UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1)39 63 55 11

Rapports Techniques

N°143

Programme 2

*Calcul symbolique, Programmation
et Génie logiciel*

GFUN: A MAPLE PACKAGE FOR THE MANIPULATION OF GENERATING AND HOLONOMIC FUNCTIONS IN ONE VARIABLE

Bruno SALVY
Paul ZIMMERMANN

Octobre 1992

GFUN: A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable

Bruno Salvy *Paul Zimmermann*

Abstract

We describe the GFUN package which contains functions for manipulating sequences, linear recurrences or differential equations and generating functions of various types. This document is intended both as an elementary introduction to the subject and as a reference manual for the package.

GFUN: Un package Maple pour la manipulation de fonctions génératrices et holonomes en une variable

Résumé

Nous décrivons le package GFUN qui contient des fonctions permettant de manipuler des suites, des récurrences ou des équations différentielles linéaires ainsi que des fonctions génératrices de types variés. Ce document est conçu à la fois comme une introduction élémentaire au domaine et comme un manuel de référence pour le package.

GFUN: A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable

Bruno Salvy *Paul Zimmermann*
Bruno.Salvy@inria.fr Paul.Zimmermann@inria.fr

Algorithms Project,
INRIA,
78153 Le Chesnay Cedex,
France.

Abstract

We describe the GFUN package which contains functions for manipulating sequences, linear recurrences or differential equations and generating functions of various types. This document is intended both as an elementary introduction to the subject and as a reference manual for the package.

Introduction

Generating functions are a very important tool for the study of sequences. Given a sequence $\{f_n\}$, one classically defines (among others) two generating functions $f(z)$ and $\hat{f}(z)$ by the following formulæ:

$$f(z) = \sum_{n\geq 0} f_n z^n, \quad \hat{f}(z) = \sum_{n\geq 0} f_n \frac{z^n}{n!}.$$

The first one is called *the ordinary generating function* of the sequence $\{f_n\}$, while the second one is its *exponential generating function*. Thus the sequence consisting of ones has $\exp(z)$ as its exponential generating function (e.g.f.) and $1/(1-z)$ as its ordinary generating function (o.g.f.). When viewed as an e.g.f, the function $1/(1-z)$ corresponds to $\{n!\}$. Other elementary generating functions include $1/(1-z)^2$ for $\{1,2,3,\ldots\}$, or $1/(1-z-z^2)$ for the Fibonacci sequence. Nice examples can be found in many references, see for instance [7].

Many properties of a sequence are reflected by analytical properties of its generating function. In particular, asymptotic properties of a sequence are usually more directly computable from its generating function. Besides, the generating function is often more compact than the "closed form" of the sequence, and generating functions are available even when no such closed form exists. Another nice feature of these functions is that there is an obvious correspondence (morphism) between operations performed on the sequences and operations performed on the generating functions. An elementary introduction to this can be found in [11, 21].

The package GFUN has been designed as an help for the manipulation and discovery of generating functions. Given the first terms of the sequence, the package contains functions that will help conjecture what the generating function is. In some cases, this answer will be "explicit". In most cases though, such an explicit expression will not exist, and the answer will be an equation (either differential or algebraic) satisfied by the generating function. The GFUN package also provides tools to compute with generating functions defined by equations. For instance, given two generating functions defined by linear differential equations with polynomial coefficients, there is a procedure to compute the differential equation satisfied by their product.

In the following sections we shall review the functions of the package, giving an idea of what they may be used for, and a short description of the algorithm being used. In Section 1 we review the procedures that can be used to discover a generating function. Section 2 describes the procedures related to the manipulation of linear differential or recurrence equations as well as algebraic equations. Finally Section 3 contains an overview of the package from the MAPLE point of view.

1 Guessing the generating function

From the first terms of a sequence, it is very tempting to try guessing the next ones. Many so-called intelligence tests include such an exercise. Of course, from the mathematical point of view no unique solution exists to such a question, unless extra constraints are added. For instance, Bergeron and Plouffe [1] have studied the problem of trying to get a rational generating function, or a function whose logarithmic derivative is rational, and several variations of these. The idea is that although nothing general can be found, it is very useful to find empirically some generating function which can be the basis of a conjecture. In most cases, the generating function which is found by GFUN is the right one. For instance, among the approximately 4500 sequences contained in the next edition of Sloane's book [16], about 22% of the generating functions can be found automatically without error.

From the algorithmic point of view, the procedures in this part of the GFUN package are very simple: they form the specified generating series from the given first terms of the sequence, they form a candidate equation for it and apply an indeterminate coefficient method, which reduces to Gaussian elimination.

Several types of generating functions are of interest. Apart from the ordinary and exponential generating functions that have already been mentioned, the package also considers their logarithmic derivatives (*lgdogf* and *lgdegf*) as well as their compositional inverses (*revogf* and *revegf*). For instance, the procedure *listtoratpoly*, which looks for a rational generating function, takes as its last argument a list of types of generating functions to consider. Besides the predefined types (*ogf, egf, lgdogf, lgdegf, revogf, revegf*), new types can be created by the user and then be used like the predefined ones. For more information about this, see `?listtolist` once the package has been loaded under MAPLE.

listtoseries Given a list of terms (numbers or polynomials or any algebraic expressions), the simplest operation is to build the series one wishes to study. This is performed by the procedure *listtoseries*. For instance,

```
> S1:=listtoseries([1,2*t,3*t^2,4*t^3,5*t^4],z,revogf);
                          2      2  3       3  4       4  5      6
            S1 := z - 2 t z  + 5 t  z  - 14 t  z  + 42 t  z  + O(z )
```

constructs the compositional inverse of the following formal power series

```
> listtoseries([0,1,2*t,3*t^2,4*t^3,5*t^4],z,ogf);
                      2      2  3      3  4      4  5      6
            z + 2 t z  + 3 t  z  + 4 t  z  + 5 t  z  + O(z )
```

The convention with the *revogf* and *revegf* options is that the list starts with the index 1, so that the series is invertible.

seriestolist is the reciprocal of the operation above: its input is a series and a type of generating function, it performs the same conversion as above but returns the output as a list.

While these two operations are useful from the computational point of view, they do not perform any mathematical operation. Thanks to them however, the procedures described from now on come in twins, one of them acting on a list and the other one on a series.

listtoratpoly and seriestoratpoly These functions are an interface to Maple's *convert/ratpoly* function which computes Padé approximants. Given the first terms of a list (or the initial part of a series) they compute a rational generating function. This function is output as a result when the sum of the degrees of its numerator and denominator is less than the number of terms given in the input. Nothing is output otherwise. This heuristically insures that the generating function which is found is not an accident.

Example: This is the Fibonacci sequence:

```
> listtoratpoly([1,1,2,3,5,8,13],z);
                               1
                      [- ------------, ogf]
                                    2
                         - 1 + z + z
```

Example: The "problème des rencontres" (see [7, pp. 180–183]) consists in studying the number of permutations of size n with no fix points. From the table [7, p. 182], we get the 13 first values, and our program does the rest:

```
> listtoratpoly([1,0,1,2,9,44,265,1854,14833,133496,1334961,14684570,176214841],
> t,[ogf,egf,lgdogf,lgdegf]);
                              t
                        [- -------, lgdegf]
                           - 1 + t

> expand(exp(int(op(1,"),t=0..z)));
                                1
                         --------------
                         exp(z) (1 - z)
```

This is precisely the generating function in [7, p. 182, Theorem B]. The problem of enumerating clouds [7, pp. 273–277] falls into the same class, with a less trivial rational function.

listtodiffeq and seriestodiffeq The rational generating functions treated by the procedures above can be viewed as solutions of linear differential equations of order 0 with polynomial coefficients (the numerator and denominator). We now turn to the more general problem of finding a linear differential equation with polynomial coefficients satisfied by the generating function. Several parameters can be used to modify the order of the equation and the degree of the coefficients one is looking for. By default, these procedures look for an equation of order less than 3 with coefficients of degree less or equal to 3.

Example: Numerators of convergents to e. It is known since Euler that the continued fraction expansion of e is very simple, the elements being 2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8,...By reducing the fractions obtained by truncating this expansion, one gets convergents to e. Doing this only at elements of index $3k+1$ yields

$$3, \frac{19}{7}, \frac{193}{71}, \frac{2721}{1001}, \frac{49171}{18089}, \frac{1084483}{398959}, \frac{28245729}{10391023}, \frac{848456353}{312129649}, \frac{28875761731}{10622799089}, \ldots$$

We input the sequence of numerators of this to *listtodiffeq*:

```
> listtodiffeq([3,19,193,2721,49171,1084483,28245729,848456353,28875761731],y(z));
[
                                                                    /   2       \
                                          /  d        \             |  d        |
{D(y)(0) = 19, y(0) = 3, 1/4 y(z) + 5/2 |---- y(z)| + (- 1/4 + z) |----- y(z)|}
                                          \ dz        /             |   2       |
                                                                    \ dz        /

    , egf]
```

In Maple V, this equation cannot be solved directly with *dsolve*, but with a little help, Maple finds:

$$y(z) = \frac{\sqrt{e^{1-\sqrt{1-4z}}}}{1-4z}\left(1+\frac{2}{\sqrt{1-4z}}\right). \tag{1}$$

As in the case of rational functions, extra care is taken so that the equation is over-constrained by the given terms of the list or series. In other words, the last few terms are not taken into account for the computation of the differential equation and are used at the end to check that the equation is "really" satisfied by the series.

listtorec and seriestorec It is well known that power series solutions of a linear differential equation with polynomial coefficients (the class of differential equations found by *listtodiffeq* and *seriestodiffeq*) are precisely those power series whose coefficients satisfy a linear recurrence with polynomial coefficients. The procedures *listtorec* and *seriestorec* search for this recurrence directly.

Example: We take the same sequence as above

```
> listtorec([3,19,193,2721,49171,1084483,28245729,848456353,28875761731],u(n));
     [{- u(n) + (- 6 - 4 n) u(n + 1) + u(n + 2), u(2) = 19, u(1) = 3}, ogf]
```

listtoalgeq and seriestoalgeq Algebraic generating functions are a very important special case of the solutions of linear differential equations. While procedures exist to find an algebraic solution of a linear differential equation [19], these have never been incorporated in computer algebra systems for an order larger than 2. It is thus very important to have a tool that will look for the minimal polynomial directly.

Example: Convex polyominoes of perimeter $2n$. It has been shown by M. Delest and X. Viennot [8] that the number of such polyominoes has an algebraic generating function of degree 2 (with large degree coefficients). To find out the minimal polynomial without their article, one can enumerate the number of such polyominoes for $n = 0, \ldots, 34$ (with a program) and use GFUN:

```
> l:=[0,0,1,2,7,28,120,528,2344,10416,46160,203680,894312,3907056,16986352,
73512288,316786960,1359763168,5815457184,24788842304,105340982248,446389242480,
1886695382192,7955156287456,33468262290096,140516110684832,588832418973280,
2463133441338048,10286493304041104,42892130604098656,178592047539343200,
742609229473744320,3083957343567791392,12792021060576424896,53000868925259947840]:
> gfun['maxdegcoeff']:=20: gfun['maxdegeqn']:=2:
> listtoalgeq(l,y(x));
       4        5        6         7         8        9        10
     [x  - 12 x  + 58 x  - 140 x  + 153 x  - 24 x  + 16 x

                   2        3         4         5         6         7
          + (- 2 x  + 28 x  - 150 x  + 376 x  - 416 x  + 128 x ) y(x)
```

```
                         2         3         4      2
     + (1 - 16 x + 96 x  - 256 x  + 256 x ) y(x) ,              ogf]
```

This type of method is used in real life by physicists, see e.g., [12] and by combinatorists [23]. Apart from their intrinsic interest, these formulæ enable one to compute as many terms of the sequences as desired (see below), or to derive easily asymptotic expansions for the nth term.

listtohypergeom and seriestohypergeom Hypergeometric sequences are those sequences such that the ratio of two successive terms is a fixed rational function of the index. Of special interest are the ${}_2F_1$ hypergeometric functions, special cases of which are the exponential function, $\log(1+x)$, $(1+x)^a$. The procedures *listtohypergeom* and *seriestohypergeom* look for a ${}_2F_1$ by indeterminate coefficients, and then call Maple's *hypergeom* procedure which sometimes simplifies the result.

Example: The sequence $\binom{2n+12}{n}$.

```
> listtohypergeom([1,14,120,816,4845,26334],z);
                                 4096
                  [---------------------------------, ogf]
                           1/2                 1/2 12
                    (1 - 4 z)     (1 + (1 - 4 z)    )
```

guessgf This function is the top-level routine for the cases when one has no intuition about the type of generating function to look for. It will first look for a rational function then a ${}_2F_1$, then an algebraic generating function and then a linear differential equation. Then it tries to solve the equation and returns a result only when Maple can find a solution. Since it tries all the possible types of generating functions, it is very time consuming and should be used only when one does not have any hint as to what the generating function can be. We give two examples which also demonstrate that the procedures above can handle sequences of polynomials.

Example: Hermite polynomials.

```
> alias(H=orthopoly[H]):
> l:=[seq(H(n,t),n=0..10)]:
> guessgf(l,z);
                      [y(z) = exp(z (- z + 2 t)), egf]
```

Example: Jacobi polynomials.

```
> l:=[seq(orthopoly[P](n,t),n=0..10)]:
> guessgf(l,z);
                                      1
                    [y(z) = ---------------------, ogf]
                                             2 1/2
                            ( 1 - 2 t z + z )
```

2 Manipulation of holonomic functions

A function of one variable is said to be *holonomic* (or D-finite) when it satisfies an ordinary linear differential equation with polynomial coefficients (we will say here a *holonomic* equation). Similarly, a sequence is called holonomic (or P-recursive) when it satisfies a linear recurrence with polynomial coefficients (a *holonomic* recurrence).

It can be shown that the generating function (either ordinary or exponential) of a holonomic sequence is holonomic and that reciprocally the sequence of Taylor coefficients of a holonomic function is holonomic [17]. This correspondence is implemented in the procedures *diffeqtorec* and *rectodiffeq*:

diffeqtorec This procedure translates a holonomic equation $c_k(z)y^{(k)}(z) + \cdots + c_1(z)y'(z) + c_0(z)y(z) + b(z) = 0$ for the function $y(z)$ into a holonomic recurrence for the coefficients u_n of its Taylor series. The algorithm consists in translating each $z^k y^{(j)}(z)$ into $\frac{(n-k+j)!}{(n-k)!}u_{n+j-k}$, the inhomogeneous terms yielding the initial conditions of the recurrence.

Example: The following differential equation came up in a "real" problem in the newsgroup `sci.math`:

```
> eq := (335*t^2+1290)*diff(f(t),t,t)+1540*t*diff(f(t),t)+468720*f(t)=544;
                             /   2      \
                2            |  d       |           /  d      \
  eq := (335 t   + 1290) |----- f(t)| + 1540 t |---- f(t)| + 468720 f(t) = 544
                             |   2      |           \ dt      /
                             \ dt       /
```

This falls into our class of ordinary linear differential equations with polynomial coefficients. Using *diffeqtorec*, we convert it into a holonomic recurrence:

```
> diffeqtorec(eq,f(t),u(n));
                         2                              2
     {(- 241 n - 67 n   - 93744) u(n) + (- 516 - 258 n   - 774 n) u(n + 2),

                   7812         136
         u(2) = - ---- u(0) + ---, u(1) = D(f)(0)}
                    43          645
```

Therefore the Taylor coefficients u_n of f satisfy the recurrence

$$u_{n+2} = -\frac{\left(241\,n + 67\,n^2 + 93744\right)}{516 + 258\,n^2 + 774\,n}u_n \quad \text{for } n \geq 1,$$

and thus the even and odd parts of f, $(f(t) + f(-t))/2$ and $(f(t) - f(-t))/2$, are hypergeometric. Since a particular solution of the inhomogeneous equation is 34/29295, this gives a full description of the sought solution.

rectodiffeq This procedure is the inverse of the procedure *diffeqtorec*: given a holonomic recurrence $a_p(n)u_{n+p} + \cdots + a_1(n)u_{n+1} + a_0(n)u_n + b_n = 0$, it computes a holonomic equation satisfied by its o.g.f. $y(z)$. This is done by translating each $n^k u_{n+i}$ into $\theta^k[y(z)z^{-i}]$ where $\theta = z\frac{d}{dz}$ and recovering the initial conditions from the inhomogeneous terms.

Example: The sequence of general term $u_k = 1/(k^2 + 1)$ is holonomic because it satisfies the inhomogeneous recurrence $(k^2 + 1)u_k = 1$, or the homogeneous recurrence $(k^2 + 1)u_k = (k^2 + 2k + 2)u_{k+1}$. The corresponding differential equation is

```
> rectodiffeq((k^2+1)*u(k)=1,u(k),y(z));
                                            /   2      \
           2  /  d      \      2    3  |  d       |
    (z - z ) |---- y(z)| + (z  - z ) |----- y(z)| + (1 - z) y(z) - 1
              \ dz      /               |   2      |
                                            \ dz       /
```

2.1 Building up equations

The class of holonomic functions and sequences enjoys nice closure properties. In particular we have the following theorem.
Theorem 1 (Closure) *[6, 13, 17, 22]*
(a) algebraic functions are holonomic;
(b) the sum of two holonomic functions is holonomic;
(c) the Cauchy product of two holonomic functions is holonomic (convolution of the sequences);
(d) the Hadamard product of two holonomic functions is holonomic (term-wise product of the sequences);
(e) if f is holonomic and g is algebraic, then $f \circ g$ is holonomic;
(f) the class of holonomic functions is closed under differentiation, integration, direct and inverse Laplace-Borel transform.
Most of these closure properties are implemented in the GFUN package. We now give a review of these, in the same order as the properties presented in the theorem. The algorithms described in this section can be found between the lines in [17] or [22].

algfuntodiffeq [Property (a)] Given an irreducible polynomial P in two variables, this procedure computes a holonomic equation satisfied by any function $y(z)$ solution of $P(z, y(z)) = 0$. The algorithm used by GFUN was pointed out by Comtet [6], it can also be found in [4, pp. 276–278]:

(α) Differentiating $P(z, y(z))$ with respect to z yields $y'(z) = A(y,z)/B(y,z)$, with A and B two polynomials.

(β) Since P and B are relatively prime (P is irreducible), the extended gcd algorithm yields three polynomials $u(y,z)$, $v(y,z)$ and $g(z)$ such that $uB + vP = g$. From this we define $C = A\,u \bmod P$ and deduce
$$y'(z) = \frac{C(y,z)}{g(z)}. \tag{2}$$

(γ) Differentiating repetitively equation (2) and reducing the right-hand side by means of (2) to suppress y' and modulo P to eliminate large powers of y, we obtain after a number of steps bounded by $\deg_y(P)$ sufficiently many linear equations to eliminate the powers of y by Gaussian elimination. This yields the desired equation.

Note: if P is not irreducible, a recursive splitting can be used as is now usual in computer algebra [9], and *algfuntodiffeq* applied to each factor. After making the resulting differential equations homogeneous, the procedure *diffeq+diffeq* (see below) gives a holonomic equation for the roots of $P(z, y(z)) = 0$.

Example: Motzkin numbers. These numbers count the number of unary-binary trees of size n. Their generating function satisfies $y = 1 + zy + z^2y^2$ and is analytic at zero, so that $M(z) = (1 - z - \sqrt{1 - 2z - 3z^2})/(2z^2)$. This expression translates into the following explicit formula for the nth Motzkin number:
$$M_n = -\frac{1}{2} \sum_{k=\lceil \frac{n+2}{2} \rceil}^{n+2} (-1)^k \binom{1/2}{k} \binom{k}{n+2-k} 2^{2k-n-2} 3^{n+2-k}. \tag{3}$$

The generating function of Motzkin numbers also satisfies the following differential equation

```
> algfuntodiffeq(y=1+z*y+z^2*y^2,y(z));
             3      2       / d      \        2
        (3 z  + 2 z  - z) |---- y(z)| + (3 z  + 3 z - 2) y(z) + 2
                            \ dz      /
```

This holonomic equation enables one, using the procedures *diffeqtorec* and *rectoproc* (see below) to compute efficiently the nth Motzkin number. In fact, the fastest known method to compute coefficients of an algebraic function [4, 5] is to use successively *algfuntodiffeq*, *diffeqtorec* and *rectoproc*.

diffeq+diffeq and rec+rec [Property (b)] The procedure *diffeq+diffeq*, given two holonomic equations $L_1(f, f', \ldots, f^{(d_1)}) = 0$ and $L_2(g, g', \ldots, g^{(d_2)}) = 0$, returns a holonomic equation satisfied by the sum $h = f + g$, using the following algorithm:

(α) Differentiate repetitively both sides of the equation $h = f + g$, reducing derivatives of order d_1 of f and d_2 of g according to L_1 and L_2.

(β) After $d_1 + d_2 + 1$ equations have been found, Gaussian elimination yields a linear dependency between $h, h', \ldots, h^{(d_1+d_2)}$.

The procedure *rec+rec* is the companion of *diffeq+diffeq* for recurrences.

diffeq*diffeq and cauchyproduct [Property (c)] These procedures are similar to *diffeq+diffeq* and *rec+rec* except that they compute the *product* of two holonomic functions, that is the Cauchy product (convolution) of two sequences. The algorithm starts from $h = fg$, then computes $d_1 d_2 + 1$ equations and Gaussian elimination yields a linear dependency between $h, h', \ldots, h^{(d_1 d_2)}$.

Example: The functions $f = 1/\sqrt{1-z}$ and $g = \cos(z)$ are holonomic because f is algebraic, thus holonomic, and $g'' + g = 0$. Thus the function $1/(1-z) + \cos(z)/\sqrt{1-z} = f(f+g)$ is holonomic by closure under sum and product:

```
> eq_f := algfuntodiffeq(y^2*(1-z)=1,y(z)): eq_g := diff(y(z),z,z)+y(z):
> `diffeq*diffeq`(eq_f,`diffeq+diffeq`(eq_f,eq_g,y(z)),y(z));
             4      3       2
        (16 z  - 64 z  + 136 z  - 144 z + 53) y(z)

                  5      4       3       2
             + (16 z  - 80 z  + 168 z  - 184 z  + 125 z - 45) D(y)(z)

                  4       3       2                 (2)
             + (32 z  - 128 z  + 240 z  - 224 z + 80) D   (y)(z)

                  5      4       3       2                      (3)
             + (16 z  - 80 z  + 172 z  - 196 z  + 116 z - 28) D   (y)(z)
```

Example: To prove that $\sin^2 + \cos^2 = 1$, it is sufficient to compute the differential equation of order 3 satisfied by the left-hand side, $f''' + 4f' = 0$, check the first three Taylor coefficients, and then use an argument of analytic continuation.

hadamardproduct and rec*rec [Property (d)] Given two holonomic equations defining respectively f and g, the procedure *hadamardproduct* computes the Hadamard product of f and g, that is the o.g.f. of $f_n g_n$. The algorithm used by GFUN is:

(α) Using *diffeqtorec*, convert the equations into two linear recurrences $L_1(f_n, \ldots, f_{n+p}) = 0$ and $L_2(g_n, \ldots, g_{n+q}) = 0$.

(β) Define $h_n = f_n g_n$.

(γ) Shift repetitively this equation ($n \mapsto n+1$), reducing f_{n+p} and g_{n+q} according to L_1 and L_2.

(δ) Once $pq + 1$ equations have been built, Gaussian elimination yields a linear dependency between $h_n, h_{n+1}, \ldots, h_{n+pq}$.

(ϵ) Using *rectodiffeq*, convert back this recurrence into a holonomic equation for the function $h(z) = \sum h_n z^n$.

The procedure *rec*rec* is the companion for recurrences: it returns a recurrence for the sequence $f_n g_n$, following steps (β) to (δ).

Example: Squares of Catalan numbers. Knowing that the generating function of Catalan numbers is solution of $y(x) = 1 + xy^2(x)$, we deduce a holonomic equation for the generating function of squares of Catalan numbers:

```
> deq := algfuntodiffeq(y = 1 + x*y^2, y(x)):
> hadamardproduct(deq,deq,y(x));
                                                  /  2       \
        2          / d       \          3    2    |  d       |
  (- 32 x  + 3 x) |---- y(x)| + (- 16 x  + x ) |----- y(x)| + (1 - 4 x) y(x)
                   \ dx      /                    |   2      |
                                                  \ dx       /

     - y(0)
```

algebraicsubs [Property (e)] Given a holonomic equation $L(f, \ldots, f^{(d_1)}) = 0$ and a square-free polynomial equation $P(z, g) = 0$ with $\deg_g(P) = d_2$, this procedure derives a holonomic equation for $h = f \circ g$. The algorithm is as follows:

(α) Use steps (α) and (β) of *algfuntodiffeq* to deduce from $P(z, g)$ an expression for g'

$$g' = \frac{A(z,g)}{B(z)} \tag{4}$$

where A, B are polynomials and $\deg_g(A) < d_2$.

(β) Rewrite $L(f, \ldots, f^{(d_1)}) = 0$ as

$$f^{(d_1)}(z) = C(z, f(z), f'(z), \ldots, f^{(d_1-1)}(z)) \tag{5}$$

where C is rational in z and linear with respect to the other variables. Replace in (5) z by $g(z)$, and like in step (β) of *algfuntodiffeq* eliminate g from the denominator. We then obtain

$$f^{(d_1)} \circ g(z) = D(z, g(z), f \circ g(z), f' \circ g(z), \ldots, f^{(d_1-1)} \circ g(z)) \tag{6}$$

where D is rational in z, polynomial of degree less than d_2 in the second variable, and linear with respect to the other variables. Now we are ready for the main loop of the algorithm:

(γ) Define $h = f(g(z))$.

(δ) Differentiate repetitively with respect to z and reduce according to equations (4) and (6) as well as modulo P. The remaining terms are of the form

$$R_{i,j}(z) g^i(z) f^{(j)}(g(z))$$

with $R_{i,j}(z)$ a rational function, $0 \le i < d_2$ and $0 \le j < d_1$.

(ϵ) After $d_1 d_2 + 1$ iterations, Gaussian elimination yields the desired equation.

Example: Given an explicit holonomic function, it is often useful to find a differential equation it satisfies. From this one can then deduce a recurrence for its Taylor coefficients which enable a computation of a Taylor series of order n in $O(dn)$ operations, where d is the order of the recurrence. For example, consider the exponential generating function of the numerators of convergents to e, given in (1):

$$y(z) = \frac{\sqrt{e^{1-\sqrt{1-4z}}}}{1-4z}\left(1+\frac{2}{\sqrt{1-4z}}\right).$$

We first write $y(z) = f(g(z))$ where $f(z) = \sqrt{e^{1-z}}(1+2/z)/z^2$ and $g(z) = \sqrt{1-4z}$. The function f is holonomic because $z(z+2)f' = -(z^2/2+3z+6)f$, and g is algebraic ($g^2(z) = 1-4z$), thus y is holonomic and satisfies

```
> eq_f:=z*(z+2)*diff(y(z),z)+(z^2/2+3*z+6)*y(z): eq_g:=y(z)^2-(1-4*z):
> algebraicsubs(eq_f, eq_g, y(z));
                                                 (2)
           y(z) + 10 D(y)(z) + (- 1 + 4 z) D    (y)(z)
```

From this one can deduce a recurrence with *diffeqtorec* and then using *rectoproc* (see below) compute very efficiently the numerators of the convergents to e. Since the denominators obey a similar recurrence, this gives a way to generate quickly the nth convergent to e, within a little amount of memory.

borel and invborel [Property (f)] Given a recurrence for a holonomic sequence $\{a_n\}$, these procedures compute a recurrence for the sequences $\{a_n/n!\}$ and $\{n!a_n\}$ respectively. Thus the Borel transform of an o.g.f. f is the associated e.g.f. $\hat{f}$, and the inverse Borel transform of $\hat{f}$ is f, which is given formally by

$$f(z) = \int_0^\infty \hat{f}(tz)e^{-t}\,dt. \tag{7}$$

By giving an extra parameter to these functions, they work directly on the differential equations.

Example: **Resummation.** Formal power series solutions of linear differential equations at an irregular point are generally divergent [20]. However, by means of Borel transform and then its inverse transform under integral form, one can get correct numerical values from these divergent series (the litterature on this subject is abundant, more precise and general statements can be found in [2, 3, 14, 15, 18]). Of particular interest are the singularities of the Borel transform which are intimately related to the Stokes phenomenon. This corresponds to taking different paths of integration in (7) so as to avoid singularities. These singularities can be obtained explicitly as the roots of the higher order coefficient of the differential equation satisfied by the Borel transform. We give the celebrated example of Euler's equation

```
> eq := {z^2*diff(y(z),z)+y(z)=z, (D(y))(0)=1};
                                  /          \
                                2 |  d       |
           eq := {D(y)(0) = 1, z  |---- y(z)| + y(z) = z}
                                  |          |
                                  \ dz       /
```

The power series solution to this equation can be found by *diffeqtorec* and solving the resulting equation: it is the divergent series $\sum(-1)^n n! z^{n+1}$. This prevents direct computation of say $y(0.1)$. We compute the Borel transform of `eq`:

```
> eq2:=borel(eq,y(z),Y,'diffeq');
                                                 /  d       \
       eq2 := {D(Y)(0) = 1, Y(0) = 0, z (z + 1) |---- Y(z)| - z}
                                                 \ dz       /
```

From this we deduce that the only singularity of the Borel transform is at -1. Besides, we do not need numerical integration to compute $\hat{y} = Y$ in (7) because fortunately this equation admits a very simple liouvillian solution:

```
> dsolve(eq2,Y(z));
                                 Y(z) = ln(z + 1)
```
We thus get a *convergent* integral representation for the solution:
```
> res:=Int(exp(-t)*ln(z*t+1),t=0..infinity);
                          infinity
                             /
                            |
                 res :=     |      exp(- t) ln(1 + z t) dt
                            |
                           /
                           0
```
And we can easily compute $y(z)$ at a specified z. This is $y(0.1)$:
```
> evalf(subs(z=0.1,res));
                                   .09156333394
```
As a check, the method of summation to the least term yields in this case 0.0915637760, whose first 6 digits are correct.

2.2 Useful tools

rectoproc Given a holonomic recurrence, *rectoproc* returns a MAPLE procedure that computes the nth term of the sequence. The initial terms of the sequence, if provided, are stored in the *remember* table of the procedure; the other terms are computed one by one, according to the recurrence, and using the memo-mechanism (`option remember`) provided by MAPLE.

Example: Motzkin numbers. Using *algfuntodiffeq*, we obtained (page 7) a holonomic equation for the o.g.f. $M(z)$ of Motzkin numbers. Now we first translate this differential equation into a recurrence
```
> rec := diffeqtorec((3*z^3+2*z^2-z)*D(y)(z)+(3*z^2+3*z-2)*y(z)+2, y(z), M(n));
{M(0) = 1, M(1) = 1, (3 n - 3) M(n - 2) + (2 n + 1) M(n  - 1)  + (- n - 2) M(n)}
```
which we convert into a procedure to compute the nth Motzkin number M_n
```
> Motzkin := rectoproc(rec, M(n));
proc(n)
options remember;
    if not type(n,nonnegint) then ERROR(`invalid arguments`) fi;
    ((3*n-3)*procname(n-2)+(2*n+1)*procname(n-1))/(n+2)
end
```
Now the MAPLE procedure *Motzkin* enables one to get as many terms of the sequence as we want
```
> Motzkin(100);
                 737415571391164350797051905752637361193303669
```
The above computation took only 0.2 second on a DecStation 5000, compared to 8.6 seconds using the formula (3). In general, for a recurrence of order k, the procedure *rectoproc* makes $O(kn)$ operations to compute the nth term of the sequence (or all terms up to the nth). As already mentioned, this is usually the fastest way to compute coefficients of algebraic functions. More generally, when faced with a sequence, it is very important to check whether it may be holonomic, because this allows for a very fast computation of the elements of the sequence. Another nice example is the rencontres problem [7, p. 180-183] already mentioned.

ratpolytocoeff Given a rational function, this procedure computes an explicit value of the nth Taylor coefficient. The algorithm consists in computing a partial fraction decomposition and then translating it term by term. It is equivalent to solving the linear recurrence with constant coefficients by the characteristic polynomial.

Example: The Fibonacci sequence. We start from the generating function obtained by *listtoratpoly*.

```
> ratpolytocoeff(1/(1-z-z^2),z,n);
                                (- 1 - n)      (- 1 - n)
                      (- 1 - %1)                %1
                    - ------------------- + -----------
                           2 %1 + 1           2 %1 + 1
                                                2
%1 :=                    RootOf(- 1 + _Z + _Z )
```

This is to be read

$$F_n = \frac{\phi^{-n-1} - (-1-\phi)^{-n-1}}{2\phi + 1},$$

where ϕ is any of the roots of $z^2 + z - 1$ (the golden ratio and its algebraic conjugate). A future version of this program will completely avoid unnecessary factorizations and thus work over the mimimal algebraic extension [10].

3 Installation and customization

As usual in Maple, the package gfun comes in two files: **gfun** and **gfun.test**. These are available by anonymous ftp from **ftp.inria.fr** or by email from the authors. From the file *gfun*, one should first create a file **gfun.m** by typing (under Unix)

`maple -s < gfun`

To use the package, the global Maple variable *_liblist* (MapleV.2 does not need this) should be set in such a way that the package is located in a visible place. See **?with** under Maple for more details. Once *_liblist* has been properly set, the package is loaded by the command **with(gfun)**. Then help on-line for the package is also loaded and can be accessed by **?gfun**. If you intend to use this package a lot, it might be desirable to set *_liblist* and to load gfun in your **.mapleinit** file (under Unix).

The behavior of the functions in the package is controlled by their arguments and by several global variables. Most notably (see also **?gfun,parameters**),

- **gfun['maxordereqn']** and **gfun['minordereqn']** maximum and minimum order of linear recurrence equation or linear differential equation which is tried in the functions *seriesto...* and *listto...*;
- **gfun['maxdegeqn']** and **gfun['mindegeqn']** maximum and minimum degree of an algebraic equation for the functions *listtoalgeq* and *seriestoalgeq*;
- **gfun['maxdegcoeff']** and **gfun['mindegcoeff']** maximum and minimum degree of the coefficients in the l.d.e., l.r.e. and algebraic equations for the functions *seriesto...* and *listto...*;
- **gfun['optionsgf']** a list of types of generating functions to try in the functions *listto...* and *seriesto...*.

It is also possible to get details of the computations taking place by setting **infolevel[gfun]** to anything between 0 and 5 (increasing it gives more information).

Acknowledgements

This program started partly from discussions with F. Bergeron and S. Plouffe, and benefited from their constant but kind urge to add new functionalities.

This work was supported in part by the ESPRIT III Basic Research Action Programme of the E.C. under contract ALCOM II (#7141).

References

[1] Bergeron, F., and Plouffe, S. Computing the generating function of a series given its first terms. *Journal of experimental mathematics* (1992).

[2] Borel, É. Leçons sur les séries divergentes. In *Collection de monographies sur la théorie des fonctions, publiée sous la direction de M. Émile Borel*. Gauthiers-Villars, Paris, 1901. Second edition, 1928. Reprinted by J. Gabay, 1988.

[3] Candelpergher, B. Une introduction à la résurgence. *Gazette des Mathématiciens 42* (Oct. 1989), 36–64.

[4] Chudnovsky, D. V., and Chudnovsky, G. V. On expansion of algebraic functions in power and Puiseux series, I. *Journal of Complexity 2* (1986), 271–294.

[5] Chudnovsky, D. V., and Chudnovsky, G. V. On expansion of algebraic functions in power and Puiseux series, II. *Journal of Complexity 3* (1987), 1–25.

[6] Comtet, L. Calcul pratique des coefficients de Taylor d'une fonction algébrique. *L'Enseignement Mathématique 10* (1964), 267–270.

[7] Comtet, L. *Advanced Combinatorics*. Reidel, Dordrecht, 1974.

[8] Delest, M.-P., and Viennot, G. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science 34* (1984), 169–206.

[9] Duval, D. *Diverses questions relatives au calcul formel avec des nombres algébriques*. Doctorat d'État, Université scientifique, technologique et médicale de Grenoble, 1987.

[10] Gourdon, X., and Salvy, B. Asymptotics of linear recurrences with rational coefficients. Tech. rep., INRIA, 1992. To appear.

[11] Graham, R., Knuth, D., and Patashnik, O. *Concrete Mathematics*. Addison Wesley, 1989.

[12] Guttmann, A. J., and Enting, I. G. The number of convex polygons on the square and honeycomb lattices. *Journal of Physics* Series A *21* (1988), 467–474.

[13] Lipshitz, L. *D*-finite power series. *Journal of Algebra 122* (1989), 353–373.

[14] Loday-Richaud, M. Introduction à la multisommabilité. *Gazette des Mathématiciens 44* (Apr. 1990), 41–63.

[15] Malgrange, B., and Ramis, J.-P. Fonctions multisommables. *Annales de l'Institut Fourier 42*, 1-2 (1992), 353–368.

[16] Sloane, N. J. A. *A Handbook of Integer Sequences*. Academic Press, 1973.

[17] Stanley, R. P. Differentiably finite power series. *European Journal of Combinatorics 1* (1980), 175–188.

[18] Thomann, J. Resommation des séries formelles. Solutions d'équations différentielles linéaires du second ordre dans le champ complexe au voisinage de singularités irrégulières. *Numerische Mathematik 58* (1990), 503–535.

[19] Ulmer, F. On algebraic solutions of linear differential equations with primitive unimodular Galois group. In *Algebraic Algorithms and Error Correcting Codes* (1991), vol. 307 of *Lecture Notes in Computer Science*, pp. 446–455.

[20] Wasow, W. *Asymptotic Expansions for Ordinary Differential Equations*. Dover, 1987. A reprint of the John Wiley edition, 1965.

[21] Wilf, H. S. *Generatingfunctionology.* Academic Press, 1990.

[22] Zeilberger, D. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics 32* (1990), 321–368.

[23] Zeilberger, D. A proof of Julian West's conjecture that the number of two-stack-sortable permutations of length n is $2(3n)!/((n+1)!(2n+1)!)$. *Discrete Mathematics 102* (1992), 85–93.