`FPS`
A Package for the Automatic Calculation of Formal Power Series

Wolfram Koepf
ZIB Berlin
Email: `Koepf@ZIB.de`

Present REDUCE form by
Winfried Neun
ZIB Berlin
Email: `Neun@ZIB.de`

1 Introduction

This package can expand functions of certain type into their corresponding Laurent-Puiseux series as a sum of terms of the form

$$\sum_{k=0}^{\infty} a_k (x - x_0)^{mk/n+s}$$

where m is the 'symmetry number', s is the 'shift number', n is the 'Puiseux number', and x_0 is the 'point of development'. The following types are supported:

- **functions of 'rational type'**, which are either rational or have a rational derivative of some order;
- **functions of 'hypergeometric type'** where $a(k+m)/a(k)$ is a rational function for some integer m;
- **functions of 'explike type'** which satisfy a linear homogeneous differential equation with constant coefficients.

The FPS package is an implementation of the method presented in [2]. The implementations of this package for Maple (by D. Gruntz) and Mathematica (by W. Koepf) served as guidelines for this one.

Numerous examples can be found in [3]–[4], most of which are contained in the test file `fps.tst`. Many more examples can be found in the extensive bibliography of Hansen [1].

2 REDUCE operator FPS

The FPS Package must be loaded first by:

```
load FPS;
```

`FPS(f,x,x0)` tries to find a formal power series expansion for `f` with respect to the variable `x` at the point of development `x0`. It also works for formal Laurent (negative exponents) and Puiseux series (fractional exponents). If the third argument is omitted, then `x0:=0` is assumed.

Examples: `FPS(asin(x)^2,x)` results in

```
         2*k  2*k             2  2
        x   *2   *factorial(k) *x
infsum(----------------------------,k,0,infinity)
        factorial(2*k + 1)*(k + 1)
```

`FPS(sin x,x,pi)` gives

```
                   2*k       k
        ( - pi + x)   *( - 1) *( - pi + x)
infsum(------------------------------------,k,0,infinity)
                factorial(2*k + 1)
```

and `FPS(sqrt(2-x^2),x)` yields

```
            2*k
         - x   *sqrt(2)*factorial(2*k)
infsum(--------------------------------,k,0,infinity)
           k              2
          8 *factorial(k) *(2*k - 1)
```

Note: The result contains one or more `infsum` terms such that it does not interfere with the REDUCE operator `sum`. In graphical oriented REDUCE interfaces this operator results in the usual $\sum$ notation.

If possible, the output is given using factorials. In some cases, the use of the Pochhammer symbol `pochhammer(a,k)`$:= a(a+1)\cdots(a+k-1)$ is necessary.

The operator `FPS` uses the operator `SimpleDE` of the next section.

If an error message of type

```
Could not find the limit of:
```

occurs, you can set the corresponding limit yourself and try a recalculation. In the computation of `FPS(atan(cot(x)),x,0)`, REDUCE is not able to find the value for the limit `limit(atan(cot(x)),x,0)` since the `atan` function is multi-valued. One can choose the branch of `atan` such that this limit equals $\pi/2$ so that we may set

```
let limit(atan(cot(~x)),x,0)=>pi/2;
```

and a recalculation of `FPS(atan(cot(x)),x,0)` yields the output `pi - 2*x` which is the correct local series representation.

3 REDUCE operator SimpleDE

`SimpleDE(f,x)` tries to find a homogeneous linear differential equation with polynomial coefficients for f with respect to x. Make sure that y is not a used variable. The setting `factor df;` is recommended to receive a nicer output form.

Examples: `SimpleDE(asin(x)^2,x)` then results in

```
                 2
df(y,x,3)*(x  - 1) + 3*df(y,x,2)*x + df(y,x)
```

`SimpleDE(exp(x^(1/3)),x)` gives

```
                    2
27*df(y,x,3)*x  + 54*df(y,x,2)*x + 6*df(y,x) - y
```

and `SimpleDE(sqrt(2-x^2),x)` yields

```
               2
df(y,x)*(x  - 2) - x*y
```

The depth for the search of a differential equation for `f` is controlled by the variable `fps_search_depth`; higher values for `fps_search_depth` will increase the chance to find the solution, but increases the complexity as well. The default value for `fps_search_depth` is 5. For `FPS(sin(x^(1/3)),x)`, or `SimpleDE(sin(x^(1/3)),x)` e. g., a setting `fps_search_depth:=6` is necessary.

The output of the FPS package can be influenced by the switch `tracefps`. Setting `on tracefps` causes various prints of intermediate results.

4 Problems in the current version

The handling of logarithmic singularities is not yet implemented.

The rational type implementation is not yet complete.

The support of special functions [5] will be part of the next version.

References

[1] E. R. Hansen, *A table of series and products.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[2] Wolfram Koepf, *Power Series in Computer Algebra*, J. Symbolic Computation 13 (1992)

[3] Wolfram Koepf, *Examples for the Algorithmic Calculation of Formal Puiseux, Laurent and Power series*, SIGSAM Bulletin 27, 1993, 20-32.

[4] Wolfram Koepf, *Algorithmic development of power series.* In: Artificial intelligence and symbolic mathematical computing, ed. by J. Calmet and J. A. Campbell, International Conference AISMC-1, Karlsruhe, Germany, August 1992, Proceedings, Lecture Notes in Computer Science **737**, Springer-Verlag, Berlin–Heidelberg, 1993, 195–213.

[5] Wolfram Koepf, *Algorithmic work with orthogonal polynomials and special functions.* Konrad-Zuse-Zentrum Berlin (ZIB), Preprint SC 94-5, 1994.