`orthpoly` — library for orthogonal polynomials

Table of contents

Introduction

The `orthpoly` package provides some standard orthogonal polynomials.

The package functions are called using the package name `orthpoly` and the name of the function. E.g., use

```
>> orthpoly::legendre(5, x)
```

to generate the fifth degree Legendre polynomial in the indeterminate `x`. This mechanism avoids naming conflicts with other library functions. If this is found to be inconvenient, then the routines of the `orthpoly` package may be exported via `export`. E.g., after calling

```
>> export(orthpoly, legendre)
```

the function `orthpoly::legendre` may be called directly:

```
>> legendre(5, x)
```

All routines of the `orthpoly` package are exported simultaneously by

```
>> export(orthpoly)
```

If the identifier `legendre` already has a value, then `export` returns a warning and does not export `orthpoly::legendre`. The value of the identifier `legendre` must be deleted before it can be exported succesfully from the `orthpoly` package.

`orthpoly::chebyshev1` – the Chebyshev polynomials of the first kind

`orthpoly::chebyshev1(n,x)` computes the value of the n-th degree Chebyshev polynomial of the first kind at the point x.

Call(s):

- `orthpoly::chebyshev1(n, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Chebyshev polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::chebyshev1` returns itself symbolically.

Related Functions: `orthpoly::chebyshev2`, `orthpoly::jacobi`

Details:

- These polynomials have integer coefficients.
- Evaluation is fast and numerically stable for real floating point values x from the interval $[-1.0, 1.0]$. Cf. example 2.
- `orthpoly::chebyshev2` implements the Chebyshev polynomials of the second kind.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::chebyshev1(2, x)

                              2
                     poly(2 x  - 1, [x])

>> orthpoly::chebyshev1(3, x[1])

                         3
               poly(4 x[1]  - 3 x[1], [x[1]])
```

However, using arithmetical expressions as input the “values” of these polynomials are returned:

```
>> orthpoly::chebyshev1(2, 6*x)

                                   2
                               72 x  - 1

>> orthpoly::chebyshev1(3, x[1] + 2)

                                                2
           2 (x[1] + 2) (2 (x[1] + 2)  - 1) - x[1] - 2
```

“Arithmetical expressions” include numbers:

```
>> orthpoly::chebyshev1(2, sqrt(2)),
   orthpoly::chebyshev1(3, 8 + I),
   orthpoly::chebyshev1(1000, 0.3)

                  3, 1928 + 761 I, -0.9991251117
```

If no integer degree is specified, `orthpoly::chebyshev1` returns itself symbolically:

```
>> orthpoly::chebyshev1(n, x), orthpoly::chebyshev1(1/2, x)

    orthpoly::chebyshev1(n, x), orthpoly::chebyshev1(1/2, x)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::chebyshev1(200, 0.3)

                              -0.316963268
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> T200 := orthpoly::chebyshev1(200, x):
>> DIGITS := 10: evalp(T200, x = 0.3)

                               56068.79149
```

This result is caused by numerical round-off. Also with increased `DIGITS` only a few leading digits are correct:

```
>> DIGITS := 20: evalp(T200, x = 0.3)

                         -0.31701395850025751393

>> delete DIGITS, T200:
```

Background:

- The Chebyshev polynomials are given by $T(n,x) = \cos(n \arccos(x))$ for real $x \in [-1,1]$. This representation is used by `orthpoly::chebyshev1` for floating point values in this range.

- These polynomials satisfy the recursion formula

$$T(n,x) = 2\,x\,T(n-1,x) - T(n-2,x)$$

  with $T(0,x) = 1$ and $T(1,x) = x$.

- They are orthogonal on the interval $[-1,1]$ with respect to the weight function $w(x) = (1-x^2)^{-1/2}$.

- $T(n,x)$ is a special Jacobi polynomial:

$$T(n,x) = \frac{2^{2n}\,(n!)^2}{(2n)!}\,P\Big(n, -\frac{1}{2}, -\frac{1}{2}, x\Big)\ .$$

Changes:

- Indexed identifiers are now regarded as indeterminates. Non-integer degrees n are now accepted, leading to an unevaluated function call. Floating point evaluations are now numerically stable. Efficiency was improved.

`orthpoly::chebyshev2` – the Chebyshev polynomials of the second kind

`orthpoly::chebyshev2(n,x)` computes the value of the n-th degree Chebyshev polynomial of the second kind at the point x.

Call(s):

- `orthpoly::chebyshev2(n, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Chebyshev polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::chebyshev2` returns itself symbolically.

Related Functions: `orthpoly::chebyshev1`, `orthpoly::gegenbauer`, `orthpoly::jacobi`

Details:

- These polynomials have integer coefficients.
- Evaluation is fast and numerically stable for real floating point values x from the interval $[-1.0, 1.0]$. Cf. example 2.
- `orthpoly::chebyshev1` implements the Chebyshev polynomials of the first kind.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::chebyshev2(2, x)

                                 2
                       poly(4 x   - 1, [x])
```

```
>> orthpoly::chebyshev2(3, x[1])

                              3
                 poly(8 x[1]   - 4 x[1], [x[1]])
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::chebyshev2(2, 6*x)

                                   2
                           144 x   - 1
```

```
>> orthpoly::chebyshev2(3, x[1] + 2)

                                                   2
               2 (2 x[1] + 4) (2 (x[1] + 2)   - 1)
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::chebyshev2(2, sqrt(2)),
   orthpoly::chebyshev2(3, 8 + I),
   orthpoly::chebyshev2(1000, 0.3)

                 7, 3872 + 1524 I, -1.012277265
```

If no integer degree is specified, then `orthpoly::chebyshev2` returns itself symbolically:

```
>> orthpoly::chebyshev2(n, x), orthpoly::chebyshev2(1/2, x)

   orthpoly::chebyshev2(n, x), orthpoly::chebyshev2(1/2, x)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::chebyshev2(200, 0.3)

                    -0.01869337443
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> U200 := orthpoly::chebyshev2(200, x):
>> DIGITS := 10: evalp(U200, x = 0.3)

                    437298.8655
```

This result is caused by numerical round-off. Also with increased `DIGITS` only a few leading digits are correct:

```
>> DIGITS := 20: evalp(U200, x = 0.3)

                    -0.018650101492067216121

>> delete DIGITS, U200:
```

Background:

- The Chebyshev polynomials of the second kind are given by

$$U(n,x) = \frac{\sin((n+1)\,\arccos(x))}{\sin(\arccos(x))}$$

  for real $x \in [-1,1]$. This representation is used by `orthpoly::chebyshev2` for floating point values in this range.

- These polynomials satisfy the recursion formula

$$U(n,x) = 2\,x\,U(n-1,x) - U(n-2,x)$$

  with $U(0,x) = 1$ and $U(1,x) = 2\,x$.

- They are orthogonal on the interval $[-1,1]$ with respect to the weight function $w(x) = \sqrt{1-x^2}$.

- $U(n,x)$ coincides with the Gegenbauer polynomial $G(n,1,x)$.

- $U(n,x)$ is a special Jacobi polynomial:

$$U(n,x) = \frac{2^{2n}\,n!\,(n+1)!}{(2n+1)!}\;P\Big(n,\frac{1}{2},\frac{1}{2},x\Big)\ .$$

Changes:

- Indexed identifiers are now regarded as indeterminates. Non-integer degrees n are now accepted, leading to an unevaluated function call. Floating point evaluations are now numerically stable. Efficiency was improved.

`orthpoly::curtz` – the Curtz polynomials

`orthpoly::curtz(n,x)` computes the value of the n-th degree Curtz polynomial at the point x.

Call(s):

- `orthpoly::curtz(n, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Curtz polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::curtz` returns itself symbolically.

Details:

- These polynomials have rational coefficients.
- Evaluation for real floating point values x is numerically stable. Cf. example 2.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::curtz(2, x)

                          2
                  poly(x  - x + 1/3, [x])

>> orthpoly::curtz(3, x[1])
```

```
                 3            2
        poly(x[1]  - 3/2 x[1]  + 11/12 x[1] - 1/4, [x[1]])
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::curtz(2, 6*x)

                    6 x (6 x - 1/2) - 3 x + 1/3

>> orthpoly::curtz(3, x[1] + 2)

 x[1]                /   x[1]     \
 ---- + (x[1] + 3/2) | - ---- - 1 | +
  3                  \    2       /

             /                           x[1]       \
   (x[1] + 2) | (x[1] + 2) (x[1] + 3/2) - ---- - 2/3 | + 5/12
             \                            2         /
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::curtz(2, sqrt(2)), orthpoly::curtz(3, 8 + I),
   orthpoly::curtz(100, 0.3)

                       1/2
  1/2   1/2           2
 2    (2    - 1/2) - ---- + 1/3, 4807/12 + 2015/12 I,
                       2

    0.001395122936
```

If no integer degree is specified, then `orthpoly::curtz` returns itself symbolically:

```
>> orthpoly::curtz(n, x), orthpoly::curtz(1/2, x)

         orthpoly::curtz(n, x), orthpoly::curtz(1/2, x)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::curtz(50, 1.2)

                                        0.0003843630923
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> orthpoly::curtz(50, x): evalp(%, x = 1.2)

                                        0.0003843575545
```

Note that due to numerical round-off only 4 digits are correct.

Background:

- The Curtz polynomials are given by the recursion formula

$$C(n,x) = x^n \; + \; x \sum_{i=1}^{n-1} \frac{(-1)^i}{i+1} \, C(n-i-1,x) \; + \; \frac{(-1)^n}{n+1}$$

  with $C(0,x) = 1$.

Changes:

- Indexed identifiers are now regarded as indeterminates. Non-integer degrees n are now accepted, leading to an unevaluated function call. Floating point evaluations are now numerically stable. Efficiency was improved.

`orthpoly::gegenbauer` – the Gegenbauer (ultraspherical) polynomials

`orthpoly::gegenbauer(n,a,x)` computes the value of the n-th degree Gegenbauer polynomial with parameter a at the point x.

Call(s):

- `orthpoly::gegenbauer(n, a, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`a` — an arithmetical expression.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Gegenbauer polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::gegenbauer` returns itself symbolically.

Related Functions: `orthpoly::chebyshev2`, `orthpoly::legendre`

Details:

- Evaluation for real floating point values x from the interval $[-1.0, 1.0]$ is numerically stable. Cf. example 2.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::gegenbauer(2, a, x)

                                  2   2
                 poly((2 a + 2 a ) x  - a, [x])

>> orthpoly::gegenbauer(3, 2, x[1])

                              3
                poly(32 x[1]  - 12 x[1], [x[1]])
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::gegenbauer(2, a, 6*x)

                         2            2  2
                   72 a x  - a + 72 a  x

>> orthpoly::gegenbauer(3, 2, x[1] + 2)

  8 (x[1] + 2) (3 (x[1] + 2) (4 x[1] + 8) - 2)   20 x[1]
  -------------------------------------------- - ------- -
40/3
                      3                              3
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::gegenbauer(2, a, sqrt(2)),
   orthpoly::gegenbauer(3, 0.4, 8 + I),
   orthpoly::gegenbauer(1000, -1/3, 0.3)

                 2
        3 a + 4 a , 865.536 + 341.152 I, 0.00006046127974
```

If no integer degree is specified, then `orthpoly::gegenbauer` returns itself symbolically:

```
>> orthpoly::gegenbauer(n, a, x), orthpoly::gegenbauer(1/2, 2, x)

 orthpoly::gegenbauer(n, a, x), orthpoly::gegenbauer(1/2, 2, x)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::gegenbauer(200, 4, 0.3)

                    165549.7263
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> G200 := orthpoly::gegenbauer(200, 4, x):
>> DIGITS := 10: evalp(G200, x = 0.3)

                    -1.537729446e11
```

This result is caused by numerical round-off. Also with increased `DIGITS` only a few leading digits are correct:

```
>> DIGITS := 20: evalp(G200, x = 0.3)

                    165541.53415992590886

>> delete DIGITS, G200:
```

Background:

- The Gegenbauer polynomials are given by the recursion formula

$$G(n,a,x) = \frac{2\,(n-1+a)}{n}\,x\,G(n-1,a,x) + \frac{n-2+2\,a}{n}\,G(n-2,a,x)$$

  with $G(0,a,x) = 1,\ G(1,a,x) = 2\,a\,x$.

- For fixed real $a > -1/2$ these polynomials are orthogonal on the interval $[-1,1]$ with respect to the weight function $w(x) = (1-x^2)^{a-1/2}$.

- $G(n,1/2,x)$ coincides with the Legendre polynomial $P(n,x)$.

- $G(n,1,x)$ coincides with the Chebyshev polynomial $U(n,x)$ of the second kind.

- The polynomials $G(n,0,x)$ are trivial.

Changes:

- Indexed identifiers are now regarded as indeterminates. Arbitrary expressions are now accepted for the parameter a. Non-integer degrees n are now accepted, leading to an unevaluated function call. Efficiency was improved.

orthpoly::hermite – the Hermite polynomials

orthpoly::hermite(n,x) computes the value of the n-th degree Hermite polynomial at the point x.

Call(s):

- orthpoly::hermite(n, x)

Parameters:

n — a nonnegative integer: the degree of the polynomial.
x — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type DOM_IDENT) or an indexed identifier (of type "_index").

Return Value: If x is an indeterminate, then a polynomial of domain type DOM_POLY is returned. If x is an arithmetical expression, then the value of the Hermite polynomial at this point is returned as an arithmetical expression. If n is not a nonnegative integer, then orthpoly::hermite returns itself symbolically.

Details:

- These polynomials have integer coefficients.

Example 1. Polynomials of domain type DOM_POLY are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::hermite(2, x)

                          2
                  poly(4 x  - 2, [x])

>> orthpoly::hermite(3, x[1])

                       3
           poly(8 x[1]  - 12 x[1], [x[1]])
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::hermite(2, 6*x)

                            2
                       144 x  - 2
```

```
>> orthpoly::hermite(3, x[1] + 2)

   2 (x[1] + 2) (2 (x[1] + 2) (2 x[1] + 4) - 2) - 8 x[1] -
16
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::hermite(2, sqrt(2)), orthpoly::hermite(3, 8 + I),
   orthpoly::hermite(1000, 0.3);

                6, 3808 + 1516 I, 2.26821486e1433
```

If no integer degree is specified, then `orthpoly::hermite` returns itself symbolically:

```
>> orthpoly::hermite(n, x), orthpoly::hermite(1/2, x)

       orthpoly::hermite(n, x), orthpoly::hermite(1/2, x)
```

Background:

- The Hermite polynomials are given by the recursion formula

$$H(n,x) = 2\,x\,H(n-1,x) - 2\,(n-1)\,H(n-2,x)$$

  with $H(0,x) = 1$ and $H(1,x) = 2\,x$.

- These polynomials are orthogonal on the real line with respect to the weight function $w(x) = e^{-x^2}$.

Changes:

- Indexed identifiers are now regarded as indeterminates. Non-integer degrees n are now accepted, leading to an unevaluated function call. Efficiency was improved.

`orthpoly::jacobi` – the Jacobi polynomials

`orthpoly::jacobi(n,a,b,x)` computes the value of the n-th degree Jacobi polynomial with parameters a and b at the point x.

Call(s):

- `orthpoly::jacobi(n, a, b, x)`

Parameters:

`n`	—	a nonnegative integer: the degree of the polynomial.
`a, b`	—	arithmetical expressions.
`x`	—	an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Jacobi polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::jacobi` returns itself symbolically.

Related Functions: `orthpoly::chebyshev1`, `orthpoly::chebyshev2`, `orthpoly::gegenbauer`, `orthpoly::legendre`

Details:

- Evaluation for real floating point values x from the interval $[-1.0, 1.0]$ is numerically stable. Cf. example 2.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::jacobi(2, a, b, x)

     / /                     2    2       \
     | | 7 a   7 b   a b   a    b         |  2
 poly| | --- + --- + --- + -- + -- + 3/2 | x  +
     \ \  8     8     4    8    8         /

    /              2    2 \     /  2                 2       \
    | 3 a   3 b   a    b  |     | a    b   a b   a   b       |
    | --- - --- + -- - -- | x + | -- - - - --- - - + -- - 1/2 |,
    \  4     4    4    4  /     \ 8    8    4    8   8       /

        \
        |
    [x] |
        /

>> orthpoly::jacobi(3, 4, 5, x[1])

                   3            2
    poly(455/8 x[1]  - 91/8 x[1]  - 91/8 x[1] + 7/8, [x[1]])
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::jacobi(2, 4, b, 6*x)

 (b + 1) (6 x - 1)
 ----------------- +
         4

                         /     / b       \    b       \
    (6 x (b + 8) - b + 2) | 6 x | - + 7/2 | - - + 5/2 |
                         \     \ 2       /    2       /
    ---------------------------------------------------
                               4
```

```
>> orthpoly::jacobi(2, 0, I, x[1] + 2)

 (1/4 + 1/4 I) x[1] + (((4 + I) x[1] + (6 + I))

    ((3/2 + 1/2 I) x[1] + (7/2 + 1/2 I))) / 4 + (1/4 + 1/4 I)
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::jacobi(2, 1/2, -1/2, sqrt(2)),
   orthpoly::jacobi(3, 2, 5, 8 + I),
   orthpoly::jacobi(1000, 1, 2, 0.3);

    1/2   1/2
 3 2    (2    + 1/2)
 ------------------- - 3/8, 31733/2 + 12859/2 I, -0.06546648097
          2
```

If no integer degree is specified, then `orthpoly::jacobi` returns itself symbolically:

```
>> orthpoly::jacobi(n, a, b, x), orthpoly::jacobi(1/2, 0, 1, 1)

  orthpoly::jacobi(n, a, b, x), orthpoly::jacobi(1/2, 0, 1, 1)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::jacobi(100, 1/2, 3/2, 0.9)

                                 0.2560339406
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> P100 := orthpoly::jacobi(100, 1/2, 3/2, x):
>> evalp(P100, x = 0.9)
```

```
2.139740624e14
```

This result is caused by numerical round-off. Also with increased `DIGITS` only a few leading digits are correct:

```
>> DIGITS := 30: evalp(P100, x = 0.9)

        0.256005789994057173724575383078

>> delete P100, DIGITS:
```

Background:

- The Jacobi polynomials are given by the recursion formula

$$\begin{aligned}2n\, c_n c_{2n-2} P(n,a,b,x) = {} & c_{2n-1}(c_{2n-2}c_{2n}x + a^2 - b^2)P(n-1,a,b,x)\\ & - 2(n-1+a)(n-1+b)c_{2n}P(n-2,a,b,x)\end{aligned}$$

  with $c_i = i + a + b$ and

$$P(0,a,b,x) = 1, \quad P(1,a,b,x) = \frac{a-b}{2} + \left(1 + \frac{a+b}{2}\right) x.$$

- For fixed real $a > -1$, $b > -1$ the Jacobi polynomials are orthogonal on the interval $[-1,1]$ with respect to the weight function $w(x) = (1-x)^a(1+x)^b$.

- For special values of the parameters a, b the Jacobi polynomials are related to the Legendre polynomials

$$P(n,x) = P(n,0,0,x)\ ,$$

  to the Chebyshev polynomials of the first kind

$$T(n,x) = \frac{2^{2n}\,(n!)^2}{(2n)!}\, P\left(n, -\frac{1}{2}, -\frac{1}{2}, x\right)\ ,$$

  to the Chebyshev polynomials of the second kind

$$U(n,x) = \frac{2^{2n}\, n!\,(n+1)!}{(2n+1)!}\, P\left(n, \frac{1}{2}, \frac{1}{2}, x\right)\ ,$$

  and to the Gegenbauer polynomials, respectively:

$$G(n,a,x) = \frac{\Gamma(a+\frac{1}{2})\,\Gamma(n+2a)}{\Gamma(2a)\,\Gamma(n+a+\frac{1}{2})}\, P\left(n, a-\frac{1}{2}, a-\frac{1}{2}, x\right)\ .$$

Changes:

- Indexed identifiers are now regarded as indeterminates. Now arbitrary expressions are accepted for the parameters a, b. Non-integer degrees n are now accepted, leading to an unevaluated function call. Efficiency was improved.

`orthpoly::laguerre` – the (generalized) Laguerre polynomials

`orthpoly::laguerre(n,a,x)` computes the value of the generalized n-th degree Laguerre polynomial with parameter a at the point x.

Call(s):

- `orthpoly::laguerre(n, a, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`a` — an arithmetical expression.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Laguerre polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::laguerre` returns itself symbolically.

Details:

- The standard Laguerre polynomials correspond to $a = 0$. They have rational coefficients.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::laguerre(2, a, x)

         /                        /         2      \       \
         |      2                 | 3 a    a       |       |
     poly| 1/2 x  + (- a - 2) x + | --- + -- + 1 |, [x] |
         \                        \  2    2        /       /

>> orthpoly::laguerre(3, a, x[1])
```

```
      /
      |            3   / a       \     2
 poly| - 1/6 x[1]  + | - + 3/2 | x[1]  +
      \                \ 2       /

    /          2     \        /              3     \          \
    |    5 a    a     |        | 11 a    2   a      |          |
    | - --- - -- - 3 | x[1] + | ---- + a  + -- + 1 |, [x[1]] |
    \    2    2      /        \  6          6      /          /
```

However, using arithmetical expressions as input the "values" of these polynomials are returned:

```
>> orthpoly::laguerre(2, 4, 6*x)

                    (5 - 6 x) (7 - 6 x)
                    ------------------- - 5/2
                             2

>> orthpoly::laguerre(2, 2/3*I, x[1] + 2)

   (- x[1] - (1 - 2/3 I)) ((1 + 2/3 I) - x[1])
   ------------------------------------------- - (1/2 + 1/3 I)
                        2
```

"Arithmetical expressions" include numbers:

```
>> orthpoly::laguerre(2, a, sqrt(2)),
   orthpoly::laguerre(3, 0.4, 8 + I),
   orthpoly::laguerre(1000, 3, 0.3);

        2
 3 a    a        1/2       1/2
 --- + -- - 2 2    - a 2    + 2, - 4.969333333 - 8.713333334 I,
  2    2

    -15691.69498
```

If no integer degree is specified, then `orthpoly::laguerre` returns itself symbolically:

```
>> orthpoly::laguerre(n, a, x), orthpoly::laguerre(1/2, a, x)

   orthpoly::laguerre(n, a, x), orthpoly::laguerre(1/2, a, x)
```

Background:

- The Laguerre polynomials are given by the recursion formula

$$L(n,a,x) = \frac{2\,n+a-1-x}{n}\,L(n-1,a,x) - \frac{n+a-1}{n}\,L(n-2,a,x)$$

  with $L(0,a,x) = 1$ and $L(1,a,x) = 1 + a - x$.

- For fixed real $a > -1$ these polynomials are orthogonal on the interval $[0,\infty)$ with respect to the weight function $w(x) = x^a e^{-x}$.

Changes:

- Indexed identifiers are now regarded as indeterminates. Now arbitrary expressions are accepted for the parameter a. Non-integer degrees n are now accepted, leading to an unevaluated function call. Efficiency was improved.

`orthpoly::legendre` – the Legendre polynomials

`orthpoly::legendre(n,x)` computes the value of the n-th degree Legendre polynomial at the point x.

Call(s):

- `orthpoly::legendre(n, x)`

Parameters:

`n` — a nonnegative integer: the degree of the polynomial.
`x` — an indeterminate or an arithmetical expression. An indeterminate is either an identifier (of domain type `DOM_IDENT`) or an indexed identifier (of type `"_index"`).

Return Value: If `x` is an indeterminate, then a polynomial of domain type `DOM_POLY` is returned. If `x` is an arithmetical expression, then the value of the Legendre polynomial at this point is returned as an arithmetical expression. If `n` is not a nonnegative integer, then `orthpoly::legendre` returns itself symbolically.

Related Functions: `numeric::gldata`, `orthpoly::gegenbauer`, `orthpoly::jacobi`

Details:

- These polynomials have rational coefficients.
- Evaluation for real floating point values x from the interval $[-1.0, 1.0]$ is numerically stable. Cf. example 2.
- Use `numeric::gldata` to compute the roots of the Legendre polynomials. Cf. example 3.

Example 1. Polynomials of domain type `DOM_POLY` are returned, if identifiers or indexed identifiers are specified:

```
>> orthpoly::legendre(2, x)

                                  2
                     poly(3/2 x  - 1/2, [x])

>> orthpoly::legendre(3, x[1])

                               3
                poly(5/2 x[1]  - 3/2 x[1], [x[1]])
```

However, using arithmetical expressions as input the “values” of these polynomials are returned:

```
>> orthpoly::legendre(2, 6*x)

                                  2
                            54 x  - 1/2

>> orthpoly::legendre(3, x[1] + 2)

                    /              2       \
                    | 3 (x[1] + 2)         |
       5 (x[1] + 2) | ------------- - 1/2 |
                    \       2              /   2 x[1]
       ------------------------------------ - ------ - 4/3
                         3                       3
```

“Arithmetical expressions” include numbers:

```
>> orthpoly::legendre(2, sqrt(2)), orthpoly::legendre(3, 8 + I),
   orthpoly::legendre(1000, 0.3)

                 5/2, 1208 + 476 I, -0.0256691675
```

If no integer degree is specified, then `orthpoly::legendre` returns itself symbolically:

```
>> orthpoly::legendre(n, x), orthpoly::legendre(1/2, x)

      orthpoly::legendre(n, x), orthpoly::legendre(1/2, x)
```

Example 2. If a floating point value is desired, then a direct call such as

```
>> orthpoly::legendre(100, 0.9)

                         0.1022658206
```

is appropriate and yields a correct result. One should not evaluate the symbolic polynomial at a floating point value, because this may be numerically unstable:

```
>> P100 := orthpoly::legendre(100, x):

>> evalp(P100, x = 0.9)

                         4.222688586e13
```

This result is caused by numerical round-off. Also with increased `DIGITS` only a few leading digits are correct:

```
>> DIGITS := 30: evalp(P100, x = 0.9)

               0.102266645970798303097499109014

>> delete P100, DIGITS:
```

Example 3. We recommend to use `numeric::gldata` for computing roots of the Legendre polynomial $P(n, x)$. This routine provides all roots of the function $Q(n, y) = P(n, 2\,y - 1)$:

```
>> QRoots := numeric::gldata(5, DIGITS)[2]

  [0.04691007703, 0.2307653449, 1/2, 0.769234655, 0.9530899229]
```

These values are easily transformed to roots of $P(n, x)$:

```
>> PRoots := map(QRoots, y -> 2*y - 1)

  [-0.9061798459, -0.5384693101, 0, 0.5384693101, 0.9061798459]

>> orthpoly::legendre(5, r) $ r in PRoots

 -1.08046818e-14, -1.084202173e-19, 0, 1.084202173e-19,

    1.08046818e-14

>> delete QRoots, PRoots:
```

Background:

- The Legendre polynomials are given by $P(n,x) = \frac{1}{2^n\, n!}\, \frac{d^n}{dx^n}(x^2-1)^n$.
- They satisfy the recursion formula

  $$P(n,x) = \frac{2\,n-1}{n}\; x\; P(n-1,x) - \frac{n-1}{n}\; P(n-2,x)$$

  with $P(0,x) = 1$ and $P(1,x) = x$.
- They are orthogonal on the interval $[-1,1]$ with respect to the weight function $w(x) = 1$.
- $P(n,x)$ coincides with the Gegenbauer polynomial $G(n,1/2,x)$.
- $P(n,x)$ coincides with the Jacobi polynomial $P(n,0,0,x)$.

Changes:

- Indexed identifiers are now regarded as indeterminates. Non-integer degrees n are now accepted, leading to an unevaluated function call. Floating point evaluations are now numerically stable. Efficiency was improved.