approximation of a certified truncation of the Taylor series. For now, only the truncation error is certified, the numerical error being neglected. This should be fixed in the near future. Then, it would be important to provide evaluation code on more general domains, notably on neighbourhoods of infinity or singularities.

4.2 Certification in Coq

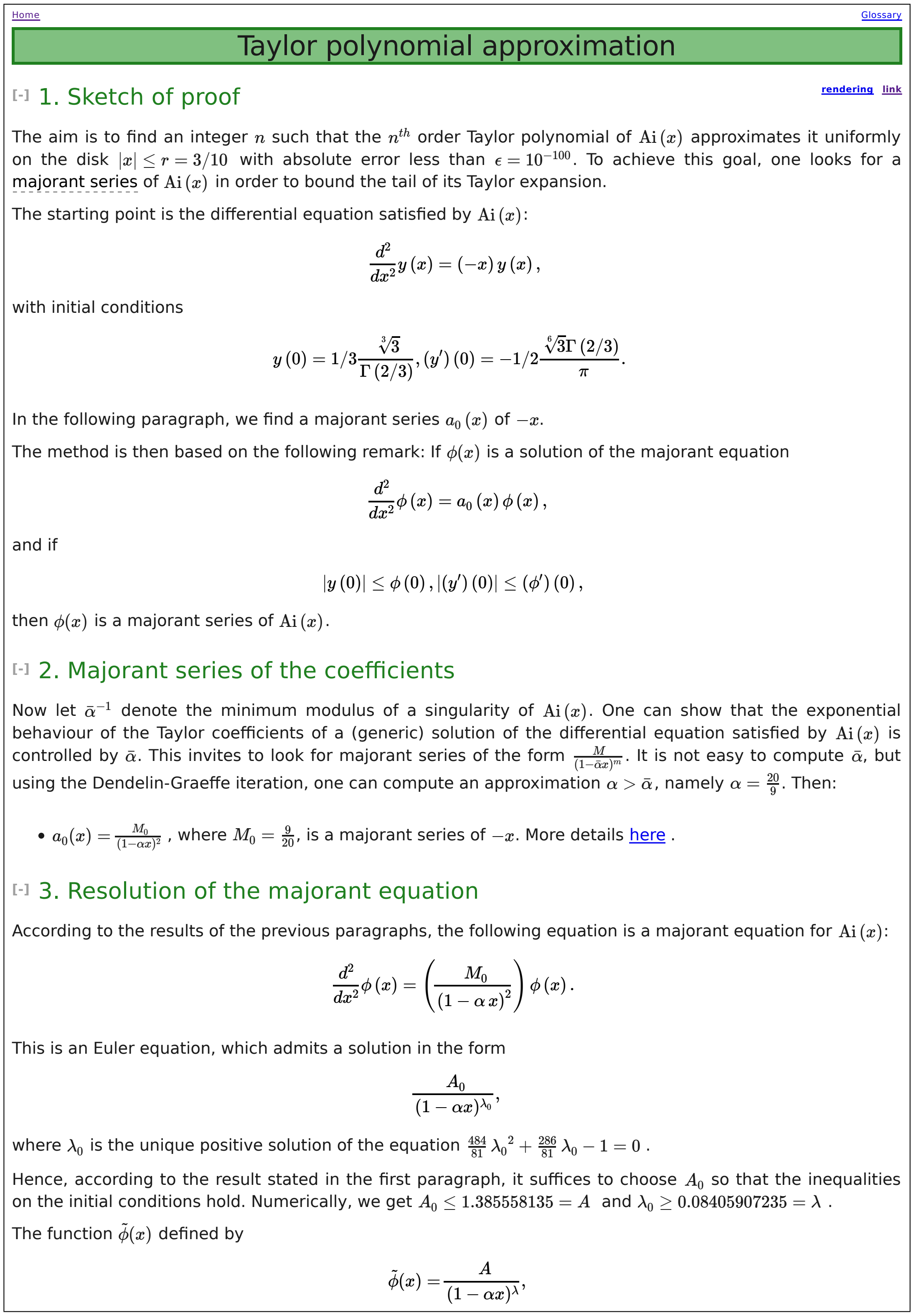
As outlined above, certification is unavoidable for critical situations in which we cannot afford to make mistakes in function evaluations. We have kept the long-term goal of Coq certification in mind. In particular we have used only elementary algorithms and theoretic tools. Therefore our module is a good candidate for this formal certification.

5 Thanks

I would like to thank Marc Mezzarobba and Nicolas Brisebarre for this nice internship, for their constant support and advice and for all the things they taught to me during those few weeks. I also would like to thank Frédéric Chyzak, Bruno Salvy and Assia Mahboubi for interesting discussions and remarks about my work.

```c
#include <mpc.h>

static mpc_t coeff[8+1];
static const char* coeff_str[8+1] =
{
  ".35502806", "-.25881940", "0.", ".5917134e-1", "-.2156829e-1",
  "0.", ".197237e-2", "-.51351e-3", "0.",
};

void init(mpfr_prec_t prec, mpc_rnd_t rnd)
{
  mpc_init2(coeff[0], prec);
  mpc_init2(coeff[1], prec);
  ...
  mpc_init2(coeff[8], prec);

  mpc_set_str(coeff[0], coeff_str[0], 10, rnd);
  ...
  mpc_set_str(coeff[8], coeff_str[8], 10, rnd);
}

void compute(mpc_t x, mpc_t acc, mpc_rnd_t rnd)
{
  int i;

  mpc_set(acc, coeff[8], rnd);

  i = 8 - 1;
  while (i >= 0)
  {
    mpc_mul(acc, acc, x, rnd);
    mpc_add(acc, acc, coeff[i], rnd);
    i--;
  }
}

void uninit(void)
{
  mpc_clear(coeff[0]);
  ...
  mpc_clear(coeff[8]);
}
```

Listing 1: An example of evaluation code in C generated by our web pages.



Home Glossary

Taylor polynomial approximation

[-] 1. Sketch of proof

rendering link

The aim is to find an integer n such that the n^{th} order Taylor polynomial of $\mathrm{Ai}\,(x)$ approximates it uniformly on the disk $|x| \leq r = 3/10$ with absolute error less than $\epsilon = 10^{-100}$. To achieve this goal, one looks for a majorant series of $\mathrm{Ai}\,(x)$ in order to bound the tail of its Taylor expansion.

The starting point is the differential equation satisfied by $\mathrm{Ai}\,(x)$:

$$\frac{d^2}{dx^2} y\,(x) = (-x)\, y\,(x)\,,$$

with initial conditions

$$y\,(0) = 1/3 \frac{\sqrt[3]{3}}{\Gamma\,(2/3)}, (y')\,(0) = -1/2 \frac{\sqrt[6]{3}\Gamma\,(2/3)}{\pi}.$$

In the following paragraph, we find a majorant series $a_0\,(x)$ of $-x$.

The method is then based on the following remark: If $\phi(x)$ is a solution of the majorant equation

$$\frac{d^2}{dx^2} \phi\,(x) = a_0\,(x)\, \phi\,(x)\,,$$

and if

$$|y\,(0)| \leq \phi\,(0)\,, |(y')\,(0)| \leq (\phi')\,(0)\,,$$

then $\phi(x)$ is a majorant series of $\mathrm{Ai}\,(x)$.

[-] 2. Majorant series of the coefficients

Now let $\bar{\alpha}^{-1}$ denote the minimum modulus of a singularity of $\mathrm{Ai}\,(x)$. One can show that the exponential behaviour of the Taylor coefficients of a (generic) solution of the differential equation satisfied by $\mathrm{Ai}\,(x)$ is controlled by $\bar{\alpha}$. This invites to look for majorant series of the form $\frac{M}{(1-\bar{\alpha}x)^m}$. It is not easy to compute $\bar{\alpha}$, but using the Dendelin-Graeffe iteration, one can compute an approximation $\alpha > \bar{\alpha}$, namely $\alpha = \frac{20}{9}$. Then:

- $a_0(x) = \frac{M_0}{(1-\alpha x)^2}$, where $M_0 = \frac{9}{20}$, is a majorant series of $-x$. More details here .

[-] 3. Resolution of the majorant equation

According to the results of the previous paragraphs, the following equation is a majorant equation for $\mathrm{Ai}\,(x)$:

$$\frac{d^2}{dx^2} \phi\,(x) = \left(\frac{M_0}{(1 - \alpha\, x)^2} \right) \phi\,(x)\,.$$

This is an Euler equation, which admits a solution in the form

$$\frac{A_0}{(1 - \alpha x)^{\lambda_0}},$$

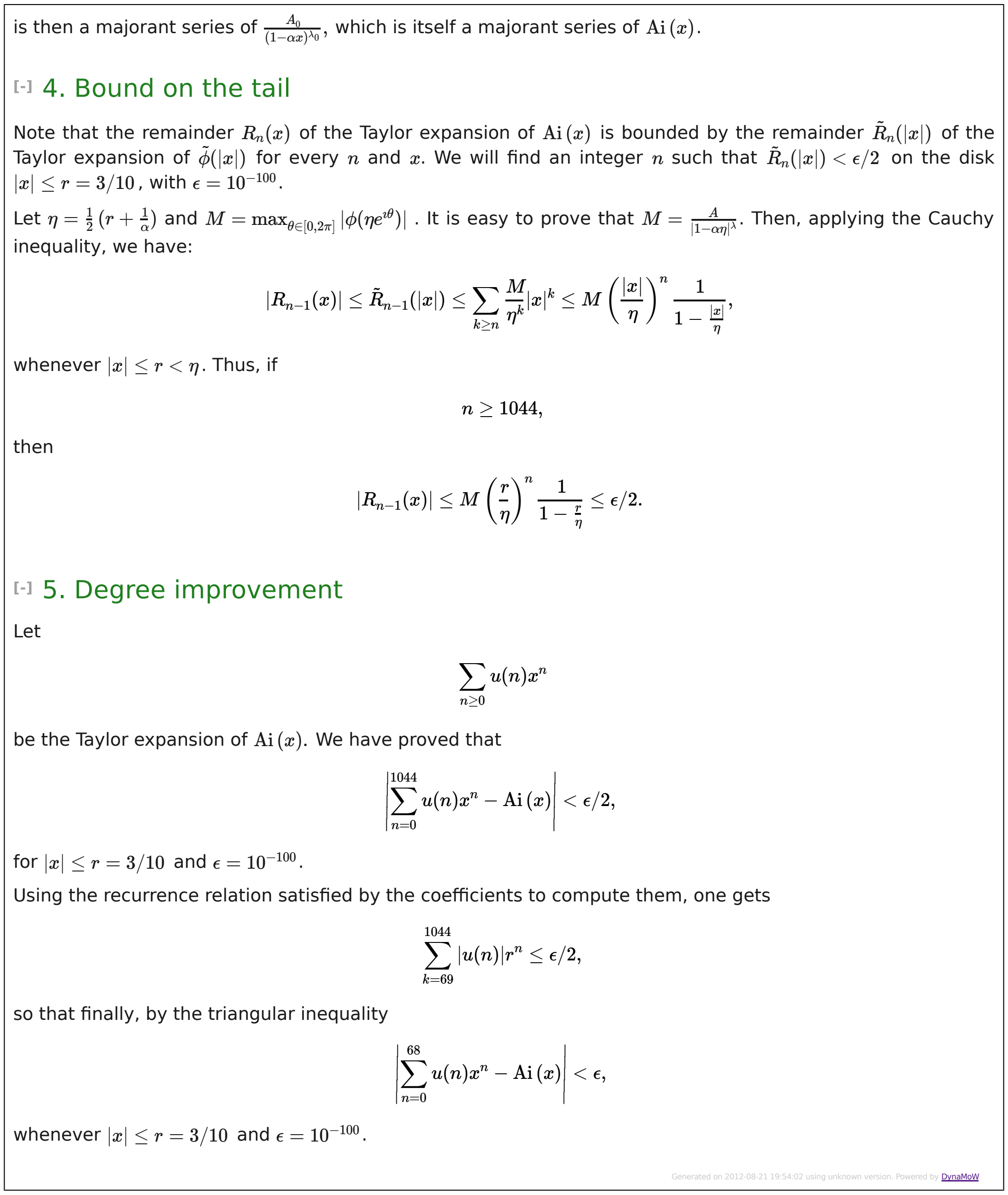
where λ_0 is the unique positive solution of the equation $\frac{484}{81} {\lambda_0}^2 + \frac{286}{81} \lambda_0 - 1 = 0$.

Hence, according to the result stated in the first paragraph, it suffices to choose A_0 so that the inequalities on the initial conditions hold. Numerically, we get $A_0 \leq 1.385558135 = A$ and $\lambda_0 \geq 0.08405907235 = \lambda$.

The function $\tilde{\phi}(x)$ defined by

$$\tilde{\phi}(x) = \frac{A}{(1 - \alpha x)^{\lambda}},$$

Figure 4: Details for a given polynomial approximation (1).



is then a majorant series of $\frac{A_0}{(1-\alpha x)^{\lambda_0}}$, which is itself a majorant series of $\mathrm{Ai}\,(x)$.

[-] 4. Bound on the tail

Note that the remainder $R_n(x)$ of the Taylor expansion of $\mathrm{Ai}\,(x)$ is bounded by the remainder $\tilde{R}_n(|x|)$ of the Taylor expansion of $\tilde{\phi}(|x|)$ for every n and x. We will find an integer n such that $\tilde{R}_n(|x|) < \epsilon/2$ on the disk $|x| \le r = 3/10$, with $\epsilon = 10^{-100}$.

Let $\eta = \frac{1}{2}\left(r + \frac{1}{\alpha}\right)$ and $M = \max_{\theta \in [0,2\pi]} |\phi(\eta e^{\imath\theta})|$. It is easy to prove that $M = \frac{A}{|1-\alpha\eta|^{\lambda}}$. Then, applying the Cauchy inequality, we have:

$$|R_{n-1}(x)| \le \tilde{R}_{n-1}(|x|) \le \sum_{k \ge n} \frac{M}{\eta^k}|x|^k \le M\left(\frac{|x|}{\eta}\right)^n \frac{1}{1-\frac{|x|}{\eta}},$$

whenever $|x| \le r < \eta$. Thus, if

$$n \ge 1044,$$

then

$$|R_{n-1}(x)| \le M\left(\frac{r}{\eta}\right)^n \frac{1}{1-\frac{r}{\eta}} \le \epsilon/2.$$

[-] 5. Degree improvement

Let

$$\sum_{n \ge 0} u(n) x^n$$

be the Taylor expansion of $\mathrm{Ai}\,(x)$. We have proved that

$$\left|\sum_{n=0}^{1044} u(n)x^n - \mathrm{Ai}\,(x)\right| < \epsilon/2,$$

for $|x| \le r = 3/10$ and $\epsilon = 10^{-100}$.

Using the recurrence relation satisfied by the coefficients to compute them, one gets

$$\sum_{k=69}^{1044} |u(n)| r^n \le \epsilon/2,$$

so that finally, by the triangular inequality

$$\left|\sum_{n=0}^{68} u(n)x^n - \mathrm{Ai}\,(x)\right| < \epsilon,$$

whenever $|x| \le r = 3/10$ and $\epsilon = 10^{-100}$.

Generated on 2012-08-21 19:54:02 using unknown version. Powered by DynaMoW

Figure 5: Details for a given polynomial approximation (2).

6 References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, ninth Dover printing, tenth GPO printing edition, 1964. `http://people.math.sfu.ca/~cbm/aands/`.

[2] George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*, volume 71 of *Encyclopedia of mathematics and its applications.* Cambridge University Press, pub-CAMBRIDGE:adr, 1999.

[3] Alexandre Benoit, Frédéric Chyzak, Alexis Darrasse, Stefan Gerhold, Marc Mezzarobba, and Bruno Salvy. The Dynamic Dictionary of Mathematical Functions (DDMF). In *Mathematical Software - ICMS*, pages 35–41, 2010.

[4] Alin Bostan, Frédéric Chyzak, Marc Giusti, Romain Lebreton, Bruno Salvy, and Éric Schost. Algorithmes efficaces en calcul formel, 2012. `http://algo.inria.fr/chyzak/mpri/poly-20120112.pdf`.

[5] Frédéric Chyzak and Alexis Darrasse. Using Camlp4 for presenting dynamic mathematics on the web: DynaMoW, an OCaml language extension for the run-time generation of mathematical contents and their presentation on the web. In Olivier Danvy, editor, *ICFP'11 (September 19–21, 2011, Tokyo, Japan)*, page 259–265. ACM, 2011.

[6] The Correctly Rounded mathematical library. `http://lipforge.ens-lyon.fr/www/crlibm/`.

[7] J. H. Davenport and M. Mignotte. On finding the largest root of a polynomial. *Mathematical Modelling and Numerical Analysis*, 24(6):693–696, 1990.

[8] Andreas Enge, Mickaël Gastineau, Philippe Théveny, and Paul Zimmermann. *mpc — A library for multiprecision complex arithmetic with exact rounding.* `http://mpc.multiprecision.org/`.

[9] P. Henrici. *Applied and computational complex analysis I.* John Wiley, New York, 1974.

[10] P. Henrici. *Applied and computational complex analysis II.* John Wiley, New York, 1977.

[11] D. E. Knuth. *The Art of Computer Programming: Volume 2, Seminumerical Algorithms.* Addison-Wesley Publishing Company, 1969.

[12] C. Lanczos. *Applied Analysis.* Van Nostrand, Prentice-Hall, 1956.

[13] Marc Mezzarobba. Génération automatique de procédures numériques pour les fonctions D-finies. Rapport de stage, Master parisien de recherche en informatique, 2007. `http://marc.mezzarobba.net/m2/Mezzarobba_MScThesisMPRI2007-1.2.pdf`.

[14] Marc Mezzarobba. NumGfun: a package for numerical and analytic computation with D-finite functions. In Wolfram Koepf, editor, *ISSAC '10*, pages 139–146. ACM, 2010.

[15] Marc Mezzarobba. *Autour de l'évaluation numérique des fonctions D-finies.* Thèse de doctorat, École polytechnique, 2011. `http://marc.mezzarobba.net/these/these-mezzarobba.pdf`.

[16] MPFI, a multiple precision interval arithmetic library based on MPFR. `http://perso.ens-lyon.fr/nathalie.revol/software.html`.

[17] The GNU MPFR Library. `http://www.mpfr.org/`.

[18] Joris van der Hoeven. Fast Evaluation of Holonomic Functions Near and in Regular Singularities. *J. Symb. Comput*, 31(6):717–743, 2001.