This manual describes the functionality of the Mathematica package HolonomicFunctions. It is a very powerful tool for the work with special functions, it can assist in solving summation and integration problems, it can automatically prove special function identities, and much more. The package has been developed in the frame of the PhD thesis [9]. The whole theory and the algorithms are described there, and it contains also many references for further reading as well as some more advanced examples; the examples in this manual are mostly of a very simple nature in order to illustrate clearly the use of the software. HolonomicFunctions is freely available from the RISC combinatorics software webpage

www.risc.uni-linz.ac.at/research/combinat/software/HolonomicFunctions/

Short references

Annihilator[expr, ops] computes annihilating operators for the expression expr with respect to the Ore operators ops (p. 5).

AnnihilatorDimension[ann] gives the dimension of the annihilating left ideal ann (p. 8).

AnnihilatorSingularities[ann, start] computes the set of singular points for a system ann of multivariate recurrences (p. 9).

ApplyOreOperator[opoly, expr] applies the operator given by the Ore polynomial opoly to expr (p. 11).

ChangeMonomialOrder[opoly, ord] changes the monomial order of the Ore polynomial opoly to ord (p. 12).

ChangeOreAlgebra[opoly, alg] translates the Ore polynomial opoly into the new Ore algebra alg (p. 13).

*partially supported by the Austrian Science Fund (FWF): P20162-N18, and by the US National Science Foundation: NSF-DMS 0070567.
CreativeTelescoping[expr, delta, ops] performs Chyzak’s algorithm to find creative telescoping relations for expr (p. 14).

Delta[n] represents the forward difference (delta) operator w.r.t. n (p. 17).

Der[x] represents the operator “partial derivative w.r.t. x” (p. 18).

DFiniteDE2RE[ann, {x_1, ..., x_d}, {n_1, ..., n_d}] computes recurrences in the variables n_1, ..., n_d for the coefficients of a power series that is solution to the given differential equations in ann (p. 19).

DFiniteOreAction[ann, opoly] executes the closure property “application of an operator” for the ∂-finite function described by ann (p. 20).

DFinitePlus[ann_1, ann_2] executes the closure property “sum” for the ∂-finite functions described by ann_1 and ann_2 (p. 22).

DFiniteQSubstitute[ann, {q, m, k}] computes an annihilating ideal of q-difference equations for the result of the substitution q → e^{2iπ/m q^{1/k}} (p. 24).

DFiniteRE2DE[ann, {n_1, ..., n_d}, {x_1, ..., x_d}] gives differential equations in x_1, ..., x_d of a generating function whose coefficients are described by the given recurrences in ann (p. 26).

DFiniteSubstitute[ann, subs] executes the closure properties “algebraic substitution” for continuous variables and “rational-linear substitution” for discrete variables (p. 27).

DFiniteTimes[ann_1, ann_2] executes the closure property “product” for the ∂-finite functions described by ann_1 and ann_2 (p. 29).

DFiniteTimesHyper[ann, expr] executes the closure property “product” when one factor expr is hypergeometric and hyperexponential in the respective variables (p. 30).

DSolvePolynomial[eqn, f[x]] determines whether the ordinary linear differential equation eqn in f[x] (with polynomial coefficients) has polynomial solutions, and in the affirmative case, computes them (p. 31).

DSolveRational[eqn, f[x]] determines whether the ordinary linear differential equation eqn in f[x] (with polynomial coefficients) has rational solutions, and in the affirmative case, computes them (p. 32).

Euler[x] represents the Euler operator θ_x = x D_x (p. 33).

FGLM[gb, order] transforms the noncommutative Gröbner basis gb into a Gröbner basis with respect to the given order. (p. 34).

FindCreativeTelescoping[expr, deltas, ops] finds creative telescoping relations for expr by ansatz (p. 36).

FindRelation[ann, opts] finds relations with certain properties in the annihilating ideal ann by ansatz (p. 39).

FindSupport[ann, opts] computes only the support of a relation in the annihilating ideal ann that satisfies the given constraints (p. 41).

GBEqual[gb_1, gb_2] whether two Gröbner bases are the same (p. 43).
HermiteTelescoping\[expr, \text{Der}[y], \text{Der}[x]\] performs Hermite telescoping to find creative telescoping relations for the hyperexponential \(expr\) (p. 44).

LeadingCoefficient\[opoly\] gives the leading coefficient of the Ore polynomial \(opoly\) (p. 45).

LeadingExponent\[opoly\] gives the exponents of the leading term of the Ore polynomial \(opoly\) (p. 46).

LeadingPowerProduct\[opoly\] gives the leading power product of the Ore polynomial \(opoly\) (p. 47).

LeadingTerm\[opoly\] gives the leading term of \(opoly\) (p. 48).

NormalizeCoefficients\[opoly\] removes the content of the Ore polynomial \(opoly\) (p. 49).

OreAction\[op\] determines how the newly defined Ore operator \(op\) acts on arbitrary expressions (p. 50).

OreAlgebra\[g_1, g_2, \ldots\] defines an Ore algebra that is generated by \(g_1, g_2, \ldots\) (p. 51).

OreAlgebraGenerators\[alg\] gives the list of generators of the Ore algebra \(alg\) (p. 53).

OreAlgebraOperators\[alg\] gives the list of Ore operators that are contained in the Ore algebra \(alg\) (p. 54).

OreAlgebraPolynomialVariables\[alg\] gives the list of variables that occur polynomially in the Ore algebra \(alg\) (p. 55).

OreDelta\[op\] defines the endomorphism \(\delta\) for the Ore operator \(op\). (p. 56).

OreGroebnerBasis\[\{P_1, \ldots, P_k\}\] computes a left Gröbner basis of the ideal generated by the Ore polynomials \(P_1, \ldots, P_k\) (p. 57).

OreOperatorQ\[expr\] tests whether \(expr\) is an Ore operator (p. 60).

OreOperators\[expr\] gives a list of Ore operators that occur in \(expr\) (p. 61).

OrePlus\[opoly_1, opoly_2\] adds the Ore polynomials \(opoly_1\) and \(opoly_2\) (p. 62).

OrePolynomial\[data, alg, order\] is the internal representation of Ore polynomials (p. 63).

OrePolynomialDegree\[opoly\] gives the total degree of the Ore polynomial \(opoly\) (p. 65).

OrePolynomialListCoefficients\[opoly\] gives a list containing all nonzero coefficients of the Ore polynomial \(opoly\). (p. 66).

OrePolynomialSubstitute\[opoly, rules\] applies the substitutions \(rules\) to the Ore polynomial \(opoly\) (p. 67).

OrePolynomialZeroQ\[opoly\] tests whether an Ore polynomial is zero (p. 68).

OrePower\[opoly, n\] gives the \(n\)-th power of the Ore polynomial \(opoly\) (p. 69).

OreReduce\[opoly, \{g_1, g_2, \ldots\}\] reduces the Ore polynomial \(opoly\) modulo the set of Ore polynomials \(\{g_1, g_2, \ldots\}\) (p. 70).
OreSigma[op] defines the endomorphism $\sigma$ for the Ore operator op (p. 72).

OreTimes[opoly1, opoly2] multiplies the Ore polynomials opoly1 and opoly2, respecting their noncommutative nature (p. 73).

Printlevel = $n$ activates and controls verbose mode (see p. 74).

QS[x, q^-n] represents the $q$-shift operator on $x$ (p. 75).

QSSolvePolynomial[eqn, f[x], q] determines whether the linear $q$-shift equation eqn in $f[x]$ (with polynomial coefficients) has polynomial solutions, and in the affirmative case, computes them (p. 76).

QSSolveRational[eqn, f[x], q] determines whether the linear $q$-shift equation eqn in $f[x]$ (with polynomial coefficients) has rational solutions, and in the affirmative case, computes them (p. 77).

RandomPolynomial[var, deg, c] gives a dense random polynomial in the variables var, of degree deg, and with integer coefficients between $-c$ and $c$ (p. 78).

RSolvePolynomial[eqn, f[n]] determines whether the linear recurrence equation eqn in $f[n]$ (with polynomial coefficients) has polynomial solutions, and in the affirmative case, computes them (p. 79).

RSolveRational[eqn, f[n]] determines whether the linear recurrence equation eqn in $f[n]$ (with polynomial coefficients) has rational solutions, and in the affirmative case, computes them (p. 80).

S[n] represents the forward shift operator w.r.t. $n$ (p. 81).

SolveCoupledSystem[eqns, {f1, ..., fk}, {v1, ..., vj}] computes all rational solutions of a coupled system of linear difference and differential equations (p. 82).

SolveOreSys[type, var, eqns, {f1[var], ..., fk[var]}, pars] computes all rational solutions of a first-order coupled linear difference or differential system (p. 84).

Support[opoly] gives the support of the OrePolynomial opoly (p. 86).

Takayama[ann, vars] performs Takayama’s algorithm for definite summation and integration with natural boundaries (p. 87).

ToOrePolynomial[expr, alg] converts expr to an Ore polynomial in the Ore algebra alg (p. 90).

UnderTheStaircase[gb] computes the list of monomials (power products) that lie under the stairs of the Gröbner basis gb (p. 92).
Annihilator

Annihilator[expr, ops]
computes annihilating relations for expr w.r.t. the Ore operator(s) ops.

Annihilator[expr]
automatically tries to determine for which operators relations exist.

More Information

The input expr can be any Mathematica expression (but not a list of expressions) and ops can be either a list of operators or a single operator; admissible operators are S, Der, Delta, and QS. The output consists of a list of OrePolynomial expressions which form a Gröbner basis of an annihilating left ideal for expr. It need not necessarily be the maximal annihilating ideal (i.e., the full annihilator), but often it is, in particular when expr is recognized to be ∂-finite.

If the input is not recognized to be ∂-finite, some heuristics to find relations are applied, but it may well be that some are missed. The relations are computed by recursively analyzing the structure of the input down to its “atomic” building blocks and then executing the ∂-finite closure properties DFinitePlus, DFiniteTimes, DFiniteSubstitute, and DFiniteOreAction. To see a complete list of mathematical functions that are recognized by Annihilator as atomic building blocks (whether ∂-finite or not) type ?Annihilator.

Annihilator has the attribute HoldFirst to prevent Mathematica from doing any simplification on the input. If expr contains the command D or ApplyOreOperator then the closure property DFiniteOreAction is performed, which is more desirable compared to first evaluate and then computing an annihilating ideal (see below for an example on this issue). Similarly, if expr contains Sum or Integrate then not Mathematica is asked to simplify the expression, but CreativeTelescoping is executed automatically on the summand (resp. integrand). For evaluating the delta part, Mathematica’s FullSimplify and Limit are used; if they fail (or if you don’t trust them), you can use the option Inhomogeneous in order to obtain inhomogeneous recurrences (resp. differential equations), where the critical components of the delta part have been wrapped with Hold. Often the problem is that the evaluation of the delta part requires additional assumptions; they can be given with the option Assumptions.

The following options can be given:

Assumptions \[\rightarrow\] $Assumptions
In cases where CreativeTelescoping is called internally, these assumptions are passed and used for simplifying the inhomogeneous parts.

Head \[\rightarrow\] None
By default, the annihilating operators are returned as OrePolynomial expressions; if some symbol (other than None) is given, e.g., Head \[\rightarrow\] f, then the output is given as relations of the specified function.
Inhomogeneous → False

Applies only if expr is a sum or an integral. In order to present the result as operators, the relations found by creative telescoping are homogenized by default. If this option is set to True then two lists are returned: one containing the homogeneous parts (given as Ore polynomials), and the other containing the corresponding inhomogeneous parts. Critical components (like Limit expressions) of the inhomogeneous parts are wrapped with Hold.

Method → Automatic

is passed when CreativeTelescoping is called internally, see p. 14.

MonomialOrder → DegreeLexicographic

the monomial order w.r.t. to which the output is given; see OreGroebnerBasis (p. 57) for a list of supported monomial orders.

\[ \text{\textbf{Examples}} \]

\begin{verbatim}
In[1]:= Annihilator[Sin[Sqrt[x^2 + y]], \{Der[x], Der[y]\}]
\text{Out}[1]= \{D_x - 2 x D_y, (4 x^2 + 4 y) D_y^2 + 2 D_y + 1\}
In[2]:= Annihilator[LegendreP[n, x]]
\text{Out}[2]= \{(n + 1) S_n + (1 - x^2) D_x + (-n x - x), (x^2 - 1) D_x^2 + 2 x D_x + (-n^2 - n)\}
In[3]:= Annihilator[ArcSin[Sqrt[x + 1]]^k, \{S[k], Der[x]\}]

Annihilator::nondf : The expression ArcSin[Sqrt[1 + x]]^k is not recognized to be \(n\)-finite.

The result might not generate a zero-dimensional ideal.
\text{Out}[3]= \{(4 x^2 + 4 x) S_k^2 D_x^2 + (4 x + 2) S_k^2 D_x + (k^2 + 3 k + 2)\}
In[4]:= Annihilator[Sum[Binomial[n, k], \{k, 0, n\}], S[n]]
\text{Out}[4]= \{S_n - 2\}
In[5]:= Annihilator[Integrate[(LegendreP[2 k + 1, x] / x)^2, \{x, -1, 1\}], S[k], Assumptions \to Element[k, Integers] \&\& k \geq 0]
\text{Out}[5]= \{-S_n + 1\}
In[6]:= Annihilator[Fibonacci[n], S[n], Head \to f]
\text{Out}[6]= \{-f(n) - f(n + 1) + f(n + 2) = 0\}
\end{verbatim}

Note the difference between the following two ways to compute a differential equation for \(J_n'(x)\). The closure property DFiniteOreAction never increases the order whereas DFinitePlus usually does.

\begin{verbatim}
In[7]:= Annihilator[D[BesselJ[n, x], x], Der[x]]
\text{Out}[7]= \{(n^2 x^2 - x^4) D_x^2 + (3 n^2 x - x^3) D_x + (-n^4 + 2 n^2 x^2 + n^2 - x^4 + x^2)\}
In[8]:= expr = D[BesselJ[n, x], x]
\text{Out}[8]= D[BesselJ[n, x], x]
In[9]:= \frac{1}{2} (BesselJ[\text{-}1 + n, x] - BesselJ[1 + n, x])
\text{Out}[9]= \frac{1}{2} (BesselJ[\text{-}1 + n, x] - BesselJ[1 + n, x])
In[10]:= Annihilator[expr, Der[x]]
\text{Out}[10]= \{x^4 D_x^2 + 6 x^3 D_x^3 + (-2 n^2 x^2 + 2 x^4 + 5 x^2) D_x^2 + (-2 n^2 x + 6 x^3 - x) D_x + (n^4 - 2 n^2 x^2 - 2 n^2 x^4 + x^2 + 1)\}
\end{verbatim}
The following, more advanced example is taken from [8], formula 7.322:
\[
\int_0^{2a} \frac{(x(2a-x))^{\nu + \frac{1}{2}}}{e^{bx}} C_n^{(\nu)} \left( \frac{x}{a} - 1 \right) \, dx = \pi \left( \frac{1}{a} \right)^{\nu} \Gamma(n + 2\nu) I_{n+\nu}(ab).
\]
In this example the inhomogeneous parts are so complicated that Mathematica needs some little help to get them simplified. Hence we use the option Inhomogeneous.

\begin{verbatim}
{lhs, inhom} = Annihilator[Integrate[
  ((x(2a - x))^(nu - 1/2) GegenbauerC[n, nu, x/a - 1])/E^(-bx),
  {x, 0, 2a}],
  {Der[a], Der[b], S[n], S[nu]}, Assumptions \rightarrow nu \geq 1, Inhomogeneous \rightarrow True];
\end{verbatim}

The annihilating ideal for the left-hand side is nice, but the inhomogeneous part is so big that we don’t want to display it here.

\begin{verbatim}
{lhs}
\end{verbatim}

\begin{verbatim}
{(-an^2 - 2anu - 2an - 2nu - a)S_n - 2bnS_n,
  (bn^2 + 4bnv + bn + 4nu^2 + 2bn)D_\nu - 2b^2nuS_n +
  (abn^2 + 4abnu + abn + 4abnu^2 + 2abv - n^3 - 4nu^2 - n^2 - 4nu^2 - 2nu),
  (an^2 + 4anu + an + 4nu^2 + 2an)D_\nu - 2b^2nuS_n + (abn^2 + 4abnu + abn +
   4abnu^2 + 2abv - n^3 - 6nu^2 - n^2 - 12nu^2 - 4nu^2 - 8nu^2 - 2nu^2),
  (4b^2nu^2 + 4b^2nu)S_n + (4nu^2 + 20nu^2 + 2nu^2 + 32nu^3 + 76nu^2 + 44nu + 16nu^4 +
   56nu^3 + 64nu^2 + 24nu)S_n + (-a^2n^2 - 8a^2nu - 6a^2n^3 - 24a^2n^2nu^2 -
   36a^2n^2nu - 32a^2nu^3 - 32a^2nu^2 - 44a^2nu^3 - 6a^2n^2 - 16a^2nu^4 -
   48a^2nu^3 - 12a^2nu^2)\}
\end{verbatim}

\begin{verbatim}
ByteCount[inhom]
\end{verbatim}

\begin{verbatim}
127664
\end{verbatim}

\begin{verbatim}
Simplify[ReleaseHold[inhom /. Limit \rightarrow myLimit]]
\end{verbatim}

\begin{verbatim}
myLimit \rightarrow Limit, Assumptions \rightarrow nu \geq 1
\end{verbatim}

\begin{verbatim}
{0, 0, 0, 0}
\end{verbatim}

\begin{verbatim}
rhs = Annihilator[{-(-1)^n Pi (Gamma[2nu + n]/Gamma[nu])
  (a/(2b))nu BesselI[nu + n, ab]) / E^(-ab),
  {Der[a], Der[b], S[n], S[nu]}]
\end{verbatim}

\begin{verbatim}
{(-an^2 + 2anu + 2an + 2nu + a)S_n + 2bnS_n,
  (bn^2 + 4bnv + bn + 4nu^2 + 2bn)D_\nu - 2b^2nuS_n +
  (abn^2 + 4abnu + abn + 4abnu^2 + 2abv - n^3 - 4nu^2 - n^2 - 4nu^2 - 2nu),
  (an^2 + 4anu + an + 4nu^2 + 2an)D_\nu - 2b^2nuS_n + (abn^2 + 4abnu + abn +
   4abnu^2 + 2abv - n^3 - 6nu^2 - n^2 - 12nu^2 - 4nu^2 - 8nu^2 - 2nu^2),
  (4b^2nu^2 + 4b^2nu)S_n + (4nu^2 + 20nu^2 + 2nu^2 + 32nu^3 + 76nu^2 + 44nu + 16nu^4 +
   56nu^3 + 64nu^2 + 24nu)S_n + (-a^2n^2 - 8a^2nu - 6a^2n^3 - 24a^2n^2nu^2 -
   36a^2n^2nu - 32a^2nu^3 - 32a^2nu^2 - 44a^2nu^3 - 6a^2n^2 - 16a^2nu^4 -
   48a^2nu^3 - 12a^2nu^2)\}
\end{verbatim}

\begin{verbatim}
GBEqual[lhs, rhs]
\end{verbatim}

\begin{verbatim}
True
\end{verbatim}

\begin{verbatim}
\[ \text{See Also:} \]
AnnihilatorDimension, AnnihilatorSingularities, CreativeTelescoping, DFinitePlus, DFiniteTimes, DFiniteSubstitute, DFiniteOreAction
\end{verbatim}
AnnihilatorDimension

\[ \text{AnnihilatorDimension}[\text{ann}] \]

gives the dimension of the annihilating ideal \( \text{ann} \).

**More Information**

The input \( \text{ann} \) has to be a list of \text{OrePolynomial} expressions that constitute a Gröbner basis (with respect to the Ore algebra and monomial order that they are represented in), and the output is a natural number. Note that the Gröbner basis condition is not tested, and if violated, the result may be wrong. What internally happens is that only the leading exponent vectors are considered (so, alternatively, a list of such can be given as input). If the ideal \( \text{ann} \) is \( \partial \)-finite, then its dimension is 0.

**Examples**

```math
In[16] := Annihilator[ChebyshevT[n, x], {S[n], Der[x]}]
Out[16] := \{nS_n + (1 - x^2)D_x - nx, (x^2 - 1)D_x + xD_x - n^2\}
```

```math
In[17] := AnnihilatorDimension[%]
Out[17] := 0
```

```math
In[18] := Annihilator[StirlingS1[k, m], {S[k], S[m]}]
```

\text{Annihilator::nondf} : The expression \( \text{StirlingS1}[k, m] \) is not recognized to be \( \partial \)-finite.

The result might not generate a zero-dimensional ideal.

```math
Out[18] := \{S_kS_m + kS_m - 1\}
```

```math
In[19] := AnnihilatorDimension[%]
Out[19] := 1
```

```math
In[20] := Annihilator[StirlingS1[k, m]StirlingS2[k, n], {S[k], S[m], S[n]}]
```

\text{Annihilator::nondf} : The expression \( \text{StirlingS1}[k, m] \) is not recognized to be \( \partial \)-finite.

The result might not generate a zero-dimensional ideal.

\text{Annihilator::nondf} : The expression \( \text{StirlingS2}[k, n] \) is not recognized to be \( \partial \)-finite.

The result might not generate a zero-dimensional ideal.

```math
Out[20] := \{S_kS_mS_n + (kn + k)S_mS_n + kS_m + (-n - 1)S_n - 1\}
```

```math
In[21] := AnnihilatorDimension[%]
Out[21] := 2
```

**See Also**

Annihilator, UnderTheStaircase
AnnihilatorSingularities

AnnihilatorSingularities[\emph{ann, start}] computes the set of singular points in the positive region above \emph{start} for a system \emph{ann} of multivariate recurrences.

\begin{itemize}
  \item \textbf{More Information}
  
  The input \emph{ann} has to be a system of (multivariate) recurrences, given as a list of \texttt{OrePolynomial} expressions. They need not to form a Gröbner basis, but note that it will not be recognized if they happen to be inconsistent (since only their leading terms are taken into account). The second argument \emph{start}, a list of integers, gives the start point of the sequence. Then all singular points in the positive region above \emph{start}, i.e., \emph{start} + \(\mathbb{N}^d\) where \(d\) is the number of variables, are determined. By singular points, we mean those points where none of the recurrences can be applied, either because it would involve values from outside the region or because its leading coefficient vanishes.

  The output is a list of pairs, each of which consists of a list of substitutions and a condition under which these substitutions give rise to singular points. It is tacitly assumed that the recurrence variables take only integer values—hence this condition is not extra stated in the output.

  The region under consideration can be further restricted by giving assumptions on the variables.

  The following options can be given:

  \begin{itemize}
  \item \textbf{Assumptions} \(\mapsto\) \texttt{Assumptions}

  further restrictions on the variables of the recurrences
  \end{itemize}

\end{itemize}

\begin{itemize}
  \item \textbf{Examples}

  \begin{verbatim}
In[22]:= AnnihilatorSingularities[ToOrePolynomial[\{(n - 5)S[n] - 1\}, \{0\}]]
Out[22]= {{\{n \rightarrow 0\}, True}, {\{n \rightarrow 6\}, True}}
\end{verbatim}

The following two recurrences annihilate \(\delta_{k,n}\); all points on the diagonal are singular because of the leading coefficients, and the point \((0, 0)\) is singular because usage of either recurrence would require values from outside the region. Note that the cases returned by \texttt{AnnihilatorSingularities} may overlap (as it appears here).

\begin{verbatim}
In[22]:= ToOrePolynomial[\{(n - k - 1)S[k] + k - n, (n - k + 1)S[n] + k - n\}]
Out[22]= \{\{(-k + n - 1)S_k + (k - n), (-k + n + 1)S_n + (k - n)\}
\end{verbatim}

\begin{verbatim}
In[24]:= AnnihilatorSingularities[\%, \{0, 0\}]
Out[24]= {{\{k \rightarrow n\}, n \geq 0}, {\{k \rightarrow 0, n \rightarrow 0\}, True}}
\end{verbatim}
\end{itemize}
In the next example we have constant coefficients, hence the only singularities correspond to values under the stairs. Due to the additional restriction on the domain, we get a parallelogram-shaped set of initial values.

In[25]:= ToOrePolynomial[{S[k]^2 - 1, S[n]^3 - 1}, OreAlgebra[S[k], S[n]]]

Out[25]= {S^2 k - 1, S^3 n - 1}

In[26]:= AnnihilatorSingularities[%, {1, 1}, Assumptions \[\rightarrow\] k \[\leq\] n]

Out[26]= {{{k \[rightarrow\] 1, n \[rightarrow\] 1}, True}, {{k \[rightarrow\] 1, n \[rightarrow\] 2}, True}, {{k \[rightarrow\] 1, n \[rightarrow\] 3}, True}, {{k \[rightarrow\] 2, n \[rightarrow\] 2}, True}, {{k \[rightarrow\] 2, n \[rightarrow\] 3}, True}, {{k \[rightarrow\] 2, n \[rightarrow\] 4}, True}}

\[\n\]

See Also

Annihilator, AnnihilatorDimension, UnderTheStaircase
ApplyOreOperator

**ApplyOreOperator**\[opoly, expr\]

applies the Ore operator \(opoly\) to the expression \(expr\).

\[\]

**More Information**

The first argument \(opoly\) can be an **OrePolynomial** expression or a plain polynomial in the existing Ore operators; also a list of the previously mentioned is admissible. How the occurring Ore operators act on \(expr\) is defined by the command **OreAction**. The second argument can be any Mathematica expression to which the necessary actions can be applied. If \(opoly\) contains \(q\)-shift operators of the form \(QS[x, q^n]\), then all occurrences of the dummy variables \(x\) are replaced by \(q^n\).

\[\]

**Examples**

```
In[27]:= opoly = ToOrePolynomial[Der[x]^2 + 1];
In[28]:= ApplyOreOperator[opoly, Sin[x]]

Out[28]= 0
```

```
In[29]:= ApplyOreOperator[S[n] - n - 1, n!]

Out[29]= (-n - 1)n! + (n + 1)!
```

```
In[30]:= ApplyOreOperator[q^n]

Out[30]= q^n
```

```
In[31]:= ApplyOreOperator[ToOrePolynomial[q^n, OreAlgebra[QS[q^n, q^n]]], q^n]

Out[31]= q^{2n}
```

\[\]

**See Also**

Delta, Der, Euler, OreAction, QS, ToOrePolynomial, S
ChangeMonomialOrder

ChangeMonomialOrder[opoly, ord]
changes the monomial order of the Ore polynomial opoly to ord.

More Information

The input opoly must be an OrePolynomial expression or a list of such. The terms of opoly are reordered according to the new monomial order ord, and the OrePolynomial expression(s) carrying the new order is returned.

The following monomial orders are supported. See the description of OreGroebnerBasis (p. 57) for more details:

- Lexicographic,
- ReverseLexicographic,
- DegreeLexicographic,
- DegreeReverseLexicographic,
- EliminationOrder[n],
- WeightedLexicographic[w]
- WeightedOrder[w, order],
- MatrixOrder.

Examples

In[32]:=
opoly = ToOrePolynomial[Sum[S[a]^i S[b]^j, {i, 0, 2}, {j, 0, 3}],
OreAlgebra[S[a], S[b]], MonomialOrder -> DegreeLexicographic]
Out[32]= S^2 a S^3 b + S^2 a S^2 b + S^2 a S + S^2 b^3 + S^2 b + S a + S^2 b + S + S + 1

In[33]:= ChangeMonomialOrder[opoly, Lexicographic]
Out[33]= S^2 a S^3 b + S^2 a S^2 b + S^2 a S + S^2 b^3 + S^2 b + S a + S^2 b + S + S + 1

In[34]:= ChangeMonomialOrder[opoly, WeightedOrder[{2, 1}, Lexicographic]]
Out[34]= S^2 a S^3 b + S^2 a S^2 b + S^2 a S + S^2 b^3 + S^2 b + S a + S^2 b + S + S + S + 1

See Also

ChangeOreAlgebra, FGLM, OrePolynomial, ToOrePolynomial
ChangeOreAlgebra

ChangeOreAlgebra[opoly, alg]
transform the Ore polynomial opoly into the Ore algebra alg.

More Information

To each OrePolynomial expression the algebra in which it is represented is attached. The command ChangeOreAlgebra can be used to change this representation if possible. Note that the transformed Ore polynomial is returned, and not the original opoly is changed. Changes can concern the order in which the generators of the algebra are given, as well as the set of generators itself. The command may fail for several reasons, e.g., if the input is not a polynomial in the new set of generators, or if the input involves some Ore operators that are not any more contained in alg; in such cases $Failed is returned.

The following options can be given:

- **MonomialOrder** → None
  if this option is used then also the monomial order is changed; None means that the monomial order of the input is taken.

Examples

In[35]:= opoly = ToOrePolynomial[
(x^3 + x^2 + x + 1) ** Der[x]’2 + (6x^2 + 4x + 2) ** Der[x] + 6x + 2,
OreAlgebra[Der[x]]]
Out[35]= (x^3 + x^2 + x + 1) D[2]^2 x + (6x^2 + 4x + 2) D[2] x + 6x + 2

In[36]:= ChangeOreAlgebra[opoly, OreAlgebra[x, Der[x]]]

In[37]:= ChangeOreAlgebra[opoly, OreAlgebra[Der[x], x]]

In[38]:= ChangeOreAlgebra[opoly, OreAlgebra[S[n]]]
ChangeOreAlgebra::ops : Some of the operators Der[x] occur in the polynomial
but are not part of the algebra.
Out[38]= $Failed

In[39]:= (1/x) ** opoly

In[40]:= ChangeOreAlgebra[%, OreAlgebra[x, Der[x]]]
ChangeOreAlgebra::nopoly : The elements of the new OreAlgebra do not occur polynomially.
Out[40]= $Failed

See Also

ChangeMonomialOrder, OreAlgebra, OrePolynomial, ToOrePolynomial
CreativeTelescoping

CreativeTelescoping[expr, delta, ops]
finds creative telescoping relations for expr with Chyzak’s algorithm, i.e.,
operators of the form \( P + \delta \cdot Q \) such that \( P \) involves only Ore operators
from ops and their variables.

CreativeTelescoping[ann, delta, ops]
finds creative telescoping relations in the annihilating ideal ann.

More Information

The first argument is either an annihilating ideal (i.e., a Gröbner basis of such
an ideal) or any mathematical expression. In the latter case, An annihilator
is internally called with expr. The second argument delta indicates whether a
summation or an integration problem has to be solved; it is then \( S[a] - 1 \) or
\( \text{Der}[a] \), respectively (where \( a \) is the summation resp. integration variable). For
\( q \)-summation use \( QS[q, qa] - 1 \). The third argument ops specifies the surviving
Ore operators, i.e., the operators that occur in the principal part \( P \) (as well as in
\( Q \)).

The output consists of two lists, the first one containing all the principle parts
(such that they constitute a Gröbner basis), and the second one containing the
corresponding delta parts.

Since the principle of creative telescoping is really one of the main aspects of
this package, we want to explain shortly what is behind. When we want to do
a definite sum of the form \( \sum_{k=a}^{b} f(k, w) \) then we search for creative telescoping
operators that annihilate \( f \) and that are of the form

\[
T = P(w, \partial_w) + (S_k - 1)Q(k, w, S_k, \partial_w)
\]

where \( \partial_w \) stands for some Ore operators that act on the variables \( w \). The
operator \( P \) is called the principal part, and \( Q \) is called the delta part. With such
an operator \( T \) we can immediately derive a relation for the definite sum:

\[
0 = \sum_{k=a}^{b} T(k, w, S_k, \partial_w) \cdot f(k, w)
\]

\[
= \sum_{k=a}^{b} P(w, \partial_w) \cdot f(k, w) + \sum_{k=a}^{b} (S_k - 1)Q(k, w, S_k, \partial_w) \cdot f(k, w)
\]

\[
= P(w, \partial_w) \cdot \sum_{k=a}^{b} f(k, w) + \left[ Q(k, w, S_k, \partial_w) \cdot f(k, w) \right]_{k=a}^{b+1}.
\]

Depending on whether the inhomogeneous part evaluates to zero or not, we
have \( P \) as an annihilating operator for the sum, or we get an inhomogeneous
relation for the sum. In the latter case, if one is not happy with that, one
can homogenize the relation by multiplying an annihilating operator for the
inhomogeneous part to $P$ from the left. Things become more complicated when the summation bounds involve the variables $w$, since then additional correction terms have to be introduced; the command **Annihilator** automatically deals with these issues.

Similarly we can derive relations for a definite integral $\int_a^b f(x, w) \, dx$. In this case we look for creative telescoping operators that annihilate $f$ and that are of the form

$$T = P(w, \partial_w) + D_x Q(x, w, D_x, \partial_w).$$

Again it is straightforward to deduce a relation for the integral

$$0 = \int_a^b T(x, w, D_x, \partial_w) \cdot f(x, w) \, dx$$

$$= \int_a^b P(w, \partial_w) \cdot f(x, w) \, dx + \int_a^b \left(D_x Q(x, w, D_x, \partial_w)\right) \cdot f(x, w) \, dx$$

$$= P(w, \partial_w) \cdot \int_a^b f(x, w) \, dx + \left[Q(x, w, D_x, \partial_w) \cdot f(x, w)\right]_a^b$$

which may be homogeneous or inhomogeneous.

The following options can be given:

**Incomplete** → **False**

If this option is set to **True** then the computation is stopped after the first creative telescoping relation is found; makes sense only if *ops* contains more than one Ore operator.

**Method** → **Automatic**

The following methods can be chosen:

"**Chyzak**": Executes Chyzak’s algorithm [6] where the uncoupling is done by Gaussian elimination using the **OreGroebnerBasis** command; usually the fastest and most reliable option.

**AbramovZima, Gauss, Zuercher, IncompleteZuercher**: Chyzak’s algorithm, but uses Stefan Gerhold’s implementation [7] of different uncoupling algorithms; his package **OreSys** has to be loaded in advance.

"**Barkatou**": Chyzak’s algorithm, but uses Barkatou’s algorithms [3, 4] to solve the system directly without uncoupling; since we implemented only some cases of these algorithms, this option does not work in most cases.


**Return** → **Automatic**

the value of this option is passed to the command **HermiteTelescoping** when **Method** → "**Hermite**" is used.

**Support** → \{\}

specify the support of the principal part $P$. 

---

15
\[\text{In}[41] := \text{CreativeTelescoping}\left[\text{Binomial}[n,k], S[k] - 1, S[n]\right]\]
\[\text{Out}[41] = \left\{ \left\{ S[n] - 2, \left\{ \frac{k}{k - n - 1} \right\} \right\} \right\}\]

\[\text{In}[42] := \text{CreativeTelescoping}\left[\text{ChebyshevT}[n, 1 - x^2]/\text{Sqrt}[1 - x^2], \text{Der}[x], \{S[n], \text{Der}[y]\}\right]\]
\[\text{Out}[42] = \left\{ \left\{ x^2 - 2 S[n] + \left( 2n^2 - 2ny + y^2 - 2y \right) \text{Der}[y] + \left( 2n^2 - 2ny + 2y \right), \right\} \right\}\]

The following allows us to write down the indefinite integral of the Hermite polynomials, namely \(\int H_n(x) \, dx = H_{n+1}(x)/(2(n+1))\).

\[\text{In}[43] := \text{CreativeTelescoping}\left[\text{HermiteH}[n, x], \text{Der}[x], S[n]\right]\]
\[\text{Out}[43] = \left\{ \left\{ 1, \left\{ -\frac{1}{2(n+1)} S[n] \right\} \right\} \right\}\]

The next example is famous for demonstrating that a recurrence found be creative telescoping need not be the minimal one that exists for the sum (in this instance, the sum for \(k\) from 0 to \(n\) evaluates to \((-3)^n\) and hence has a first-order recurrence.

\[\text{In}[44] := \text{CreativeTelescoping}\left[\left(-1\right)^k \text{Binomial}[n,k] \text{Binomial}[3k,n], \text{Delta}[k], S[n]\right]\]
\[\text{Out}[44] = \left\{ \left\{ -6S[n] + (-15n - 21)S[n] + (-9n - 9), \right\} \right\}\]

\[\text{See Also}\]
Annihilator, FindCreativeTelescoping, HermiteTelescoping, SolveCoupledSystem, Takayama
Delta

\[ \Delta[n] \]
represents the forward difference (delta) operator with respect to \( n \).

\[ \Delta \]

More Information

When this operator occurs in an OrePolynomial object, it is displayed as \( \Delta_n \). The symbol Delta receives its meaning from the definitions of OreSigma, OreDelta, and OreAction.

Examples

```
In[45]:=
OreDelta[Delta[n]]
Out[45]= (\#1 /. n \to n + 1) - \#1 &

In[46]:=
ApplyOreOperator[Delta[k], k!]
Out[46]= (k + 1)! - k!
```

You can use the command \texttt{ChangeOreAlgebra} to switch between the delta and the shift representation of a recurrence:

```
In[47]:=
ToOrePolynomial[Delta[n]^3, OreAlgebra[Delta[n]]]
Out[47]= \Delta^3_n

In[48]:=
ChangeOreAlgebra[%, OreAlgebra[S[n]]]
Out[48]= S^3_n - 3S^2_n + 3S_n - 1
```

See Also

ApplyOreOperator, Der, Euler, OreAction, OreDelta, OreSigma, QS, S, ToOrePolynomial
Der

Der[x]

represents the operator “partial derivative w.r.t. x”.

▼ More Information

When this operator occurs in an OrePolynomial object, it is displayed as $D_x$. The symbol Der receives its meaning from the definitions of OreSigma, OreDelta, and OreAction.

▼ Examples

In[49]:= OreDelta[Der[x]]
Out[49]= $\partial_x #1 &$

In[50]:= ApplyOreOperator[Der[z]~2, f[z]]
Out[50]= f''[z]

The symbol Der itself does not do anything. In order to perform noncommutative arithmetic, it first has to be embedded into an OrePolynomial object.

In[51]:= Der[x] ** x
Out[51]= Der[x] ** x

In[52]:= ToOrePolynomial[Der[x]]
Out[52]= $D_x$

In[53]:= % ** x
Out[53]= $x D_x + 1$

▼ See Also

ApplyOreOperator, Delta, Euler, OreAction, OreDelta, OreSigma, QS, S, ToOrePolynomial
DFiniteDE2RE

DFiniteDE2RE[ann, \{x_1, \ldots, x_d\}, \{n_1, \ldots, n_d\}]
computes recurrences in \(n_1, \ldots, n_d\) for the coefficients of a power series that
is solution to the given differential equations in \(x_1, \ldots, x_d\) in \(ann\).

More Information

The input \(ann\) is an annihilating ideal, given as a list of \(OrePolynomial\) expre-
sions that form a Gröbner basis. It is assumed that the operators \(D_{x_1}, \ldots, D_{x_d}\)
are part of the Ore algebra in which \(ann\) is represented. In the output alge-
bra, these differential operators are replaced by the shift operators \(S_{n_1}, \ldots, S_{n_d}\).
The recurrences for the coefficients of the generating function are obtained by
relatively simple rewrite rules. Finally, a Gröbner basis of the resulting oper-
ators is computed and returned. Alternatively it is conceivable to use CreativeTelescoping or Takayama on Cauchy’s integral formula that extracts
the coefficients of the generating function (see the examples below).

Examples

\begin{verbatim}
In[54]:= DFiniteDE2RE[Annihilator[Log[1 - x], Der[x]], x, n]
Out[54]= \{(−n − 1)S_n + n\}

In[55]:= DFiniteDE2RE[Annihilator[E^(x + y), \{Der[x], Der[y]\}], \{x, y\}, \{m, n\}]
Out[55]= \{(−n − 1)S_n + 1, (−m − 1)S_m + 1\}

In[56]:= DFiniteDE2RE[Annihilator[BesselJ[n, x]], x, m]
Out[56]= \{S_n + (m − n + 1)S_m, (m^2 + 4m − n^2 + 4)S_m^2 + 1\}

In[57]:= CreativeTelescoping[BesselJ[n, x]/x^(m + 1), Der[x], \{S[n], S[m]\}]
Out[57]= \{\{(S_n + (m − n + 1)S_m, (−m^2 − 4m + n^2 − 4)S_m^2 − 1), \{1, S_n + \frac{−m − n − 2}{x}\}\}\}

In[58]:= ann = Annihilator[BesselJ[n, x]/x^(m + 1), \{Der[x], S[n], S[m]\}];

In[59]:= Takayama[ann, \{x\}]
Out[59]= \{S_n + (m − n + 1)S_m, (m^2 + 4m − n^2 + 4)S_m^2 + 1\}
\end{verbatim}

See Also

CreativeTelescoping, DFiniteRE2DE, Takayama
DFiniteOreAction

`DFiniteOreAction[ann, opoly]`

computes an annihilating ideal for the function that is obtained when the operator `opoly` is applied to the function described by `ann` (using the closure property "application of an operator").

More Information

The closure property "application of an operator" reads as follows: Let \( f \) be a function that is \( \partial \)-finite with respect to the operators \( \partial_1, \ldots, \partial_d \); then \( P \cdot f \) is \( \partial \)-finite as well, where \( P \) is an operator in the Ore algebra generated by the \( \partial_1, \ldots, \partial_d \).

In this sense, `ann` is an annihilating ideal (i.e., a list of `OrePolynomial` expression that form a Gr"obner basis) in some Ore algebra \( O \) and `opoly` is an operator in \( O \) (which can be given either as an `OrePolynomial` expression, or as a plain polynomial in the Ore operators of \( O \).

Note that the dimension of the vector space under the stairs of the resulting Gröbner basis is always smaller or equal than the one of the input `ann`. This fact is particularly useful when a sum of two expressions can be written as an operator applied to a single expression; then usually `DFiniteOreAction` delivers bigger annihilating ideals than `DFinitePlus`, see the example below.

This command is called by `Annihilator` if its input contains `D` or `ApplyOreOperator`.

The following options can be given:

- **MonomialOrder** → `None`
  specifies the monomial order in which the output should be given; `None` means that the monomial order of the input is taken.

Examples

Sine and cosine have the same differential equation:

\[
\text{In[60]} := \text{DFiniteOreAction[ToOrePolynomial[{Der[x]^2 + 1}], Der[x]]}
\]

\[
\text{Out}[60] = \{D_x^2 + 1\}
\]

The next example shows a situation where `DFiniteOreAction` is preferable to `DFinitePlus`: find an annihilating ideal for \( P_{n+1}(x) + P_n(x) \).

\[
\text{In[61]} := \text{DFiniteOreAction[Annihilator[LegendreP[n, x], \{S[n], Der[x]\}], S[n] + 1]}\]

\[
\text{Out}[61] = \{(2n^2 + 6n + 4) S_n + (-2n x^2 + 2n - 3 x^2 + 3) D_x + (-2n^2 x - 5n x - n - 3 x - 1),
\]

\[
(x^2 - 1) D_x^2 + (x + 1) D_x + (-n^2 - 2n - 1)\}
\]

\[
\text{In[62]} := \text{UnderTheStaircase[%]}
\]
\[\text{In[62]} = \{1, D_x\}\]

\[\text{In[63]} = \text{DFinitePlus}[	ext{Annihilator}[\text{LegendreP}[n + 1, x], \{S[n], \text{Der}[x]\}]],
\quad	ext{Annihilator}[\text{LegendreP}[n, x], \{S[n], \text{Der}[x]\}]\]

\[\text{Out[62]} = \{1, D_x, S[n], D_x^2\}\]

\[\text{In[64]} = \text{UnderTheStaircase}[]\]

\[\text{Out[62]} = \{1, D_x, S[n], D_x^2\}\]

Sometimes the dimension can even drop:

\[\text{In[65]} = \text{ann} = \text{Annihilator}[\text{HarmonicNumber}[n], S[n]]\]

\[\text{Out[62]} = \{(n + 2)S[n]^2 + (-2n - 3)S[n] + (n + 1)\}\]

\[\text{In[66]} = \text{DFiniteOreAction}[\text{ann}, S[n] - 1]\]

\[\text{Out[62]} = \{(n + 2)S[n] + (-n - 1)\}\]

\[\text{See Also}\]

Annihilator, DFinitePlus, DFiniteSubstitute, DFiniteTimes
DFinitePlus

DFinitePlus[{ann1, ann2, ...}]
computes an annihilating ideal for the sum of the functions described by
ann1, ann2, etc. using the \(\partial\)-finite closure property “sum”.

More Information

The \(ann_i\) are annihilating ideals, given as lists of OrePolynomial expressions
that form Gröbner bases (this property is assumed and not checked separately).
The generators of all the \(ann_i\) have to be elements in the same Ore algebra and
with the same monomial order. Assume that \(ann_i\) annihilates the function \(f_i\),
then the output is an annihilating ideal for \(f_1 + f_2 + \ldots\) (or more precisely for
any linear combination \(c_1 f_1 + c_2 f_2 + \ldots\), again given as a Gröbner basis.

The dimension of the vector space under the stairs of the output equals at most
to the sum of the vector space dimensions of the \(ann_i\). Note also that the output
corresponds to the intersection of the input ideals (which is the left LCM in case
of univariate recurrences or differential equations).

DFinitePlus is called by Annihilator whenever a sum of nontrivial expres-
sions is encountered.

The following options can be given:

MonomialOrder \(\rightarrow\) None
specifies the monomial order in which the output should be given; None
means that the monomial order of the input is taken.

Examples

\[\text{In[67] := \{ann1 = Annihilator[HermiteH[n, x], \{S[n], Der[x]\}]\}\]
\[\text{Out[67] = \{S[n] + Der[x] - 2 x, Der[x]^2 - 2 x Der[x] + 2 n\}\}}\]
\[\text{In[68] := UnderTheStaircase[ann1]}\]
\[\text{Out[68] = \{1, Der[x]\}}\]
\[\text{In[69] := \{ann2 = Annihilator[x^n, \{S[n], Der[x]\}]\}}\]
\[\text{Out[69] = \{x Der[x] - n, S[n] - x\}\}}\]
\[\text{In[70] := UnderTheStaircase[ann2]}\]
\[\text{Out[70] = \{1\}\}}\]
\[\text{In[71] := \{DFinitePlus[ann1, ann2] \}}\]
\[\text{Out[71] = \{(nx - x^3) Der[x]^2 + (n - n^2) S[n] + (-n^2 - 2 n x^2 + n + 2 x^4) Der[x] + (n^2 x - 2 n x^3 - 2 n x),}\]
\[\text{(n - x^3) S[n] Der[x] + (n x + x) S[n] + (n x + x) Der[x] + (-2 n^2 - 2 n),}\]
\[\text{(x^2 - n) S[n]^2 + (4 n x - 3 x^2 + 2 x) S[n] + (2 n x - x^3 + 2 x) Der[x] + (-2 n^2 - 2 n x^2 - 2 n + 2 x^3 - 2 x^2))}\]
\[\text{In[72] := UnderTheStaircase[\%]}\]
\[\text{Out[72] = \{1, Der[x], S[n]\}}\]
The expression \((n + 1)! - n! = n \cdot n!\) is obviously annihilated by a first-order recurrence. But the output of the closure property \textbf{DFinitePlus} is a recurrence that annihilates any linear combination \(c_1(n+1)! + c_2n!\), and hence is of order 2. Note that in this case also \textbf{DFiniteOreAction} could be used to obtain the first-order recurrence.

In[73]:= \textbf{DFinitePlus} @ @ ToOrePolynomial[\{\{S[n] - n - 2\}, \{S[n] - n - 1\}\}]

Out[73]= \{\textit{S}2\ n + (-2n - 4)\textit{S}n + (n^2 + 3n + 2)\}

In[74]:= Annihilator[(n + 1)! - n!, S[n]]

Out[74]= \{\textit{S}2\ n + (-2n - 4)\textit{S}n + (n^2 + 3n + 2)\}

In[75]:= Annihilator[n n!, S[n]]

Out[75]= \{n\textit{S}n + (-n^2 - 2n - 1)\}

In[76]:= \textbf{DFiniteOreAction}[Annihilator[n n!, S[n]], S[n] - 1]

Out[76]= \{n\textit{S}n + (-n^2 - 2n - 1)\}

\textbf{See Also}

Annihilator, \textbf{DFiniteOreAction}, \textbf{DFiniteSubstitute}, \textbf{DFiniteTimes}
DFiniteQSubstitute

\[\text{DFiniteQSubstitute}[\text{ann}, \{q, m\}]\]

computes an annihilating ideal of \(q\)-difference equations for the result of the substitution \(q \rightarrow e^{2i\pi/m}q\).

\[\text{DFiniteQSubstitute}[\text{ann}, \{q, m, k\}]\]

computes an annihilating ideal of \(q\)-difference equations for the result of the substitution \(q \rightarrow e^{2i\pi/m}q^{1/k}\).

\[\text{DFiniteQSubstitute}[\text{ann}, \{\{q_1, m_1, k_1\}, \ldots, \{q_d, m_d, k_d\}\}]\]

doing the same for several such substitutions.

▼ More Information

\(\text{ann}\) is an annihilating ideal, given as a list of \text{OrePolynomial} expressions that form a Gröbner basis. Typically, \(\text{ann}\) consists of \(q\)-difference equations involving the Ore operator \(QS\). The command \text{DFiniteQSubstitute} first finds some elements in \(\text{ann}\) whose coefficients are of a special form: the variable that represents \(q^n_j\) occurs only with powers that are multiples of \(\text{lcm}(m_j, k_j)\) and \(q_j\) itself occurs only with powers divisible by \(k_j\) (for all \(1 \leq j \leq d\) respectively).

Then in these recurrences the substitutions \(q_j \rightarrow e^{2i\pi/m_j}q^{1/k_j}\) can be safely performed (note that performing these substitutions in \(\text{ann}\) directly would in general lead out of the underlying Ore algebra).

Let \(u\) denote the number of monomials under the stairs of \(\text{ann}\), then the output will have at most \(u \prod_{j=1}^{d} (k_j \text{lcm}(m_j, k_j))^{N(j)}\) monomials under the stairs, where \(N(j)\) is the number of operators of the form \(QS[\ldots q_j^\hat{\ldots}]\) in the algebra.

The following options can be given:

\text{ModuleBasis} \rightarrow \{\}

when dealing with annihilating modules, give here a list of natural numbers indicating the location of the position variables among the generators of the Ore algebra. Can be used for dealing with inhomogeneous recurrences.

\text{Return} \rightarrow \text{Annihilator}

specifies the type of the output:

\text{Annihilator}: By default, a Gröbner basis for the annihilating ideal of the resulting sequence is returned.

\text{Backsubstitution}: Returns some elements of \(\text{ann}\) in whose coefficients the powers of \(q^n_j\) are multiples of \(\text{lcm}(m_j, k_j)\) and the powers of \(q_j\) are divisible by \(k_j\). Substituting \(q_j \rightarrow e^{2i\pi/m_j}q^{1/k_j}\) in this result yields exactly the output of \text{Return} \rightarrow \text{Annihilator}.

\text{Support}: Only the support of the annihilating ideal of the resulting sequence is returned.
\[\] \textbf{Examples}

We study a very simple example, the sequence \((q; q)_n:\)
\[
\text{ann} = \text{Annihilator}[\text{QPochhammer}[q, q, n], \text{QS}[qn, q^{-n}]]
\]
\[
\text{ann} = (1 + q)S_{q^n} + (1 + q - q q^n - q^2 q^n^2)
\]

The command \textit{DFiniteQSubstitute} is now employed to study the sequence \((-q; -q)_n\) which corresponds to the substitution \(q \rightarrow e^{2i\pi/3}q\).
\[
\text{ann2} = \text{DFiniteQSubstitute}[\text{ann}, \{q, 2\}]
\]
\[
\text{ann2} = (1 + q)S_{q^n} + (-1 - q)S_{q^n} + (q - q^2 q^n^2)
\]

Now we demonstrate how this procedure can be applied to an inhomogeneous recurrence, e.g.,
\[
f_{n+1}(q) + (q^n + 1)f_n(q) = 0:
\]
\[
\text{rec} = \text{ToOrePolynomial}([[\text{QS}[qn, q^{-n}] - 1]b, \text{QS}[qn, q^{-n}] + q^n + 1 + b]], \text{OreAlgebra}[[\text{QS}[qn, q^{-n}], b]])
\]
\[
\text{rec} = \{(1 + q)S_{q^n} + b, S_{q^n} + 1 + q^n\}
\]

The result again is to be interpreted as an inhomogeneous recurrence:
\[
f_{n+3}(-q) + f_{n+2}(-q) + (q^{2n+3} - q^2)f_{n+1}(-q) + (q^{2n+2} - q^2)f_n(-q) = q^2 - 1.
\]

\[\] \textbf{See Also}

\textit{DFiniteSubstitute, QS}
DFiniteRE2DE

DFiniteRE2DE[ann, \{n_1, \ldots, n_d\}, \{x_1, \ldots, x_d\}]

computes differential equations in \(x_1, \ldots, x_d\) of a generating function whose
coefficients satisfy the recurrences in \(n_1, \ldots, n_d\) that are contained in the
annihilating ideal \(ann\).

\[\text{More Information}\]

\(ann\) is an annihilating ideal, given as a list of \textit{OrePolynomial} expressions
that form a Gröbner basis. It is assumed that the operators \(S_{n_1}, \ldots, S_{n_d}\) are
part of the Ore algebra in which \(ann\) is represented. In the output algebra,
these shift operators are replaced by the differential operators \(D_{x_1}, \ldots, D_{x_d}\).
The differential equations for the generating function are obtained by relatively
simple rewrite rules. In general, they are inhomogeneous, and hence in the next
step they are homogenized. Finally, a Gröbner basis of the resulting operators
is computed and returned. This procedure does not always deliver the maximal
annihilating ideal for the generating function, and therefore it is conceivable to
use \textit{CreativeTelescoping} or \textit{Takayama} for this task (see the example below).

\[\text{Examples}\]

\begin{verbatim}
In[87]:= DFiniteRE2DE[ToOrePolynomial[{S[n] - 1}], n, x]
	now[87] = \{(x - 1)D_x + 1\}

The following example shows how much \textbf{DFiniteRE2DE} can overshoot. With
\textbf{CreativeTelescoping}, we find the maximal annihilating ideal for the generating function.

In[88]:= DFiniteRE2DE[Annihilator[LegendreP[n, x], {S[n], Der[x]}], {n}, {y}]

\begin{verbatim}
now[88] = \{(x - 1)(x + 1)D_x + (n - nxy)y, (x - 1)D_x - ny\}

In[89]:= CreativeTelescoping[LegendreP[n, x]y^n, S[n] - 1, \{Der[x], Der[y]\}]

\begin{verbatim}
now[89] = \{(x^2 - 1)D_x + (x - 1)D_y + x, \(-2xy + y^2 + 1\)D_y + (xy - 1)D_x + x, \(-2xy + y^2 + 1\)D_y + (xy - 1)D_x + x, \(-2xy + y^2 + 1\)D_y + (xy - 1)D_x + x\}

In[90]:= Takayama[Annihilator[LegendreP[n, x]y^n, \{S[n], Der[x], Der[y]\}], {n}]

\begin{verbatim}
now[90] = \{(x^2 - 1)D_x + (x - 1)D_y + x, \(-2xy + y^2 + 1\)D_y + (xy - 1)D_x + x, \(-2xy + y^2 + 1\)D_y + (xy - 1)D_x + x\}
\end{verbatim}
\end{verbatim}

\[\text{See Also}\]

CreativeTelescoping, DFiniteDE2RE, Takayama

26
DFiniteSubstitute

\[ \text{DFiniteSubstitute}[\text{ann}, \text{subs}] \]
computes an annihilating ideal for the function that is obtained by performing the substitutions \( \text{subs} \) on the function described by \( \text{ann} \).

\[ \nabla \text{ More Information} \]

The input \( \text{ann} \) is an annihilating ideal, given as a list of \text{OrePolynomial} expressions that form a Gröbner basis. The second argument \( \text{subs} \) has to be a list of \text{Rule} expressions. \text{DFiniteSubstitute} executes the closure properties “algebraic substitution” for continuous variables and “rational-linear substitution” for discrete variables.

An algebraic substitution is given in the form \( x \to \text{expr} \) where \( x \) is a continuous variable (i.e., \( \text{Der}[x] \) is part of the algebra of \( \text{ann} \)) and where \( \text{expr} \) is algebraic in \( z_1, z_2, \ldots \) (the continuous variables of the output, specified by the option \text{Algebra}), i.e., there exists a polynomial \( p(a, z_1, z_2, \ldots) \) such that \( p(\text{expr}, z_1, z_2, \ldots) = 0 \). Note that the variable \( x \) must not appear on the right-hand side of the substitution rule: substitutions like \( x \to \sqrt{x} \) or \( x \to x + y \) are not admissible. Instead, new variables have to be introduced such as \( x \to \sqrt{z} \) or \( x \to y + z \).

A discrete variable \( n \) (i.e., \( S[n] \) is part of the algebra of \( \text{ann} \)) can be replaced via \( n \to \text{expr} \) where the expression \( \text{expr} \) is rational-linear in the discrete variables \( k_1, k_2, \ldots \) of the output: \( \text{expr} = c + r_1 k_1 + r_2 k_2 + \ldots \) where \( r_i \in \mathbb{Q} \) and \( c \) is an arbitrary constant. Note that for discrete substitutions, the variable to be replaced is allowed to appear on the right-hand side, i.e., substitutions like \( n \to 2n \) or \( n \to (k + n)/2 \) are valid. In cases where the set of variables is not changed, the option \text{Algebra} needs not to be given.

\text{DFiniteSubstitute} is called by \text{Annihilator} whenever a substitution has to be performed.

The following options can be given:

\[ \text{Algebra} \to \text{None} \]
specifies the Ore algebra in which the output should be given; \text{None} means that the Ore algebra of the input is taken.

\[ \text{MonomialOrder} \to \text{None} \]
specifies the monomial order in which the output should be given; \text{None} means that the monomial order of the input is taken.

\[ \nabla \text{ Examples} \]
\[ \text{Input} = \text{ann1} = \text{Annihilator}[	ext{Log}[x], \text{Der}[x]] \]

27
Continuous and discrete substitutions can be performed in one single step:

\[ \text{ann2 = DFiniteSubstitute[ann1, \{ x \to y^2 - z^{(1/3)} \},} \]
\[ \text{Algebra \to OreAlgebra[Der[y], Der[z]]} \]
\[ \text{ann2 = \{ } \]
\[ yD_yD_b + 6zD_y^2 + 6D_y, -36z^2D_b^2 + yD_b - 36zD_b, \]
\[ y^2D_y^2(27y^6z^2 - 27z^3)D_b^2 + (54y^6z - 108z^2)D_b^2 + yD_b + (6y^6 - 54z)D_b \]\n\[ \text{Simplify[ApplyOreOperator[ann2, Log[y^2 - z^{(1/3)}]]]]} \]
\[ \text{ann1 = Annihilator[LegendreP[n, x], \{S[n], Der[x] \},} \]
\[ \text{Algebra \to OreAlgebra[S[n], Der[y]]} \]
\[ \text{ann2 = DFiniteSubstitute[ann1, \{ n \to 3n - k, x \to Sqrt[y^2 - 1] \},} \]
\[ \text{Algebra \to OreAlgebra[S[k], S[n], Der[y]]}; \]
\[ \text{FullSimplify[ApplyOreOperator[ann2, LegendreP[3n - k, Sqrt[y^2 - 1]]]]} \]
\[ \text{A recurrence for (n/2)!:\n}\]
\[ \text{DFiniteSubstitute[ToOrePolynomial[\{S[n] - n - 1 \}], \{ n \to n/2 \}}] \]
\[ \text{The closure property substitution includes computing diagonals:\n}\]
\[ \text{DFiniteSubstitute[Annihilator[Binomial[2n, k], \{S[k], S[n] \}], \{ k \to n \},} \]
\[ \text{Algebra \to OreAlgebra[S[n]]} \]
\[ \text{Annihilator, DFiniteOreAction, DFinitePlus, DFiniteTimes} \]
DFiniteTimes

DFiniteTimes[{\text{ann}_1, \text{ann}_2, \ldots}]
computes an annihilating ideal for the product of the functions described
by \text{ann}_1, \text{ann}_2, etc. using the $\partial$-finite closure property "product".

\begin{itemize}
  \item \textbf{More Information}
  \end{itemize}

The \text{ann}_i are annihilating ideals, given as lists of \text{OrePolynomial} expressions
that form Gröbner bases (this property is assumed and not checked separately).
The generators of all the \text{ann}_i have to be elements in the same Ore algebra and
with the same monomial order. Assume that \text{ann}_i annihilates the function \text{f}_i,
then the output is an annihilating ideal for \text{f}_1 \cdot \text{f}_2 \cdots, again given as a Gröbner
basis.

The dimension of the vector space under the stairs of the output equals at most
to the product of the vector space dimensions of the \text{ann}_i.

\textbf{DFiniteTimes} is called by \textbf{Annihilator} whenever a product of nontrivial ex-
pressions is encountered.

\begin{itemize}
  \item \textbf{Examples}
\end{itemize}

\begin{verbatim}
In[99]:= \text{ann1} = \text{Annihilator}[\text{HermiteH}[n,x], \{S[n], \text{Der}[x]\}]
Out[99]= \{S[n] + \text{Der}[x] - 2x, D_2^2 - 2x D_2 + 2n\}
In[100]:= \text{ann2} = \text{Annihilator}[x^n, \{S[n], \text{Der}[x]\}]
Out[100]= \{x \text{Der}[x] - n, S[n] - x\}
In[101]:= \text{DFiniteTimes}[\text{ann1}, \text{ann2}]
Out[101]= \{S[n] + \text{Der}[x] - (n - 2x), x^2 D_2^2 + (-2nx - 2x^3) D_2 + (n^2 + 4nx^2 + n)\}
In[102]:= \text{UnderTheStaircase}[\%]
Out[102]= \{1, \text{Der}[x]\}
In[103]:= \text{DFiniteTimes} \@ \text{Table}[\text{ann2}, \{10\}]
Out[103]= \{x \text{Der}[x] - 10n, S[n] - x^{10}\}
\end{verbatim}

\begin{itemize}
  \item \textbf{See Also}
\end{itemize}

Annihilator, DFiniteOreAction, DFinitePlus, DFiniteSubstitute, DFiniteTimesHyper
**DFiniteTimesHyper**

**DFiniteTimesHyper[ann, expr]**

computes an annihilating ideal for the product \( f \cdot \text{expr} \), where \( f \) is annihilated by \( \text{ann} \) and \( \text{expr} \) is hypergeometric and hyperexponential in all variables under consideration.

\[
\text{ann, expr}
\]

The input \( \text{ann} \) is a list of **OrePolynomial** expressions (or a single such expression) which—for this special command—need not form a Gröbner basis.

The expression \( \text{expr} \) has to be hypergeometric with respect to all discrete variables of \( \text{ann} \) and hyperexponential with respect to all continuous variables of \( \text{ann} \). Assume that \( \text{ann} \) annihilates some function \( f \), then the output operators annihilate the product \( f \cdot \text{expr} \).

The result is obtained by simple rewrite rules, namely to multiply to each term some rational function that is determined by the exponents of the shift and differential operators and that compensates the rational function that appears by shifting and differentiating \( \text{expr} \). In standard situations where \( \text{ann} \) is given as a Gröbner basis, the command **DFiniteTimes** is preferable.

**Examples**

Of course, **DFiniteTimesHyper** works also when the input is a Gröbner basis. Then it delivers the same output as **DFiniteTimes**.

\[
\text{In}[104] := \text{Out}[104] = \{−S_n − xD_x + (n + 2x^2), x^2D_x^2 + (−2nx − 2x^3)D_x + (n^2 + 4nx^2 + n)\}
\]

But it also works for inputs that do not form a Gröbner basis (note that the elements of \( \text{ann} \) do not even belong to the same Ore algebra):

\[
\text{In}[105] := \text{Out}[105] = \{S_n + xD_x + (−n − 2x^2), x^2D_x^2 + (−2nx − 2x^3)D_x + (n^2 + 4nx^2 + n)\}
\]

**See Also**

Annihilator, **DFiniteOreAction**, **DFinitePlus**, **DFiniteSubstitute**, **DFiniteTimes**
DSolvePolynomial

DSolvePolynomial[eqn, f[x]]
determines whether the ordinary linear differential equation eqn in f[x]
(with polynomial coefficients) has polynomial solutions, and in the affir-
mative case, computes them.

More Information

The first argument eqn can be given either as an equation (with head Equal),
or as the left-hand side expression (which is then understood to be equal to
zero). If the coefficients of eqn are rational functions, it is multiplied by their
common denominator. The second argument is the function to be solved for.
The algorithm works by determining a degree bound and then making an ansatz
for the solution with undetermined coefficients.

The command DSolvePolynomial is able to deal with parameters; these have
to occur linearly in the inhomogeneous part. Call the parameterized version
using the option ExtraParameters.

The following options can be given:

ExtraParameters → {}
specify some extra parameters for which the equation has to be solved.

Examples

In[109]:= DSolvePolynomial[f'[x] == 5, f[x]]

In[110]:= DSolvePolynomial[
   x^2 f''[x] - f'[x] - (2 x - 1) f[x] == -2 x^4 + cx^3 + 3 x^2 + 3 x,
   f[x], ExtraParameters → {c}]
Out[110]= {{f[x] → x^3 - 3 x - 3, c → 7}}

See Also

DSolveRational, QSolvePolynomial, QSolveRational,
RSolvePolynomial, RSolveRational
DSolveRational

\[ \text{DSolveRational}[eqn, f[x]] \]
determines whether the ordinary linear differential equation \( eqn \) in \( f[x] \) has rational solutions, and in the affirmative case, computes them.

\[ \text{More Information} \]

The first argument \( eqn \) can be given either as an equation (with head \texttt{Equal}), or as the left-hand side expression (which is then understood to be equal to zero). If the coefficients of \( eqn \) are rational functions, it is multiplied by their common denominator. The second argument is the function to be solved for. Following Abramov’s algorithm [1], first the denominator of the solution is determined. Then \texttt{DSolvePolynomial} is called to find the numerator polynomial.

The following options can be given:

\[ \texttt{ExtraParameters} \to \{\} \]
specify some extra parameters for which the equation has to be solved; these have to occur linearly in the inhomogeneous part.

\[ \text{Examples} \]

\[ \begin{align*}
\text{\texttt{In[11]}=} & \quad \text{DSolveRational}[x^2 + x f'[x] == f[x], f[x]] \\
\text{\texttt{Out[11]}=} & \quad \{\{f[x] \to C[1] x + 1\}\} \\
\text{\texttt{In[12]}=} & \quad \text{DSolveRational}[(a^2 + 2 a^3 + a^4) f''[a] + (8 a + 21 a^2 + 13 a^3) f''[a] + (12 + 53 a + 44 a^2) f'[a] + (27 + 36 a) f[a], f[a]] \\
\text{\texttt{In[13]}=} & \quad \text{DSolveRational}[u^2 (1 + u^2)^2 f''[u] - u (1 + u^2)(-1 + 2 u + 2 a u + u^2 + 2 u^3 + 2 a u^3) f'[u] - (1 + u + a u + 3 u^2 - 2 a u^2 - a^2 u^2 - 3 u^4 - 4 a u^4 - 2 a^2 u^4 - u^5 - a u^6 - u^6 - 2 a u^6 - a^2 u^8) f[u] == u^2 (1 + u^2)^2 (c[0] - u c[1] + u^2 c[2]), f[u], \texttt{ExtraParameters} \to \{c[0], c[1], c[2]\}] \\
\end{align*} \]

\[ \text{See Also} \]

\texttt{DSolvePolynomial}, \texttt{QSolvePolynomial}, \texttt{QSolveRational}, \texttt{RSolvePolynomial}, \texttt{RSolveRational}
Euler

Euler \[ x \] represents the Euler operator \( \theta_x = xD_x \).

\[ \square \] **More Information**

When this operator occurs in an OrePolynomial object, it is displayed as \( \theta_x \). The symbol Euler receives its meaning from the definitions of OreSigma, OreDelta, and OreAction. Functions like Annihilator cannot deal with mcEuler; use Der[\(x\)] instead to represent differential equations.

\[ \square \] **Examples**

\[ \text{In}[114]:= \text{ToOrePolynomial}[\text{Euler}[x]] \]
\[ \text{Out}[114]= \theta_x \]

\[ \text{In}[115]:= \%^8 \]
\[ \text{Out}[115]= \theta_x^8 \]

\[ \text{In}[116]:= \text{ChangeOreAlgebra}[\%, \text{OreAlgebra}[\text{Der}[x]]] \]
\[ \text{Out}[116]= x^8 D_x^8 + 28 x^7 D_x^7 + 260 x^6 D_x^6 + 1050 x^5 D_x^5 + 1701 x^4 D_x^4 + 966 x^3 D_x^3 + 127 x^2 D_x^2 + x D_x \]

\[ \text{In}[117]:= \text{ChangeOreAlgebra}[\%, \text{OreAlgebra}[\text{Euler}[x]]] \]
\[ \text{Out}[117]= \theta_x^8 \]

\[ \square \] **See Also**

ApplyOreOperator, Delta, Der, OreAction, OreDelta, OreSigma, QS, S, ToOrePolynomial
FGLM

\[ \text{FGLM}[gb, order] \]
transforms the Gröbner basis \( gb \) of a zero-dimensional ideal into a Gröbner basis for this ideal with respect to \( order \), using the FGLM algorithm.

\[ \text{FGLM}[gb, alg, order] \]
translates the input into the new Ore algebra \( alg \) and then performs the FGLM algorithm.

▼ More Information

The input \( gb \) must be a list of OrePolynomial expressions that form a Gröbner basis with respect to the monomial order that is attached to these Ore polynomials (the Gröbner basis property is not checked by FGLM!).

Note that this implementation works only for zero-dimensional ideals.

The following options can be given:

\[ \text{ModuleBasis} \rightarrow \{ \} \]
when you deal with Gröbner bases over a module, give here a list of natural numbers indicating the location of the position variables among the generators of the Ore algebra.

▼ Examples

We start with a left ideal that is computed by Annihilator; hence it is a Gröbner basis with respect to degree order. To extract the pure ODE resp. recurrence, we use the FGLM algorithm to get a Gröbner basis with lexicographic order, first with \( D_x \prec S_n \) and then with \( S_n \prec D_x \).

\[ \begin{align*}
\text{gb} &= \text{Annihilator}[n \text{Sin}[x] + x \text{HarmonicNumber}[n] + 1, \{ S[n], \text{Der}[x] \}] \\
\text{gb} &= \{ (n^2 + 2n)S_n^2 + D_x^2 + (-2n^2 - 3n)S_n + (n^2 + n), \\
xD_x^2 + nxS_n D_x - D_x^2 - nS_n - nxD_x + n, \\
nS_nD_x^2 + (-n - 1)D_x^2 \}
\end{align*} \]

\[ \begin{align*}
\text{FGLM}[\text{gb}, \text{Lexicographic}] \\
\text{FGLM}[\text{gb}, \text{OreAlgebra}[\text{Der}[x], S[n]], \text{Lexicographic}]
\end{align*} \]

\[ \begin{align*}
\text{gb} &= \{ (n^2 + 3)S_n^2 + (−3n - 7)S_n^2 + (3n + 5)S_n + (−n - 1), \\
xD_x S_n^2 - 2x D_x S_n + xD_x - S_n^2 + 2S_n - 1, \\
D_x^2 + (n^2 + 2n)S_n^2 + (−2n^2 - 3n)S_n + (n^2 + n) \}
\end{align*} \]
In the next example, we compute in a module whose elements have 2 entries. The two positions are indicated by the position variables $p_0$ and $p_1$. Among the generators of the Ore algebra, they sit on position 1 and 2.

```
In[121]:= ToOrePolynomial[{-np_0 + 2p_1, (2 + 2n)p_1 - np_1S[n]}, OreAlgebra[p_0, p_1, S[n]]]
Out[121]= {-np_0 + 2p_1, -np_1S[n] + (2n + 2)p_1}
```

```
In[122]:= FGLM[%, OreAlgebra[p_1, p_0, S[n]], Lexicographic, ModuleBasis -> {1, 2}]
Out[122]= {p_0S[n] - 2p_0, 2p_1 - np_0}
```

<table>
<thead>
<tr>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnnihilatorDimension, OreGroebnerBasis, GBEqual</td>
</tr>
</tbody>
</table>

35
FindCreativeTelescoping

FindCreativeTelescoping[expr, deltas, ops]
finds creative telescoping relations for expr by making an ansatz with explicit (heuristically determined) denominators in the delta parts. With this command multiple summations and integrations can be done in one step by giving several deltas.

FindCreativeTelescoping[ann, deltas]
finds creative telescoping relations in the annihilating ideal ann.

More Information

The first argument is either an annihilating ideal (i.e., a Gröbner basis of such an ideal) or any mathematical expression. In the latter case, Annihilator is internally called with expr. The second argument deltas indicates which summations and integrations have to be performed; this means Sz[a] – 1 (resp. QSZ[q, q^a] – 1) for (q-) summation w.r.t. the variable a and Der[a] for integration w.r.t. a. If expr is given, then the third argument ops specifies the surviving Ore operators, i.e., the operators that occur in the principal part (as well as in the delta part). For a more detailed explanation of the method of creative telescoping, see CreativeTelescoping, p. 14. For a more detailed description of how the ansatz used in this command is constructed, see the paper [10].

The output consists of two lists \{L1, L2\}, where L1 contains all the principle parts (such that they constitute a Gröbner basis), and L2 contains the corresponding delta parts. In particular, L1 and L2 have the same length, and the i-th element of L2 is the list of delta parts (corresponding to the given deltas) for the i-th principal part in L1.

The following options can be given:

Mode → Automatic
This command can be used in different ways:

Automatic: everything is done automatically, i.e., the minimal ansatz is determined in a homomorphic image and then the solution is computed in the original domain.

FindSupport: only modular computations are done and the minimal ansatz is returned as a list of options that can be given to FindCreativeTelescoping again in order to perform the final computation or a modular one.

Modular: this mode requires to specify an ansatz (in the way as it is returned by Mode → FindSupport) and is supposed to be used together with the options OrePolynomialSubstitute, Modulus, and FileNames.
Support → Automatic
Specify the support of the principal part. By default, FindCreativeTelescoping loops over the support until a set of principal parts is found that generates a zero-dimensional ideal.

Degree → Automatic
Specify the degree of the summation and integration variables to be used in the numerators of the delta parts. By default, FindCreativeTelescoping loops over the degree up to a heuristically determined degree bound. If the support is given with the option Support then it loops ad infinitum (if no creative telescoping operator is found).

Denominator → Automatic
By default the denominators of the delta parts are determined automatically. Setting Denominator → d for a polynomial d uses d as denominator in each delta part. Setting Denominator → \{\{d_{11}, \ldots, d_{1n}\}, \ldots, \{d_{m1}, \ldots, d_{mn}\}\} uses \(d_{ij}\) as the denominator of the \(j\)-th term in the \(i\)-th delta part; hence \(n\) is the number of monomials under the stairs of the input Gröbner basis and \(m\) is the number of deltas.

Modulus → 0
All computations are done modulo this number. In connection with the option Mode → Modular, setting Modulus → \{p_1, \ldots, p_n\} computes the result \(n\) times modulo all the prime numbers \(p_i\).

OrePolynomialSubstitute → {} If a list of rules \(\{a \rightarrow a_0, b \rightarrow b_0, \ldots\}\) is given, then the result is computed with these substitutions (the variables \(a, b, \ldots\) must not be summation or integration variables). In connection with the option Mode → Modular one can give a list of such substitutions: \(\{\{a \rightarrow a_1, b \rightarrow b_1, \ldots\}, \{a \rightarrow a_2, b \rightarrow b_2, \ldots\}, \ldots\}\). Then the computation is done for each of these substitutions.

"Ansatz" → Automatic
To be used in connection with Mode → Modular, and the value of this option is best produced using Mode → FindSupport.

Variables → {}
To be used in connection with Mode → Modular, and the value of this option is best produced using Mode → FindSupport.

"DenominatorName" → None
To be used in connection with Mode → Modular, and the value of this option is best produced using Mode → FindSupport.

FileNames → ""
When using Mode → Modular then this option can give a StringForm to specify the locations where all the results have to be stored. Each variable in the substitution list (see option OrePolynomialSubstitute), requires a wild-card, and an additional wild-card is needed if several prime numbers are given in option Modulus. See the example below.
Examples

In[123]:= FindCreativeTelescoping[Binomial[n, k], S[k] - 1, S[n]]

Out[123]= {{-S[n] + 2}, {{k/(k - n) - 1}}}

When using uncoupling to compute creative telescoping relations (as it is done in CreativeTelescoping), then the following example is very hard to solve, but using FindCreativeTelescoping, it is solved in a few seconds:

In[124]:= ann = Annihilator[
   x BesselJ[1, ax] BesselI[1, ax] BesselY[0, x] BesselK[0, x],
   {Der[x], Der[a]}];

In[125]:= Timing[First[FindCreativeTelescoping[ann, Der[x]]]]

Out[125]= {10.7167, {a D[a] + 2}}

Compute a creative telescoping relation for the Andrews-Paule double sum:

In[126]:= FindCreativeTelescoping[
   Binomial[i + j, i]^2 Binomial[4n - 2i - 2j, 2n - 2i],
   {S[i] - 1, S[j] - 1, S[n]}]

Out[126]= {{1}, {2i^2 j + i^2 n - i^2 - 2ij^2 + 3ijn - 2ij + 3in,
   (j + 1)(i + j - 2n),
   -2i^2 j - 2ij^2 + 3ijn - 2ij + j^2 n - j^2 + 3jn
   (i + 1)(i + j - 2n)}}

The following example illustrates the use of the options FindSupport and Modular. Assume ann is the annihilating ideal for some function $f(n, k)$ that is to be summed over $k$, and the direct computation of the creative telescoping relation is too expensive (e.g., requires more memory than available). Then the following commands can be used to generate the data that is necessary for reconstructing the solution from homomorphic images. In the example, 80 interpolation points and 10 prime numbers are used, so in total 800 files will be stored in the given directory. The reconstruction itself has to be done by other means (this functionality is not provided by FindCreativeTelescoping).

In[127]:= ans = FindCreativeTelescoping[ann, S[k] - 1, Mode -> FindSupport];

In[128]:= FindCreativeTelescoping[ann, S[k] - 1, Mode -> Modular,
   Sequence @@ ans, FileNames -> "/temp/fct_1`2.m",
   OrePolynomialSubstitute -> Table[{{n -> n0}, {n0, 20, 99}}],
   Modulus -> Table[NextPrime[2^31, -i], {i, 10}]]

See Also

Annihilator, CreativeTelescoping, FindRelation, Printlevel
FindRelation

FindRelation[ann, opts]
computes relations with certain properties (that have to be specified by the
options opts) in the annihilating ideal ann.

More Information

The input ann is a list of OrePolynomial expressions that have to form a
Gröbner basis (this property is not checked by FindRelation). It then makes
an ansatz with undetermined coefficients in order to find elements in ann that
obey the constraints that have been given in opts. In particular, this com-
mand can be used for elimination (e.g., when executing Zeilberger’s slow algo-
rithm [12]) and it is usually faster than elimination via Gröbner basis compu-
tation.

The following options can be given:

Eliminate → {}
forces the coefficients to be free of the given variables; note that this option
refers only to the coefficients, not to the generators of the Ore algebra (these
can be influenced by the option Support).

ModuleBasis → {}
when ann is a Gröbner basis over a module, give here a list of natural num-
bers indicating the location of the position variables among the generators
of the Ore algebra.

Pattern →
include only monomials into the ansatz whose exponent vectors match the
given pattern.

Support → Automatic
in the default setting the total degree of the support is increased until some-
thing is found; alternatively you can give a list of power products (of the
generators of the Ore algebra) that are used in the ansatz.

Examples

Find contiguous relations of the 2F1 hypergeometric function:

\[\text{Annihilator}[\text{Hypergeometric2F1}[a, b, c, x], \{S[a], S[b], S[c], \text{Der}[x]\}]\]

\[\{(ab - ac + bc - c^2)S_c + (c - cx)D_x + (-ac - bc + c^2),\]
\[bS_c - xD_x - b,\]
\[aS_a - xD_x - a,\]
\[(x^2 - x)D_x^2 + (ax + bx - c + x)D_x + ab\}\]

\[\text{FindRelation}[\%, \text{Support → \{1, S[a], S[c]\}}\]

\[\{(acx - ac)S_a + (abx - acx - bcx + c^2 x)S_c + (ac + bcx + c^2 (-x))\}\]
For solving Strang’s integral \( \int_1^2 (P_{2k+1}(x)/x)^2 \, dx \) with Zeilberger’s slow algorithm, we would like to eliminate \( x \) from the annihilating ideal of the integrand (of course, this is not the best way to solve this integral, see CreativeTelescoping, p. 14). The output is of considerable size; that’s why we display only its support to illustrate why FindRelation takes some time here.

\[\text{Out}\[134\] = \text{Annihilator}[(\text{LegendreP}[2k + 1, x]/x)^2, \{\text{Der}[x], \text{S}[k]\}];\]

\[\text{Timing}[\text{rel} = \text{First}[\text{FindRelation}[\text{ann}, \text{Eliminate } \rightarrow x]]];\]

\[\text{Out}\[133\] = \{80.169, \text{Null}\}\]

\[\text{ByteCount}[\text{rel}]\]

\[\text{Out}\[135\] = 118360\]

\[\text{Support}[\text{rel}]\]

\[\text{Out}\[136\] = \{D^6_s, S^6_k, D^6_s S^6_k, D^6_s S^5_k, D^6_s S^4_k, D^6_s S^3_k, D^6_s S^2_k, D^6_s S^1_k, D^6_s S^0_k, D^6_s, S^6_k, S^5_k, S^4_k, S^3_k, S^2_k, S^1_k, S^0_k, D^5_s, S^5_k, S^4_k, S^3_k, S^2_k, S^1_k, S^0_k, D^4_s, S^4_k, S^3_k, S^2_k, S^1_k, S^0_k, D^3_s, S^3_k, S^2_k, S^1_k, S^0_k, D^2_s, S^2_k, S^1_k, S^0_k, D^1_s, S^1_k, S^0_k, D^0_s, \}

The next example shows how relations for basis functions can be found that are needed in finite element methods. In this instance, we are looking for a relation that involves the function itself and its first derivative (both can be arbitrarily shifted) and whose coefficients are free of \( x \) and \( y \).

\[\text{Out}\[137\] = \text{Factor}[\text{FindRelation}[\text{Annihilator}[(1 - x)^{-i} \text{LegendreP}[i, 2(y/(1 - x)) - 1] \text{JacobiP}[j, 2i + 1, 0, 2x - 1], \{\text{S}[i], \text{S}[j], \text{Der}[x]\}], \text{Eliminate } \rightarrow \{x, y\}, \text{Pattern } \rightarrow \{\_, \_, 0(1)\}]\]

\[\text{Out}\[138\] = \{-2(i + j + 5)(2i + 2j + 5)S_j^0D_x - (j + 3)(2i + 2j + 5)S_j^0D_x - 2(2i + 3)(i + j + 3)S_j^0D_x + 2(2i + 1)(i + j + 3)S_j^0D_x + 2(i + j + 3)(2i + 2j + 5)(2i + 2j + 7)S_j^0 + (j + 1)(2i + 2j + 7)S_j^0D_x + 2(i + j + 3)(2i + 2j + 5)(2i + 2j + 7)S_j^0 + (j + 3)(2i + 2j + 7)S_j^0D_x\}\]

\[\text{SEE ALSO}\]

Annihilator, FindCreativeTelescoping, FindSupport, OreGroebnerBasis
FindSupport

FindSupport[ann, opts]
computes only the support of a relation in the ideal ann that satisfies the
constraints given by opts.

More Information

The input ann is a list of OrePolynomial expressions that have to form a
Gröbner basis (this property is not checked by FindSupport). It then makes
an ansatz with undetermined coefficients in order to find elements in ann that
obey the constraints that have been given in opts. In contrast to FindRelation
it computes only the support of such elements. In fact, FindSupport is called
by FindRelation if the support is not specified.

Especially for bigger examples, one might be interested in the support of the
result before it is actually computed, in order to get a feeling for its size. On the
other hand, FindSupport can be used to make statements of the form “there
exists no \( k \)-free recurrence of order up to 100”; use Printlevel for switching to
verbose mode in order to see which supports have unsuccessfully been tried.

FindSupport uses homomorphic images for fastly obtaining its results. In
unlucky cases this can lead to fake solutions.

The following options can be given:

Eliminate \[ \rightarrow \{\} \]
forces the coefficients to be free of the given variables; note that this option
refers only to the coefficients, not to the generators of the Ore algebra.

ModuleBasis \[ \rightarrow \{\} \]
when ann is a Gröbner basis over a module, give here a list of natural num-
bers indicating the location of the position variables among the generators
of the Ore algebra.

Pattern \[ \rightarrow \_\]
include only monomials into the ansatz whose exponent vectors match the
given pattern.

Modulus \[ \rightarrow 2147483629 \]
the prime number with respect to which the modular computations are
done.

Examples

See FindRelation for more information about the following example. Note
also that computing the full relation there takes 80 seconds, whereas finding the
support only is much faster.
\textbf{In[136]} := ann = Annihilator[(LegendreP[2k + 1, x]/x)^2, \{Der[x], S[k]\}];
\textbf{In[137]} := Timing[FindSupport[ann, Eliminate \rightarrow x]]
\textbf{Out[137]} = {3.10019, 
\textbf{For more examples, see } \textbf{FindRelation}; \textbf{ its use is very similar to } \textbf{FindSupport}.  
\[\text{\textbf{\textcopyright \quad See Also}}\]
Annihilator, FindCreativeTelescoping, FindRelation, 
OreGroebnerBasis, Printlevel, Support

42
GBEqual

\[ \text{GBEqual}[gb_1, gb_2] \]

compares two Gröbner bases \( gb_1 \) and \( gb_2 \) whether they are the same.

More Information

The two Gröbner bases \( gb_1 \) and \( gb_2 \) have to be given with respect to the same monomial order, and they have to be completely reduced. Both are lists of OrePolynomial expressions. \texttt{GBEqual} does not perform any advanced mathematics: it just goes through the two lists and checks whether either the sum or the difference of the corresponding elements gives 0. Hence it gives True if the corresponding Gröbner basis elements differ only by sign. Nevertheless, it is very useful, since this test cannot be done by using \texttt{Equal} (==) or \texttt{SameQ} (===).

Examples

\begin{verbatim}
In[138]:= gb1 = Annihilator[Sum[Binomial[n,k] Binomial[m+n-k,n], {k,0,n}], {S[m], S[n]}]
Out[138]= {\((m+1)S_m + (-n-1)S_n + (m-n), (n+2)S_n^2 + (-2m-1)S_n + (-n-1)\)}
In[139]:= gb2 = Annihilator[Sum[2^k Binomial[m,k] Binomial[n,k], {k,0,n}], {S[m], S[n]}]
Out[139]= \{(m+1)S_m + (-n-1)S_n + (m-n), (-n-2)S_n^2 + (2m+1)S_n + (n+1)\}
In[140]:= gb1 === gb2
Out[140]= False
In[141]:= FullSimplify[gb1 == gb2]
Out[141]= \{0, \((2(n+2)S_n^2 + (-4m-2)S_n - 2(n+1))\) \} == \{0, 0\}
In[142]:= GBEqual[gb1, gb2]
Out[142]= True
In[143]:= gb3 = FGLM[gb2, Lexicographic]
Out[143]= \{(n+2)S_n^2 + (-2m-1)S_n + (-n-1), (m+1)S_m + (-n-1)S_n + (m-n)\}
In[144]:= GBEqual[gb1, gb3]
Out[144]= False
\end{verbatim}

See Also

\texttt{FGLM, OreGroebnerBasis, OrePolynomial}
HermiteTelescoping

HermiteTelescoping[expr, Der[y], Der[x]]
computes telescoper and certificate for a hyperexponential function expr
using the algorithm Hermite telescoping.

HermiteTelescoping[ann, Der[y], Der[x]]
computes a creative telescoping relation in the annihilating ideal ann (which
must be the annihilator of a hyperexponential function).

\[ \text{More Information} \]

For general information about creative telescoping, see CreativeTelescoping.
The algorithm Hermite telescoping has been developed in [5]. The output is of
the form \{(t), \{c\}\} where t is the telescoper and c the certificate (the output
format is the same as in CreativeTelescoping).

The following options can be given:

Return \rightarrow \text{Automatic}
returns by default the above form \{(t), \{c\}\}, "telescoper" returns \{t\} (and
is faster since the certificate is not computed at all), and "bound" returns
an integer, the precomputed bound for the order of the telescoper.

\[ \text{Examples} \]

\[ \begin{align*}
\text{In[145]} &= \text{HermiteTelescoping}[\text{Exp}[xy + x^2y + y^2x^2], \text{Der}[y], \text{Der}[x]] \\
\text{Out[145]} &= \{\{2xyD_x + (x^2 + x + 2)\}, \{-2y - 1\}\} \\
\text{In[146]} &= \text{HermiteTelescoping}[\text{E}^{(xy^10 + y)}, \text{Der}[y], \text{Der}[x], \text{Return} \rightarrow \text{"bound"}] \\
\text{Out[146]} &= 9
\end{align*} \]

\[ \text{See Also} \]

CreativeTelescoping, FindCreativeTelescoping
LeadingCoefficient

LeadingCoefficient[opoly]
gives the leading coefficient of the Ore polynomial opoly.

More Information

The input opoly has to be an OrePolynomial expression. What is considered
as the leading coefficient depends on the Ore algebra in which opoly is pre-
sented and of course on the monomial order. The leading coefficient of the zero
polynomial is not defined; LeadingCoefficient returns Indeterminate in this
case.

Examples

In[47]:= opoly = ToOrePolynomial[(1 - x^2) ** Der[x]^2 + x ** Der[x] + 4x^3,
OreAlgebra[Der[x]]]

Out[47]= (1 - x^2) D_x^2 + x D_x + 4x^3

In[48]:= LeadingCoefficient[opoly]

Out[48]= 1

In[49]:= opoly = ChangeOreAlgebra[opoly, OreAlgebra[x, Der[x]]]

Out[49]= -x^2 D_x^2 + 4x^3 + x D_x + D_x^2

In[50]:= LeadingCoefficient[opoly]

Out[50]= -1

In[51]:= opoly = ChangeMonomialOrder[opoly, Lexicographic]

Out[51]= 4x^3 - x^2 D_x^2 + x D_x + D_x^2

In[52]:= LeadingCoefficient[opoly]

Out[52]= 4

See Also

ChangeMonomialOrder, ChangeOreAlgebra, LeadingExponent, LeadingPowerProduct, LeadingTerm, NormalizeCoefficients, OrePolynomial, OrePolynomialListCoefficients, ToOrePolynomial, Support
LeadingExponent

LeadingExponent[opoly]
gives the exponent vector of the leading term of the Ore polynomial opoly.

More Information

The input opoly has to be an OrePolynomial expression. What is considered as the leading term depends on the Ore algebra in which opoly is presented and of course on the monomial order. To match the definition of the degree of the zero polynomial, the leading exponent of the zero polynomial is a list of $-\infty$'s, its length corresponding to the number of variables (generators of the Ore algebra).

Examples

In[153]:= opoly = ToOrePolynomial[(1 - x^2) Der[x]^2 + x Der[x] + 4 x^3, OreAlgebra[Der[x]]]
Out[153]= (1 - x^2) D^2 x + x D x + 4 x^3

In[154]:= LeadingExponent[opoly]
Out[154]= {2}

In[155]:= opoly = ChangeOreAlgebra[opoly, OreAlgebra[x, Der[x]]]
Out[155]= -x^2 D_x^2 + 4 x^3 + x D_x + D_x^2

In[156]:= LeadingExponent[opoly]
Out[156]= {2, 2}

In[157]:= opoly = ToOrePolynomial[a b S[a] S[b], OreAlgebra[a, b, S[a], S[b]]]
Out[157]= a b S_S

In[158]:= LeadingExponent[opoly]
Out[158]= {1, 1, 1, 1}

In[159]:= LeadingExponent[0 * opoly]
Out[159]= {-\infty, -\infty, -\infty, -\infty}

See Also

ChangeMonomialOrder, ChangeOreAlgebra, LeadingCoefficient, LeadingPowerProduct, LeadingTerm, NormalizeCoefficients, OrePolynomial, OrePolynomialListCoefficients, ToOrePolynomial, Support
LeadingPowerProduct

LeadingPowerProduct[opoly]
gives the leading power product of the Ore polynomial opoly.

More Information

The input opoly has to be an OrePolynomial expression. What is considered as the leading term depends on the Ore algebra in which opoly is presented and of course on the monomial order. The leading power product of the zero polynomial is not defined; LeadingPowerProduct returns Indeterminate in this case.

Examples

In[160]:= opoly = ToOrePolynomial[(1 - x^2) ** Der[x]^2 + x ** Der[x] + 4 x^3]
Out[160]= (1 - x^2) D^2_x + x D_x + 4 x^3

In[161]:= LeadingPowerProduct[opoly]
Out[161]= D^2_x

In[162]:= opoly = ChangeOreAlgebra[opoly, OreAlgebra[x, Der[x]]]
Out[162]= -x^2 D^2_x + 4 x^3 + x D_x + D^2_x

In[163]:= LeadingPowerProduct[opoly]
Out[163]= x^2 D^2_x

In[164]:= Annihilator[StruveH[n, x], {S[n], Der[x]}]
Out[164]= {x^2 D^2_x + (-2 n x - x) S[n] - 2 n x D_x + (n^2 + n + x^2) D_x + (n + 1) S[n] - x,
  (2 n x + 3 x) S[n] + (-4 n^2 - 10 n - x^2 - 6) S[n] - x^2 D_x + (3 n x + 3 x) D_x}

In[165]:= LeadingPowerProduct[Annihilator[StruveH[n, x], {S[n], Der[x]}]]
Out[165]= {D^2_x, S[n] D_x, S[n]^3}

See Also

ChangeMonomialOrder, ChangeOreAlgebra, LeadingCoefficient, LeadingExponent, LeadingTerm, NormalizeCoefficients, OrePolynomial, OrePolynomialListCoefficients, ToOrePolynomial, Support
LeadingTerm

**LeadingTerm[opoly]**

gives the leading term of the Ore polynomial *opoly*.

\[\text{LeadingTerm}[\text{opoly}]\]

The input *opoly* has to be an **OrePolynomial** expression. What is considered as the leading term depends on the Ore algebra in which *opoly* is presented and of course on the monomial order. By “leading term” we understand the leading power product (or leading monomial) multiplied by the leading coefficient. The leading term of the zero polynomial is not defined; **LeadingTerm** returns **Indeterminate** in this case.

\[\text{LeadingTerm}[\text{opoly}]\]

\[\text{Indeterminate}\]

**Examples**

\[\text{In}[166]:= \text{opoly} = \text{ToOrePolynomial}[(1 - x^2)^2 \text{Der}[x]^2 + x \text{Der}[x] + 4x^3, \text{OreAlgebra}[\text{Der}[x]]]\]

\[\text{Out}[166]= (1 - x^2)\text{Der}^2 x + x \text{Der} x + 4x^3\]

\[\text{In}[167]:= \text{LeadingTerm}[\text{opoly}]\]

\[\text{Out}[167]= (1 - x^2)\text{Der}^2 x\]

\[\text{In}[168]:= \text{opoly} = \text{ChangeOreAlgebra}[\text{opoly}, \text{OreAlgebra}[x, \text{Der}[x]]]\]

\[\text{Out}[168]= -x^2 \text{Der}^2 x + 4x^3 + x \text{Der} x + \text{Der}^2 x\]

\[\text{In}[169]:= \text{LeadingTerm}[\text{opoly}]\]

\[\text{Out}[169]= -x^2 \text{Der}^2 x\]

\[\text{In}[170]:= \text{Annihilator}[\text{StruveH}[n, x], \{S[n], \text{Der}[x]\}]\]

\[\text{Out}[170]= \{x^2 \text{Der}^2 x + (-2nx - x)S_n - 2nx \text{Der} x + (n^2 + n + 1)S_n - x, xS_n \text{Der} x + (n^2 + n + 1)S_n - x, (2nx + 3x)S_n^2 + (-4n^2 - 10n - x^2 - 6)S_n - x^2 \text{Der} x + (3nx + 3x)\}\]

\[\text{In}[171]:= \text{LeadingTerm}[\%]\]

\[\text{Out}[171]= \{x^2 \text{Der}^2 x, xS_n \text{Der} x, (2nx + 3x)S_n^2\}\]

**See Also**

ChangeMonomialOrder, ChangeOreAlgebra, LeadingCoefficient, LeadingExponent, LeadingPowerProduct, NormalizeCoefficients, OrePolynomial, OrePolynomialListCoefficients, ToOrePolynomial, Support
NormalizeCoefficients

**NormalizeCoefficients[opoly]**

removes the content of the Ore polynomial *opoly.*

\[\text{\text{\textup{\texttt{NormalizeCoefficients}}}}\text{\texttt{[opoly]}}}\]

**More Information**

The input *opoly* has to be an *OrePolynomial* expression. The output is a multiple of the input such that all denominators are cleared and the content is 1. When the option **Extended** is used, then the output is a list, containing the content and the normalized Ore polynomial.

The following options can be given:

**Denominator \rightarrow True**

setting this option to **False** deactivates the computation of a common denominator; can be used for better performance when it is known in advance that the input has polynomial coefficients.

**Extended \rightarrow False**

when you also need the content that has been removed from *opoly* then set this option to **True**; the output then is of the form \{c, op\} where c is the content (given as a standard Mathematica expression) and p is the normalized Ore polynomial (given as an *OrePolynomial* expression).

**Modulus \rightarrow 0**

if a prime number is given with this option, then all polynomial operations (like *PolynomialGCD* or *Together*) are performed modulo this prime.

**Examples**

\[n_{[172]} = \text{opoly} = \text{Together}[\text{ToOrePolynomial[}(n(n - 1))**S[n]**2 + ((n - 1)/(n + 1))**S[n] + n**2 - 1]]\]

\[n_{[173]} = \text{NormalizeCoefficients[opoly]}\]

\[n_{[174]} = \text{NormalizeCoefficients[opoly, Extended \rightarrow True]}\]

**See Also**

*OrePolynomialListCoefficients, ToOrePolynomial*
**OreAction**

\[ \text{OreAction}[op] \]

defines how the Ore operator \( op \) acts on arbitrary expressions.

More Information

An Ore operator like \( S_n \) or \( D_x \), in the first place is defined by the two endomorphisms \( \sigma \) and \( \delta \). But also the action of this operator has to be fixed; usually it is either equal to \( \sigma \) (e.g., in the case of a shift operator) or to \( \delta \) (e.g., in the case of a differential operator). The definition of \texttt{OreAction} is used by \texttt{ApplyOreOperator}.

The standard Ore operators (shift, differential, delta, Euler, \( q \)-shift) are pre-defined in \textit{HolonomicFunctions} (using \texttt{OreSigma}, \texttt{OreDelta}, and \texttt{OreAction}). If you want to define your own Ore operators, use \texttt{OreAction} to define how they should act on expressions. Note that \( op \) can be a pattern as well as a fixed expression.

Examples

\begin{verbatim}
In[175]:= OreAction[Der[x]]
Out[175]= \( \partial_x \) \#1 &
In[176]:= OreSigma[MyOp] = MySigma;
In[177]:= OreDelta[MyOp] = MyDelta;
In[179]:= ToOrePolynomial[MyOp^2]
Out[179]= MyOp^2
In[180]:= ApplyOreOperator[%, a + 1]
Out[180]= MyAction[MyAction[a + 1]]
\end{verbatim}

Now we introduce the double-shift, i.e., the shift by 2.

\begin{verbatim}
In[181]:= OreSigma[S2[a_]] := \#/ a \to a + 2 &;
In[182]:= OreDelta[S2[a_]] := 0 &;
In[183]:= OreAction[S2[a_]] := OreSigma[S2[a]]; In[184]:= ToOrePolynomial[S2[n] + n^2]
Out[184]= S2n + n^2
In[185]:= ApplyOreOperator[%, f[n]]
Out[185]= n^2 f(n) + f(n + 2)
\end{verbatim}

See Also

\texttt{ApplyOreOperator}, \texttt{OreDelta}, \texttt{OreOperatorQ}, \texttt{OreOperators}, \texttt{OreSigma}
<table>
<thead>
<tr>
<th>OreAlgebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>OreAlgebra([g_1, g_2, \ldots])</td>
</tr>
<tr>
<td>creates an Ore algebra with the generators (g_1, g_2, \ldots).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OreAlgebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>OreAlgebra([o\text{poly}])</td>
</tr>
<tr>
<td>gives the Ore algebra in which the Ore polynomial (o\text{poly}) is represented.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OreAlgebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>OreAlgebra([a\text{nn}])</td>
</tr>
<tr>
<td>gives the Ore algebra in which the annihilating ideal (a\text{nn}) is represented.</td>
</tr>
</tbody>
</table>

More Information

In the above specification, \(o\text{poly}\) is assumed to be an OrePolynomial expression and \(a\text{nn}\) to be a list of such. The elements of \(a\text{nn}\) usually will be represented in the same Ore algebra, so this algebra will be returned. However, if you give a list of OrePolynomial expressions that do not share the same algebra, you get back a list of Ore algebras, corresponding to the elements of the input \(a\text{nn}\).

The command OreAlgebra is also used to create an Ore algebra. For this purpose, the generators of the desired Ore algebra have to be given as arguments. By the “generators of an Ore algebra” we understand all variables and Ore operators that occur polynomially. An Ore algebra in general has the form \(K(x_1, \ldots, x_k)[y_1, \ldots, y_n][\partial_1; \sigma_1, \delta_1] \cdots [\partial_d; \sigma_d, \delta_d]\); the generators of this Ore algebra are \(y_1, \ldots, y_n, \partial_1, \ldots, \partial_d\). Note that the Ore operators \(\partial_i\) always have to be included into the set of generators (no division by those is allowed!).

The endomorphisms \(\sigma_i\) and \(\delta_i\) are not given explicitly when creating an Ore algebra. They have to be “attached” to the symbols \(\partial_i\) before, using the commands OreSigma, OreDelta, and OreAction. For standard applications this is not necessary since the common Ore operators are already predefined.

The order in which the generators \(g_1, g_2, \ldots\) are given is used when an Ore polynomial is represented in this algebra: all power products are written according to this order. For example, when dealing with the Weyl algebra, you can choose whether \(x\) appears always on the left and \(D_x\) on the right (the standard way), or the other way round. The coefficients (in the general form as stated above they are elements in \(K(x_1, \ldots, x_k)\)) are always written on the left.

When working in a module over some Ore algebra, use position variables: to represent the element \((T_1, \ldots, T_m), T_i \in \mathcal{O}\), write it either as \(p_1 T_1 + \cdots + p_m T_m\) with new indeterminates \(p_1, \ldots, p_m\) (“position variables”), or as \(p T_1 + p^2 T_2 + \cdots + p^m T^m\) with a single position variable \(p\). The position variables have to be added to the generators of \(\mathcal{O}\) and their positions among the generators have to be given with the option ModuleBasis to the relevant commands that can deal with modules (OreGroebnerBasis, FGLM, FindRelation, etc.).

An Ore algebra is displayed using the standard mathematical notation for such algebras, but internally it is represented as a Mathematica expression of the form OreAlgebraObject\([gens, cNormal, cPlus, cTimes, ext]\). Herein, \(gens\) is a list containing the generators of the algebra. The next three arguments are
functions that determine the coefficient arithmetic: \textit{cNormal} specifies in which form the coefficients should be kept (e.g., in expanded or factored form); in particular, \textit{cNormal} is supposed to yield 0 whenever a coefficient is actually zero (recall that expressions like $1 - \frac{x}{x+1} - \frac{1}{x+1}$ are not automatically simplified to 0). The functions \textit{cPlus} and \textit{cTimes} are used for adding resp. multiplying two coefficients. Finally, \textit{ext} tells whether to work in an algebraic extension.

For easily setting the coefficient representation, certain upvalues have been defined for Ore algebras, e.g. \textit{Together[alg]} changes the functions \textit{cNormal}, \textit{cPlus}, and \textit{cTimes} to \textit{Together}. Analogously for \textit{Expand} and \textit{Factor}.

The following options can be given:

\textbf{CoefficientNormal} $\rightarrow$ \textit{Expand}

specifies in which form the coefficients of Ore polynomials in this algebra are represented.

\textbf{CoefficientPlus} $\rightarrow$ (\#1 + \#2 &)

specifies how to add two coefficients of Ore polynomials in this algebra.

\textbf{CoefficientTimes} $\rightarrow$ (\textit{Expand}[\#1 \#2] &)

specifies how to multiply two coefficients in this algebra.

\textbf{Extension} $\rightarrow$ None
give an algebraic extension \textit{ext}.

\begin{itemize}
  \item \textbf{Examples}
  \begin{itemize}
    \item \texttt{In[186]= \textcolor{red}{alg} = OreAlgebra[x, Der[x]]}
    \item \texttt{Out[186]= K[x]\{D_x; 1, D_x\}}
    \item \texttt{In[187]= \textcolor{red}{FullForm[alg]}}
    \item \texttt{Out[187]= OreAlgebraObject[List[x, Der[x]], Expand, Function[Plus[Slot[1], Slot[2]]], Function[Expand[Times[Slot[1], Slot[2]]]], None]}
    \item \texttt{In[188]= \textcolor{red}{ToOrePolynomial}[x ** Der[x] + 1, \textcolor{red}{alg}]}
    \item \texttt{Out[188]= D_x x + 1}
    \item \texttt{In[189]= \textcolor{red}{ChangeOreAlgebra}[%, OreAlgebra[Der[x], x]]}
    \item \texttt{Out[189]= D_x x}
    \item \texttt{In[190]= \textcolor{red}{OreAlgebra}[\%]}
    \item \texttt{Out[190]= D_x x}
    \item \texttt{In[191]= \textcolor{red}{OreAlgebra}[S[k], Delta[n], Der[x], Euler[z], QS[qn, q^n]]}
    \item \texttt{Out[191]= K(k, n, qn, x, z)[S_k; S_k, 0][\Delta_n; S_n, \Delta_n][D_x; 1, D_x][\theta_z; 1, \theta_z][S_{qn, q}; S_{qn, q}, 0]}
    \item \texttt{In[192]= \textcolor{red}{OreAlgebra}[\{n, S[n], \text{Der}[x]\}]}
    \item \texttt{Out[192]= K(x)[n][S_n; S_n, 0][D_x; 1, D_x]}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \textbf{See Also}
  \begin{itemize}
    \item \texttt{OreAction}, \texttt{OreAlgebraGenerators}, \texttt{OreAlgebraOperators}, \texttt{OreAlgebraPolynomialVariables}, \texttt{OreDelta}, \texttt{OreOperatorQ}, \texttt{OreOperators}, \texttt{OrePolynomial}, \texttt{OreSigma}, \texttt{ToOrePolynomial}
  \end{itemize}
\end{itemize}
OreAlgebraGenerators

\( \text{OreAlgebraGenerators[alg]} \)

gives the list of generators of the Ore algebra \( \text{alg} \).

\( \text{More Information} \)

The generators are in the same order as they have been given when creating \( \text{alg} \) using \text{OreAlgebra}. Taking into account the internal structure how an Ore algebra is represented, \text{OreAlgebraGenerators} is the same as \text{First}.

\( \text{Examples} \)

\( \text{In[193]:=} \text{alg} = \text{OreAlgebra} \{p, \text{Der}[x], n, S[n], \text{Delta}[m], x\} \)

\( \text{Out[193]} = K(m)[p, n, x][D_x; 1, D_x][S_n; S_n, 0][\Delta_m; S_m, \Delta_m] \)

\( \text{In[194]:=} \text{OreAlgebraGenerators[alg]} \)

\( \text{Out[194]} = \{p, \text{Der}[x], n, S[n], \text{Delta}[m], x\} \)

\( \text{In[195]:=} \text{First[alg]} \)

\( \text{Out[195]} = \{p, \text{Der}[x], n, S[n], \text{Delta}[m], x\} \)

\( \text{See Also} \)

\text{OreAlgebra, OreAlgebraOperators, OreAlgebraPolynomialVariables}
OreAlgebraOperators

\texttt{OreAlgebraOperators[\textit{alg}]}

gives the list of Ore operators that are contained in the Ore algebra \textit{alg}.

\textbf{More Information}

An Ore operator is an expression for which \texttt{OreSigma} and \texttt{OreDelta} are defined, e.g., \texttt{S[n]} or \texttt{Der[x]} are predefined Ore operators. The list returned by \texttt{OreAlgebraOperators} is a subset of that one returned by \texttt{OreAlgebraGenerators}.

\textbf{Examples}

\begin{verbatim}
\texttt{In[196]:=} \texttt{alg = OreAlgebra[p, Der[x], n, S[n], Delta[m], x]}
\texttt{Out[196]= K(m)[p, n, x]|D_1, D_2|S_a; S_b, S_c|\Delta_a; S_b, \Delta_c]
\texttt{In[197]:=} \texttt{OreAlgebraOperators[alg]}
\texttt{Out[197]= \{Der[x], S[n], Delta[m]\}}
\end{verbatim}

\textbf{See Also}

\texttt{OreAlgebra, OreAlgebraGenerators, OreAlgebraPolynomialVariables, OreOperatorQ, OreOperators}
OreAlgebraPolynomialVariables

OreAlgebraPolynomialVariables[\textit{alg}]

gives the list of variables that occur polynomially in the Ore algebra \textit{alg}.

\begin{itemize}
\item More Information
\end{itemize}

Every generator of the Ore algebra \textit{alg} that is not an Ore operator, i.e., for which \texttt{OreSigma} and \texttt{OreDelta} are not defined, is called a “polynomial variable”.

\begin{itemize}
\item Examples
\end{itemize}

\begin{verbatim}
\texttt{In[198]= alg = OreAlgebra[p, Der[x], n, S[n], Delta[m], x]}
\texttt{Out[198]= K[(m)]\{p, n, x\}[Dp; 1, Dx][Sn; S, 0][Dm; S, M]}
\texttt{In[199]= OreAlgebraPolynomialVariables[alg]}
\texttt{Out[199]= \{p, n, x\}}
\end{verbatim}

\begin{itemize}
\item See Also
\end{itemize}

\texttt{OreAlgebra, OreAlgebraGenerators, OreAlgebraOperators}
OreDelta

OreDelta[op]
defines the endomorphism δ for the Ore operator op.

More Information

Ore operators like Sn or Dx are defined by two endomorphisms σ and δ such that δ is a σ-derivation, i.e., that satisfies the skew Leibniz law

δ(fg) = σ(f)δ(g) + δ(f)g.

Then the commutation rule for the newly introduced symbol ∂ is

∂a = σ(a)∂ + δ(a).

The standard Ore operators (shift, differential, delta, Euler, q-shift) are pre-defined in HolonomicFunctions (using OreSigma, OreDelta, and OreAction). If you want to define your own Ore operators, use OreSigma and OreDelta to define their commutation properties. Note that op can be a pattern as well as a fixed expression.

Examples

In[200]:= OreDelta[Der[x]]
Out[200]= ∂x #1 &

In[201]:= OreDelta[S[n]]
Out[201]= 0 &

We show how a generic Ore operator can be defined.

In[203]:= ToOrePolynomial[(n**2*S2[n^2] + 1)^2]
Out[203]= (n^2 + 2n) S2[n^2] + 2n S2[n] + 1

ApplyOreOperator[%], f[n]
Out[208]= (n^2 + 2n) f(n + 4) + f(n) + 2nf(n + 2)

See Also

OreAction, OreOperatorQ, OreOperators, OreSigma
**OreGroebnerBasis**

\[
\text{OreGroebnerBasis}\left[\{P_1, \ldots, P_k\}\right]
\]

computes a left Gröbner basis for the left ideal that is generated by the Ore polynomials \(P_1, \ldots, P_k\).

\[
\text{OreGroebnerBasis}\left[\{P_1, \ldots, P_k\}, \text{alg}\right]
\]

translates \(P_1, \ldots, P_k\) into the Ore algebra \(\text{alg}\) and then computes a left Gröbner basis.

**More Information**

The input polynomials \(P_1, \ldots, P_k\) have to be given as \(\text{OrePolynomial}\) expressions. If they are not in this format, or if they belong to different Ore algebras, then the second argument has to be given to specify in which algebra the computation should take place.

\(\text{OreGroebnerBasis}\) executes Buchberger’s algorithm and it makes use of the chain criterion (Buchberger’s second criterion). The product criterion (Buchberger’s first criterion) cannot be used in noncommutative polynomial rings.

Per default, the output is a completely auto-reduced Gröbner basis, all its elements are normalized to have content 1, and they are ordered by their leading monomials according to the monomial order.

The following options can be given:

**Method → "sugar"**

specifies the pair selection strategy in Buchberger’s algorithm. The following methods can be chosen:

"sugar": the sugar strategy has been proposed by Giovini, Mora, Niesi, Robbiano, and Traverso in 1991 and is one of the most popular pair selection strategies. It mimics the Gröbner basis computation in an homogeneous ideal.

"normal": the normal strategy has been proposed by Bruno Buchberger himself; it takes the pair with the smallest lcm of the leading monomials.

"elimination": a greedy strategy that is designed for elimination problems: it prefers pairs where the total degree of the variables to be eliminated is minimal (in the lcm of the leading monomials).

"pairsize": pairs of small size are treated first; we take as the size of a pair the product of the \(\text{ByteCounts}\) of the two polynomials.
MonomialOrder \rightarrow \text{None}

specifies the monomial order; \text{None} means that the same monomial order
is taken in which the input is given. If the input is not given as \text{OrePolynomial}
expressions, then \text{DegreeLexicographic} is taken per default. The
following monomial orders can be chosen:

\text{DegreeLexicographic}:
 total degree lexicographic order (= graded lexicographic ordering). The terms of higher total degree come first; two
terms of the same degree are ordered lexicographically (see below).

\text{DegreeReverseLexicographic}:
 total degree reverse lexicographic ordering.

\text{Lexicographic}:
 the terms with the highest power of the first variable come first: for two terms with the same power of the first variable the power
of the second variable is compared, and so on.

\text{ReverseLexicographic}:
 reverse lexicographic ordering: The term with the higher power of the last variable comes last; for terms with the same
power of the last variable, the exponent on the next to last variable is
compared, and so on.

Note: Under this ordering the monomials are not well-ordered!

\text{EliminationOrder}[n]: a block order s.t. the first \(n\) variables are eliminated, i.e., they are lexicographically greater than the rest. In the two
blocks, the variables are ordered by \text{DegreeLexicographic}.

\text{WeightedLexicographic}[w]: weighted lexicographic ordering: the total
degrees weighted by the weight vector \(w\) are compared first, and if they
are equal, the two terms are compared lexicographically.

\text{WeightedLexicographic}[w,\text{order}]: similar as before, but now the two
terms are compare by \text{order} in case that their weighted total degrees
are the same.

\text{MatrixOrder}[m]: matrix ordering: the exponent vectors are multiplied by
the matrix \(m\) and the results are compared lexicographically.

You can define your own monomial ordering by giving a pure function that
takes as input two exponent vectors and yields \text{True} or \text{False}.

\underline{Extended} \rightarrow \text{False}

setting this option to \text{True} computes also the cofactors of the Gröbner basis
that allow to write each of its elements as a linear combination of the \(P_i\). In
this case, a list of length 2 is returned, its first element being the Gröbner
basis, and its second element being the matrix of cofactors.

\underline{Incomplete} \rightarrow \text{False}

setting this option to \text{True} causes that the computation is interrupted as
soon as an element free of the elimination variables is found (works only in
connection with \text{EliminationOrder}).

\underline{ModuleBasis} \rightarrow \{\}

when computing in a module, give here a list of natural numbers indicating
where the position variables are located among the generators of the Ore
algebra.
Modulus → 0
give a prime number here for modular computations.

NormalizeCoefficients → True
whether the content of intermediate and final results shall be removed.

Reduce → True
whether autoreduction should be performed at the end of Buchberger’s algorithm.

ReduceTail → True
whether also the tail (i.e., the terms that come behind the first non-reducible term) should be reduced for each S-polynomial.

\[\text{Examples}\]

\[\text{In}\[209\]:=} \text{rels} = \text{Flatten}[\text{Annihilator}[\text{GegenbauerC}[n, m, x], \#] & /@ \{S[n], S[m], \text{Der}[x]\}]
\[\text{Out}\[209\]= \{(n + 2)S_n^2 + (-2mx - 2nx - 2x)S_n + (2m + n), (4m^2x^2 - 4m^2 + 4mx^2 - 4m)S_m^2 + (-4m^2x^2 + 8m^2 - 4mnx^2 + 4mn - 4mx^2 + 6m)S_m + (-4m^2 - 4mn - 2m - n^2 - n), (x^2 - 1)D_x^2 + (2mx + x)D_x + (-2mn - n^2)\}\]
\[\text{In}\[210\]:=} \text{OreGroebnerBasis}[\text{rels}, \text{OreAlgebra}[S[m], S[n], \text{Der}[x]]]
\[\text{Out}\[210\]= \{(−n - 1)S_n + (x^2 - 1)D_x + (2mx + nx), -2mS_n + xD_x + (2m + n), (1 - x^2)D_x^2 + (-2mx - x)D_x + (2mn + n^2)\}\]

By Gröbner basis computation we can show that relations are not compatible.
\[\text{In}\[211\]:=} \text{opolys} = \text{ToOrePolynomial}[\{S[k]^2 + (n - k) ** S[k] + n^2 - k^2, S[n]^2 + (2n) ** S[n] - k\}, \text{OreAlgebra}[S[k], S[n]]]
\[\text{Out}\[211\]= \{S[k]^2 + (n - k)S_k + (n^2 - k^2), S[n]^2 + 2nS_n - k\}\]
\[\text{In}\[212\]:=} \text{OreGroebnerBasis}[\text{opolys}]
\[\text{Out}\[212\]= \{1\}\]

We demonstrate the use of the option Extended:
\[\text{In}\[213\]= \{\text{gb}, \text{cofs}\} = \text{OreGroebnerBasis}[\{x^3, \text{Der}[x]^2\}, \text{OreAlgebra}[x, \text{Der}[x]], \text{Extended} \rightarrow \text{True}]\]
\[\text{Out}\[213\]= \{(1), \{(−1/8)D_x^4 + 1/6 D_x^3, \frac{1}{8}x^4D_x^2 + \frac{4}{3}x^3D_x + 3x^2\}\}\]
\[\text{In}\[214\]= \text{Inner}[\#1 ** \#2 &, \text{cofs}, \{x^3, \text{Der}[x]^2\}, \text{Plus}]\]
\[\text{Out}\[214\]= \{1\}\]

\[\text{See Also}\]
FGLM, FindRelation, GBEqual, OreAlgebra, OrePolynomial, OreReduce, Printlevel, UnderTheStaircase
OreOperatorQ

\[\text{OreOperatorQ}[\text{expr}]\]

tests whether \text{expr} is an Ore operator or not.

\[\text{More Information}\]

This command gives \textbf{True} if \text{expr} is an Ore operator; this is the case when the two endomorphisms \textbf{OreSigma}[expr] and \textbf{OreDelta}[expr] are defined. Otherwise \textbf{False} is returned.

Note that by the term “Ore operator” we mean a single symbol that has been introduced by an Ore extension. In particular we do not mean an operator in the sense of a recurrence or a differential equation; those are (Ore) polynomials involving some Ore operators.

\[\text{Examples}\]

\[
\text{In}[213]= \text{OreOperatorQ}[\text{Der}[x]]
\text{Out}[213]= \text{True}
\]

\[
\text{In}[214]= \text{OreOperatorQ}["\partial"]
\text{Out}[214]= \text{False}
\]

\[
\text{In}[215]= \text{OreSigma}["\partial"] = \sigma; \text{OreDelta}["\partial"] = \delta;
\text{In}[216]= \text{OreOperatorQ}["\partial"]
\text{Out}[216]= \text{True}
\]

\[
\text{In}[217]= \text{ToOrePolynomial}["\partial" \ast^a]
\text{Out}[217]= \sigma^a\partial + \delta^a
\]

\[\text{See Also}\]

\textbf{OreDelta, OreOperators, OreSigma}
OreOperators

OreOperators[expr] gives a list of Ore operators that are contained in expr.

More Information

Every expression \( e \) for which the two endomorphisms OreSigma[\( e \)] and OreDelta[\( e \)] are defined, is considered as an Ore operator.

Note that by the term “Ore operator” we mean a single symbol that has been introduced by an Ore extension. In particular we do not mean an operator in the sense of a recurrence or a differential equation; those are (Ore) polynomials involving some Ore operators.

Examples

\begin{verbatim}
In[221]:= OreOperators[S[n]^2 + S[k]^2 + x Der[x] S[n]]
Out[221]= {Der[x], S[k], S[n]}

In[222]:= OreOperators[D[f[x], x]]
Out[222]= {}

In[223]:= Annihilator[Fibonacci[n, x]]
Out[223]= {2 n S_n + (-x^2 - 4) D_x + (-n x - x), (x^2 + 4) D_x^2 + 3x D_x + (1 - n^2)}

In[224]:= OreOperators[%]
Out[224]= {Der[x], S[n]}
\end{verbatim}

See Also

OreDelta, OreOperatorQ, OreSigma
OrePlus

OrePlus[opoly1, poly2, ...]
computes the sum of the Ore polynomials poly1, poly2, etc.

OrePlus[opoly1, poly2, ..., alg]
translates poly1, poly2, etc. into the Ore algebra alg and then computes their sum.

▼ More Information

The input polynomials can be either given as OrePolynomial expressions or as standard Mathematica polynomials. OrePlus then tries to figure out which Ore algebra is best suited for representing the output. Alternatively, this algebra can be explicitly given as the last argument.

To make the work with Ore polynomials more convenient, we have defined an upvalue for addition. This means that the standard notation poly1 + poly2 can be used for addition of Ore polynomials; if at least one of the summands is of type OrePolynomial then OrePlus will be called.

▼ Examples

The output of OrePlus is always an OrePolynomial expression; sometimes (as here in output line 2) this cannot be directly be seen.

In[223]:= OrePlus[Der[x]**x, (1 - x)**Der[x], OreAlgebra[x, Der[x]]]
Out[223]= Der[x] + 1

In[226]:= % - Der[x]
Out[226]= 1

In[227]:= Head[%]
Out[227]= OrePolynomial

Observe how the representation format of the coefficients (here in factored form) is preserved by OrePlus.

In[228]:= ToOrePolynomial[(n**2 + n)**S[n] + n**2 - 1]
Out[228]= (n^2 + n)S[n] + (n^2 - 1)

In[229]:= Factor[%]
Out[229]= n(n + 1)S[n] + (n - 1)(n + 1)

In[230]:= OrePlus[%, (S[n] + 1)**n]
Out[230]= (n + 1)^2S[n] + (n^2 + n - 1)

▼ See Also

OrePolynomial, OrePolynomialSubstitute, OrePower, OreTimes, ToOrePolynomial
OrePolynomial

\[\text{OrePolynomial}[\text{data}, \text{alg}, \text{order}]\]

is the internal representation of an Ore polynomial in the Ore algebra \(\text{alg}\) with monomial order \(\text{order}\).

\[\text{More Information}\]

The above format is the FullForm representation of an Ore polynomial. Thanks to the command ToOrePolynomial, you never have to type it explicitly. Unless you type FullForm, you also will never see this representation, since Ore polynomials are displayed in a very nice format: each term is displayed as \(cg_1^{e_1}g_2^{e_2}\ldots g_d^{e_d}\) where \(c\) is the coefficient and the \(g_i\) are the generators of the algebra (their order is preserved). Additionally, the terms are ordered according to the monomial order, i.e., the leading term is always in front.

The first argument \(\text{data}\) contains a list of terms, each term being a pair consisting of a coefficient and an exponent vector. The exponents are given in the order as the generators of the algebra are given. The zero polynomial has the empty list in this place. The second argument \(\text{alg}\) is an Ore algebra, i.e., an expression of type OreAlgebraObject. The third argument contains the monomial order with respect to which the entries in \(\text{data}\) are ordered; see the description of OreGroebnerBasis (p. 57) for a list of supported monomial orders. Note that \(\text{data}\) is always kept ordered such that the leading term corresponds to the first list element.

In the following, let \(p, p_1\) and \(p_2\) be OrePolynomial expressions. The algebra and the monomial order of an Ore polynomial can be extracted by OreAlgebra\[p\] and MonomialOrder\[p\], respectively.

For the type OrePolynomial numerous upvalues have been defined to make life easier. First this concerns the arithmetical operations Plus, Times, NonCommutativeMultiply, and Power. This means that you can type \(p_1 + p_2\) instead of the more cumbersome OrePlus\[p_1, p_2\] and \(p_1** p_2\) instead of OreTimes\[p_1, p_2\] (the ** symbol is Mathematica’s notation for NonCommutativeMultiply). Attention when using the commutative Times: it should only be used in cases where noncommutativity does not play any rôle, e.g., for \(2 * p_1\) or \(-p_1\). Otherwise it can happen that Mathematica reorders the factors and the output is not what you originally wanted.

Next, the commands Expand, Factor, and Together can be applied to an OrePolynomial expression. However, they affect only the coefficients. In particular, Factor\[p\] does not perform any operator factorization. It just causes that the coefficients will be given in factored form. Note that this operation also changes the OreAlgebraObject of \(p\) in order to remember that the coefficients are factored. In all subsequent operations this property then will persist.

Some standard polynomial operations can be performed on OrePolynomial expressions like on standard polynomials. They include Coefficient, Exponent, and Variables. The command PolynomialMod is applied to the coefficients.
only. Finally, `Normal[p]` converts an Ore polynomial to a standard commutative polynomial.

\[ \text{Examples} \]

\begin{verbatim}
In[233]:= alg = OreAlgebra[S[n], Der[x]]
Out[233]= K[n, x][S[n], 0][Dx; 1, Dx]

In[234]:= p1 = ToOrePolynomial[S[n] + n, alg]
Out[234]= S[n] + n

In[235]:= p2 = ToOrePolynomial[x Der[x] + S[n] - 1, alg]
Out[235]= S[n] + x*Dx - 1

In[236]:= p1 + p2
Out[236]= 2*S[n] + x*Dx + (n - 1)

In[237]:= p1 - p2
Out[237]= -xD[x] + (n + 1)

In[238]:= p1 ** p2
Out[238]= S[n]^2 + x*S[n]*Dx + (n - 1)*S[n] + n*x*Dx - n

In[239]:= p2 ** p1
Out[239]= S[n]^2 + x*S[n]*Dx + n*S[n] + n*x*Dx - n

In[240]:= First[%]
Out[240]= {{1, {2, 0}}, {x, {1, 1}}, {n, {1, 0}}, {nx, {0, 1}}, {-n, {0, 0}}}

In[241]:= (n + 1) ** p1^3
Out[241]= (n + 1)*S[n]^3 + (3*n^2 + 6*n + 3)*S[n]^2 + (3*n^3 + 6*n^2 + 4*n + 1)*S[n] + (n^4 + n^3)

In[242]:= Factor[%]
Out[242]= (n + 1)*S[n]^3 + 3*(n + 1)^2*S[n]^2 + (n + 1)*S[n]^2 + n*(n + 1)

In[243]:= % - 3*S[n]^2 + n^2
Out[243]= (n + 1)*S[n]^3 + 3*(n + 2)*S[n]^2 + (n + 1)*(3*n^2 + 3*n + 1)*S[n] + n^2*(n + 1)

In[244]:= Exponent[%, n]
Out[244]= 4

In[245]:= Variables[p2]
Out[245]= {x, Der[x], S[n]}

In[246]:= MonomialOrder[p1]
Out[246]= DegreeLexicographic

In[247]:= Normal[p2]
Out[247]= x Der[x] + S[n] - 1
\end{verbatim}

\[ \text{See Also} \]

`ApplyOreOperator`, `ChangeMonomialOrder`, `ChangeOreAlgebra`, `LeadingCoefficient`, `LeadingExponent`, `LeadingPowerProduct`, `LeadingTerm`, `NormalizeCoefficients`, `OreAlgebra`, `OrePlus`, `OrePower`, `OreTimes`, `OrePolynomialDegree`, `OrePolynomialListCoefficients`, `OrePolynomialSubstitute`, `OrePolynomialZeroQ`, `Support`, `ToOrePolynomial`
OrePolynomialDegree

OrePolynomialDegree[opoly]
gives the total degree of the Ore polynomial oopoly with respect to all generators of the algebra.

OrePolynomialDegree[opoly, vars]
gives the total degree of oopoly with respect to vars.

\[ \text{More Information} \]

The input oopoly must be an OrePolynomial expression, and vars a list of (or a single) indeterminates. This list must either be a subset of the generators of the algebra, or contain only elements that do not belong to the algebra.

\[ \text{Examples} \]

\begin{verbatim}
In[246]:= oopoly = ToOrePolynomial[
(n + x) S[n] Der[x] + n^2 x^2 S[n] + nx Der[x] - 1,
OreAlgebra[S[n], Der[x]]]
Out[246]= (n + x) S[n] Der[x] + n^2 x^2 S[n] + nx Der[x] - 1

In[247]:= OrePolynomialDegree[oopoly]
Out[247]= 2

In[248]:= OrePolynomialDegree[oopoly, S[n]]
Out[248]= 1

In[249]:= OrePolynomialDegree[oopoly, {n, x}]
Out[249]= 4

In[250]:= oopoly = ChangeOreAlgebra[oopoly, OreAlgebra[S[n], Der[x], n, x]]

In[251]:= OrePolynomialDegree[oopoly]
Out[251]= 5
\end{verbatim}

\[ \text{See Also} \]

LeadingExponent, LeadingPowerProduct, OrePolynomial, Support, ToOrePolynomial
OrePolynomialListCoefficients

OrePolynomialListCoefficients[opoly]
gives a list containing all coefficients of the Ore polynomial opoly, ordered according to the monomial order.

More Information

The output contains the coefficients as they appear in the OrePolynomial expression opoly. In particular, it does not contain zeros, and hence is different from Mathematica’s CoefficientList.

Examples

In[252] := Annihilator[(k + n)!StirlingS1[k, m]StirlingS2[m, n], {S[k], S[m], S[n]}]
Annihilator::nondf : The expression StirlingS1[k, m] is not recognized to be \(\partial\)-finite.
The result might not generate a zero-dimensional ideal.
Annihilator::nondf : The expression StirlingS2[m, n] is not recognized to be \(\partial\)-finite.
The result might not generate a zero-dimensional ideal.

Out[252] = {S[k] S[m] S[n] + (k^2 + kn + 2k) S[m] S[n] + (-kn - k - n^2 - 3n - 2) S[n] + (-k^2 - 2kn - 3k - n^2 - 3n - 2)}

In[253] := opoly = Factor[First[%]]


In[254] := OrePolynomialListCoefficients[opoly]

Out[254] = {1, 1, -1, 2, -1, 1, -1, -2, -1, 1, -3, -3, -3, -2, -2}

See Also

ChangeMonomialOrder, ChangeOreAlgebra, LeadingCoefficient, NormalizeCoefficients, OrePolynomial, ToOrePolynomial
OrePolynomialSubstitute

OrePolynomialSubstitute[opoly, rules]
  applies the substitutions rules to the Ore polynomial opoly.

▼ More Information

The input opoly has to be an OrePolynomial expression, and rules a list of rules of the form \( a \rightarrow b \) or \( a \rightarrow b \).

Never try to use Mathematica’s ReplaceAll for substitutions on Ore polynomials. This may cause results that are not well-formed OrePolynomial expressions (as the last example demonstrates).

The following options can be given:

Algebra \( \rightarrow \) None
  the Ore algebra in which the output should be represented; None means that the algebra of opoly is taken.

MonomialOrder \( \rightarrow \) None
  the monomial order in which the output should be represented; None means that the monomial order of opoly is taken.

▼ Examples

\[\text{In}[257]:= \text{opoly} = \text{ToOrePolynomial}[S[n]\text{Der}[x] + nxS[n] + \text{Der}[x] + 1, \text{OreAlgebra}[\text{Der}[x], S[n]]]\]
\[\text{Out}[257]= \text{D}[x]S[n] + \text{D}[x] + nxS[n] + 1\]
\[\text{In}[258]:= \text{OrePolynomialSubstitute}[\text{opoly}, \{\text{Der}[x] \rightarrow 1\}]\]
\[\text{Out}[258]= (nx + 1)S[n] + 2\]
\[\text{In}[259]:= \text{OreAlgebra}[\%]\]
\[\text{Out}[259]= K(n, x)[\text{D}[x]; 1, \text{D}[x]][S[n]; S[n], 0]\]
\[\text{In}[260]:= \text{OrePolynomialSubstitute}[\text{opoly}, \{\text{Der}[x] \rightarrow 1\}, \text{Algebra} \rightarrow \text{OreAlgebra}[S[n]]]\]
\[\text{Out}[260]= (nx + 1)S[n] + 2\]
\[\text{In}[261]:= \text{OrePolynomialSubstitute}[\text{opoly}, \{n \rightarrow 0\}]\]
\[\text{Out}[261]= \text{D}[x]S[n] + \text{D}[x] + 1\]

The following gives nonsense!
\[\text{In}[262]:= \text{opoly} /. n \rightarrow 0\]
\[\text{Out}[262]= \text{D}[x]S[0] + \text{D}[x] + 0S[0] + 1\]

▼ See Also

OrePolynomial, ToOrePolynomial
OrePolynomialZeroQ

OrePolynomialZeroQ[opoly]

tests whether the Ore polynomial opoly is zero.

More Information

Note that this command does not do any simplification on the coefficients.

Examples

In[263]:= opoly = ToOrePolynomial[n S[n] + S[n], OreAlgebra[S[n]]]
Out[263]= (n + 1) S

In[264]:= OrePolynomialZeroQ[opoly]
Out[264]= False

In[265]:= OrePolynomialZeroQ[opoly - S[n]**n]
Out[265]= True

In[266]:= opoly = ToOrePolynomial[Der[x] + 1 - Sin[x]**2 - Cos[x]**2, OreAlgebra[Der[x]]]
Out[266]= Der[x] + (1 - Cos[x]**2 - Sin[x]**2)

In[267]:= OrePolynomialZeroQ[opoly - Der[x]]
Out[267]= False

See Also

OrePolynomial, ToOrePolynomial
OrePower

OrePower[opoly, n] computes the n-th power of the Ore polynomial opoly.

More Information

The input opoly is an OrePolynomial expression and n has to be an integer. Negative integers are only admissible in degenerate cases when opoly does not contain any Ore operators.

To make the work with Ore polynomials more convenient, we have defined an upvalue for Power. This means that the standard notation opoly^n can be used and OrePower will be called automatically.

Examples

In[268] := opoly = ToOrePolynomial[n x S[n] Der[x]]
Out[268] = nxD x S n

In[269] := OrePower[opoly, 3]
Out[269] = (n^3 x^3 + 3 n^2 x^3 + 2 n x x^2) D^3 S + (3 n^3 x^2 + 9 n^2 x^2 + 6 n x^2) D^2 S
+ (n^2 x + 3 n x x + 2 n x) D S

In[270] := 1/opoly
   OrePower::negpow : Negative power of an OrePolynomial.
Out[270] = $Failed

In[271] := OrePolynomialSubstitute[opoly, {S[n] -> 1, Der[x] -> 1}]^(-1)
Out[271] = 1/nx

See Also

OrePolynomial, OrePolynomialSubstitute, OrePlus, OreTimes, ToOrePolynomial
OreReduce

\textbf{OreReduce}\{opoly, \{g_1, g_2, \ldots \}\}

reduces the Ore polynomial \textit{opoly} modulo the set of Ore polynomials \{g_1, g_2, \ldots \}.

\begin{itemize}
  \item \textbf{More Information}
  \end{itemize}

The \( g_i \) are \texttt{OrePolynomial} expressions and \textit{opoly} may be either an \texttt{OrePolynomial} expression, or a standard Mathematica polynomial; in the latter case it is translated into the Ore algebra in which the \( g_i \) are given. Note that the set \{\( g_1, g_2, \ldots \)\} needs not to form a Gröbner basis. However, if it is not, the result of the reduction may not be uniquely defined.

As always, the operations in \texttt{OreReduce} involve only multiplications from the left.

By default, no content is removed during the reduction. Thus denominators may appear and therefore it is recommended to switch to togethered coefficient representation in advance.

The following options can be given:

\begin{itemize}
  \item \texttt{Extended} \rightarrow \texttt{False}
    \begin{itemize}
      \item setting this option to \texttt{True} computes also the cofactors of the reduction.
      \item The output then is of the form \{\( r, f, \{c_1, c_2, \ldots \}\)\} such that \( r \) is the reductum of \textit{opoly} and \textit{opoly} can be written as \( f \cdot r + c_1 g_1 + c_2 g_2 + \ldots \).
    \end{itemize}
  \item \texttt{ModuleBasis} \rightarrow \{\}
    \begin{itemize}
      \item when computing in a module, give here a list of natural numbers indicating where the position variables are located among the generators of the Ore algebra.
    \end{itemize}
  \item \texttt{Modulus} \rightarrow 0
    \begin{itemize}
      \item give a prime number here for modular computations.
    \end{itemize}
  \item \texttt{NormalizeCoefficients} \rightarrow \texttt{False}
    \begin{itemize}
      \item whether the content of intermediate and final results shall be removed.
    \end{itemize}
  \item \texttt{OrePolynomialSubstitute} \rightarrow \{\}
    \begin{itemize}
      \item If a list of rules \{\( a \rightarrow a_0, b \rightarrow b_0, \ldots \)\} is given, then the reduction is computed with these substitutions (note that it takes care of noncommutativity as the substitutions are performed at a point where the noncommuting nature of these variables is not relevant any more).
    \end{itemize}
  \item \texttt{ReduceTail} \rightarrow \texttt{True}
    \begin{itemize}
      \item whether also the tail (i.e., the terms that come behind the first non-reducible term) should be reduced.
    \end{itemize}
\end{itemize}
In[272]:= gb = Annihilator[LaguerreL[n, x], {Der[x], S[n]}]

Out[272]= {xD + (-n - 1)S, + (n - x + 1), (n + 2)S^2 + (-2n + x - 3)S, + (n + 1)}

In[273]:= OreReduce[(S[n] Der[x])^3, gb]

Out[273]= (-n^2 + 2n + n^2/x + n^2/x + 1/x) S, + (n^3 - 2n^2 + 2x^2 + 2x^2 + x^2)

In[274]:= OreReduce[(S[n] Der[x])^3, Together[gb]]

Out[274]= n^2(-x) + nx^2 + 2n + x^2 + x + 2 S, + (n + 1)(nx + x - 2)

In[275]:= OreReduce[(S[n] Der[x])^3, gb, NormalizeCoefficients → True]

Out[275]= (-nx + x^2 + x + 2)S, + (nx + x - 2)

In[276]:= OreReduce[(S[n] Der[x])^3, gb, NormalizeCoefficients → True, Extended → True]

Out[276]= {(-nx + x^2 + x + 2)S, + (nx + x - 2), \frac{x^3}{n + 1}, 
\frac{x^2}{n + 1} D_s^2 S^3 + \frac{(n + 4)x}{n + 1} D_s S^4 + \frac{n^2 + 9n + 20}{n + 1} S^5 + \frac{-nx + x^2 - 6x}{n + 1} D_s S^3 - 2 \frac{n^2 - nx + 10n - 4x + 24}{n + 1} S^4 + \frac{n^2 - 2nx + 11n + x^2 - 9x + 30}{n + 1} S^5, 
\frac{n^2 + 9n + 20}{n + 1} S^6 + \frac{-nx + x^2 - 6x}{n + 1} D_s S^3 + \frac{-nx + x^2 - 5x + 2}{n + 1} S^2 
+ \frac{-nx - 3x + 2}{n + 1} S, + \frac{-nx - x - 2}{n + 1}}

In[277]:= Inner[#1 ** #2 &, Last[%], gb, Plus]

Out[277]= \frac{x^3}{n + 1} D_s^3 S^3 + (nx - x^2 - x - 2)S, + (-nx - x + 2)

\[\text{\textbf{See Also}}\]

NormalizeCoefficients, OreGroebnerBasis
OreSigma

\textbf{OreSigma}[\textit{op}]

defines the endomorphism \( \sigma \) for the Ore operator \( \textit{op} \).

\subsection*{More Information}

Ore operators like \( S_n \) or \( D_x \) are defined by two endomorphisms \( \sigma \) and \( \delta \) such that \( \delta \) is a \( \sigma \)-derivation, i.e., that satisfies the skew Leibniz law

\[ \delta(fg) = \sigma(f)\delta(g) + \delta(f)g. \]

Then the commutation rule for the newly introduced symbol \( \partial \) is

\[ \partial a = \sigma(a)\partial + \delta(a). \]

The standard Ore operators (shift, differential, delta, Euler, \( q \)-shift) are predefined in \texttt{HolonomicFunctions} (using \texttt{OreSigma}, \texttt{OreDelta}, and \texttt{OreAction}). If you want to define your own Ore operators, use \texttt{OreSigma} and \texttt{OreDelta} to define their commutation properties. Note that \textit{op} can be a pattern as well as a fixed expression.

\subsection*{Examples}

\begin{verbatim}
In[278]:= OreSigma[Der[x]]
Out[278]= #1 &
In[279]:= OreSigma[S[n]]
Out[279]= #1 /. n -> n + 1 &
\end{verbatim}

We show how a generic Ore operator can be defined.

\begin{verbatim}
In[292]:= ToOrePolynomial[MyOp^2 ** a]  
Out[292]= MySigma(MySigma(a)) MyOp^2  
+ (MyDelta(MySigma(a)) + MySigma(MyDelta(a))) MyOp  
+ MyDelta(MyDelta(a))
\end{verbatim}

Now we introduce the double-shift, i.e., the shift by 2.

\begin{verbatim}
In[293]:= OreSigma[S2[a_,]] := #1 /. a -> a + 2 &;
In[294]:= OreDelta[S2[a_,]] := 0 &;
In[295]:= ToOrePolynomial[(n ** S2[n] + 1)^2]
Out[295]= (n^2 + 2 n) S2[n]^2 + 2 n S2[n] + 1
\end{verbatim}

\begin{verbatim}
In[296]:= ApplyOreOperator[%, f[n]]
Out[296]= (n^2 + 2 n) f(n + 4) + f(n) + 2 n f(n + 2)
\end{verbatim}

\subsection*{See Also}

\texttt{OreAction}, \texttt{OreDelta}, \texttt{OreOperatorQ}, \texttt{OreOperators}
OreTimes

\[\text{OreTimes}\left[\text{opoly}_1, \text{opoly}_2, \ldots\right]\]
computes the product of the Ore polynomials \(\text{opoly}_1, \text{opoly}_2, \ldots\).

\[\text{OreTimes}\left[\text{opoly}_1, \text{opoly}_2, \ldots, \text{alg}\right]\]
translates \(\text{opoly}_1, \text{opoly}_2, \ldots\) into the Ore algebra \(\text{alg}\) and then computes their product.

More Information

The input polynomials can be either given as \text{OrePolynomial} expressions or as standard Mathematica polynomials. \text{OreTimes} then tries to figure out which Ore algebra is best suited for representing the output. Alternatively, this algebra can be explicitly given as the last argument.

To make the work with Ore polynomials more convenient, we have defined an upvalue for multiplication. This means that the notation \(\text{opoly}_1 \ast \ast \text{opoly}_2\) can be used for multiplying two Ore polynomials: if at least one of the factors is of type \text{OrePolynomial} then \text{OreTimes} will be called. The same works for the commutative \text{Times}. But since Mathematica might reorder the factors, this should only be used if the factors in fact commute, e.g., in \(2 \ast \text{opoly}_1\).

Examples

\[\text{In}[287]= \text{OreTimes}[\text{Der}[x], x]\]
\[\text{Out}[287]=xD_x + 1\]

The following does not work since none of the factors is of type \text{OrePolynomial}.

\[\text{In}[288]= \text{Der}[x] \ast \ast x\]
\[\text{Out}[288]=\text{Der}[x] \ast \ast x\]

This is another way how to do it.

\[\text{In}[289]= \text{ToOrePolynomial}[\text{Der}[x]]\]
\[\text{Out}[289]=D_x\]
\[\text{In}[290]= \% \ast \ast x\]
\[\text{Out}[290]=xD_x + 1\]

Shift a recurrence by 1:

\[\text{In}[291]= \text{ToOrePolynomial}\left[(2 + n)h[2 + n] + (-3 - 2n)h[1 + n] + (1 + n)h[n], h[n]\right]\]
\[\text{Out}[291]= (n + 2)S_n + (-2n - 3)S_n + (n + 1)\]
\[\text{In}[292]= S[n] \ast \ast \%\]
\[\text{Out}[292]= (n + 3)S_n^3 + (-2n - 5)S_n^2 + (n + 2)S_n\]

See Also

\text{OrePolynomial}, \text{OrePolynomialSubstitute}, \text{OrePlus}, \text{OrePower}, \text{ToOrePolynomial}
Printlevel

Printlevel = \(n\)
activates and controls verbose mode, displaying information about the current computation up to recursion level \(n\).

\[\text{Examples}\]

\begin{verbatim}
In[293]:= ann = Flatten[Annihilator[LegendreP[n, x], #] & /@ {S[n], Der[x]}]
Out[293]= \(\{(n + 2) S^2_n + (-2 n x - 3 x) S_n + (n + 1), (x^2 - 1) D^2_x + 2 x D_x + (-n^2 - n)\}\)
\end{verbatim}

\begin{verbatim}
In[294]:= Printlevel = 2; OreGroebnerBasis[ann, OreAlgebra[S[n], Der[x]]]
OreGroebnerBasis: Number of pairs: 1
OreGroebnerBasis: Taking \{4, \{2, 2\}, 1, 2\}
OreGroebnerBasis: Does not reduce to zero \(\rightarrow\) number 3 in the basis.
The lpp is \(\{1, 1\}\). The ByteCount is 1316.
OreGroebnerBasis: Number of pairs: 2
OreGroebnerBasis: Taking \{5, \{2, 1\}, 1, 3\}
OreGroebnerBasis: Does not reduce to zero \(\rightarrow\) number 4 in the basis.
The lpp is \(\{1, 0\}\). The ByteCount is 1300.
OreGroebnerBasis: Number of pairs: 3
OreGroebnerBasis: Taking \{5, \{1, 2\}, 2, 3\}
OreGroebnerBasis: Number of pairs: 2
OreGroebnerBasis: Taking \{6, \{2, 0\}, 1, 4\}
OreGroebnerBasis: Number of pairs: 1
OreGroebnerBasis: Taking \{6, \{1, 1\}, 3, 4\}
OreGroebnerBasis: Reducing no. 1 of 2
OreGroebnerBasis: Reducing no. 2 of 2
Out[294]= \(\{(n - 1) S_n + (x^2 - 1) D_x + (n x + x), (1 - x^2) D^2_x - 2 x D_x + (n^2 + n)\}\)
\end{verbatim}

\begin{verbatim}
In[295]:= Printlevel = Infinity; Annihilator[Sum[Binomial[n, k], \{k, 0, n\}], S[n]]
Entering Annihilator[Sum]
Annihilator called with Binomial[n, k].
Annihilator: The factors that contain not-to-be-evaluated elements are \{\}
Annihilator: The remaining factors are \{Gamma[1 + k]**(-1), Gamma[1 + n], Gamma[1 - k + n]**(-1)\}
Annihilator: Factors that are not hypergeometric and hyperexponential: \{\}
CreativeTelescoping: Trying \(d = 0\), ansatz = eta[0] + (-1 + S[k])**phi[1][k]
LocalOreReduce: Reducing \{0, 0\}
LocalOreReduce: Reducing \{1, 0\}
Start to solve scalar equation...
RSolveRational: got a recurrence of order 1
RSolvePolynomial: degree bound = 0
Solved scalar equation.
CreativeTelescoping: Trying \(d = 1\), ansatz = eta[0]**1 + eta[1]**S[n]
\((-1 + S[k])**phi[1][k]**1\)
LocalOreReduce: Reducing \{0, 1\}
Start to solve scalar equation...
RSolveRational: got a recurrence of order 1
RSolvePolynomial: degree bound = 1
Solved scalar equation.
Out[295]= \{S_n - 2\}
\end{verbatim}

74
The $q$-shift on $x$ is defined to be $q x$. Often in practice the variable $x$ is in fact a power of $q$, e.g., $x = q^n$, and then the $q$-shift of $x$ corresponds to a shift in $n$. That’s why the Ore operator $\text{QS}$ takes two arguments, namely the variable on which the $q$-shift acts and the power of $q$ to which this variable corresponds.

Since for polynomial arithmetic it is problematic to deal with indeterminates like $q^n$, such occurrences will always be replaced by the variable $x$, e.g., when creating an OrePolynomial expression. When it is part of an OrePolynomial, the $q$-shift operator is displayed as $S_{x,q}$. When applying it to some expression, the variable $x$ is again replaced by $q^n$.

The operator $\text{QS}[x,q^n]$ acts on $x$ via $x \mapsto qx$ as well as on $n$ via $n \mapsto n + 1$. This behaviour is important when you deal with expressions like $\text{QBinomial}[n,k,q]$ where the variable $x$ or the power $q^n$ do not appear explicitly.

Example:

```math
In[296]:= \text{OreSigma}[\text{QS}[x,q^n]]
Out[296]= #1/. x \rightarrow qx /. n \rightarrow n + 1 &
```

```math
In[297]:= \text{ToOrePolynomial}[(q^n - q^3)\text{QS}[x,q^n] - q^2(2n) - q^2(n + 1),
\text{OreAlgebra}[\text{QS}[x,q^n]]]
Out[297]= (x - q^3)S_{x,q} + (-qx - x^2)
```

```math
In[298]:= \text{ApplyOreOperator}[%, f[q^n] + g[q^n]]
Out[298]= (-q^{n+1} - q^{2n}) (f[q^n] + g[q^n]) + (q^n - q^2) (f[q^{n+1}] + g[q^{n+1}])
```

```math
In[299]:= \text{Annihilator}[\text{QBinomial}[n,k,q]]
Out[299]= \{ (qk - q q^n)S_{q^n,q} + (q qk qn - qk), (q qk^2 - qk)S_{k,q} + (qk - q n) \}
```

```math
In[300]:= \text{ApplyOreOperator}[%, \text{QBinomial}[n,k,q]]
Out[300]= \{ (q^n - q^n) \text{QBinomial}[n,1,k,q] + (q^{k+1} - q^k) \text{QBinomial}[n,k,q],
\text{QBinomial}[n,k+1,q] + \text{QBinomial}[n,k,q] \}
```

```math
In[312]:= \text{FunctionExpand}[%, n \rightarrow 3 /. k \rightarrow 2]
Out[312]= \{ (q^2 + q + 1)(q^6 - q^3) + (q^2 + 1)(q^2 + q + 1)(q^2 - q^4),
q^5 - q^2 + (q^2 + q + 1)(q^2 - q^3) \}
```

```math
In[313]= \text{Expand}[%]
Out[313]= \{ 0, 0 \}
```

See Also:

ApplyOreOperator, Delta, Der, Euler, OreAction, OreDelta, OreSigma, S, ToOrePolynomial
QSolvePolynomial

\texttt{QSolvePolynomial[eqn, f[x], q]}

determines whether the linear \( q \)-shift equation \( eqn \) in \( f[x] \) (with polynomial coefficients) has polynomial solutions, and in the affirmative case, computes them.

\[ \text{More Information} \]

A \( q \)-shift equation is an equation that involves \( f[x] \), \( f[qx] \), \( f[q^2x] \), etc. It may be given as an equation (with head \texttt{Equal}) or as the left-hand side expression (which is then understood to be equal to zero). If the coefficients of \( eqn \) are rational functions, it is multiplied by their common denominator. The algorithm works by determining a degree bound and then making an ansatz for the solution with undetermined coefficients.

The command \texttt{QSolvePolynomial} is able to deal with parameters; these have to occur linearly in the inhomogeneous part. Call the parameterized version using the option \texttt{ExtraParameters}.

The following options can be given:

\texttt{ExtraParameters} \rightarrow \{\}

- specify some extra parameters for which the equation has to be solved.

\[ \text{Examples} \]

\texttt{In[303]:= QSolvePolynomial[f[qx] - q^{10} f[x] == 0, f[x], q]}
\texttt{Out[303]= \{\{f[x] \rightarrow C[1] x^{10}\}\}}

\texttt{In[304]:= QSolvePolynomial[f[q^2 x] + f[qx] + f[x] ==}
\texttt{\quad (q^4 + q^2 + 1)x^2 + (q^3 + q^2 + q)x + 3q^7, f[x], q]}
\texttt{Out[304]= \{\{f[x] \rightarrow q^7 + qx + x^2\}\}}

\texttt{In[305]:= QSolvePolynomial[f[q^3 x] + f[x] - cx^3, f[x], q, ExtraParameters -> c]}
\texttt{Out[305]= \{\{f[x] \rightarrow C[1] x^3, c \rightarrow C[1] (q^{9} + 1)\}\}}

\[ \text{See Also} \]

\texttt{DSolvePolynomial, DSolveRational, QSolveRational, RSolvePolynomial, RSolveRational}
QSolveRational

QSolveRational[eqn, f[x], q]

determines whether the linear q-shift equation eqn in f[x] (with polynomial coefficients) has rational solutions, and in the affirmative case, computes them.

More Information

A q-shift equation is an equation that involves f[x], f[qx], f[q^2x], etc. It may be given as an equation (with head Equal) or as the left-hand side expression (which is then understood to be equal to zero). If the coefficients of eqn are rational functions, it is multiplied by their common denominator. Following Abramov's algorithm [2], first the denominator of the solution is determined. Then QSolvePolynomial is called to find the numerator polynomial.

The command QSolvePolynomial is able to deal with parameters; these have to occur linearly in the inhomogeneous part. Call the parameterized version using the option ExtraParameters.

The following options can be given:

ExtraParameters → {}  
specify some extra parameters for which the equation has to be solved.

Examples

In[306]:= QSolveRational[q^2 f[qx] - f[x] + (a1 x + a0)/(x + 1), f[x], q, ExtraParameters → {a0, a1}]

In[307]:= QSolveRational[q^3(qx + 1) f[q^2 x] - 2q^2(x + 1) f[qx] + (x + q) f[x]  
   == (q^6 - 2q^3 + 1)x^2 + (q^5 - 2q^3 + 1)x, f[x], q]
   - q x^3 - q x^2 - x^3 - x^2) / ((q - 1)(q + 1)(q + q x))}}

See Also

DSolvePolynomial, DSolveRational, QSolvePolynomial, 
RSolvePolynomial, RSolveRational
RandomPolynomial

RandomPolynomial[var, deg, c]
gives a dense random polynomial in the variable(s) var of degree deg with integer coefficients between −c and c.

\[\text{RandomPolynomial[x, 5, 10^6]}\]
\[\text{RandomPolynomial[{x, y, z}, 2, 1000]}\]
\[\text{RandomPolynomial[{x, y, z}, {1, 2, 3}, 100]}\]
\[\text{RandomPolynomial[{a, z, S[a], Der[z]}, 3, 1]}\]
\[\text{ToOrePolynomial[%, OreAlgebra[S[a], Der[z]]]}\]

\[\text{OrePolynomial, ToOrePolynomial}\]
RSolvePolynomial

RSolvePolynomial[eqn, f[n]]
determines whether the linear recurrence equation eqn in f[n] (with polynomial coefficients) has polynomial solutions, and in the affirmative case, computes them.

More Information

The first argument eqn can be given either as an equation (with head Equal), or as the left-hand side expression (which is then understood to be equal to zero). If the coefficients of eqn are rational functions, it is multiplied by their common denominator. The second argument is the function to be solved for. The algorithm works by determining a degree bound and then making an ansatz for the solution with undetermined coefficients.

The command RSolvePolynomial is able to deal with parameters; these have to occur linearly in the inhomogeneous part. Call the parameterized version using the option ExtraParameters.

The following options can be given:

ExtraParameters → {}
specifies some extra parameters for which the equation has to be solved.

Examples

 RSolvePolynomial[(n^3 + 1)f[n + 1] - 3(n^3 + n^2 + n)f[n] == 2 - 2n^6, f[n]]
Out[313]= \{f[n] \rightarrow n^3 + 1\}

 RSolvePolynomial[eqn, f[k]]
Out[314]= {}

 RSolvePolynomial[eqn, f[k], ExtraParameters \rightarrow \{x[0], x[1]\}]
Out[315]= \{\{f[k] \rightarrow C[1]k, x[0] \rightarrow -2C[1], x[1] \rightarrow C[1]\}\}

See Also

DSolvePolynomial, DSolveRational, QSolvePolynomial, QSolveRational, RSolveRational
RSolveRational

**RSolveRational**[eqn, f[n]]
determines whether the linear recurrence equation eqn in f[n] (with polynomial coefficients) has rational solutions, and in the affirmative case, computes them.

\[\begin{align*}
\text{More Information} \\
\text{The first argument } eqn \text{ can be given either as an equation (with head } \text{Equal}, \text{ or as the left-hand side expression (which is then understood to be equal to zero). If the coefficients of } eqn \text{ are rational functions, it is multiplied by their common denominator. The second argument is the function to be solved for. Following Abramov’s algorithm [2], first the denominator of the solution is determined. Then } \text{RSolvePolynomial} \text{ is called to find the numerator polynomial.}
\text{The command } \text{RSolveRational} \text{ is able to deal with parameters; these have to occur linearly in the inhomogeneous part. Call the parameterized version using the option } \text{ExtraParameters}. \\
\text{The following options can be given:}
\begin{align*}
\text{ExtraParameters} & \rightarrow \{\} \\
& \text{specifies some extra parameters for which the equation has to be solved.}
\end{align*}
\text{Examples}
\begin{align*}
\text{In}[317]:= & \text{RSolveRational}[\text{(2} n + 7) f[n + 2] + (2 n^2 + 7 n + 5) f[n + 1] - (6 n^2 + 7 n - 3) f[n] == 3 - 2 n, f[n]] \\
\text{Out}[317]= & \{\{ f[n] \rightarrow \frac{1}{2n + 3}\}\}
\text{In}[318]:= & \text{eqn = (} k - 1 - n\}(n - k) f[1 + k] - (k + 1)(k - n - 1) f[k] + (k + 1)(k - 1 - n)x[0] - (k + 1)(n + 1)x[1] \\
\text{Out}[318]= & -(k - n - 1)(k - n) f[k + 1] - (k + 1)(k - n - 1)f[k] + (k + 1)(k - n - 1)x[0] - (k + 1)(n + 1)x[1] \\
\text{In}[319]:= & \text{RSolveRational}[\text{eqn, } f[k]] \\
\text{Out}[319]= \{\}
\text{In}[320]:= & \text{RSolveRational}[\text{eqn, } f[k], \text{ExtraParameters} \rightarrow \{x[0], x[1]\}] \\
\text{Out}[320]= \{\{ f[k] \rightarrow -C[1]k \text{, } x[0] \rightarrow -2C[1], x[1] \rightarrow C[1]\}\}
\end{align*}
\text{See Also}
\begin{align*}
\text{DSolvePolynomial, DSolveRational, QSolvePolynomial, QSolveRational, RSolvePolynomial}
\end{align*}
\text{80}
\[S\]

\[S[n]\]

represents the forward shift operator with respect to \(n\).

\[\n\]

\[\n\]

More Information

When this operator occurs in an \texttt{OrePolynomial} object, it is displayed as \(S\).

The symbol \(S\) receives its meaning from the definitions of \texttt{OreSigma}, \texttt{OreDelta}, and \texttt{OreAction}.

\[\n\]

Examples

\(\texttt{In[321]= OreSigma[S[n]]}\)

\(\texttt{Out[321]= #1 /. n \to n + 1 &}\)

\(\texttt{In[322]= OreDelta[S[n]]}\)

\(\texttt{Out[322]= 0 &}\)

\(\texttt{In[323]= ApplyOreOperator[S[n]^3, f[n]]}\)

\(\texttt{Out[323]= f[n + 3]}\)

The symbol \(S\) itself does not do anything. In order to perform noncommutative arithmetic, it first has to be embedded into an \texttt{OrePolynomial} object.

\(\texttt{In[324]= S[n] ** n^2}\)

\(\texttt{Out[324]= S[n] ** n^2}\)

\(\texttt{In[325]= ToOrePolynomial[S[n]]}\)

\(\texttt{Out[325]= S_n}\)

\(\texttt{In[326]= % ** n^2}\)

\(\texttt{Out[326]= (n^2 - 2n + 1)S_n}\)

See Also

\texttt{ApplyOreOperator, Delta, Der, Euler, OreAction, OreDelta, OreSigma, QS, ToOrePolynomial}
SolveCoupledSystem

\[ \text{SolveCoupledSystem}[\text{eqns}, \{f_1, \ldots, f_k\}, \{v_1, \ldots, v_j\}] \]
computes all rational solutions of a coupled system of linear difference and
differential equations.

More Information
The first argument \textit{eqns} is a list of linear equations in the functions \( f_1, \ldots, f_k \),
their shifts and/or their derivatives, the second argument are the names of these
functions, and in the third argument the variables on which the \( f_i \) depend.
\textbf{SolveCoupledSystem} uses \textbf{OreGroebnerBasis} to uncouple the given system
(which corresponds to Gaussian elimination). The advantage (compared to
\textbf{SolveOreSys}) is that more general types of systems can be dealt with, i.e.,
there are no restrictions concerning the order, and even mixed difference and
differential systems can be addressed. Additionally, this command is also more
reliable than \textbf{SolveOreSys}.

Note that \textbf{SolveCoupledSystem} addresses only linear systems; if the input is not
linear (which is not checked), the result most probably will be wrong.

The following options can be given:

\textbf{ExtraParameters} \rightarrow \{\}

specifies some extra parameters for which the equations have to be solved;
these parameters are allowed to occur in \textit{eqns} only linearly and only in
the inhomogenous parts.

\textbf{Return} \rightarrow "solution"

specifies what result should be returned; by default this is the solution of
the \textit{eqns}. \textbf{Return} \rightarrow "uncoupled" returns the uncoupled system without
solving.

Examples

\begin{verbatim}
s[27] := SolveCoupledSystem[ 
   \{(n + 1)f[x] + (x + nx)g[x] + (x^2 - 1)g'[x] == 0, 
   (-x - nx)f[x] - (n + 1)g[x] + (x^2 - 1)f'[x] == 1 - x^2\}, 
   \{f, g\}, x] 
\end{verbatim}

\begin{verbatim}
o[28] := SolveCoupledSystem[ 
   \{-(n + 2)a[1 + n] + (n + 2)b[n] - (2n + 3)b[1 + n] == 0, 
   (n + 2)a[n] + (n + 1)b[1 + n] - n - 2 == 0\}, \{a, b\}, n] 
\end{verbatim}

\begin{verbatim}
o[29] := SolveCoupledSystem[ 
\end{verbatim}

\begin{verbatim}
s[30] := SolveCoupledSystem[ 
   \{f[k + 1, x] - f[k, x] + D[f2[k, x], x] - 3kx^2 - x^3, 
   x^2[f[k, x] - a D[f2[k, x], x]] \}, \{f1, f2\}, \{k, x\}] 
\end{verbatim}

82
Out[328]= \{\{f1[k, x] \rightarrow k x^3, f2[k, x] \rightarrow k x^3 - C[1]\}\}

\[\text{See Also}\]

DSolveRational, RSolveRational, SolveOreSys
SolveOreSys

\[ \text{SolveOreSys[type, var, eqns, \{f_1[var], \ldots, f_k[var]\}, \text{pars}] } \]

computes all rational solutions of the first-order coupled linear difference or
differential system eqns.

\[ \text{\textbf{More Information}} \]

The input consists of five arguments: type is either \textbf{S} or \textbf{Der}, indicating that
a difference resp. differential system has to be solved, var is the variable, eqns
is a list of equations, \( f_1[var], \ldots, f_k[var] \) are the functions to be solved for,
and pars is a list of extra parameters, that are allowed to occur linearly in the
inhomogeneous parts.

The system of equations is uncoupled using the corresponding uncoupling pro-
cedure from Stefan Gerhold’s OreSys package \cite{7} (this package has to be loaded
in advance). Then the scalar equations are solved with the functions DSolve-
Rational and RSolveRational, respectively, and by backwards substitution.

For the uncoupling, some dummy functions psi are created; they also show up
in the output.

This command is kind of obsolete since SolveCoupledSystem offers more
powerful solving abilities.

The following options can be given:

\[ \text{Method } \rightarrow \text{OreSys'Zuercher} \]
this option is passed to the uncoupling procedure.

\[ \text{\textbf{Examples}} \]

\[ \text{In[330]:= << OreSys.m} \]

OreSys Package by Stefan Gerhold – ©RISC Linz – V 1.1 (12/02/02)

\[ \text{In[331]:= SolveOreSys[Der, x,} \]
\{\( (1 + n)[x] + (x + nx)[g[x] + (-1 + x^2)g'[x] == 0, \)
\( 1 - x^2 == (-x - nx)f[x] + (-1 - n)g[x] + (-1 + x^2)f'[x], \}
\{f[x], g[x]\}, \{\} \}

\[ \text{Out[331]= } \{\{\text{HolonomicFunctions’Private’psi}\$24336[1][x] \rightarrow \frac{x}{n}, \}
\text{HolonomicFunctions’Private’psi}\$24336[2][x] \rightarrow \frac{n + 1}{n}, \}
\{f[x] \rightarrow \frac{x}{n}, g[x] \rightarrow \frac{1}{n} \}\} \}

\[ \text{In[332]:= SolveOreSys[S, n,} \]
\]

84
\[\{(−2−n)a[1+n]+(2+n)b[n]+(−3−2n)b[1+n]==0,−2−n+(2+n)a[n]+(1+n)b[1+n]==0\}\{a[n],b[n]\}\}\]

\[oa[nn]=\left\{\begin{array}{l} \text{HolonomicFunctions'Private'psi$24516[1][n] \rightarrow C[1]n + C[1] - n^2 + 2,} \\
\text{HolonomicFunctions'Private'psi$24516[2][n] \rightarrow C[1]n + C[1] - 2n^2 - n + 2,} \\
\text{a[n] \rightarrow C[1]n + C[1] - n^2 + 2, b[n] \rightarrow (n + 1) (-C[1] + n - 2)} \end{array}\right\}\]

\(\text{See Also}\)

DSolveRational, RSolveRational, SolveCoupledSystem
Support

Support[opoly]
gives the support of the OrePolynomial opoly.

More Information

The input has to be given as an OrePolynomial expression. By its support, we understand the list of power products which have nonzero coefficient. These power products are returned again as OrePolynomial expressions.

Examples

In[333]:= opoly = ToOrePolynomial[(S[n] + Der[x] + nx)^2, OreAlgebra[S[n], Der[x]]]
Out[333]= S^2 n + 2 S n Der[x] + Der[x]^2 + (2 nx + x) S[n] + 2 n x Der[x] + (n^2 x^2 + n)
In[334]:= Support[opoly]
Out[334]= {S^2, S[n], Der[x], S[n], Der[x], 1}
In[335]:= opoly = ChangeOreAlgebra[opoly, OreAlgebra[n, x, S[n], Der[x]]]
Out[335]= n^2 x^2 + 2 nx S[n] + 2 n x Der[x] + x S[n] + S[n]^2 + 2 S[n] Der[x] + Der[x]^2 + n
In[336]:= Support[opoly]
Out[336]= {n^2 x^2, nx S[n], nx Der[x], x S[n], S[n]^2, S[n] Der[x], Der[x]^2, n}
In[337]:= Support[opoly - opoly]
Out[337]= {}

See Also

FindRelation, LeadingPowerProduct, OrePolynomial, OrePolynomialListCoefficients
Takayama

\texttt{Takayama[\textit{ann, vars}]}  
performs Takayama's algorithm for definite summation and integration with  
natural boundaries on a function annihilated by \textit{ann}, summing and integrating  
with respect to \textit{vars}.

\texttt{Takayama[\textit{ann, vars, alg}]}  
converts all elements of \textit{ann} into the Ore algebra \textit{alg} and then performs  
Takayama's algorithm.

\begin{itemize}
  \item \textbf{More Information}
  \end{itemize}

The input \textit{ann} is a list of \texttt{OrePolynomial} expressions or a list of standard  
Mathematica polynomials (then the third argument, an Ore algebra into which  
these are translated, has to be given). \textit{vars} is the list of variables with respect  
to which the summation and integration is done. More generally, \textit{vars} contains  
all variables to be eliminated (Takayama's algorithm is based on solving an  
elimination problem \cite{11}).

The variables for which a shift operator belongs to the Ore algebra are interpreted as summation variables. The variables for which a differential operator belongs to the Ore algebra are interpreted as integration variables.

The algorithm loops over the degree of \textit{vars}, trying to find an operator free of  \textit{vars} in the module that is truncated at this degree. \texttt{Takayama} can run into an  
infinite loop for two reasons: either the input is not holonomic, or it is holonomic  
but this property is lost by extension/contraction. The fact which degrees are  
tried can be influenced by the options \texttt{StartDegree} and \texttt{MaxDegree}.

The following options can be given:

\begin{itemize}
  \item \textbf{Extended} \rightarrow \texttt{False}  
setting this option to \texttt{True} computes also the delta parts (which can be  
very costly).
  \item \textbf{Incomplete} \rightarrow \texttt{False}  
setting this option to \texttt{True} causes that the computation is interrupted as  
soon as an element free of the elimination variables \textit{vars} is found.
  \item \textbf{Method} \rightarrow "sugar"  
this option is passed to \texttt{OreGroebnerBasis} and specifies the pair selection  
strategy.
  \item \textbf{Modulus} \rightarrow 0  
give a prime number here for modular computations.
  \item \textbf{Reduce} \rightarrow \texttt{False}  
if this is set to \texttt{True} then the delta parts are reduced to normal form with  
respect to the input annihilator (works only in connection with \texttt{Extended}).
\end{itemize}
Saturate → False
when this option is set to True then it is tried to saturate the annihilating ideal in the polynomial algebra (“Weyl closure”) by an additional Gröbner basis computation; this can increase the number of solutions, and/or decrease the order of the result.

StartDegree → 0
the degree with respect to vars from which on it is tried to find operators free of vars in the truncated module.

MaxDegree → Infinity
the maximal degree with respect to vars for which it is tried to find operators free of vars in the truncated module; set it to a finite number to prevent an infinite loop.

\[ \text{Examples} \]
We start with Moll’s quartic integral
\[
\int_0^\infty \frac{1}{(x^4 + 2ax^2 + 1)^{m+1}} \, dx.
\]

\[ \text{In[338]:=} \text{ann} = \text{Annihilator}[1/(x^4 + 2ax^2 + 1)^{(m + 1)}, \{S[m], \text{Der}[x], \text{Der}[a]\}] \]
\[ \text{Out[338]} = \{D_a (2ax^2 + x^4 + 1), \text{Der}[x] (4ax + 4mx + 4mx^3 + 4x^3), \text{Der}[a] (2ax^2 + x^4 + 1)S_m - 1\} \]

\[ \text{In[339]:=} \text{Takayama}[\text{ann}, \{x\}] \]
\[ \text{Out[339]} = \{-4m - 4S_m + 2aD_a + (4m + 3)\} \]

\[ \text{In[340]:=} \text{Takayama}[\text{ann}, \{x\}, \text{Extended} \rightarrow \text{True}] \]
\[ \text{Out[340]} = \{-4m - 4S_m + 2aD_a + (4m + 3), \{2ax^3 + x^5 + x\}\} \]

\[ \text{In[341]:=} \text{Takayama}[\text{ann}, \{x\}, \text{Extended} \rightarrow \text{True}, \text{Reduce} \rightarrow \text{True}] \]
\[ \text{Out[341]} = \{-4m - 4S_m + 2aD_a + (4m + 3), \{x\}\} \]

\[ \text{In[342]:=} \text{Takayama}[\text{ann}, \{x\}, \text{Saturate} \rightarrow \text{True}] \]
\[ \text{Out[342]} = \{-D_s + 2r, -D_r + 2s\} \]

With Takayama’s algorithm, multiple sums and integrals can be done in one stroke, e.g.,
\[
\int_{-\infty}^{\infty} \left( \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{e^{-x^2} x^m s^n H_m(x) H_n(x)}{m!n!} \right) \, dx = \sqrt{\pi} e^{2rs}.
\]

\[ \text{In[343]:=} \text{ann} = \text{Annihilator}[\text{HermiteH}[m, x] \text{HermiteH}[n, x] r^m s^n (\text{Exp}[-x^2]/m!)/n!],\]
\[ \{S[m], S[n], \text{Der}[x], \text{Der}[r], \text{Der}[s]\}\]
For $q$-summation, either the summation variables can be given or their corresponding $q$-powers. In the first example, the fourth-order recurrence is found at degree 6 (can be seen when switching to verbose mode using Printlevel). Trying a higher degree results in a shorter recurrence.

\[
\text{In[345]:=} \text{ann} = \text{Annihilator}[(i + j)^2 + j^2]/\text{QPochhammer}[q, q, n - i - j] \\
/\text{QPochhammer}[q, q, i]/\text{QPochhammer}[q, q, j] \\
\{\text{QS}[qi, q^i], \text{QS}[qj, q^j], \text{QS}[qn, q^n]\}
\]

\[
\text{Out[345]} = \{(qi qj - q^{2n})S[q n, q^2 - qi qj] \\
(qi - 1)S[q, q^n + (q j q^i - q^2 qi q^3 q^n) \\
(qi - 1)S[q, q^n + (q j q^i - q^2 qi q^n)]\}
\]

\[
\text{In[346]:=} \text{Takayama}[\text{ann}, \{i, j\}]
\]

\[
\text{Out[346]} = \{(1 - q^{2n})S^{\text{ii}} + (q j q^i - q^3 q^n - q^3 q^n + q^2 q^n + q^n + q^n q^n q^n - q^n - q^n - q^n - q^n)S^{\text{ii}} + \\
(q j q^i - q^3 q^n + q j q^n + 2 q^n + q^n - q^n - q^n - q^n + q^n)S^{\text{ii}} + \\
(q j q^i - q^3 q^n + q j q^n - q^n - q^n)S^{\text{ii}} + q^n\}
\]

\[
\text{In[347]:=} \text{Takayama}[\text{ann}, \{qi, qj\}, \text{StartDegree} \to 7]
\]

\[
\text{Out[347]} = \{(q j q^n - 1)S^{\text{ii}} + (q j q^n - q^n q^n + q^2 q^n q^n - q^4 q^n + q^n + q^n + q^n + q^n + q^n + q^n)S^{\text{ii}} + \\
(q j q^n - q^n q^n q^n - q^n q^n - q^n - q^n - q^n + q^n)S^{\text{ii}} + q^n\}
\]

In the last example, the input is not holonomic, and hence Takayama would run for ever if not given the option MaxDegree.

\[
\text{In[348]:=} \text{Takayama}[\text{Annihilator}[1/(k^2 + n^2), \{S[k], S[n]\}], k, \text{MaxDegree} \to 20]
\]

\[
\text{Out[348]} = $\text{Failed}$
\]

\[\text{See Also}\]

Annihilator, CreativeTelescoping, OreGroebnerBasis, Printlevel
ToOrePolynomial

ToOrePolynomial[expr]
converts expr to an OrePolynomial expression.

ToOrePolynomial[expr, alg]
converts expr to an OrePolynomial expression in the Ore algebra alg.

ToOrePolynomial[eqn, {f[v1, ..., vk]}]
converts the equation of f into operator notation.

More Information

The input expr must be a polynomial in the involved Ore operators and all other generators of alg (if specified); in particular these are not allowed in the denominator. Also all Ore operators that occur in expr must be part of alg. If an equation eqn is given, then the Ore operators are determined by the occurrences of f. The equation eqn must be linear and homogeneous.

If expr is a standard Mathematica polynomial, i.e., if it does not involve NonCommutativeMultiply, then it is assumed that in its expanded form all monomials are in standard form according to the order of the generators of alg (regardless how Mathematica sorts the factors).

Otherwise use NonCommutativeMultiply, written as **, to fix the order of the factors in a product.

If no Ore algebra is given, then the rational Ore algebra that is generated by all Ore operators in expr is chosen.

The following options can be given:

MonomialOrder → DegreeLexicographic
the monomial order in which the terms of the Ore polynomial are ordered; see the description of OreGroebnerBasis (p. 57) for a list of supported monomial orders.

Examples

In[349]:= ToOrePolynomial[(1 - x^2) Der[x] + (n + 1) S[n] - x - nx]
Out[349]= (1 - x^2) D[x] + (n + 1) S[n] + (-nx - x)

In[350]:= ToOrePolynomial[(-x - nx) f[x, n] + (1 + n) f[x, 1 + n] +
(1 - x^2) D[f[x, n], x], f[x, n]]
Out[350]= (-x - nx) f[x, n] + (1 + n) f[x, 1 + n] +
(1 - x^2) D[f[x, n], x], f[x, n]]

In[351]:= ToOrePolynomial[Der[x] x, OreAlgebra[Der[x], x]]
Out[351]= x D[x]

In[352]:= ToOrePolynomial[Der[x] x, OreAlgebra[Der[x], x]]
\[ \text{Out}[352] = D_x \] 
\[ \text{In}[353] := \text{ChangeOreAlgebra}[\%, \text{OreAlgebra}['%']] \] 
\[ \text{Out}[353] = xD_x + 1 \] 
\[ \text{In}[354] := \text{ToOrePolynomial}[S[n] ** (1/n^3)] \] 
\[ \text{Out}[354] = \frac{1}{(n+1)^3} S_n \] 
\[ \text{In}[355] := \text{ToOrePolynomial}[S[n] ** (1/n^3), \text{OreAlgebra}[n, S[n]]] \] 
\[ \text{ToOrePolynomial::nopoly} : \text{The input is not a polynomial w.r.t. generators of the algebra.} \] 
\[ \text{Out}[355] = \text{$Failed} \] 
\[ \text{In}[356] := \text{ToOrePolynomial}[\text{Der}[x]^2 ** (f[x] + x^3)] \] 
\[ \text{Out}[356] = (f[x] + x^3) D_x^2 + (2f'[x] + 6x^2) D_x + (f''[x] + 6x) \] 
\[ \text{In}[357] := \text{ToOrePolynomial}[S[n] ** n] \] 
\[ \text{OrePower::exp} : \text{Invalid exponent n.} \] 
\[ \text{ToOrePolynomial::badalg} : \text{The input cannot be represented in the corresponding algebra.} \] 
\[ \text{Out}[357] = \text{$Failed} \] 

\[ \text{See Also} \] 
\text{OreAlgebra, OreOperatorQ, OreOperators, OrePolynomial} \]
UnderTheStaircase

UnderTheStaircase[gb]
computes the list of monomials (power products) that lie under the stairs
of the Gröbner basis gb.

UnderTheStaircase[exps]
computes the list of exponent vectors that lie under the stairs defined by
the exponent vectors exps.

More Information

The input gb is a list of OrePolynomial expressions. The command UnderTheStaircase just looks at their leading power products and assumes that they in fact form a Gröbner basis (this property is not checked). Alternatively, the exponent vectors of the leading power products can be given.

If the left ideal that is generated by gb is zero-dimensional, then the power
products that lie under its stairs are returned as a list of OrePolynomial expressions. If the ideal is not zero-dimensional, the symbol Infinity is returned.

If exponent vectors are given, then also a list of exponent vectors (or Infinity) is returned.

If the input consists of OrePolynomial expressions, the output is sorted ac-
cording to the monomial order in which gb is given. If the input consists of
exponent vectors, the output is not sorted.

Examples

In[358]:= gb = Annihilator[StruveH[n, x]]
Out[358]= {x \cdot D^2 x + (2 nx - x) S_n - 2 n x \cdot S_n + (n^2 + n + x^2),
x \cdot S_n - (n + 1) S_n - x,
S_n \cdot D_n + (2 n x + 3 x) S_n^2 + (-4 n^2 - 10 n - x^2 - 6) S_n - x^2 \cdot D_n + (3 n x + 3 x)}
In[359]:= UnderTheStaircase[gb]
Out[359]= {1, D_n, S_n}
In[360]:= UnderTheStaircase[Take[gb, 2]]
Out[360]= Infinity
In[361]:= UnderTheStaircase[ToOrePolynomial[{1}, OreAlgebra[Der[x], S[y]]]]
Out[361]= {}
References


