Relaxed resolution of implicit equations

Joris van der Hoeven

Relaxed resolution of implicit equations*

Joris van der Hoeven

LIX, CNRS
École polytechnique
91128 Palaiseau Cedex
France

Email: vdhoeven@lix.polytechnique.fr
Web: http://lix.polytechnique.fr/~vdhoeven

December 17, 2009

The technique of relaxed power series expansion provides an efficient way to solve equations of the form $F=\Phi(F)$, where the unknown F is a vector of power series, and where the solution can be obtained as the limit of the sequence $0, \Phi(0), \Phi(\Phi(0)), \ldots$. With respect to other techniques, such as Newton's method, two major advantages are its generality and the fact that it takes advantage of possible sparseness of Φ. In this paper, we extend the relaxed expansion mechanism to more general implicit equations of the form $\Phi(F)=0$.

Keywords: Implicit equation, relaxed power series, algorithm

A.M.S. subject classification: 68W25, 42-04, 68W30, 30B10, 33F05, 11Y55

1. Introduction

Let $\mathbb{K}$ be an effective field of constants of characteristic zero. Let $F=(F^{[1]}, \ldots, F^{[r]})$ be a column vector of r indeterminate series in $\mathbb{K}[[z]]$. We may also consider F as a power series $F_0+F_1 z+\cdots \in \mathbb{K}^r[[z]]$. Let $\Phi(F)=(\Phi(F)^{[1]}, \ldots, \Phi(F)^{[r]})$ be a column vector of expressions built up from F, z and constants in $\mathbb{K}$ using ring operations, differentiation and integration (with constant term zero). Finally, let $C_0, \ldots, C_{k-1} \in \mathbb{K}^r$ be a finite number of initial conditions. Assume that the system

$$\left\{\begin{array}{rcl} \Phi(F) & = & 0 \\ F_0 & = & C_0 \\ & \vdots & \\ F_{k-1} & = & C_{k-1} \end{array}\right. \tag{1}$$

admits a unique solution $f \in \mathbb{K}[[z]]^r$. In this paper, we are interested in the efficient computation of this solution up to a given order n.

In the most favourable case, the equation $\Phi(F)=0$ is of the form

$$F-\Psi(F) \;=\; 0, \tag{2}$$

where the coefficient $\Psi(F)_n$ of z^n in $\Psi(F)$ only depends on earlier coefficients $F_0, \ldots, F_{n-1}$ of F, for each $n \in \mathbb{N}$. In that case,

$$F_n \;=\; \Psi(F)_n$$

[*]. This work has been supported by the ANR-09-JCJC-0098-01 MaGiX project, as well as a Digiteo 2009-36HD grant and Région Ile-de-France.

actually provides us with a recurrence relation for the computation of the solution. Using the technique of relaxed power series expansions [vdH02a, vdH07], which will briefly be recalled in section 2, it is then possible to compute the expansion $F_{;n}=F_0+\cdots+F_{n-1}z^{n-1}$ up till order n in time

$$\mathsf{T}(n) \;=\; s\,\mathsf{R}(n)+O(t\,n), \tag{3}$$

where s is the number of multiplications occurring in Ψ, where is t the total size of Ψ as an expression, and $\mathsf{R}(n)$ denotes the complexity of relaxed multiplication of two power series up till order n. Here we assume that Ψ is represented by a directed acyclic graph, with possible common subexpressions. For large n, we have $\mathsf{R}(n)=O(\mathsf{M}(n)\log n)$, where $\mathsf{M}(n)=O(n\log n\log\log n)$ denotes the complexity [CT65, SS71, CK91] of multiplying two polynomials of degrees $< n$. If $\mathbb{K}$ admits sufficiently many 2^p-th roots of unity, then we even have $\mathsf{R}(n)=O(\mathsf{M}(n)\,\mathrm{e}^{2\sqrt{\log 2\log\log n}})$ and $\mathsf{M}(n)=O(n\log n)$. For moderate n, when polynomial multiplication is done naively or using Karatsuba's method, relaxed multiplication is as efficient as the truncated multiplication of polynomials at order n.

One particularly important example of an equation of the above type is the integration of a dynamical system

$$F \;=\; F_0+\textstyle\int\Psi(F), \tag{4}$$

where Ψ is algebraic (i.e. does not involve differentiation or integration). In that case, given the solution f up till order n, we may consider the linearized system

$$E' \;=\; \Psi(f)+J_\Psi(f)\,E+O(z^{2n})$$

up till order $2\,n$, where $J_\Psi(f)$ stands for the Jacobian matrix associated to Ψ at f. If we also have a fundamental system of solutions of $E'=J_\Psi(f)\,E$ up till order n, then one step of Newton's method allows us to find the solution of (4) and a new fundamental system of solutions of the linearized equation up till order $2\,n$ [BK78, BCO+06]. A careful analysis shows that this leads to an algorithm of time complexity

$$\mathsf{T}(n) \;=\; \mathsf{M}(n)\,(2\,s\,r+2\,s+13/6\,r^2+4/3\,r+o(1))+O(t\,r\,n). \tag{5}$$

In [vdH06], this bound has been further improved to

$$\mathsf{T}(n) \;=\; \mathsf{M}(n)\,(2\,s+4/3\,r+o(1))+O(t\,n), \tag{6}$$

under the assumptions that $\mathbb{K}$ admits sufficiently many 2^p-th roots of unity and that $r=O(\log n)$.

Although the complexity (5) is asymptotically better than (3) for very large n, the relaxed approach often turns out to be more efficient in practice. Indeed, Newton's method both suffers from a larger constant factor and the fact that we profit less from the potential sparsity of the system. Moreover, as long as multiplications are done in the naive or Karatsuba model, the relaxed approach is optimal in the sense that the computation of the solution takes roughly the same time as its verification. Another advantage of the relaxed approach is that it generalizes to more general functional equations and partial differential equations.

Let us now return to our original implicit system (1). A first approach for its resolution is to keep differentiating the system with respect to F until it becomes equivalent to a system of the form (2). For instance, if Φ is algebraic, then differentiation of (1) yields

$$J_\Phi(F)\,F'+\frac{\partial\Phi}{\partial z}(F) \;=\; 0.$$

Consequently, if $J_{\Phi}(f)_0$ is invertible, then

$$F = F_0 - \textstyle\int J_{\Phi}(F)^{-1} \frac{\partial \Phi}{\partial z}(F)$$

provides us with an equivalent system which can be solved by one of the previous methods. Unfortunately, this method requires the computation of the Jacobian, so we do not longer exploit the potential sparsity of the original system.

If Φ is a system of differentially algebraic equations, then we may also seek to apply Newton's method. For non degenerate systems and assuming that we have computed the solution f and a fundamental system of solutions for the linearized equation up till order n, one step of Newton's method yields an extension of the solutions up till order $2n - i$, for a fixed constant $i \in \mathbb{N}$. From an asymptotic point of view, this means that the complexities (5) and (6) remain valid.

It is natural to ask for a relaxed algorithm for the resolution of (1), with a similar complexity as (3). We will restrict our attention to so-called "quasi-linear equations", for which the linearized system is "non degenerate". This concept will be introduced formally in section 3 and studied in more detail in section 6. In section 4, we present the main algorithm of this paper for the relaxed resolution of (1).

The idea behind the algorithm is simple: considering not yet computed coefficients of F as formal unknowns, we solve the system of equations $\Phi(F)_0 = \cdots = \Phi(F)_n = 0$ for increasing values of n. In particular, the coefficients of the power series involved in the resolution process are no longer in $\mathbb{K}$, but rather polynomials in F_0, F_1, ... For each subexpression $\Psi(F)$ of $\Phi(F)$ and modulo adequate substitution of known coefficients F_n by their values f_n, it turns out that there exist constants $s \in \mathbb{Z}$ and $i \in \mathbb{N}$, such that $\Psi(F)_n$ is a constant plus a linear combination of $F_{n-s-i+1}, \ldots, F_{n-s}$, for large n. Moreover, each relaxed multiplication with symbolic coefficients can be reduced to a relaxed multiplication with constant coefficients and a finite number of scalar multiplications with symbolic coefficients. The main result is stated in theorem 5 and generalizes the previous complexity bound (3).

In section 6, we provide a more detailed study of the linearized system associated to (1). This will allow us to make the dependency of $\Psi(F)_n$ on $F_{n-s-i+1}, \ldots, F_{n-s}$ more explicit. On the one hand, given a quasi-linear system on input, this will enable us to provide a certificate that the system is indeed quasi-linear. On the other hand, the asymptotic complexity bounds can be further sharpened in lucky situations (see theorem 11). Finally, in the last section 7, we outline how to generalize our approach to more general functional equations and partial differential equations.

2. Relaxed power series

Throughout this article, $\mathbb{K}$ will denote an effective field of characteristic zero. This means that elements in $\mathbb{K}$ can be encoded by data structures on a computer and that we have algorithms for performing the field operations in $\mathbb{K}$.

Let us briefly recall the technique of relaxed power series computations, which is explained in more detail in [vdH02a]. In this computational model, a power series $f \in \mathbb{K}[[z]]$ is regarded as a stream of coefficients $f_0, f_1, \ldots$. When performing an operation $g = \Phi(f_1, \ldots, f_k)$ on power series it is required that the coefficient g_n of the result is output as soon as sufficiently many coefficients of the inputs are known, so that the computation of g_n does not depend on the further coefficients. For instance, in the case of a multiplication $h = f g$, we require that h_n is output as soon as $f_0, \ldots, f_n$ and $g_0, \ldots, g_n$ are known. In particular, we may use the naive formula $h_n = \sum_{i=0}^{n} f_i \, g_{n-i}$ for the computation of h_n.

The additional constraint on the time when coefficients should be output admits the important advantage that the inputs may depend on the output, provided that we add a small delay. For instance, the exponential $g = \exp f$ of a power series $f \in z\,\mathbb{K}[[z]]$ may be computed in a relaxed way using the formula

$$g \;=\; \int f'\, g.$$

Indeed, when using the naive formula for products, the coefficient g_n is given by

$$g_n \;=\; \tfrac{1}{n}\,(f_1\, g_{n-1} + 2\, f_2\, g_{n-2} + \cdots + n\, f_n'\, g_0),$$

and the right-hand side only depends on the previously computed coefficients $g_0, \ldots, g_{n-1}$.

The main drawback of the relaxed approach is that we cannot directly use fast algorithms on polynomials for computations with power series. For instance, assuming that $\mathbb{K}$ has sufficiently many 2^p-th roots of unity and that field operations in $\mathbb{K}$ can be done in time $O(1)$, two polynomials of degrees $< n$ can be multiplied in time $\mathsf{M}(n) = O(n \log n)$, using FFT multiplication [CT65]. Given the truncations $f_{;n} = f_0 + \cdots + f_{n-1}\, z^{n-1}$ and $g_{;n} = g_0 + \cdots + g_{n-1}\, z^{n-1}$ at order n of power series $f, g \in \mathbb{K}[[z]]$, we may thus compute the truncated product $(f\, g)_{;n}$ in time $\mathsf{M}(n)$ as well. This is much faster than the naive $O(n^2)$ relaxed multiplication algorithm for the computation of $(f\, g)_{;n}$. However, the formula for $(f\, g)_0$ when using FFT multiplication depends on all input coefficients $f_0, \ldots, f_{n-1}$ and $g_0, \ldots, g_{n-1}$, so the fast algorithm is not relaxed. Fortunately, efficient relaxed multiplication algorithms do exist:

Theorem 1. [vdH97, vdH02a] *Let $\mathsf{M}(n)$ be the time complexity for the multiplication of polynomials of degrees $< n$ in $\mathbb{K}[z]$. Then there exists a relaxed multiplication algorithm for series in $\mathbb{K}[[z]]$ of time complexity $\mathsf{R}(n) = O(\mathsf{M}(n) \log n)$.*

Theorem 2. [vdH07] *If $\mathbb{K}$ admits a primitive 2^p-th root of unity for all p, then there exists a relaxed multiplication algorithm of time complexity $\mathsf{R}(n) = O(n \log n\, \mathrm{e}^{2\sqrt{\log 2 \log \log n}})$. In practice, the existence of a 2^{p+1}-th root of unity with $2^p \geqslant n$ suffices for multiplication up to order n.*

An efficient C++ implementation of relaxed power series is available in the Mathemagix system [vdH+02b]. Leaving low-level pointer and memory management details apart, we will outline how the implementation works, using an informal pseudo-language. Relaxed power series in $\mathbb{K}[[z]]$ are implemented as an abstract base class $\mathsf{Series}_{\mathbb{K}}$ which contains the already computed coefficients and a protected virtual method next for computing the next coefficient. For instance, the naive product of f, g: $\mathsf{Series}_{\mathbb{K}}$ can be implemented using the following concrete derived class $\mathsf{ProductSeries}_{\mathbb{K}}$:

```
Class $\mathsf{ProductSeries}_{\mathbb{K}} \trianglerighteq \mathsf{Series}_{\mathbb{K}}$
    Fields $f, g\colon \mathsf{Series}_{\mathbb{K}}$
    Constructor product $(\tilde{f}\colon \mathsf{Series}_{\mathbb{K}}, \tilde{g}\colon \mathsf{Series}_{\mathbb{K}})$
        $f := \tilde{f}, g := \tilde{g}$
    Method next $(n\colon \mathbb{N})$
        Return $\sum_{i=0}^{n}$ coefficient $(f, i)$ coefficient $(g, n-i)$
```

Let us briefly explain this code. In addition to the vector with the already computed coefficients (which is derived from $\mathsf{Series}_{\mathbb{K}}$), the class $\mathsf{ProductSeries}_{\mathbb{K}}$ contains two data fields for the multiplicands f and g. The constructor product (f, g) returns the product of two series f, g: $\mathsf{Series}_{\mathbb{K}}$ and the method next computes $(f\, g)_n$ using the naive relaxed method. The method next does not take care of remembering previously computed coefficients and does not make sure that coefficients are computed in order. Therefore, a different public function coefficient is used for the computation of the coefficients f_i and g_i:

```
Function coefficient ($f$: $\mathrm{Series}_{\mathbb{K}}$, $n$: $\mathbb{N}$)
    Let $f_0, \ldots, f_{\# f}$ denote the already computed coefficients of $f$
    If $n > \# f$ then for $i = \# f + 1, \ldots, n$ do $f_i := f.\mathrm{next}(i)$
    Return $f_n$
```

In the case of implicitly defined power series f, the method next involves the series $f = \mathrm{this}$ itself. For instance, exponentiation $f := \exp g$ can be implemented as follows:

```
Class $\mathrm{ExpSeries}_{\mathbb{K}} \trianglerighteq \mathrm{Series}_{\mathbb{K}}$
    Fields $f, g$: $\mathrm{Series}_{\mathbb{K}}$
    Constructor product ($\tilde{g}$: $\mathrm{Series}_{\mathbb{K}}$)
        $g := \tilde{g}$, $f :=$ integrate (product (derive ($g$), this))
    Method next ($n$: $\mathbb{N}$)
        If $n = 0$ then return 1
        Else return coefficient ($f, n$)
```

EXAMPLE 3. Let us expand the exponential of $g = z + z^2 + z^3 + \cdots$ using the above algorithms. Besides f and g, three auxiliary series $\varphi = g'$, $\psi = \varphi f$ and $\chi = \int \psi$ are created. Now the computation of f_4 gives rise to the following sequence of assignments:

$$\begin{array}{ccccccccccccccccc}
f_0 & & & := & & & 1 & \varphi_0 & := & g_0 & = & 1 & \psi_0 & := & \varphi_0 f_0 & = & 1\\
 & & \chi_0 & := & 0 & & & & & & & & & & & & \\
f_1 & := & \chi_1 & := & \psi_0 & = & 1 & \varphi_1 & := & 2 g_1 & = & 2 & \psi_1 & := & \varphi_0 f_1 + \varphi_1 f_0 & = & 3\\
f_2 & := & \chi_2 & := & \frac{\psi_1}{2} & = & \frac{3}{2} & \varphi_2 & := & 3 g_2 & = & 3 & \psi_2 & := & \varphi_0 f_2 + \varphi_1 f_1 + \varphi_2 f_0 & = & \frac{13}{2}\\
f_3 & := & \chi_3 & := & \frac{\psi_2}{3} & = & \frac{13}{6} & \varphi_3 & := & 4 g_3 & = & 4 & \psi_3 & := & \varphi_0 f_3 + \varphi_1 f_2 + \varphi_2 f_1 + \varphi_3 f_0 & = & \frac{73}{6}\\
f_4 & := & \chi_4 & := & \frac{\psi_3}{4} & = & \frac{73}{24} & \cdots & & & & & & & & &
\end{array}$$

3. IMPLICIT POWER SERIES EQUATIONS

Let $F = (F^{[1]}, \ldots, F^{[r]})$ be a column vector of r indeterminate series in $\mathbb{K}[[z]]$. Alternatively, F may be considered as a series with formal coefficients

$$\begin{aligned}
F &= F_0 + F_1 z^1 + \cdots,\\
F_n &= (F_n^{[1]}, \ldots, F_n^{[r]})
\end{aligned}$$

We will denote by $\mathbb{E}$ the set of expressions built up from F, z and constants in $\mathbb{K}$ using ring operations, differentiation and integration (with $(\int f)_0 = 0$ for all $f \in \mathbb{K}[[z]]$). Setting

$$\begin{aligned}
\mathbb{P} &= \mathbb{K}[F_0, F_1, \ldots]\\
&= \mathbb{K}[F_0^{[1]}, \ldots, F_0^{[r]}, F_1^{[1]}, \ldots, F_1^{[r]}, \ldots],
\end{aligned}$$

any expression in $\mathbb{E}$ may then be regarded as an element of $\mathbb{P}[[z]]$.

For each $\Phi \in \mathbb{E}$, we define $v_\Phi \in \mathbb{Z} \cup \{\infty\}$ using the following rules:

$$\begin{aligned}
\Phi \in \mathbb{K}[z] &\Longrightarrow v_\Phi = \mathrm{val}_z \Phi\\
\Phi \in \{F^{[1]}, \ldots, F^{[r]}\} &\Longrightarrow v_\Phi = 0\\
\Phi \in \{\Psi + \Omega, \Psi - \Omega\} &\Longrightarrow v_\Phi = \min \{v_\Psi, v_\Omega\}\\
\Phi = \Psi \Omega &\Longrightarrow v_\Phi = v_\Psi + v_\Omega\\
\Phi = \Psi' &\Longrightarrow v_\Phi = v_\Psi - 1\\
\Phi = \int \Psi &\Longrightarrow v_\Phi = v_\Psi + 1.
\end{aligned}$$

By induction, we have

$$\begin{aligned}\operatorname{val}_z \Phi &\geqslant v_\Phi\\ \Phi_n &\in \mathbb{K}[F_0, \ldots, F_{n-v_\Phi}],\end{aligned}$$

for all $\Phi \in \mathbb{E}$ and $n \in \mathbb{N}$.

Let $\Phi = (\Phi^{[1]}, \ldots, \Phi^{[r]})$ be a column vector of r expressions in $\mathbb{E}$. We will assume that Φ depends on each of the indeterminates $F^{[1]}, \ldots, F^{[r]}$. Given $C_0, \ldots, C_{k-1} \in \mathbb{K}^r$, consider the implicit system with initial conditions

$$\left\{\begin{aligned}\Phi &= 0\\ F_0 &= C_0\\ &\vdots\\ F_{k-1} &= C_{k-1}\end{aligned}\right. \tag{7}$$

For any $n \geqslant k$, this system implies the following system Σ_n of equations in $\mathbb{K}[F_0, \ldots, F_n]$:

$$\Sigma_n = \left\{\begin{aligned}\Phi^{[1]}_{n-v_{\Phi^{[1]}}} &= 0\\ &\vdots\\ \Phi^{[r]}_{n-v_{\Phi^{[r]}}} &= 0\end{aligned}\right. \tag{8}$$

The system (7) is equivalent to the systems $\Sigma_0, \Sigma_1, \ldots$ together with the initial conditions $F_0 = C_0, \ldots, F_{k-1} = C_{k-1}$.

In what follows, we will assume that (7) admits a unique solution $f \in \mathbb{K}[[z]]^r$. Given $\Psi \in \mathbb{P}[[z]]$ and $i \in \mathbb{N}$, we will denote by $\sigma_i(\Psi)$ the series in $\mathbb{P}[[z]]$ such that $\sigma_i(\Psi)_n$ is the result of the substitution of F_j by f_j in Ψ_n, for all $n \in \mathbb{N}$ and $j \leqslant n + v_\Phi - i$. If, for all $i \in \mathbb{N}$, there exists an $N_i \in \mathbb{N}$ such that $\sigma_i(\Phi^{[j]})_{n-v_{\Phi^{[j]}}}$ is linear in $F_{n-i+1}, \ldots, F_n$ for all $n \geqslant N_i$ and $j = 1, \ldots, r$, then we say that (7) is *ultimately linear*. In that case, $\sigma_i(\Sigma_n)$ becomes a linear system of equations in $F_{n-i+1}, \ldots, F_n$. More generally, the combined system

$$\Sigma_{n,i} = \left\{\begin{array}{l}\sigma_1(\Sigma_n)\\ \sigma_2(\Sigma_{n+1})\\ \vdots\\ \sigma_i(\Sigma_{n+i-1})\end{array}\right. \tag{9}$$

is a linear system of equations in $F_n, \ldots, F_{n+i-1}$ for all sufficiently large n. If $\Sigma_{n,i}$ is linear and F_n can be eliminated from $\Sigma_{n,i}$ for all sufficiently large, then we say that (7) is *quasi-linear*. The minimal i for which F_n can be eliminated from $\Sigma_{n,i}$ for all sufficiently large n will then be called the *index* of (7). The minimal m such that F_n can be eliminated from $\Sigma_{n,i}$ for all $n \geqslant m$ will be called the *offset*.

Example 4. Let $\Phi \in \mathbb{K}[z, F]^r$, $k = 1$ and assume that

$$J = \begin{pmatrix}\left(\frac{\partial \Phi^{[1]}}{\partial F^{[1]}}\right)_0 & \cdots & \left(\frac{\partial \Phi^{[1]}}{\partial F^{[r]}}\right)_0\\ \vdots & & \vdots\\ \left(\frac{\partial \Phi^{[r]}}{\partial F^{[1]}}\right)_0 & \cdots & \left(\frac{\partial \Phi^{[r]}}{\partial F^{[r]}}\right)_0\end{pmatrix}$$

is invertible. Then for all $n > 0$, we have

$$\Phi_n = J F_n + R_n,$$

with $R_n \in \mathbb{K}[F_0, \ldots, F_{n-1}]$. Hence, f_n can be computed from the previous coefficients using

$$f_n = -J^{-1} R_n(f_0, \ldots, f_{n-1}).$$

The system $\sigma_1(\Sigma_n)$ consists of the equation

$$J F_n + R_n(f_0, \ldots, f_{n-1}) \;=\; 0,$$

from which F_n can be eliminated. We conclude that (7) is quasi-linear, of index 1.

4. Relaxed resolution of implicit equations

Consider a quasi-linear system (7) of index i with unique solution $f \in \mathbb{K}[[z]]^r$. We want to solve the system in a relaxed way, by computing the systems $\Sigma_{n,i}$ for increasing values $n = k, k+1, \ldots$ and eliminating F_n from $\Sigma_{n,i}$ using linear algebra. For each subexpression Ψ of Φ, we need to evaluate $\check{\Psi} = \sigma_i(\Psi) \in \mathbb{P}[[z]]$ in a relaxed way. The main challenge is to take advantage of the fact that $\sigma_i(\Psi)_n$ is really a constant plus a linear combination of $F_{n-v_\Psi-i+1}$, ..., F_{n-v_Ψ} and to integrate the necessary substitutions of newly computed coefficients by their values into the relaxed resolution process.

Denote the inner product of vectors by $\cdot$. For each $v \in \mathbb{Z}$ and $i \in \mathbb{N}$, let $\mathbb{P}[[z]]_{v,i}$ be the subset of $\Psi \in \mathbb{P}[[z]]$ such that

$$\begin{aligned}
\Psi_{n+v} &= \Psi^*_{n+v} + \Psi^{(i-1)}_{n+v} \cdot F_{n-i+1} + \cdots + \Psi^{(0)}_{n+v} \cdot F_n, \\
\Psi^*_{n+v} &\in \mathbb{K} \\
\Psi^{(j)}_{n+v} &\in \mathbb{K}^r \qquad (j = 0, \ldots, i-1),
\end{aligned} \tag{10}$$

for all $n \geqslant -v$. Then $\mathbb{P}_{v,i}$ is a $\mathbb{K}$-vector space and

$$\begin{aligned}
\Psi \in \mathbb{P}[[z]]_{v,i} &\Longrightarrow z\,\Psi \in \mathbb{P}[[z]]_{v+1,i} \\
\Psi \in \mathbb{P}[[z]]_{v,i} &\Longrightarrow \Psi' \in \mathbb{P}[[z]]_{v-1,i} \\
\Psi \in \mathbb{P}[[z]]_{v,i} &\Longrightarrow \textstyle\int \Psi \in \mathbb{P}[[z]]_{v+1,i}.
\end{aligned}$$

Given $\Psi \in \mathbb{P}_{v,i}$ with $i > 0$, we define the *one-step substitution* $\tau(\Psi) \in \mathbb{P}[[z]]_{v+1,i-1}$ by

$$\tau(\Psi)_{n+v} \;=\; \Psi^*_{n+v} + \Psi^{(i-1)}_{n+v} \cdot f_{n-i+1} + \Psi^{(i-2)}_{n+v} \cdot F_{n-i+2} + \cdots + \Psi^{(0)}_{n+v} \cdot F_n.$$

In particular, $\tau(\sigma_i(\Psi)) = \sigma_{i-1}(\Psi)$ and the the iterate $\tau^i(\Psi)$ coincides with the full substitution $\sigma_0(\Psi) \in \mathbb{K}[[z]]$ of F by f in Ψ.

It remains to be shown how to multiply two series $\Psi \in \mathbb{P}[[z]]_{v,i}$ and $\Omega \in \mathbb{P}[[z]]_{w,i}$ in a suitable way, without introducing quadratic terms in F. Given $t \in \mathbb{N}$, it will be convenient to introduce shift operators

$$\begin{aligned}
\Psi_{\ll t} &= \Psi\, z^t \\
\Psi_{\gg t} &= \Psi_t + \Psi_{t+1}\, z^1 + \cdots.
\end{aligned}$$

We recursively define the *substitution product* $\Psi *_i \Omega \in \mathbb{P}[[z]]_{v+w,i}$ of Ψ and Ω by

$$\begin{aligned}
\Psi *_0 \Omega &= \Psi\,\Omega \\
\Psi *_i \Omega &= \sigma_0(\Psi_0)\,\sigma(\Omega_0) + [\sigma_0(\Psi_0)\,\Omega_{\gg 1} + \Psi_{\gg 1}\,\sigma_0(\Omega_0)]_{\ll 1} + \\
&\quad\; [\tau(\Psi_{\gg 1}) *_{i-1} \tau(\Omega_{\gg 1})]_{\ll 2}
\end{aligned} \tag{11}$$

using the fact that $\tau(\Psi_{\gg 1}), \tau(\Omega_{\gg 1}) \in \mathbb{P}[[z]]_{v+w,i-1}$. Unrolling (11), we have

$$\begin{aligned}
\Psi *_i \Omega \;=\; & \sum_{j<i} [\sigma_0(\Psi_j)\,\sigma_0(\Omega_j)]_{\ll 2j} + [\sigma_0(\Psi_j)\,\tau^j(\Omega_{\gg j+1}) + \tau^j(\Psi_{\gg j+1})\,\sigma_0(\Omega_j)]_{\ll 2j+1} + \\
& [\tau^i(\Psi_{\gg i})\,\tau^i(\Omega_{\gg i})]_{\ll 2i}.
\end{aligned} \tag{12}$$

The substitution product satisfies the important property

$$\sigma_i(\Psi) *_i \sigma_i(\Omega) \;=\; \sigma_i(\Psi\,\Omega).$$

Moreover, it respects the constraint that $\sigma_i(\Psi\ \Omega)_{n-v_{\Psi\Omega}}$ can be computed as soon as $\sigma_i(\Psi)_{j-v_\Psi}$ and $\sigma_i(\Omega)_{j-v_\Omega}$ are known for $j \leqslant n$. Recall that the computation of $\sigma_i(\Psi)_{n-v_\Psi}$ requires the previous computation of $f_0, \ldots, f_{n-i}$.

From the implementation point of view, we proceed as follows. We introduce a new data type $\mathbb{D}$, whose instances are of the form

$$\begin{aligned} c &= (f_c, n_c, i_c, c^*, c^{(0)}, \ldots, c^{(i_c-1)}) \\ n_c, i_c &\in \mathbb{N} \\ c^* &\in \mathbb{K} \\ c^{(j)} &\in \mathbb{K}^r \qquad (j = 0, \ldots, i-1), \end{aligned}$$

where $f_c = f$ stands for the relaxed power series solution of (7). Such an instance c represents

$$c \;\equiv\; c^* + c^{(i_c-1)} \cdot F_{n_c-i_c+1} + \cdots + c^{(0)} \cdot F_{n_c}.$$

Denoting by $\mathbb{D}_i$ the subtype of instances c in $\mathbb{D}$ with $i_c \leqslant i$, we may thus view series $\Psi \in \mathbb{P}[[z]]_{v,i}$ as elements of $\mathbb{D}_i[[z]]$. We have a natural inclusion $\mathbb{K} \to \mathbb{D}; a \mapsto (0, 0, 0, a)$, where we notice that f_c does not matter if $i_c = 0$, and a constructor $(f, n) \mapsto (f, n, 1, 0, 1)$ for the unknown F_n. The $\mathbb{K}$-vector space operations on $\mathbb{D}$ are implemented in a straightforward way. The one-step substitution operator τ is implemented by

$$\tau(c) \;=\; (f_c, n_c, i_c - 1, c^* + c^{(i_c-1)} \cdot f_{n_c-i_c+1}, c^{(0)}, \ldots, c^{(i_c-2)})$$

if $i_c > 0$ and $\tau(c) = c$ otherwise. On a fixed $\mathbb{D}_i$, this allows us to implement the substitution product $*_i$ using (11). Moreover, by casting $\tau^i(\Psi_{\gg i})$ and $\tau^i(\Omega_{\gg i})$ to relaxed series in $\mathsf{Series}_\mathbb{K}$, we may compute the product $\tau^i(\Psi_{\gg i})\ \tau^i(\Omega_{\gg i})$ using a fast relaxed product in $\mathsf{Series}_\mathbb{K}$. We are now in a position to state our relaxed algorithm for solving (7).

```
Class ImplicitSeries$_{\mathbb{K}^r} \trianglerighteq$ Series$_{\mathbb{K}^r}$
    Fields $\varphi$: Series$^r_{\mathbb{D}_i}$, $\Sigma$: Set$_{\mathbb{D}_i}$, $p$: $\mathbb{N}$
    Constructor implicit ($\Phi$: $\mathbb{E}^r$, $C_0$: $\mathbb{K}^r, \ldots, C_{k-1}$: $\mathbb{K}^r$)
        $F :=$ UnknownSeries$_{\mathbb{D}_i^r}$(this, $C_0, \ldots, C_{k-1}$)
        $\varphi := \Phi[F]$
        $\Sigma := \varnothing$ and $p := k$
    Method next ($n$: $\mathbb{N}$)
        While true
            If $p > n + i$ then raise an error
            $\Sigma := \Sigma \cup \{\varphi^{[j]}_{p-v_{\Phi^{[j]}}}: j \in \{1, \ldots, r\}, p \geqslant v_{\Phi^{[j]}}\}$ and $p := p + 1$
            Triangularize $\Sigma$ by eliminating $F_l$ with large $l$ first
            $\Sigma := \Sigma \setminus \{0\}$
            If $\Sigma \cap \mathbb{K} \neq \varnothing$ then raise an error
            If $\Sigma = \Sigma_1 \amalg \Sigma_2$ where card $\Sigma_2 = r$ and $\Sigma_2$ only involves $F^{[1]}_n, \ldots, F^{[r]}_n$ then
                Let $c \in \mathbb{K}^r$ be the unique solution to $\Sigma_2$ as a system in $F_n$
                Let $\Sigma := \Sigma_1$ and substitute $c$ for $F_n$ in $\Sigma$
                Return $c$
```

The following subalgorithm is used for the symbolic construction of the unknown series $F = C_0 + \cdots + C_{k-1} z^{k-1} + F_k z^k + F_{k+1} z^{k+1} + \cdots$:

```
Class UnknownSeries$_{\mathbb{D}_i^r} \trianglerighteq$ Series$_{\mathbb{D}_i^r}$
    Fields $f$: Series$_{\mathbb{K}^r}$, $C_0$: $\mathbb{K}^r, \ldots, C_{k-1}$: $\mathbb{K}^r$
    Constructor implicit ($\tilde{f}$: Series$_{\mathbb{K}^r}$, $\tilde{C}_0$: $\mathbb{K}^r, \ldots, \tilde{C}_{k-1}$: $\mathbb{K}^r$)
        $f := \tilde{f}$, $C_0 := \tilde{C}_0, \ldots, C_{k-1} := \tilde{C}_{k-1}$
```

```
Method next $(n:\mathbb{N})$
    If $n<k$ then return $C_n$
    Else return $(f,n,1,0,1)\equiv F_n$
```

These algorithms require a few comments. First of all, we denoted by $\Phi[F]$ the replacement of F by $C_0+\cdots+C_{k-1}\,z^{k-1}+F_k\,z^k+F_{k+1}\,z^{k+1}+\cdots$ in the expression Φ, modulo suitable implicit conversions $\mathsf{Series}_{\mathbb{D}_i^r}\leftrightarrow\mathsf{Series}_{\mathbb{D}_i}^r$. Throughout the algorithm, the set Σ stands for the current system of "not yet solved equations". Each equation is represented by an instance in $\mathbb{D}_i$ which only involves F_{n+j} with $0\leqslant j<i$. New equations are progressively added while keeping Σ in a triangular form. As soon as $F_n^{[1]}$, ..., $F_n^{[r]}$ can be eliminated from Σ, then the next coefficient f_n can be computed.

THEOREM 5. *Let* (7) *be an equation of index i and offset k. Then the above algorithm for the resolution of* (7) *is correct. If Φ involves s and is of size t as an expression, then $f_0,\ldots,f_{n-1}$ are computed in time*

$$\mathsf{T}(n)\;=\;s\,\mathsf{R}(n)+O((t+i\,r^2)\,i\,r\,n).$$

PROOF. By construction, just after setting $\Sigma:=\Sigma\cup\{\varphi_{p-v_{\Phi^{[j]}}}^{[j]}:j\in\{1,\ldots,r\},p\geqslant v_{\Phi^{[j]}}\}$, the system Σ is equivalent to $\Sigma_{n,p-n}$ from (9). If $n\geqslant k$ and $p\geqslant n+i$, we may thus eliminate F_n from the system Σ and compute f_n. This proves the correctness. As to the complexity bound, we observe that the substitution product in $\mathsf{Series}_{\mathbb{D}_i}$ amounts to $2\;i\;r\,+\,2$ scalar products and one relaxed product in $\mathsf{Series}_{\mathbb{K}}$. Similarly, each linear operation (addition, subtraction, derivation and integration) in $\mathsf{Series}_{\mathbb{D}_i}$ amounts to $i\;r+1$ similar operations in $\mathsf{Series}_{\mathbb{K}}$. If Σ is a triangular system of size $i\;r\,\times\,i\;r$ and we add r new rows, then the triangularization of the new system can be done in time $O(i^2\,r^3)$. □

REMARK 6. In practice, the user does not necessarily have any *a priori* bound for the index of (7). Instead of assuming a fixed index i for the substitution products, it is actually possible to automatically increase i whenever $\tau^i(\check{\Psi}_{\ll i})$ contains a non constant coefficient for some subexpression Ψ of Φ.

REMARK 7. A prototype of the algorithm has been implemented in the MATHEMAGIX system. For various systems of low index with rational coefficients, we have compared the time T_c to compute the solution up to order n with the time T_v to verify its correctness. For small orders $n\leqslant 100$, we observed ratios $T_c/T_v\approx 2$. For large orders $n\geqslant 1000$, we achieved $T_c\lessapprox\frac{4}{3}\,T_v$, in correspondence with the asymptotic complexity bound.

5. A WORKED EXAMPLE

Consider the system

$$\begin{cases}\Phi &=& F-G+z\,F\,G &=& 0\\ \Psi &=& z\,(F'-G')+z\,F\,G &=& 0\\ F_0 &=& 1\\ G_0 &=& 1\end{cases}$$

It is not hard to find the unique explicit solution $(f,g)\in\mathbb{K}[[z]]^2$ of this system. Indeed,

$$z\,\Phi'-\Psi\;=\;z^2\,(F\,G)',$$

whence $f\,g\in\mathbb{K}$. Since $f_0=g_0=1$, it follows that $f\,g=1$. Plugging this into the first equation $f-g+z\,f\,g=0$, we get $f^2-1+z=0$, whence $f=(1-z)^{1/2}$ and $g=(1-z)^{-1/2}$.

During the computations below, we will see that the system is quasi-linear of index 2. Denoting the relaxed solution by $(f, g) \in \mathbb{K}[[z]]^2$, we will have to compute f, g and the series $\check{F}$, $\check{G}$, $\widecheck{FG} \in \mathbb{D}_2[[z]]$.

Initialization. At the very start, we have $f_0 = g_0 = \check{F}_0 = \check{G}_0 = 1$.

Step 1. The evaluations of $\check{\Phi}_1$ and $\check{\Psi}_1$ yield

$$\begin{aligned}
\check{\Phi}_1 &= \check{F}_1 - \check{G}_1 + f_0\, g_0\\
&= 1 + F_1 - G_1\\
\check{\Psi}_1 &= \check{F}_1 - \check{G}_1 + f_0\, g_0\\
&= 1 + F_1 - G_1\\
\Sigma &:= \{F_1 - G_1 + 1\}
\end{aligned}$$

These relations do not yet enable us to determine f_1 and g_1.

Step 2. The evaluations of $\check{\Phi}_2$ and $\check{\Psi}_2$ yield

$$\begin{aligned}
\check{\Phi}_2 &= \check{F}_2 - \check{G}_2 + \check{F}_1\, g_0 + f_0\, \check{G}_1\\
&= F_1 + G_1 + F_2 - G_2\\
\check{\Psi}_2 &= 2\, \check{F}_2 - 2\, \check{G}_2 + \check{F}_1\, g_0 + f_0\, \check{G}_1\\
&= F_1 + G_1 + 2\, F_2 - 2\, G_2\\
\Sigma &:= \{F_2 - G_2 + F_1 + G_1, 2\, F_2 - 2\, G_2 + F_1 + G_1, F_1 - G_1 + 1\}
\end{aligned}$$

After triangularization, we get

$$\Sigma := \{F_2 - G_2 + F_1 + G_1, F_1 + G_1, -2\, G_1 + 1\}.$$

The two last equations imply $f_1 = -\frac{1}{2}$ and $g_1 = \frac{1}{2}$.

Step 3. Evaluations of $\check{\Phi}_3$ and $\check{\Psi}_3$ and triangularization of Σ yield

$$\begin{aligned}
\check{\Phi}_3 &= \check{F}_3 - \check{G}_3 + \check{F}_2\, g_0 + f_1\, g_1 + f_0\, \check{G}_2\\
&= -\tfrac{1}{4} + F_2 + G_2 + F_3 - G_3\\
\check{\Psi}_3 &= -\tfrac{1}{4} + F_2 + G_2 + 3\, F_3 - 3\, G_3\\
\Sigma &:= \{F_3 - G_3 + F_2 + G_2 - \tfrac{1}{4}, 2\, F_2 + 2\, G_2 - \tfrac{1}{2}, 4\, G_2 - \tfrac{1}{2}\}
\end{aligned}$$

From the equations $3\, \check{\Phi}_3 - \check{\Psi}_3 = 0$ and $\check{\Psi}_2 - \check{\Phi}_2 = 0$, we get $f_2 = g_2 = \frac{1}{8}$.

Further steps. For $n \geqslant 4$, the evaluations of $\check{\Phi}_n$ and $\check{\Psi}_n$ yield

$$\begin{aligned}
\check{\Phi}_n &= \check{F}_n - \check{G}_n + \check{F}_{n-1}\, g_0 + (f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2}) + f_0\, \check{G}_{n-1}\\
&= (f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2}) + F_{n-1} + G_{n-1} + F_n - G_n\\
\check{\Psi}_n &= (f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2}) + F_{n-1} + G_{n-1} + n\, F_n - n\, G_n\\
n\, \check{\Phi}_n - \check{\Psi}_n &= (n-1)\, F_{n-1} + (n-1)\, G_{n-1} + (f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2})\\
\check{\Psi}_n - \check{\Phi}_n &= (n-1)\, F_n - (n-1)\, G_n
\end{aligned}$$

After triangularization, we thus get

$$\begin{aligned}
\Sigma := \{&F_n - G_n + F_{n-1} + G_{n-1} + f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2},\\
&(n-1)\, F_{n-1} + (n-1)\, G_{n-1} + f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2}\\
&2\, (n-1) G_{n-1} + f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2}\}.
\end{aligned}$$

Consequently, $f_{n-1} = g_{n-1} = -\frac{1}{2\, (n-1)}\, (f_{n-2}\, g_1 + \cdots + f_1\, g_{n-2})$.

6. Symbolic linearization

Assume that the system (7) is quasi-linear. Given a subexpression Ψ of Φ an integer $i \in \mathbb{N}$ and $j \in \{1, \ldots, r\}$, we claim that the coefficient $[F_{n-v_\Psi-i}^{[j]}] \sigma_{i+1}(\Psi)_n$ of $F_{n-v_\Psi-i}^{[j]}$ in $\sigma_{i+1}(\Psi)_n$ (and which corresponds to $[\Psi_n^{(i)}]^{[j]}$ in (10)) is a rational function in n, for sufficiently large n. There are two ways to see this.

Let $\mathbb{E}^{\mathrm{rat}}$ denote the set of expressions Ψ, such that for all $i \in \mathbb{N}$ there exist vectors of rational functions $\Psi^{(0)}, \ldots, \Psi^{(i-1)} \in \mathbb{K}(N)^r$ and a sequence Ψ_n^* with

$$\sigma_i(\Psi)_n = \Psi_n^* + \Psi^{(i-1)}(n) \cdot F_{n-v_\Psi-i+1} + \cdots + \Psi^{(0)}(n) \cdot F_{n-v_\Psi}, \tag{13}$$

for all sufficiently large n. In other words,

$$[\Psi^{(i)}(n)]^{[j]} = [F_{n-v_\Psi-i}^{[j]}] \sigma_{i+1}(\Psi)_n,$$

for $j = 1, \ldots, r$ and sufficiently large n. We define $\Psi^{(i)} = 0$ if $i < 0$. We clearly have $\mathbb{K}[z] \subseteq \mathbb{E}^{\mathrm{rat}}$ and $F^{[1]}, \ldots, F^{[r]} \in \mathbb{E}^{\mathrm{rat}}$. Assume that $\Psi, \Omega \in \mathbb{E}^{\mathrm{rat}}$. Then $\Psi + \Omega, \Psi - \Omega, \Psi\,\Omega, \Psi', \int \Psi \in \mathbb{E}^{\mathrm{rat}}$ and we may explicitly compute the corresponding rational functions using

$$\begin{aligned}
(\Psi + \Omega)^{(i)} &= \Psi^{(i + v_{\Psi+\Omega} - v_\Psi)} + \Omega^{(i + v_{\Psi+\Omega} - v_\Psi)} \\
(\Psi - \Omega)^{(i)} &= \Psi^{(i + v_{\Psi+\Omega} - v_\Psi)} - \Omega^{(i + v_{\Psi+\Omega} - v_\Psi)} \\
(\Psi\,\Omega)^{(i)}(n) &= \Psi_{v_\Psi}\,\Omega^{(i)}(n - v_\Psi) + \cdots + \Psi_{v_\Psi+i}\,\Omega^{(0)}(n - v_\Psi - i) + \\
&\quad\ \Psi^{(i)}(n - v_\Omega)\,\Omega_{v_\Omega} + \cdots + \Psi^{(0)}(n - v_\Omega - i)\,\Omega_{v_\Omega+i} \\
(\Psi')^{(i)}(n) &= (n+1)\,\Psi^{(i)}(n+1) \\
\left(\int \Psi\right)^{(i)}(n) &= \frac{1}{n}\,\Psi^{(i)}(n-1).
\end{aligned}$$

If Ψ is a polynomial, then we notice that $(\Psi^{(i)})^{[j]} \in \mathbb{K}^r$ for all i and j. If Ψ is a differential polynomial of order q, then $(\Psi^{(i)})^{[j]}$ is a polynomial in $\mathbb{K}[N]$ of degree $\leqslant q$. In general, the degrees of the numerator and denominator of $(\Psi^{(i)})^{[j]}$ are bounded by the maximal number of nested differentiations resp. integrations occurring in Ψ.

An alternative way to determine the $\Psi^{(j)}$ is to consider $F = f - E$ as a perturbation of the solution and perform a Taylor series expansion

$$\Psi(f + E) = \Psi(f) + (D\,\Psi)(f)(E) + \frac{1}{2}\,(D^2\,\Psi)(f)(E, E) + \cdots.$$

The coefficients $\Psi^{(i)}$ can then be read off from the linear term using

$$\begin{aligned}
\Psi^{(i)}(n)^{[j]} &= [F_{n-v_\Psi-i}^{[j]}]\,(D\,\Psi)(f)(F - f) \\
&= [F_{n-v_\Psi-i}^{[j]}]\,(D\,\Psi)(f)(F).
\end{aligned} \tag{14}$$

For instance, consider the expression

$$\begin{aligned}
\Psi &= \int F\,G' + z^2\,F'' \\
(D\,\Psi)\begin{pmatrix} f \\ g \end{pmatrix} &= \int f\,G' + \int F\,g' + z^2\,F''.
\end{aligned}$$

Then we have

$$\begin{aligned}
\Psi^{(i)}(n) &= \begin{pmatrix} [F_{n-i}]\,(\int f\,G' + \int F\,g' + z^2\,F'')_n \\ [G_{n-i}]\,(\int f\,G' + \int F\,g' + z^2\,F'')_n \end{pmatrix} \\
&= \begin{pmatrix} \frac{i}{n}\,g_i + \delta_{0,i}\,n^2 \\ \frac{n-i}{n}\,f_i \end{pmatrix},
\end{aligned}$$

with $\delta_{0,i}=1$ if $i=0$ and $\delta_{0,i}=0$ otherwise.

A first theoretical consequence of our ability to compute symbolic expressions for $\Psi^{(i)}$ is the following:

Theorem 8. *There exists an algorithm which takes a quasi-linear system* (7) *on input and computes its index and its offset.*

Proof. The system (9) can be rewritten as a matrix-vector equation

$$M_n X_n = Y_n. \tag{15}$$

Here $X_n \in \mathbb{K}^{ir}$ is a column vector with entries $F_{n+i-1}^{[1]}, \ldots, F_{n+i-1}^{[r]}, \ldots, F_n^{[1]}, \ldots, F_n^{[r]}$ and $Y_n \in \mathbb{K}^{ir}$. The entries of the matrix $M_n \in \mathbb{K}^{ir \times ir}$ are coefficients of the form $(\Phi_{n-p}^{(i')})^{[j]}$. In particular, we can compute a matrix $M \in \mathbb{K}(N)^{ir \times ir}$ such that the matrix M_n is given by the specialization $M(n)$ of M at $N=n$ for sufficiently large n.

Let $T \in \mathbb{K}(N)^{ir \times ir}$ be the symbolic triangularization of M. For sufficiently large n, the triangularization T_n of M_n coincides with $T(n)$ for $n \geqslant n_0$. Now F_n may be eliminated from the equation (15) if and only if the last r non zero rows and the last r columns of T_n are an invertible triangular matrix. This is the case for all sufficiently large n if and only if the last r non zero rows and the last r columns of T are an invertible triangular matrix in $\mathbb{K}(N)^{r \times r}$. We compute the index of (7) as being the smallest i for which this is the case.

As to the offset, we first observe that we may explicitly compute an $n_0 \in \mathbb{N}$ such that $M_n = M(n)$ and $T_n = T(n)$ for all $n \geqslant n_0$, since the values n for which these equations do not hold are roots of a polynomial with coefficients in $\mathbb{K}$. Using the algorithm from section 4, we may compute the solution f up to any given order n. We thus compute the offset as being the smallest $k \in \mathbb{N}$ such that F_n can be eliminated from (15) for all $k \leqslant n < n_0$. □

Example 9. For the example from section 5, the equation (15) becomes

$$\begin{pmatrix} 1 & -1 & 1 & 1 \\ n & -n & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & n-1 & -n+1 \end{pmatrix} \begin{pmatrix} F_n \\ G_n \\ F_{n-1} \\ G_{n-1} \end{pmatrix} = \begin{pmatrix} -(f_{n-2}\,g_1 + \cdots + f_1\,g_{n-2}) \\ -(f_{n-2}\,g_1 + \cdots + f_1\,g_{n-2}) \\ 0 \\ 0 \end{pmatrix}.$$

Remark 10. An interesting theoretical question is how to test whether a given system is quasi-linear. The proof technique of the theorem at least gives us a partial answer: *if* a system is quasi-linear, then we will be able to provide a certificate for this fact. Most systems encountered in practice are indeed quasi-linear, provided that we have sufficiently many initial conditions. In the contrary case, it is usually possible to find a simpler equivalent quasi-linear system. It would be nice to prove a general theorem in this direction.

A second consequence of our ability to compute symbolic expressions for $\Psi^{(i)}$ is that we can avoid the systematic computation of the coefficients $\Psi_n^{(i)}$ using arithmetic in $\mathbb{D}$: we rather compute $\Psi_n^{(i)}$ on demand, by evaluating $\Psi^{(i)}$ at $N=n$. The coefficients $\Psi_n^{(i)}$ are essentially needed at two places: for the computation of substitution products and for the computation of the system Σ. Let q be the cost of an evaluation of $\Psi^{(i)}$: if Ψ is a polynomial, then $q=1$; if Ψ is a differential polynomial, then q is its order plus one; etc..

When computing $\Psi_n^{(i)}$ by evaluating $\Psi^{(i)}$ at $N=n$, the computation of one coefficient of a one-step substitution τ amounts to r evaluations of rational functions of the form $(\Psi^{(i)})^{[j]}$. Consequently, every substitution product amounts to a cost $\mathsf{R}(n) + O(i\,r\,q\,n)$ in the final complexity.

As to the computation of a coefficients F_n, we may compute M_n as in (15) using $(i\ r)^2$ evaluations of cost q and then solving a linear system of size $i\ r$. This gives rise to a cost $O((i\ r\ +\ q)\ (i\ r)^2\ n)$ in the final complexity. Alternatively, as a side effect of the triangularization of M with the notations from (15), we may compute a symbolic matrix $S \in \mathbb{K}(N)^{r \times ir}$ such that $F_n = S(n)\, Y_n$ for all sufficiently large n. If the system (7) is algebraic, then $S \in \mathbb{K}^{ir \times ir}$ actually has constant coefficients, so the complexity further reduces to $O(i\, r^2)$. In general, the evaluation of S will be more expensive, so it is not clear whether this strategy pays off. Altogether, we have proved:

Theorem 11. *Let* (7) *be an equation of index i of total size t and containing at most s multiplications. Assume that the equations involve strictly less than q nested derivations or integrations. Then $f_0, \ldots, f_{n-1}$ can be computed in time*

$$\mathsf{T}(n) \;=\; s\,\mathsf{R}(n) + O((s\,q + i\,r\,q + i^2\,r^2)\,i\,r\,n + t\,n).$$

If (7) *is algebraic, then the complexity further drops down to*

$$\mathsf{T}(n) \;=\; s\,\mathsf{R}(n) + O((s + r)\,i\,r\,n + t\,n).$$

Remark 12. The algorithm from this section has not been implemented yet. The new complexity bound constitutes an improvement mainly in the algebraic case. Since the manipulation of non constant coefficients in $\mathbb{D}$ gives rise to a small but non negligible amount of overhead for small and moderate n (see remark 7), we indeed expect further gains in this case.

7. Generalizations

For simplicity, the presentation of this paper has been focused on ordinary differential equations. Nevertheless, the techniques admit generalizations in several directions. We will outline two such generalizations.

Functional equations. Let $\mathbb{G}$ be a set of relaxed power series $g \in \mathbb{K}[[z]]$ with $g_0 = 0$ and replace $\mathbb{E}$ by the set of expressions built up from F, z and constants in $\mathbb{K}$ using ring operations, differentiation, integration and right composition with series in $\mathbb{G}$.

Assume first that $\operatorname{val} g = 1$ for all $g \in \mathbb{G}$. In a similar way as in section 6, there exists a symbolic expression of the form (13) for each $\Psi \in \mathbb{E}$, except that we now have

$$\Psi^{(i)} \;\in\; \mathbb{K}(N)[(g_1)_1^N, \ldots, (g_l)_1^N], \tag{16}$$

where $g_1, \ldots, g_l \in \mathbb{G}$ are the functions which occur as postcomposers in Ψ. In particular, if $(g_1)_1, \ldots, (g_l)_1 \in \mathbb{R}$, then the $\Psi^{(i)}$ are contained in a Hardy field, and theorem 8 generalizes.

The above observation further generalizes to the case when $\operatorname{val} g > 1$ for certain $g \in \mathbb{G}$. In non degenerate cases, expressions $F^{(i)} \circ g$ with $\operatorname{val} g > 1$ only occur as perturbations, and (16) still holds. In general, we also have to consider degenerate situations, such as the case when

$$\Psi^{(i)}(\alpha\, N + \beta) \;\in\; \mathbb{K}(N)[(g_1)_1^N, \ldots, (g_l)_1^N]$$

for a certain $\alpha > 1$ and all $\beta = 0, \ldots, \alpha - 1$.

One may even consider functional equations in which we also allow postcompositions with general expressions $g \in \mathbb{E}$ with $g_0 = 0$. Although the theory from section 6 becomes more and more intricate, the algorithm from section 4 generalizes in a straightforward way.

Partial differential equations. We may also consider power series in several variables $z_1, \ldots, z_d$. Given a multivariate power series $f \in \mathbb{K}[[z]] = \mathbb{K}[[z_1, \ldots, z_d]]$ and $n \in \mathbb{N}^d$, we denote by f_n the coefficient of $z^n = z_1^{n_1} \cdots z_d^{n_d}$ in f, and let $\mathbb{P} = \mathbb{K}[F_n: n \in \mathbb{N}^d]$. The expressions in $\mathbb{E}$ are built up from F, $z_1, \ldots, z_d$ and constants in $\mathbb{K}$ using ring operations and partial differentiation or integration with respect to the $z_1, \ldots, z_d$. The number v_Φ now becomes a vector in $(\mathbb{Z} \cup \{\infty\})^d$.

The simplest, blockwise generalization proceeds as follows. Indices $i, j \in \mathbb{N}^d$ are compared using the product ordering $i \leqslant j \Leftrightarrow i_1 \leqslant j_1 \wedge \cdots \wedge i_d \leqslant j_d$. Given $i \in \mathbb{N}^d$ and $\Psi \in \mathbb{E}$, we let $\sigma_i(\Psi)$ be the series in $\mathbb{P}[[z]]$ such that $\sigma_i(\Psi)_n$ is the result of the substitution of F_j by f_j in Ψ_n, for all $n \in \mathbb{N}^d$ and $j \leqslant n + v_\Phi - i$. Given $v \in \mathbb{Z}^d$ and $i \in \mathbb{N}^d$, we let $\mathbb{P}[[z]]_{v,i}$ be the subset of $\Psi \in \mathbb{P}[[z]]$ such that

$$\begin{aligned}
\Psi_{n+v} &= \Psi^*_{n+v} + \sum_{0 \leqslant j \leqslant i-1} \Psi^{(j)}_{n+v} \cdot F_{n-j}, \qquad (17)\\
\Psi^*_{n+v} &\in \mathbb{K}\\
\Psi^{(j)}_{n+v} &\in \mathbb{K}^r \qquad (0 \leqslant j \leqslant i-1).
\end{aligned}$$

For each $l \in \{1, \ldots, d\}$, let $\mathrm{e}_l \in \mathbb{N}^d$ be such that $(\mathrm{e}_l)_l = 1$ and $(\mathrm{e}_l)_j = 0$ for $j \neq l$. We define the one-step partial substitution $\tau_l(\Psi) \in \mathbb{P}[[z]]_{v+\mathrm{e}_l, i-\mathrm{e}_l}$ by

$$\tau(\Psi)_{n+v} = \Psi^*_{n+v} + \sum_{(i_l-1)\mathrm{e}_l \leqslant j \leqslant i-1} \Psi^{(j)}_{n+v} \cdot f_{n-j} + \sum_{0 \leqslant j \leqslant i-1-\mathrm{e}_l} \Psi^{(j)}_{n+v} \cdot F_{n-j}.$$

The partial shifts $\Psi_{z_l \ll j}$ and $\Psi_{z_l \gg j}$ are defined similarly and we denote by $\Psi_{z_l=0}$ the substitution of 0 for z_l in Ψ. The substitution product is defined recursively. If $i = 0$, then we set $\Psi *_i \Omega = \Psi\, \Omega$. Otherwise, we let $l \in \{1, \ldots, r\}$ be smallest such that i_l is maximal and take

$$\begin{aligned}
\Psi *_i \Omega = {} & \sigma_0(\Psi_{z_l=0})\, \sigma(\Omega_{z_l=0}) + [\sigma_0(\Psi_{z_l=0})\, \Omega_{z_l \gg 1} + \Psi_{z_l^0 \gg 1}\, \sigma_0(\Omega_{z_l=0})]_{z_l \ll 1} + \\
& [\tau_l(\Psi_{z_l \gg 1}) *_{i-\mathrm{e}_l} \tau_l(\Omega_{z^l \gg 1})]_{z_l \ll 2}.
\end{aligned}$$

Using this substitution product, the algorithm from section 4 generalizes. The theory from section 6 can also be adapted. However, theorem 8 admits no simple analogue, due to the fact that there is no algorithm for determining the integer roots of a system of multivariate polynomials.

Several variants are possible depending on the application. For instance, it is sometimes possible to consider only the $\Psi^{(j)}_{n+v}$ up till a certain total degree $j_1 + \cdots + j_d \leqslant i$ in (17), instead of a block of coefficients. For some applications, it may also be interesting to store the $\Psi^{(j)}_{n+v}$ in a sparse vector.

Bibliography

[BCO+06] A. Bostan, F. Chyzak, F. Ollivier, B. Salvy, É. Schost, and A. Sedoglavic. Fast computation of power series solutions of systems of differential equation. preprint, april 2006. submitted, 13 pages.

[BK78] R.P. Brent and H.T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

[CK91] D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

[CT65] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[vdH97] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Küchlin, editor, *Proc. ISSAC '97*, pages 17–20, Maui, Hawaii, July 1997.

[vdH02a] J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.

[vdH+02b] J. van der Hoeven et al. Mathemagix, 2002. `http://www.mathemagix.org`.

[vdH06] J. van der Hoeven. Newton's method and FFT trading. Technical Report 2006-17, Univ. Paris-Sud, 2006. Submitted to JSC.

[vdH07] J. van der Hoeven. New algorithms for relaxed multiplication. *JSC*, 42(8):792–802, 2007.