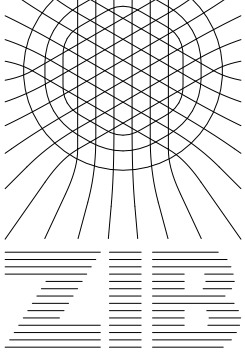
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-10711 Berlin - Wilmersdorf



Wolfram Koepf (Ed.)

REDUCE Packages on Power Series, Z-Transformation, Residues and Trigonometric Simplification

Preprint TR 95–3 (June 95)

REDUCE Packages on Power Series, Z-Transformation, Residues and Trigonometric Simplification

Wolfram Koepf
koepf@zib-berlin.de

Abstract

In this report, we present a collection of new REDUCE packages that recently have been developed. These are the packages FPS, ZTRANS, RESIDUE and TRIGSIMP on the following topics:

FPS Calculation of differential equations and formal power series representations, in particular for orthogonal polynomials and special functions of the hypergeometric type.

ZTRANS Calculation of the Z-Transformation and its inverse.

RESIDUE Calculation of residues of meromorphic functions.

TRIGSIMP Simplification of expressions involving trigonometric and hyperbolic functions.

For each of these packages, a description in form of a LaTeX file is distributed together with the package. These documentations are collected here.
In a final chapter, we show how to solve some difficult problems with these packages, and how they usefully can be combined.

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

Wolfram Koepf
REDUCE implementation by: Winfried Neun

1 Introduction

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

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

where 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, the 'symmetry number';
- **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 [4]. The implementations of this package for MAPLE (by D. Gruntz [2]) and MATHEMATICA (by W. Koepf [7]) served as guidelines for this one.
Numerous examples can be found in [5]–[6], most of which are contained in the test file `fps.tst`. Many more examples can be found in the extensive bibliography of Hansen [3].

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. For example, 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. Avoid the use of y as a variable since the output is given in terms of $y(x)$. 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 global 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., it is necessary to set `fps_search_depth:=6`.
The output of the FPS package can be influenced by the switch `tracefps`. Setting `on tracefps` causes various prints of intermediate results.

4 Orthogonal Polynomials and Special Functions

The package supports the work with the classical families of orthogonal polynomials, and other families of special functions. The orthogonal polynomials supported are given by `JacobiP(n,alpha,beta,x)`, `GegenbauerP(n,lambda,x)`, `LegendreP(n,x)`, `ChebyshevT(n,x)`, `ChebyshevU(n,x)`, `LaguerreL(n,x)`, `LaguerreL(n,alpha,x)`, and `HermiteP(n,x)`, and further the special functions `BesselJ(n,x)`, `BesselY(n,x)`, `BesselI(n,x)`, `BesselK(n,x)`, `Hankel1(n,x)`, `Hankel2(n,x)`, `KummerM(n,m,x)`, `KummerU(n,m,x)`, `WhittakerM(n,m,x)`, `WhittakerW(n,m,x)`, `LegendreP(n,m,x)`, `LegendreQ(n,m,x)`, `StruveH(n,x)`, and `StruveL(n,x)` (see [1]) are supported.
We get e. g.

```
10: FPS(LaguerreP(n,x),x);

          k
         x *pochhammer( - n,k)
infsum(-----------------------,k,0,infinity)
                          2
             factorial(k)

11: SimpleDE(ChebyshevT(n,x),x);

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

```
12: SimpleDE(BesselJ(n,x),x);

           2                          2    2
df(y,x,2)*x  + df(y,x)*x + y*( - n  + x )

13: SimpleDE(WhittakerW(n,m,x),x);

             2            2            2
4*df(y,x,2)*x  + y*( - 4*m  + 4*n*x - x  + 1)


14: SimpleDE(StruveH(n,x),x);

           3              2
df(y,x,3)*x  + df(y,x,2)*x *( - n + 2)

                  2        2        3    2      2    2
 + df(y,x)*x*( - n  - n + x ) + y*(n  + n  - n*x  + x )

15: SimpleDE(GegenbauerP(n+1,a,x)-GegenbauerP(n,a,x),x);

            2
df(y,x,2)*(x  - 1) + df(y,x)*(2*a*x - 1)

                         2
 + y*( - 2*a*n - 2*a - n  - n)
```

5 Problems in the Current Version

The handling of logarithmic singularities is not yet implemented.
The rational type implementation is not yet complete.
The calculation of power series representations of special functions is incomplete since the required limit calculations may fail.

References

[1] M. Abramowitz, and I. A. Stegun: *Handbook of Mathematical Functions.* Dover Publ., New York, 1964.

[2] D. Gruntz, W. Koepf, *Maple package on formal power series.* Maple Technical Newsletter, 1995, to appear.

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

[4] Wolfram Koepf, *Power Series in Computer Algebra*, J. Symbolic Computation 13, 1992, 581–603.

[5] Wolfram Koepf, *Examples for the algorithmic calculation of formal Puiseux, Laurent and power series*, SIGSAM Bulletin 27, 1993, 20–32.

[6] 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.

[7] Wolfram Koepf, *A package on formal power series.* Mathematica Journal 4, 1994, 62–69.

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

Z-Transform Package for REDUCE

Wolfram Koepf
Lisa Temme

1 Z-Transform

The Z-Transform of a sequence $\{f_n\}$ is the dicrete analogue of the Laplace Transform, and

$$\mathcal{Z}\{f_n\} = F(z) = \sum_{n=0}^{\infty} f_n z^{-n} .$$

This series converges in the region outside the circle $|z| = |z_0| = \limsup_{n\to\infty} \sqrt[n]{|f_n|}$.

SYNTAX: `ztrans(`f_n`, n, z)` where f_n is an expression, and n,z are identifiers.

2 Inverse Z-Transform

The calculation of the Laurent coefficients of a regular function results in the following inverse formula for the Z-Transform:
If $F(z)$ is a regular function in the region $|z| > \rho$ then $\exists$ a sequence $\{f_n\}$ with $\mathcal{Z}\{f_n\} = F(z)$ given by

$$f_n = \frac{1}{2\pi i} \oint F(z) z^{n-1} dz$$

SYNTAX: `invztrans(`$F(z)$`, z, n)` where $F(z)$ is an expression, and z,n are identifiers.

3 Input for the Z-Transform

This package can compute the Z-Transforms of the following list of f_n, and certain combinations thereof.

1	$e^{\alpha n}$	$\frac{1}{(n+k)}$
$\frac{1}{n!}$	$\frac{1}{(2n)!}$	$\frac{1}{(2n+1)!}$
$\frac{\sin(\beta n)}{n!}$	$\sin(\alpha n + \phi)$	$e^{\alpha n}\sin(\beta n)$
$\frac{\cos(\beta n)}{n!}$	$\cos(\alpha n + \phi)$	$e^{\alpha n}\cos(\beta n)$

$\frac{\sin(\beta(n+1))}{n+1}$ $\quad$ $\sinh(\alpha n+\phi)$ $\quad$ $\frac{\cos(\beta(n+1))}{n+1}$

$\cosh(\alpha n+\phi)$ $\quad$ $\binom{n+k}{m}$

Other Combinations

Linearity $\quad \mathcal{Z}\{af_n+bg_n\} = a\mathcal{Z}\{f_n\}+b\mathcal{Z}\{g_n\}$

Multiplication by n $\quad \mathcal{Z}\{n^k\cdot f_n\} = -z\frac{d}{dz}\left(\mathcal{Z}\{n^{k-1}\cdot f_n, n, z\}\right)$

Multiplication by λ^n $\quad \mathcal{Z}\{\lambda^n\cdot f_n\} = F\left(\frac{z}{\lambda}\right)$

Shift Equation $\quad \mathcal{Z}\{f_{n+k}\} = z^k\left(F(z)-\sum_{j=0}^{k-1} f_j z^{-j}\right)$

Symbolic Sums $\quad \mathcal{Z}\left\{\sum_{k=0}^{n} f_k\right\} = \frac{z}{z-1}\cdot\mathcal{Z}\{f_n\}$

$\mathcal{Z}\left\{\sum_{k=p}^{n+q} f_k\right\}$ $\quad$ combination of the above

where k,$\lambda \in \mathbf{N}-\{0\}$; and a,b are variables or fractions; and p,$q \in \mathbf{Z}$ or are functions of n; and α, β & ϕ are angles in radians.

4 Input for the Inverse Z-Transform

This package can compute the Inverse Z-Transforms of any rational function, whose denominator can be factored (over $\mathbf{Q}$), in addition to the following list of $F(z)$.

$\sin\left(\frac{\sin(\beta)}{z}\right)e^{\left(\frac{\cos(\beta)}{z}\right)}$ $\quad$ $\cos\left(\frac{\sin(\beta)}{z}\right)e^{\left(\frac{\cos(\beta)}{z}\right)}$

$\sqrt{\frac{z}{A}}\sin\left(\sqrt{\frac{z}{A}}\right)$ $\quad$ $\cos\left(\sqrt{\frac{z}{A}}\right)$

$\sqrt{\frac{z}{A}}\sinh\left(\sqrt{\frac{z}{A}}\right)$ $\quad$ $\cosh\left(\sqrt{\frac{z}{A}}\right)$

$z\ \log\left(\frac{z}{\sqrt{z^2-Az+B}}\right)$ $\quad$ $z\ \log\left(\frac{\sqrt{z^2+Az+B}}{z}\right)$

$\arctan\left(\frac{\sin(\beta)}{z+\cos(\beta)}\right)$

where k,$\lambda \in \mathbf{N}-\{0\}$ and A,B are fractions or variables ($B > 0$) and α,β, & ϕ are angles in radians.

5 Application of the Z-Transform

Solution of recurrence equations

In the same way that a Laplace Transform can be used to solve differential equations, so Z-Transforms can be used to solve recurrence equations.

Given a linear recurrence equation of k-th order

$$f_{n+k} + a_1 f_{n+k-1} + \ldots + a_k f_n = g_n \tag{1}$$

with initial conditions $f_0 = h_0$, $f_1 = h_1$, ..., $f_{k-1} = h_{k-1}$ (where h_j are given), it is possible to solve it in the following way. If the coefficients $a_1, \ldots, a_k$ are constants, then the Z-Transform of (1) can be calculated using the shift equation, and results in a solvable linear equation for $\mathcal{Z}\{f_n\}$. Application of the Inverse Z-Transform then results in the solution of (1).
If the coefficients $a_1, \ldots, a_k$ are polynomials in n then the Z-Transform of (1) constitutes a differential equation for $\mathcal{Z}\{f_n\}$. If this differential equation can be solved then the Inverse Z-Transform once again yields the solution of (1). Some examples of this method of solution can be found in §6.

6 EXAMPLES

Here are some examples for the Z-Transform

```
1: ztrans((-1)^n*n^2,n,z);

    z*( - z + 1)
--------------------
  3      2
 z  + 3*z  + 3*z + 1

2: ztrans(cos(n*omega*t),n,z);

   z*(cos(omega*t) - z)
--------------------------
                      2
 2*cos(omega*t)*z - z  - 1

3: ztrans(cos(b*(n+2))/(n+2),n,z);
```

```
                                    z
z*( - cos(b) + log(------------------------------)*z)
                                          2
                        sqrt( - 2*cos(b)*z + z  + 1)

4: ztrans(n*cos(b*n)/factorial(n),n,z);

  cos(b)/z       sin(b)                  sin(b)
 e        *(cos(--------)*cos(b) - sin(--------)*sin(b))
                   z                       z
-------------------------------------------------------
                                z
5: ztrans(sum(1/factorial(k),k,0,n),n,z);

  1/z
 e   *z
--------
 z - 1

6: operator f$

7: ztrans((1+n)^2*f(n),n,z);

                              2
df(ztrans(f(n),n,z),z,2)*z  - df(ztrans(f(n),n,z),z)*z
+ ztrans(f(n),n,z)
```

Here are some examples for the Inverse Z-Transform

```
8: invztrans((z^2-2*z)/(z^2-4*z+1),z,n);

              n       n                 n
 (sqrt(3) - 2) *( - 1)  + (sqrt(3) + 2)
-----------------------------------------
                    2

9: invztrans(z/((z-a)*(z-b)),z,n);

  n    n
 a  - b
---------
  a - b
```

```
10: invztrans(z/((z-a)*(z-b)*(z-c)),z,n);

  n      n      n      n      n      n
 a *b - a *c - b *a + b *c + c *a - c *b
----------------------------------------
  2      2        2      2    2        2
 a *b - a *c - a*b  + a*c  + b *c - b*c

11: invztrans(z*log(z/(z-a)),z,n);

  n
 a *a
-------
 n + 1

12: invztrans(e^(1/(a*z)),z,n);

        1
-----------------
  n
 a *factorial(n)

13: invztrans(z*(z-cosh(a))/(z^2-2*z*cosh(a)+1),z,n);

cosh(a*n)
```

Examples: Solutions of Recurrence Equations

I (See [2], p. 651, Example 1).
Consider the homogeneous linear recurrence equation

$$f_{n+5} - 2f_{n+3} + 2f_{n+2} - 3f_{n+1} + 2f_n = 0$$

with initial conditions $f_0 = 0$, $f_1 = 0$, $f_2 = 9$, $f_3 = -2$, $f_4 = 23$. The Z-Transform of the left hand side can be written as $F(z) = P(z)/Q(z)$ where $P(z) = 9z^3 - 2z^2 + 5z$ and $Q(z) = z^5 - 2z^3 + 2z^2 - 3z + 2 = (z-1)^2(z+2)(z^2+1)$, which can be inverted to give

$$f_n = 2n + (-2)^n - \cos\tfrac{\pi}{2}n \ .$$

The following REDUCE session shows how the present package can be used to solve the above problem.

```
14: operator f$ f(0):=0$ f(1):=0$ f(2):=9$ f(3):=-2$ f(4):=23$


20: equation:=ztrans(f(n+5)-2*f(n+3)+2*f(n+2)-3*f(n+1)+2*f(n),n,z);

                                  5                          3
equation := ztrans(f(n),n,z)*z  - 2*ztrans(f(n),n,z)*z

                                     2
            + 2*ztrans(f(n),n,z)*z  - 3*ztrans(f(n),n,z)*z

                                       3      2
            + 2*ztrans(f(n),n,z) - 9*z  + 2*z  - 5*z


21: ztransresult:=solve(equation,ztrans(f(n),n,z));

                                              2
                                       z*(9*z  - 2*z + 5)
ztransresult := {ztrans(f(n),n,z)=-----------------------------}
                                     5      3      2
                                    z  - 2*z  + 2*z  - 3*z + 2

22: result:=invztrans(part(first(ztransresult),2),z,n);

                    n    n        n    n
          2*( - 2)  - i *( - 1)  - i  + 4*n
result := ----------------------------------
                         2
```

II (See [2], p. 651, Example 2).
Consider the inhomogeneous recurrence equation:

$$f_{n+2} - 4f_{n+1} + 3f_n = 1$$

with initial conditions $f_0 = 0$, $f_1 = 1$. Giving

$$\begin{aligned} F(z) &= \mathcal{Z}\{1\}\left(\tfrac{1}{z^2-4z+3} + \tfrac{z}{z^2-4z+3}\right) \\ &= \tfrac{z}{z-1}\left(\tfrac{1}{z^2-4z+3} + \tfrac{z}{z^2-4z+3}\right). \end{aligned}$$

The Inverse Z-Transform results in the solution

$$f_n = \frac{1}{2}\left(\frac{3^{n+1}-1}{2} - (n+1)\right).$$

The following REDUCE session shows how the present package can be used to solve the above problem.

```
23: clear(f)$ operator f$ f(0):=0$ f(1):=1$


27: equation:=ztrans(f(n+2)-4*f(n+1)+3*f(n)-1,n,z);

                                   3                           2
equation := (ztrans(f(n),n,z)*z  - 5*ztrans(f(n),n,z)*z

                                                                  2
    + 7*ztrans(f(n),n,z)*z - 3*ztrans(f(n),n,z) - z )/(z - 1)

28: ztransresult:=solve(equation,ztrans(f(n),n,z));

                                        2
                                       z
result := {ztrans(f(n),n,z)=----------------------}
                                3      2
                               z  - 5*z  + 7*z - 3

29: result:=invztrans(part(first(ztransresult),2),z,n);

                n
           3*3  - 2*n - 3
result := ----------------
                  4
```

III Consider the following recurrence equation, which has a differential equation for $\mathcal{Z}\{f_n\}$.

$$(n+1)\cdot f_{n+1} - f_n = 0$$

with initial conditions $f_0 = 1$, $f_1 = 1$. It can be solved in REDUCE using the present package in the following way.

```
30: clear(f)$ operator f$ f(0):=1$ f(1):=1$


34: equation:=ztrans((n+1)*f(n+1)-f(n),n,z);

                                          2
equation :=  - (df(ztrans(f(n),n,z),z)*z  + ztrans(f(n),n,z))

35: operator tmp;

36: equation:=sub(ztrans(f(n),n,z)=tmp(z),equation);

                                 2
equation :=  - (df(tmp(z),z)*z  + tmp(z))

37: load(odesolve);

38: ztransresult:=odesolve(equation,tmp(z),z);

                           1/z
ztransresult := {tmp(z)=e   *arbconst(1)}

39: preresult:=invztrans(part(first(ztransresult),2),z,n);

              arbconst(1)
preresult := --------------
              factorial(n)

40: solve({sub(n=0,preresult)=f(0),sub(n=1,preresult)=f(1)},
arbconst(1));

{arbconst(1)=1}

41: result:=preresult where ws;

                1
result := --------------
           factorial(n)
```

References

[1] Bronstein, I.N. and Semedjajew, K.A., *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt (Main), 1981.

RESIDUE Package for REDUCE

Wolfram Koepf

This package supports the calculation of residues. The residue $\operatorname*{Res}_{z=a} f(z)$ of a function $f(z)$ (that is analytic in a punctured neighborhood of $z = a$) at the point $a \in \mathbb{C}$ is defined as [2]

$$\operatorname*{Res}_{z=a} f(z) = \frac{1}{2\pi i} \oint f(z)\, dz \ ,$$

with integration along a closed curve around $z = a$ with winding number 1.
If $f(z)$ is given by a Laurent series development at $z = a$

$$f(z) = \sum_{k=-\infty}^{\infty} a_k \, (z-a)^k \ ,$$

then

$$\operatorname*{Res}_{z=a} f(z) = a_{-1} \ . \tag{2}$$

If $a = \infty$, one defines on the other hand

$$\operatorname*{Res}_{z=\infty} f(z) = -a_{-1} \tag{3}$$

for given Laurent representation

$$f(z) = \sum_{k=-\infty}^{\infty} a_k \, \frac{1}{z^k} \ .$$

The package is loaded by the statement

```
1: load residue;
```

It contains two REDUCE operators:

- `residue(f,z,a)` determines the residue of f at the point $z = a$ if f is meromorphic at $z = a$. The calculation of residues at essential singularities of f is not supported.
- `poleorder(f,z,a)` determines the pole order of f at the point $z = a$ if f is meromorphic at $z = a$.

Note that both functions use the `taylor` package in connection with representations (2)–(3). Here are some examples:

```
2: residue(x/(x^2-2),x,sqrt(2));

 1
---
 2
```

```
3: poleorder(x/(x^2-2),x,sqrt(2));

1

4: residue(sin(x)/(x^2-2),x,sqrt(2));

 sqrt(2)*sin(sqrt(2))
----------------------
          4

5: residue(1/(x-1)^m/(x-2)^2,x,2);

 - m

6: poleorder(1/(x-1)/(x-2)^2,x,2);

2

7: residue(sin(x)/x^2,x,0);

1

8: residue((1+x^2)/(1-x^2),x,1);

-1

9: residue((1+x^2)/(1-x^2),x,-1);

1

10: residue(tan(x),x,pi/2);

-1

11: poleorder(tan(x),x,pi/2);

1

12: residue((x^n-y^n)/(x-y),x,y);

0

13: poleorder((x^n-y^n)/(x-y),x,y);
```

```
0

14: residue((x^n-y^n)/(x-y)^2,x,y);

  n
 y *n
------
  y

15: residue(tan(x)/sec(x-pi/2)+1/cos(x),x,pi/2);

-2

16: poleorder(tan(x)/sec(x-pi/2)+1/cos(x),x,pi/2);

1

17: for k:=1:2 sum residue((a+b*x+c*x^2)/(d+e*x+f*x^2),x,
    part(part(solve(d+e*x+f*x^2,x),k),2));

 b*f - c*e
-----------
     2
    f

18: residue(x^3/sin(1/x)^2,x,infinity);

  - 1
------
  15

19: residue(x^3*sin(1/x)^2,x,infinity);

-1
```

Note that the residues of factorial and Γ function terms are not yet supported.

References

[1] Bronstein, I.N. and Semedjajew, K.A., *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt (Main), 1981.

TRIGSIMP

A REDUCE Package for the Simplification and Factorization of Trigonometric and Hyperbolic Functions

Wolfram Koepf
Andreas Bernig
Herbert Melenk

1 Introduction

The REDUCE package TRIGSIMP is a useful tool for all kinds of trigonometric and hyperbolic simplification and factorization. There are three procedures included in TRIGSIMP: trigsimp, trigfactorize and triggcd. The first is for finding simplifications of trigonometric or hyperbolic expressions with many options, the second for factorizing them and the third for finding the greatest common divisor of two trigonometric or hyperbolic polynomials.
To start the package it must be loaded by:

```
1: load trigsimp;
```

2 REDUCE operator `trigsimp`

As there is no normal form for trigonometric and hyperbolic functions, the same function can convert in many different directions, e.g. $\sin(2x) \leftrightarrow 2\sin(x)\cos(x)$. The user has the possibility to give several parameters to the procedure `trigsimp` in order to influence the direction of transformations. The decision whether a rational expression in trigonometric and hyperbolic functions vanishes or not is possible using the (default) `expand` direction.
To simplify a function `f`, one uses `trigsimp(f[,options])`. Example:

```
2: trigsimp(sin(x)^2+cos(x)^2);
```

```
1
```

Possible options are (* denotes the default):

1. `sin` (*) or `cos`
2. `sinh` (*) or `cosh`
3. `expand` (*) or `combine` or `compact`
4. `hyp` or `trig` or `expon`
5. `keepalltrig`

From each group one can use at most one option, otherwise an error message will occur. The first group fixes the preference used while transforming a trigonometric expression:

```
3: trigsimp(sin(x)^2);

      2
sin(x)

4: trigsimp(sin(x)^2,cos);

         2
 - cos(x)  + 1
```

The second group is the equivalent for the hyperbolic functions. The third group determines the type of transformations. With the default `expand`, an expression is written in a form only using single arguments and no sums of arguments:

```
5: trigsimp(sin(2x+y));

                                    2
2*cos(x)*cos(y)*sin(x) - 2*sin(x) *sin(y) + sin(y)
```

With `combine`, products of trigonometric functions are transformed to trigonometric functions involving sums of arguments:

```
6: trigsimp(sin(x)*cos(y),combine);


 sin(x - y) + sin(x + y)
------------------------
            2
```

With `compact`, the REDUCE operator `compact` [2] is applied to `f`. This leads often to a simple form, but in contrast to `expand` one doesn't get a normal form. Example for `compact`:

```
7: trigsimp((1-sin(x)^2)^20*(1-cos(x)^2)^20,compact);

      40       40
cos(x)  *sin(x)
```

With the fourth group each expression is transformed to a trigonometric, hyperbolic or exponential form:

```
8: trigsimp(sin(x),hyp);

 - sinh(i*x)*i

9: trigsimp(sinh(x),expon);
```

```
  2*x
 e    - 1
----------
      x
   2*e

10: trigsimp(e^x,trig);

     x          x
cos(---) + sin(---)*i
     i          i
```

Usually, `tan`, `cot`, `sec`, `csc` are expressed in terms of `sin` and `cos`. It can be sometimes useful to avoid this, which is handled by the option `keepalltrig`:

```
11: trigsimp(tan(x+y),keepalltrig);

  - (tan(x) + tan(y))
----------------------
  tan(x)*tan(y) - 1
```

It is possible to use the options of different groups simultaneously:

```
12: trigsimp(sin(x)^4,cos,combine);

 cos(4*x) - 4*cos(2*x) + 3
---------------------------
            8
```

Sometimes, it is necessary to handle an expression in different steps:

```
13: trigsimp((sinh(x)+cosh(x))^n+(cosh(x)-sinh(x))^n,expon);

  2*n*x
 e      + 1
------------
     n*x
    e

14: trigsimp(ws,hyp);

2*cosh(n*x)

15: trigsimp((cosh(a*n)*sinh(a)*sinh(p)+cosh(a)*sinh(a*n)*sinh(p)+
    sinh(a - p)*sinh(a*n))/sinh(a));

cosh(a*n)*sinh(p) + cosh(p)*sinh(a*n)

16: trigsimp(ws,combine);

sinh(a*n + p)
```

3 REDUCE operator trigfactorize

With `trigfactorize(p,x)` one can factorize the trigonometric or hyperbolic polynomial `p` with respect to the argument x. Example:

```
17: trigfactorize(sin(x),x/2);

        x        x
{2,cos(---),sin(---)}
        2        2
```

If the polynomial is not coordinated or balanced [1], the output will equal the input. In this case, changing the value of the second argument can help to find a factorization:

```
18: trigfactorize(1+cos(x),x);

{cos(x) + 1}

19: trigfactorize(1+cos(x),x/2);

        x        x
{2,cos(---),cos(---)}
        2        2
```

The polynomial can consist of both trigonometric and hyperbolic functions:

```
20: trigfactorize(sin(2x)*sinh(2x),x);

{4, cos(x), sin(x), cosh(x), sinh(x)}
```

4 REDUCE operator triggcd

The operator `triggcd` is an application of `trigfactorize`. With its help the user can find the greatest common divisor of two trigonometric or hyperbolic polynomials. It uses the method described in [1]. The syntax is: `triggcd(p,q,x)`, where p and q are the polynomials and x is the smallest unit to use. Example:

```
21: triggcd(sin(x),1+cos(x),x/2);

     x
cos(---)
     2

22: triggcd(sin(x),1+cos(x),x);

1
```

The polynomials p and q can consist of both trigonometric and hyperbolic functions:

```
23: triggcd(sin(2x)*sinh(2x),(1-cos(2x))*(1+cosh(2x)),x);

cosh(x)*sin(x)
```

5 Further Examples

With the help of the package the user can create identities:

```
24: trigsimp(tan(x)*tan(y));

 sin(x)*sin(y)
---------------
 cos(x)*cos(y)

25: trigsimp(ws,combine);

 cos(x - y) - cos(x + y)
-------------------------
 cos(x - y) + cos(x + y)

26: trigsimp((sin(x-a)+sin(x+a))/(cos(x-a)+cos(x+a)));

 sin(x)
--------
 cos(x)

27: trigsimp(cosh(n*acosh(x))-cos(n*acos(x)),trig);

0

28: trigsimp(sec(a-b),keepalltrig);

  csc(a)*csc(b)*sec(a)*sec(b)
------------------------------
 csc(a)*csc(b) + sec(a)*sec(b)

29: trigsimp(tan(a+b),keepalltrig);

  - (tan(a) + tan(b))
----------------------
  tan(a)*tan(b) - 1

30: trigsimp(ws,keepalltrig,combine);

tan(a + b)
```

Some difficult expressions can be simplified:

```
31: df(sqrt(1+cos(x)),x,4);

                               4            2       2            2
(sqrt(cos(x) + 1)*( - 4*cos(x)  - 20*cos(x) *sin(x)  + 12*cos(x)

                      2                         4           2
     - 4*cos(x)*sin(x)  + 8*cos(x) - 15*sin(x)  + 16*sin(x) ))/(16

          4           3           2
   *(cos(x)  + 4*cos(x)  + 6*cos(x)  + 4*cos(x) + 1))

32: trigsimp(ws);

 sqrt(cos(x) + 1)
------------------
        16

33: load taylor;

34: taylor(sin(x+a)*cos(x+b),x,0,4);

cos(b)*sin(a) + (cos(a)*cos(b) - sin(a)*sin(b))*x

                                     2
 - (cos(a)*sin(b) + cos(b)*sin(a))*x

    2*( - cos(a)*cos(b) + sin(a)*sin(b))   3
 + ------------------------------------*x
                    3

    cos(a)*sin(b) + cos(b)*sin(a)   4      5
 + -----------------------------*x  + O(x )
                 3

35: trigsimp(ws,combine);

sin(a - b) + sin(a + b)                                  2    2*cos(a + b)   3
------------------------ + cos(a + b)*x - sin(a + b)*x  - --------------*x
           2                                                      3

    sin(a + b)   4      5
 + ------------*x  + O(x )
        3
```

Certain integrals whose calculation was not possible in REDUCE (without preprocessing), are now computable:

```
36: int(trigsimp(sin(x+y)*cos(x-y)*tan(x)),x);

       2                                                          2
 cos(x) *x - cos(x)*sin(x) - 2*cos(y)*log(cos(x))*sin(y) + sin(x) *x
-------------------------------------------------------------------
                                 2

37: int(trigsimp(sin(x+y)*cos(x-y)/tan(x)),x);

                                    x  2
(cos(x)*sin(x) - 2*cos(y)*log(tan(---)  + 1)*sin(y)
                                    2

                      x
  + 2*cos(y)*log(tan(---))*sin(y) + x)/2
                      2
```

Without the package, the integration fails, in the second case one doesn't receive an answer for many hours.

```
38: trigfactorize(sin(2x)*cos(y)^2,y/2);

{2*cos(x)*sin(x),

      y          y
 cos(---) + sin(---),
      2          2

      y          y
 cos(---) + sin(---),
      2          2

      y          y
 cos(---) - sin(---),
      2          2

      y          y
 cos(---) - sin(---)}
      2          2

39: trigfactorize(sin(y)^4-x^2,y);

          2              2
{ - sin(y)  + x, - (sin(y)  + x)}

40: trigfactorize(sin(x)*sinh(x),x/2);
```

```
        x        x         x         x
{4,cos(---),sin(---),cosh(---),sinh(---)}
        2        2         2         2

41: triggcd(-5+cos(2x)-6sin(x),-7+cos(2x)-8sin(x),x/2);

       x        x
2*cos(---)*sin(---) + 1
       2        2

42: triggcd(1-2cosh(x)+cosh(2x),1+2cosh(x)+cosh(2x),x/2);

        x  2
2*sinh(---)  + 1
        2
```

References

[1] Roach, Kelly: Difficulties with Trigonometrics. Notes of a talk.

[2] Hearn, A.C.: COMPACT User Manual.

Some Difficult Problems and
How these Packages can be Combined

1 Complicated Ztransformations

One might try to calculate the following inverse Z-Transformation according to the Table 4.4.4 in [1]:

```
1: load ztrans;

2: invztrans(z*(z*sinh(p)+sinh(a-p))/(z^2-2*z*cosh(a)+1),z,n);

 cosh(a*n)*sinh(a)*sinh(p) + cosh(a)*sinh(a*n)*sinh(p) + sinh(a - p)*sinh(a*n)
------------------------------------------------------------------------------
                                    sinh(a)
```

Be aware, that the result doesn't look like in the Table 4.4.4. A trigonometric simplication might be necessary. Therefore:

```
3: load trigsimp;

4: trigsimp(ws);

cosh(a*n)*sinh(p) + cosh(p)*sinh(a*n)
```

We see that the result looks better now. In particular, the package realized the common factor in numerator and denominator, and cancelled it. Finally, we may like to combine:

```
5: trigsimp(ws,combine);

sinh(a*n + p)
```

and therefore have the result according to Table 4.4.4. Another example is given by

```
6: invztrans(z*(z^2-1)*sin(b)/(z^2-2*z*cos(b)+1)^2,z,n);

           2
(sqrt(cos(b)  - 1)*sin(b)*n

                           2      n                          2      n
 *( - (cos(b) - sqrt(cos(b)  - 1))  + (cos(b) + sqrt(cos(b)  - 1)) ))/(2

          2
   *(cos(b)  - 1))

7: trigsimp(ws,expon);
```

```
          2*b*i*n
 i*n*( - e        + 1)
----------------------
          b*i*n
       2*e

8: trigsimp(ws,trig);

sin(b*n)*n
```

Here, an intermediate conversion in exponential notation was necessary to obtain a normal form since powers were involved.

2 The Hofstadter Problem

Recently, Hofstadter introduced a new geometric problem on plane triangles, and was able to find an infinite number of collinear point triples. His result is equivalent to the determinant statement

$$\begin{vmatrix} \dfrac{\sin(r\alpha)}{\sin((1-r)\alpha)} & \dfrac{\sin(r\beta)}{\sin((1-r)\beta)} & \dfrac{\sin(r\gamma)}{\sin((1-r)\gamma)} \\ \dfrac{\sin(2\alpha)}{\sin(-\alpha)} & \dfrac{\sin(2\beta)}{\sin(-\beta)} & \dfrac{\sin(2\gamma)}{\sin(-\gamma)} \\ \dfrac{\sin((2-r)\alpha)}{\sin(-(1-r)\alpha)} & \dfrac{\sin((2-r)\beta)}{\sin(-(1-r)\beta)} & \dfrac{\sin((2-r)\gamma)}{\sin(-(1-r)\gamma)} \end{vmatrix} = 0$$

for $\gamma = \pi - \alpha - \beta$. Let's deduce this result with REDUCE:

```
9: procedure term1(r); sin(r*alpha)/sin((1-r)*alpha)$

10: procedure term2(r); sin(r*beta)/sin((1-r)*beta)$

11: procedure term3(r); sin(r*(pi-alpha-beta))/sin((1-r)*(pi-alpha-beta))$

12: term:=det mat((term1(r),term2(r),term3(r)),(term1(2),term2(2),term3(2)),
    (term1(2-r),term2(2-r),term3(2-r)));

term := ( - sin(alpha*r - 2*alpha + beta*r - 2*beta - pi*r + 2*pi)

         *sin(alpha*r - alpha)*sin(alpha + beta)*sin(beta*r)*sin(2*alpha)

         *sin(beta) + sin(alpha*r - 2*alpha + beta*r - 2*beta - pi*r + 2*pi)

         *sin(beta*r - beta)*sin(alpha + beta)*sin(alpha*r)*sin(2*beta)

         *sin(alpha) - sin(alpha*r - 2*alpha)

         *sin(alpha*r - alpha + beta*r - beta - pi*r + pi)*sin(2*alpha + 2*beta)
```

```
         *sin(beta*r)*sin(alpha)*sin(beta) - sin(alpha*r - 2*alpha)

         *sin(alpha*r + beta*r - pi*r)*sin(beta*r - beta)*sin(alpha + beta)

         *sin(2*beta)*sin(alpha) +

         sin(alpha*r - alpha + beta*r - beta - pi*r + pi)*sin(beta*r - 2*beta)

         *sin(2*alpha + 2*beta)*sin(alpha*r)*sin(alpha)*sin(beta) +

         sin(alpha*r - alpha)*sin(alpha*r + beta*r - pi*r)*sin(beta*r - 2*beta)

         *sin(alpha + beta)*sin(2*alpha)*sin(beta))/(

           sin(alpha*r - alpha + beta*r - beta - pi*r + pi)*sin(alpha*r - alpha)

           *sin(beta*r - beta)*sin(alpha + beta)*sin(alpha)*sin(beta))
```

```
13: trigsimp(term);
```

```
0
```

3 Polynomial Simplification of Trigonometrics

Whereas the following list is not simplified by REDUCE

```
14: for k:=1:5 collect cos(i*k^2*acosh(x));
```

```
{cos(acosh(x)*i),

 cos(4*acosh(x)*i),

 cos(9*acosh(x)*i),

 cos(16*acosh(x)*i),

 cos(25*acosh(x)*i)}
```

with `trigsimp`, we get

```
15: for k:=1:5 collect trigsimp(cos(i*k^2*acosh(x)),trig);
```

```
{x,

    4      2
 8*x  - 8*x  + 1,
```

```
        8        6        4        2
x*(256*x  - 576*x  + 432*x  - 120*x  + 9),

       16           14           12           10          8          6
32768*x   - 131072*x   + 212992*x   - 180224*x   + 84480*x  - 21504*x

         4        2
 + 2688*x  - 128*x  + 1,

             24              22              20              18              16
x*(16777216*x   - 104857600*x   + 288358400*x   - 458752000*x   + 466944000*x

                 14              12             10            8           6
    - 317521920*x   + 146227200*x   - 45260800*x   + 9152000*x  - 1144000*x

             4         2
    + 80080*x  - 2600*x  + 25)}
```

Note that trigonometric expansion is implemented by a Divide-and-Conquer approach which makes it possible to expand for example $\sin(500x)$ without memory problems. Note that furthermore the input can be recovered by combining. However, since collected trigonometric expressions do not constitute a normal form, we cannot be sure to construct the original term. We have for example[1]

```
16: trigsimp(sin(5*x+6*y)*sin(8*w+9*x),expand)$

17: trigsimp(ws,combine);

  - cos(8*w + 14*x + 6*y) + cos(8*w + 4*x - 6*y)
-------------------------------------------------
                        2

18: trigsimp(ws-sin(5*x+6*y)*sin(8*w+9*x),combine);

0
```

4 Residue Calculation

In [4], we studied the Bateman functions $F_k(t)$, and developed the residue representation

$$F_k(t) = 2i \operatorname*{Res}_{z=i} \left(e^{itz} \frac{(z+i)^{k-1}}{(z-i)^{k+1}} \right) .$$

With the `residue` package, we easily can calculate the first six Bateman functions using this representation:

[1] We don't reproduce the huge output after expansion here.

```
19: load residue;

20: for k:=0:5 collect 2*i*residue(exp(i*t*z)*(z+i)^(k-1)/(z-i)^(k+1),z,i);

  1
{----,
   t
  e

   - 2*t
 --------,
     t
    e

  2*t*(t - 1)
 -------------,
       t
      e

             2
  2*t*( - 2*t  + 6*t - 3)
 ------------------------,
              t
           3*e
        3      2
  2*t*(t  - 6*t  + 9*t - 3)
 --------------------------,
               t
            3*e
             4       3       2
  2*t*( - 2*t  + 20*t  - 60*t  + 60*t - 15)
 ------------------------------------------}
                       t
                    15*e
```

We can compare this result with the representation $F_k(t) = e^{-t}\, L_k^{(-1)}(2t)$ ([4], (12)) in terms of the generalized Laguerre polynomials

```
21: for k:=0:5 collect exp(-t)*LaguerreP(k,-1,2*t);

  1
{----,
   t
  e

   - 2*t
 --------,
     t
    e
```

```
 2*t*(t - 1)
------------,
      t
     e
            2
 2*t*( - 2*t  + 6*t - 3)
------------------------,
             t
          3*e
       3      2
 2*t*(t  - 6*t  + 9*t - 3)
--------------------------,
              t
           3*e

             4       3       2
 2*t*( - 2*t  + 20*t  - 60*t  + 60*t - 15)
-----------------------------------------}
                      t
                  15*e
```

5 Identities for Orthogonal Polynomials

To prove the identities (see [1], (22.5.33)–(22.5.34))

$$T_n(x) = \frac{n}{2} C_n^{(0)}(x)$$

and

$$U_n(x) = C_n^{(1)}(x)$$

between the Chebyshev and Gegenbauer polynomials, for example, it is enough to show that the left and right hand sides satisfy the same differential equations, and have the same initial values. This is done by the calculations

```
22: load fps;

23: SimpleDE(ChebyshevT(n,x),x);

            2                       2
df(y,x,2)*(x  - 1) + df(y,x)*x - n *y

24: SimpleDE(n/2*GegenbauerP(n,0,x),x);

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

```
25: sub(x=0,ChebyshevT(n,x)-n/2*GegenbauerP(n,0,x));

0

26: sub(x=0,df(ChebyshevT(n,x)-n/2*GegenbauerP(n,0,x),x));

0

27: SimpleDE(ChebyshevU(n,x),x);

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

28: SimpleDE(GegenbauerP(n,1,x),x);

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

29: load zeilberg;

30: sub(x=0,ChebyshevU(n,x)-GegenbauerP(n,1,x));

            n        n*pi          n*pi          n + 2
 factorial(---)*cos(------) - cos(------)*gamma(-------)
            2         2             2              2
-------------------------------------------------------
                              n
                    factorial(---)
                              2

31: simplify_combinatorial(ws);

0

32: sub(x=0,df(ChebyshevU(n,x)-GegenbauerP(n,1,x),x));

            n - 1        n*pi - pi
(factorial(-------)*cos(-----------)*(n + 1)
              2             2

         n*pi - pi          n + 1                        n - 1
  - cos(-----------)*gamma(-------)*(n + 1))/factorial(-------)
            2               2                            2

33: simplify_combinatorial(ws);

0
```

For the simplification of factorial and Γ terms, we have loaded the package `ZEILBERG` which is described elsewhere [3].

References

[1] M. Abramowitz, and I. A. Stegun: *Handbook of Mathematical Functions.* Dover Publ., New York, 1964.

[2] Bronstein, I.N. and Semedjajew, K.A., *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt (Main), 1981.

[3] W. Koepf: REDUCE package for indefinite and definite summation. Konrad-Zuse-Zentrum Berlin (ZIB), Technical Report TR 94-9, 1994, SIGSAM Bulletin **29**, 1995, 14–30.

[4] W. Koepf, and D. Schmersau: Bounded nonvanishing functions and Bateman functions. Complex Variables **25**, 1994, 237–259.