Hypergeometric term solutions of holonomic recurrence equations

A variant of van Hoeij's algorithm

Bertrand Teguia Tabuguia (https://www.bertrandteguia.com)

1 Package HyperTermHoloRE

1.1 Introduction to HyperTermHoloRE

A term `a[n]` is said to be of hypergeometric term if the ration `a[n+1]/a[n]` is a rational function in the variable n. Note that this includes polynomial and rational functions over the field considered. Examples: `(-1)^n/n!`, `n/(2*n)!`.

A recurrence equation is said to be holonomic if it is linear homogeneous with polynomial coefficients (constant coefficients included) in the variable index. One could also include the case of rational coefficients given that multiplication by the common denominator makes the equation holonomic. Thus the package considers homogeneous linear recurrence with rational or polynomial coefficients.

The hypergeometric term solutions of a given holonomic recurrence equation form a subspace of the vector space of solutions. The main goal of this package is to compute this subspace in a basis representation form so that, every hypergeometric term solution of the given recurrence equation can be written as a linear combination of the element of that basis. The code is based on a variant of Mark van Hoeij's algorithm (see Chapter 6 in https://kobra.uni-kassel.de/handle/123456789/11598) which is proved to be much more efficient than the original Petkovsek algorithm. The corresponding function in the package is `HypervanHoeij`. The outputs are given in terms of factorials or Pochhammer symbols (when a representation with factorials is not easily accessible); Gamma terms are avoided and evaluations can be checked with non-negative integers.

Given a list of hypergeometric terms, the package provides the function `sumhyperRE` which computes a holonomic recurrence equation of lowest order satisfied by these hypergeometric terms. Up to multiplicative constants, `sumhyperRE` and `HypervanHoeij` work as reverse procedures for one another.

Examples:

```
(%i1) load(HyperTermHoloRE)$
Version 1.0
Execute 'Infos()' for details
Copyright (c) September 2020, Bertrand Teguia T.
https://www.bertrandteguia.com
University of Kassel, Germany
(%i2) RE:sumhyperRE([(-1)^n/n!,n/(2*n)!],a[n]);
                                         2
(%o2) (- 2 (n + 1) (n + 2) (2 n + 3) (4 n  + 3 n + 1) a     )
                                                       n + 2
                        3       2
         - (n + 1) (16 n  + 32 n  + 11 n - 2) a
                                               n + 1
               2
         + (4 n  + 11 n + 8) a  = 0
                               n
```

```
(%i3) HypervanHoeij(RE,a[n]);
                                      n
                                (- 1)     n
(%o3)                         {------, ------}
                                  n!   (2 n)!
```

At a step of `HypervanHoeij` the computation of the set of non-negative integer roots of the resultant of two polynomials viewed in a certain order is computed. This is called the dispersion set of these polynomials; users of this package can access it with the function `dispersionSet`. The implementation is based on the method described by Man Yiu-Kwong and Wright Francis J., which is more efficient than that based on the computation of the resultant.

Example:

```
(%i4) dispersionSet(n^27-1,(n+7)^27-1,n);
(%o4)                                {7}
```

1.2 Function definitions for HyperTermHoloRE

`HypervanHoeij` (*RE, u,* [*F*]) [Function]

Returns a basis of the subspace of hypergeometric term solutions of the recurrence equation *RE*. *RE* is a recurrence equation in the variable *u* which must be in the form *a[n]*, with atomic variables *a* and *n* (the variable index), to get a correct answer. The variable *F* which can be omitted can take two possible values: *Q* (default value) for solutions over the field of rational with non-parametric arguments, and *C* for solutions over extension fields of the rationals.

Examples:

```
(%i1) HypervanHoeij(-(2*n+1)*a[n+1]+2*(n+1)*a[n]=0,a[n]);
                                     n   2
                                    4  n!
(%o1)                             {------}
                                    (2 n)!
(%i2) RE:-n*(n+2)*a[n+2]+%i*(2*n+1)*a[n+1]+(n^2+1)*a[n]=0$
(%i3) HypervanHoeij(RE,a[n],C);
                                                      n
                          (%i + 1)     (1 - %i)  (- 1)
                                  n            n
(%o3)                  {------------, ----------------}
                         (n + %i) n!     (n - %i) n!
```

`sumhyperRE` (*L*,*u*) [Function]

Returns a recurrence equation of least order in the variable *u* (preferably in the form *a[n]*, with atomic variables *a* and *n* (the variable index)) for a given list *L* of hypergeometric terms.

Examples:

```
(%i1) RE1:sumhyperRE([n/n!,n^2/n!,n^3/n!],a[n]);
(%o1) (- n (n + 1) (n + 2) a     ) + 3 n (n + 1) a
                            n + 3                 n + 2
         - 3 n a      + a  = 0
                n + 1    n
(%i2) RE2:sumhyperRE([(1-sqrt(5))^n/2^n,(1+sqrt(5))^n/2^n],a[n]);
(%o2)                    (- a     ) + a      + a  = 0
                             n + 2     n + 1    n

One can recover the inputs of RE1 and RE2 with HypervanHoeij.

(%i3) HypervanHoeij(RE1,a[n]);
                                       2   3
                                  n   n   n
(%o3)                           {--, --, --}
                                  n!  n!  n!
(%i4) HypervanHoeij(RE2,a[n],C);
                          1 - sqrt(5)  n   sqrt(5) + 1  n
(%o4)                 {(-------------), (-------------) }
                               2                2
```

dispersionSet (*P1*, *P2*, *n*, [*F*]) [Function]

Returns the set of non-negative integers *k* such that `subst`(*n*+*k*,*n*,*P1*) has a common irreducible factor with *P2*. The optioanl variable *F* can take two possible values: *Q* (default value) to consider factors over the field of rational, and *C* for factorization over extension fields of the rationals. Note that more often extension fields are not necessary because the algorithm is based on a unique factorization domain.

Examples:

```
(%i1) dispersionSet((n-1)*(n+2)*(n+4),n*(n+20)*(n+30)*(n+10),n);
(%o1)              {1, 6, 8, 11, 16, 18, 21, 26, 28, 31}
(%i2) dispersionSet(n-%i,(n+1)^2+1,n,C);
(%o2)                               {1}
```