```
> restart
```

New results in computing Formal Power Series in Maple: Hypergeometric type and non-holonomic power series

February 2021
(Initially done using Maple 2019 - Maple 2020)

Bertrand Teguia Tabuguia and Wolfram Koepf

This worksheet presents examples that show how the new package FPS will be a great addition to Maple. It starts by explaining why it is worth to have a direct approach. Then it gives some problems related to the current Maple implementation. This is followed by results that could not be obtained previously. The list of examples can be extended at will, the goal here is just to present a few with emphasis on new results. Some byproducts are also presented.

```
> libname:=currentdir(),libname
> with(FPS)
> with(CodeTools):
```

convert/FormalPowerSeries is absolutely non-linear and misses to work for sums. Nevertheless, we would like to show why mapping convert/FormalPowerSeries on the input summands is not a good idea.

Indeed, by doing so we lose some aspects of the algorithm that might gather in its simplification aspect. In fact mapping convert/FormalPowerSeries on the input summands will force outputs to have at least the same length as their inputs, which is not an appropriate behavior for a symbolic algorithm. Let us give some examples.

```
> f:=cos(z)^2+sin(z)^2:convert(f,FormalPowerSeries,z,n)
> map(g->convert(g,FormalPowerSeries,z,n),f)
> f:=cos(2*z)-(cos(z)^2-sin(z)^2):convert(f,FormalPowerSeries,z,n)
> map(g->convert(g,FormalPowerSeries,z,n),f)
> f:=(z+3*z^2+z^3)*exp(z):convert(f,FormalPowerSeries,z,n)
> map(g->convert(g,FormalPowerSeries,z,n),expand(f))
> f:=sin(z)+z*cos(z):convert(f,FormalPowerSeries,z,n)
> map(g->convert(g,FormalPowerSeries,z,n),f)
> f:=(-1/2*z + 1/6*z^3)*arctan(z):convert(f,FormalPowerSeries,z,n)
> map(g->convert(g,FormalPowerSeries,z,n),expand(f))
```

The outputs that one should get in the three latter examples

```
> FPS((z+3*z^2+z^3)*exp(z),z,n)
> FPS(sin(z)+z*cos(z),z,n)
> FPS((-1/2*z + 1/6*z^3)*arctan(z),z,n)
```

It is therefore worth to think of a direct approach and this is what the function FPS does. However, before getting into such computations, we give examples for which the current Maple implementation yields more complicated or even wrong results due to its lack of an algorithm to compute m-fold hypergeometric term coefficients.

Complicated results

```
> convert((sin(z)+cos(z))^3,FormalPowerSeries,z,n)
> convert(ln(1+z+z^2+z^3),FormalPowerSeries,z,n)
> convert((sin(z)+cos(z))^3,FormalPowerSeries,z,n)
> convert(tan(3*arctan(z^2)),FormalPowerSeries,z,n)
> convert(1/(q[1]-z^2)/(q[2]-z^3),FormalPowerSeries,z,n)
```

Observe that mapping convert/FormalPowerSeries on these inputs will not make the results any better.

Much simpler outputs

```
> FPS((sin(z)+cos(z))^3,z,n)
> FPS(ln(1+z+z^2+z^3),z,n)
> FPS((sin(z)+cos(z))^3,z,n)
> FPS(tan(3*arctan(z^2)),z,n)
> FPS(1/(q[1]-z^2)/(q[2]-z^3),z,n)
```

Wrong outputs: wrong computation of initial coefficients with Maple 2019.

The following examples clearly show that computing linear combinations in the current Maple code is not well handled.

```
> convert((-1/2*z + 1/6*z^3)*arctan(z) + (-1/4*z^2 + 1/12)*ln(z^2 +
  1) + 5/12*z^2 + 1/4,FormalPowerSeries,z,n)
> series((-1/2*z + 1/6*z^3)*arctan(z) + (-1/4*z^2 + 1/12)*ln(z^2 +
  1) + 5/12*z^2 + 1/4,z=0,1)
> convert(exp(z)+log(1+z),FormalPowerSeries,z,n)
> series(exp(z)+log(1+z),z=0,1)
> convert(arctan(z+2),FormalPowerSeries,z,n)
> series(arctan(z+2),z=0,1)
> convert(arctanh(z)+cosh(z),FormalPowerSeries,z,n)
> series(arctanh(z)+cosh(z),z=0,1)
> convert(exp(z)+1/(1-z),FormalPowerSeries,z,n)
> series(exp(z)+1/(1-z),z,1)
```

Correct outputs

```
> FPS((-1/2*z + 1/6*z^3)*arctan(z) + (-1/4*z^2 + 1/12)*ln(z^2 + 1)
  + 5/12*z^2 + 1/4,z,n)
> FPS(exp(z)+log(1+z),z,n)
> FPS(arctan(z+2),z,n)
> FPS(arctanh(z)+cosh(z),z,n)
> FPS(exp(z)+1/(1-z),z,n)
```

Hypergeometric type power series

This is a generalization of all types of series that Maple can currently handle. It includes linear combinations of Laurent-Puiseux series by default. All this is made possible thanks to the algorithm **mfoldHyper** which computes m-fold hypergeometric term solutions of holonomic recurrence equations. Taking into account a precise computation of linear combinations without neglecting initial terms, FPS computes hypergeometric type series linearly.

convert/FormalPowerSeries fails in all the examples of this section.

We recall some steps of the algorithm

```
> f:=exp(z^2)+1/(1-z)^2
> HolonomicDE(f,F(z))
> RE:=FindRE(f,z,a(n))
```

The use of mfoldHyper. This algorithm is new and is not yet implemented in Maple at the current time.

```
> mfoldHyper(RE,a(n))
```

Since there is a 2-fold hypergeometric term solution, one can compute the coefficient of the odd term as follows.

```
> mfoldHyper(RE,a(n),ml=[2,1])
```

Finally the procedure searches for the needed linear combination and gets the following series representation

```
> FPS(f,z,n)
```

Note that this latter step consisting of finding the linear combination also requires a careful algorithmic consideration. This is to avoid misrepresentation as we highligthed in the current Maple implementation.

For the next computations, the maximum value of differential equations to be computed is fixed to 10 because it is the default value used in FPS.

```
> convert(exp(z^2)+1/(1-z)^2,FormalPowerSeries,z,n)
> CPUTime(convert(arctan(z)+arcsin(z),FormalPowerSeries,
  differentialorder=10))
> CPUTime(convert(exp(z)+arcsinh(z)+cosh(z),FormalPowerSeries,
  differentialorder=10))
> CPUTime(convert(sin(2*arcsin(z))+cos(3*arccos(z)),
  FormalPowerSeries,differentialorder=10))
> CPUTime(convert(1/sqrt(1-z^3)+ln(1+z^2),FormalPowerSeries,z,n,
  differentialorder=10))
> CPUTime(convert(cosh(ln(1+z))+sin(z)+exp(z^2)+z*cos(z^3),
  FormalPowerSeries,differentialorder=10))
> CPUTime(convert(log(1+sqrt(z)+z+z^(3/2)),FormalPowerSeries,z,n,
  differentialorder=10))
> CPUTime(convert(cos(sqrt(z))+sin(z^(3/4)),FormalPowerSeries,z,n,
  differentialorder=10))
> CPUTime(convert(arctan(sqrt(z))+arcsinh(z^(1/3)),
  FormalPowerSeries,differentialorder=10))
> CPUTime(convert(arcsin(z^(1/7))+ln(1+z^(1/3)),FormalPowerSeries,
  differentialorder=11))
> CPUTime(convert(cos(sqrt(z))+sin(z^(1/3))+exp(z^(3/4)),
  FormalPowerSeries,differentialorder=12))
```

Corresponding results

```
> FPS(arctan(z)+arcsin(z),z,n)
> FPS(exp(z)+arcsinh(z)+cosh(z),z,n)
> FPS(sin(2*arcsin(z))+cos(3*arccos(z)),z,n)
> CPUTime(FPS(1/sqrt(1-z^3)+ln(1+z^2),z,n))
> CPUTime(FPS(cosh(ln(1+z))+sin(z)+exp(z^2)+z*cos(z^3),z,n))
> FPS(log(1+sqrt(z)+z+z^(3/2)),z,n)
> FPS(cos(sqrt(z))+sin(z^(3/4)),z,n)
> CPUTime(FPS(arctan(sqrt(z))+arcsinh(z^(1/3)),z,n))
> CPUTime(FPS(arcsin(z^(1/7))+ln(1+z^(1/3)),z,n,maxdeorder=11))
> CPUTime(FPS(cos(sqrt(z))+sin(z^(1/3))+exp(z^(3/4)),z,n,
  maxdeorder=12))
```

Bonus: special functions

```
> CPUTime(FPS(exp(z)+hypergeom([a, b], [c], z^2),z,n,fpstype=
  specialfunctions))
```

In principle, Maple's conbert/FormalPowerSeries should be able to compute the following power series since the underlying recurrence equations have only two terms. This is related to finding the linear combination and the Puiseux numbers which are generalized in the new FPS command.

```
> convert(1/(sqrt(1-4*z))*((1-sqrt(1-4*z))/2*z)^2,
  FormalPowerSeries,z,n)
> FindRE(1/(sqrt(1-4*z))*((1-sqrt(1-4*z))/2*z)^2,z,a(n))
> convert(sqrt(sqrt(8*z^3+1)-1),FormalPowerSeries,z,n)
> FindRE(sqrt(sqrt(8*z^3+1)-1),z,a(n))
```

Expected results

```
> FPS(1/(sqrt(1-4*z))*((1-sqrt(1-4*z))/2*z)^2,z,n)
> FPS(sqrt(sqrt(8*z^3+1)-1),z,n)
```

Non-hypergeometric type but holonomic series

For such input we give the output in a recursive form. This approach is important for computing larger-degree Taylor polynomials of holonomic functions. This is implemented in the command FPS[Taylor].

```
> infolevel[FPS]:=2:
> FPS(arcsin(z)^3,z,n)
> FPS(arcsin(z)^3,z,n,fpstype=holonomic)
> Taylor(arcsin(z)^3,z,0,10)
> series(arcsin(z)^3,z,11)
> FPS(exp(z+z^2)*cos(z),z,n,fpstype=holonomic)
> Taylor(exp(z+z^2)*cos(z),z,0,10)
> series(exp(z+z^2)*cos(z),z,11)
> T:=Time():Taylor(exp(z+z^2)*cos(z),z,0,1000):Time()-T
> T:=Time():series(exp(z+z^2)*cos(z),z,1000):Time()-T
```

Non-holonomic case

New approach on what to do when the given input does not satisfy a linear recurrence equation. The algorithm seeks for homogeneous quadratic differential equations.

```
> FPS(tan(z),z,n)
> QDE(tan(z),F(z))
> FindQRE(tan(z),z,a(n))
> FPS(tan(z)/z^2,z,n,fpstype=quadratic)
> FPS(1/(exp(z)-1),z,n,fpstype=quadratic)
> FPS(1/(1+sin(z)),z,n,fpstype=quadratic)
> FPS(1/cos(z)^2,z,n,fpstype=quadratic)
> FPS(1+tan(z)^2,z,n,fpstype=quadratic)
```

Proving non-trivial identities

A consequence of this new approach is the detection of difficult identities. The algorithm finds equivalences that are hidden to the Maple simplify command.

```
> f1:=2*arctanh(sin(2*z)/(1+cos(2*z)))
> g1:=ln((1+tan(z))/(1-tan(z)))
> FPS(f1,z,n,fpstype=quadratic)
> FPS(g1,z,n,fpstype=quadratic)
> FPS(f1-g1,z,n,fpstype=quadratic)
> simplify(f1-g1)
> f2:=log(tan(z/2)+sec(z/2))
> g2:=arcsinh(sin(z)/(1+cos(z)))
> CPUTime(FPS(f2,z,n,fpstype=quadratic))
> CPUTime(FPS(g2,z,n,fpstype=quadratic))
> CPUTime(FPS(f2-g2,z,n,fpstype=quadratic))
> simplify(f2-g2)
```

DEtools[FindODE] vs FPS[HolonomicDE]: faster computation of holonomic differential equations

A different strategy to compute holonomic differential equations is used. Linear algebra is better exploited. This increases the speed of FPS for computing hypergeometric type series.

Below are some examples for comparison with FormalPowerSeries [HolonomicDE] which is equivalent to DEtools[FindODE] apart from the simplification applied for special functions in the latter.

```
> t,DE:=CPUTime(DEtools[FindODE](arctan(z)^2+sin(z)^4+log(1+z)^5,F
  (z),12)):
> t,PDEtools[difforder](DE,z)
> t,DE:=CPUTime(HolonomicDE(arctan(z)^2+sin(z)^4+log(1+z)^5,F(z),
  12)):
> t,PDEtools[difforder](DE,z)
> t,DE:=CPUTime(DEtools[FindODE]((3*z+5*z^7+11*z^13)*log(1+z^3+z^7)
  +arctanh(z)*cos(z)^5,F(z),15)):
> t,PDEtools[difforder](DE,z)
> t,DE:=CPUTime(HolonomicDE((3*z+5*z^7+11*z^13)*log(1+z^3+z^7)+
  arctanh(z)*cos(z)^5,F(z),15)):
> t,PDEtools[difforder](DE,z)
```

For special functions one can use FPS[HolonomicDE] as follows

```
> HolonomicDE(Int(Int(2*hypergeom([3/4, 5/4], [2], 64*x^3*(1 + x)/
  (-4*x^2 + 1)^2)/(-4*x^2 + 1)^(3/2), x), x)/x^2,F(x),
  specialfunctions)
> HolonomicDE(BesselI(0, x) + x*BesselI(2, x), F(x),
  specialfunctions)
> HolonomicDE(KummerM(1/4, 1, x) + KummerM(-1/4, 1, x),F(x),
  specialfunctions)
> HolonomicDE(LegendreP(1/4, x)*LegendreP(1/2, x),F(x),
  specialfunctions)
```

Detected issue on LREtools[hypergeomsols] for computing hypergeometric terms, a particular case of mfoldHyper implemented in FPS[rectohyperterm]

The function implements a variant of the one implemented in LREtools [hypergeomsols] with a further step consisting of writing outputs using factorial and Pochhammer symbols. The code FPS[sumhyperRE] computes a holonomic recurrence equation satisfied by a list of hypergeometric terms.

```
> RE[1]:=sumhyperRE([(a^(1/3))^n/2^n,(1+sqrt(5))^n/2^n],v(n))
> LREtools[hypergeomsols](RE[1],v(n),{},output=basis)
> RE[2]:=sumhyperRE([(1-sqrt(7))^n/n!,(1+sqrt(7))^n/(n+1)!],v(n))
> LREtools[hypergeomsols](RE[2],v(n),{},output=basis)
```

FPS[rectohyperterm]
the argument "complex" is used to allow computations for solutions involving non-rational numbers, which is bounded by the field of complex numbers.

```
> rectohyperterm(RE[1],v(n),complex)
> rectohyperterm(RE[2],v(n),complex)
```

Mark van Hoeij, the author of LREtools[hypergeomsols], proposed an alternative for this issue by using convert/RootOf

```
> LREtools[hypergeomsols](convert(RE[1],RootOf),v(n),{},output=
  basis)
> LREtools[hypergeomsols](convert(RE[2],RootOf),v(n),{},output=
  basis)
```

Below are examples where this does not lead to results

```
> RE[3]:=sumhyperRE([cos(x)^(3*n)/n!,(-sin(x))^(3*n)/n!],v(n))
> LREtools[hypergeomsols](convert(RE[3],RootOf),v(n),{},output=
  basis)
> RE[4]:=sumhyperRE([ln(x)^n,ln(y*x)^n],v(n))
> LREtools[hypergeomsols](convert(RE[4],RootOf),v(n),{},output=
  basis)
```

FPS[rectohyperterm]

```
> rectohyperterm(RE[3],v(n),complex)
> rectohyperterm(RE[4],v(n),complex)
```

An important fact for solutions over algebraic extension fields: the code gathers conjugate hypergeometric term solutions in a single representation. This is made possible thanks to the Maple standard representation of algebraic numbers: "RootOf". When needed, as in the case of computing power series, the values can be recovered by applying "allvalues" to the solutions as shown below.

```
> RE[5]:=FindRE(1/(1+z+z^7),z,v(n))
> rectohyperterm(RE[5],v(n),complex)
> map(allvalues,%)
> LREtools[hypergeomsols](RE[5],v(n),{},output=basis)
```

The advantage is that this makes it possible to avoid the reconstruction of solutions which is a further step considered by LREtools[hypergeomsols].

```
> S:=[solve(z^3+7*z^2+z+28,z)]:
> RE[6]:=sumhyperRE([S[1]^n,S[2]^n,S[3]^n],v(n))
> rectohyperterm(RE[6],v(n),complex)
> map(allvalues,%)
> LREtools[hypergeomsols](RE[6],v(n),{},output=basis)
> RE[7]:=sumhyperRE([I^n/n!,(-I)^n/n!,n!*(1+I*sqrt(7))^n/a^n,n!*(1-
  I*sqrt(7))^n/a^n],v(n))
> rectohyperterm(RE[7],v(n),complex)
> LREtools[hypergeomsols](RE[7],v(n),{},output=basis)
```

Note, however, that these remarks are not meant to show that FPS [rectohyperterm] is better than LREtools[hypergeomsols]. The idea is to present some tools that could improve LREtools[hypergeomsols] whose speed remains the best in most of the successful examples.

This presentation of the package FPS ends here. We hope you found it interesting. Feel free to send questions to
bertrand.teguia@aims-cameroon.org/bteguia@mathematik.uni-kassel.de