All packages presented here are avaible for download at
https : // combinatorics.risc.jku.at/software

In[1]:=
```
<< RISC`GeneratingFunctions`
```

```
Package GeneratingFunctions version 0.8 written by Christian Mallinger
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria
```

In[2]:=
```
? REPlus
```

```
RecurrenceEquationPlus[re1,re2,a[n]] gives a recurrence equation that
is satisfied by the sum of solutions of the recurrences re1 and re2.
All recurrences are given in a[n].

Alias: REPlus

See also: REInfo, DEPlus
```

defining the sequences of Perrin and Lucas numbers

In[3]:=
```
perrin = {a[n + 3] - a[n + 1] - a[n] == 0, a[0] == 3, a[1] == 0, a[2] == 2};
lucas = {a[n + 2] - a[n + 1] - a[n] == 0, a[0] == 2, a[1] == 1};
```

computing a recurrence for the addition of Perrin and Lucas numbers

In[5]:=
```
REPlus[perrin, lucas, a[n]]
```

Out[5]=
```
{a[n] + 2 a[1 + n] - 2 a[3 + n] - a[4 + n] + a[5 + n] == 0,
 a[0] == 5, a[1] == 1, a[2] == 5, a[3] == 7, a[4] == 9}
```

In[6]:=
```
REPlus[perrin[[1]], lucas[[1]], a[n]]
```

Out[6]=
```
a[n] + 2 a[1 + n] - 2 a[3 + n] - a[4 + n] + a[5 + n] == 0
```

In[7]:=
```
? RE*
```

▼ RISC`GeneratingFunctions`

RE	RE2L	REHadamard	REInterlace	REPlus	RESubsequence
RE2DE	RECauchy	REInfo	REOut	REShadow	

In[8]:=
```
? RESubsequence
```

RecurrenceEquationSubsequence[re,a[n],m*n+k] gives a recurrence that
is satisfied by a subsequence of the form a[m*n+k] of every solution a[n] of
the input recurrence re.

Alias: RESubsequence

See also: REInfo, REInterlace

Computing a recurrence for all odd - indexed Lucas numbers

In[9]:=
```
RESubsequence[lucas, a[n], 2 n + 1]
```

Out[9]=
```
{a[n] - 3 a[1 + n] + a[2 + n] == 0, a[0] == 1, a[1] == 4}
```

Cauchy product of Perrin and Lucas numbers

In[10]:=
```
RECauchy[perrin, lucas, a[n]]
```

Out[10]=
```
{a[n] + 2 a[1 + n] - 2 a[3 + n] - a[4 + n] + a[5 + n] == 0,
 a[0] == 6, a[1] == 3, a[2] == 13, a[3] == 20, a[4] == 34}
```

defining the Fibonacci sequence

In[11]:=
```
fib = {F[n + 2] - F[n + 1] - F[n] == 0, F[0] == 0, F[1] == 1};
```

Computing the recurrence for the Cauchy product of Fibonacci numbers with the constant sequence 1, i.e., for the partial sum of Fibonacci numbers

In[12]:=
```
RECauchy[fib, {F[n] == 1}, F[n]]
```

Out[12]=
```
{F[n] - 2 F[2 + n] + F[3 + n] == 0, F[0] == 0, F[1] == 1, F[2] == 2}
```

Fibonacci, Perrin and Lucas numbers also exist as built -in functions in Mathematica

In[13]:=
```
? Fibonacci
```

Fibonacci[n] gives the Fibonacci number F_n.
Fibonacci[n, x] gives the Fibonacci polynomial $F_n(x)$. ≫

In[14]:=
```
? Perrin
```

Perrin[n] gives the nth Perrin number.

In[15]:=
```
? LucasL
```

LucasL[n] gives the Lucas number L_n.
LucasL[n, x] gives the Lucas polynomial $L_n(x)$. ≫

In[16]:=
```
Fibonacci[0]
```

Out[16]=
```
0
```

creating data for guessing

In[17]:=
```
dataL = Table[LucasL[nn], {nn, 0, 20}];
dataP = Table[Perrin[nn], {nn, 0, 20}];
```

In[19]:=
```
dataL
```

Out[19]=
```
{2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199,
 322, 521, 843, 1364, 2207, 3571, 5778, 9349, 15 127}
```

In[20]:=
```
? GuessRE
```

GuessRecurrenceEquation[list,a[n],{minorder,maxorder},{mindeg,maxdeg}]
tries to guess a linear recurrence equation (RE) in a[n] with polynomial
coefficients, which is satisfied by the elements in the input list.
The orders that are tried range from "minorder" to "maxorder",
the coefficient polynomials are tried with degrees "mindeg" up to
"maxdeg". The output contains a RE (or FAIL, if no recurrence could be
found) together with a transformation that had to be performed on the
generating function of the input list. Short forms for the function call are

GuessRecurrenceEquation[list,a[n]] and

GuessRecurrenceEquation[list,a[n],maxorder,maxdeg],

where the default values minorder=1, maxorder=2, mindeg=0, maxdeg=3 are used.

GuessRecurrenceEquation has the following options (and default values):

AdditionalEquations ("All") In order to avoid accidental results,

"All" elements in the input list are used

to build the equations for the coefficients

of the RE. Setting this parameter to

a positive integer k, causes the function

to build just d+k equations, where d is the

number of indeterminants. This option can

be used to get a speed up.

Hypergeom (False) whether to search for m–hypergeometric

recurrences only.

Transform ({"ogf","egf"}) transformations that are tried.

Note: The first element in the list gives the term a[0] in the sequence.

Alias: GuessRE

See also: GuessDE, GuessAE, GuessRatF, ListOfTransformations

Guessing the recurrence for Lucas numbers

In[21]:=
```
GuessRE[dataL, a[n]]
```

Out[21]=
```
{{-a[n] - a[1 + n] + a[2 + n] == 0, a[0] == 2, a[1] == 1}, ogf}
```

Guessing the recurrence for Perrin numbers -> need to increase the maximal order used!

In[22]:=
```
GuessRE[dataP, a[n]]
```

Out[22]=
```
FAIL
```

In[23]:=
```
GuessRE[dataP, a[n], {0, 3}, {0, 0}]
```

Out[23]=
```
{{-a[n] - a[1 + n] + a[3 + n] == 0, a[0] == 3, a[1] == 0, a[2] == 2}, ogf}
```

In[24]:=
```
<< RISC`HolonomicFunctions`
<< RISC`Guess`
```

HolonomicFunctions Package version 1.7.3 (21-Mar-2017)
written by Christoph Koutschan
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria

```
--> Type  ?HolonomicFunctions  for help.
```

Guess Package version 0.52
written by Manuel Kauers
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria

In[26]:=
```
GuessMinRE[dataP, a[n]]
```

Out[26]=
$$-a[n]-a[1+n]+a[3+n]$$

In[27]:=
```
? Guess*
```

▼ RISC`GeneratingFunctions`

Guess	GuessAlgebraic-Equation	GuessDifferenti-alEquation	GuessRational-Function	GuessRecurren-ceEquation
GuessAE	GuessDE	GuessRatF	GuessRE	

▼ RISC`Guess`

GuessCurveRE	GuessMinDE	GuessMultDE	GuessSymmetr-icRE	GuessUnivDE
GuessMinAE	GuessMinRE	GuessMultRE	GuessUnivAE	GuessUnivRE

GuessUnivAE[list, f[x], dy, dx, mod] finds an algebraic equation of degree dy with polynomial coefficients of degree x over either QQ or the finite field with mod elements.

Example from lecture : both C - finite and hypergeometric

In[28]:=
```
data = Table[2^(n-1) (n+2), {n, 0, 30}];
```

In[29]:=
```
GuessMinRE[data, a[n]]
```

Out[29]=
$$(-6-2n)\,a[n]+(2+n)\,a[1+n]$$

In[30]:=
```
? GuessMultRE
```

GuessMultRE[data, strset, vars, deg] computes a vector
space basis of the space of all recurrences satisfied by the array data.

data... a d–dimensional rectangular array of numbers or rational functions
strset... a list of expressions of the form f[n+int,m+int,...]
with f being a symbol, n,m,... being variables and int being integers
vars... a list of variables, e.g., {n,m,k}, or a list of blocks, e.g., {{n,m},{k}}, indicating
that the polynomial coefficients in the recurrence equations should be symmetric
wrt. all the variables belonging to the same block. {n,m,k} is equivalent to
{{n},{m},{k}}. If a variable is of the form q^n, then the q–shift is applied in direction n.
deg... total degree bound for the polynomial coefficients in the recurrence
equations, or a list of individual degree bounds for each block.

Options:
Modulus –> prime number used for solution shape prediction. (0 to skip this step.)
AdditionalEquations –> number of extra equations to be used, Infinity to take all data.
AdditionalTerms –> list of terms to be included in the ansatz
in addition to those following from structure set and degree specification.
StartPoint –> point corresponding to data[[1,1,1...]], default {0,0,0,...}
Extension –> minimal polynomial of a single parameter, or {} if there is not algebraic extension.
Except –> list of terms, e.g., n^3 m f[n+2, m+1], that must not occur in the recurrences. If a term is
embraced with Cone[ ], e.g., Cone[n^3 m f[n+2, m+1]], all multiplies of it will be excluded, too.
MustHaveOneOf –> list of terms, e.g., n^3 m f[n+2, m+1], of which
at least one must occur in the recurrences, default All
Constraints –> logical expression constraining the points from which
the sample values should be chosen, e.g., Abs[n–m]<=3
Sort –> internal ordering for terms to be used in the ansatz
Factor –> True or False depending on whether coefficients of resulting recurrences should be factored.

In[31]:=
```
GuessMultRE[data, {a[n], a[n + 1], a[n + 2]}, {n}, 0]
```
Out[31]=
```
{4 a[n] - 4 a[1 + n] + a[2 + n]}
```

In[32]:=
```
GuessMinRE[Table[Fibonacci[n], {n, 0, 50}], f[n]]
```
Out[32]=
```
-f[n] - f[1 + n] + f[2 + n]
```

In[34]:=
```
Flatten[Table[s2[n + nn, k + kk], {nn, 0, 1}, {kk, 0, 1}]]
```
Out[34]=
```
{s2[n, k], s2[n, 1 + k], s2[1 + n, k], s2[1 + n, 1 + k]}
```

In[33]:=
```
GuessMultRE[Table[StirlingS2[nn, kk], {nn, 0, 20}, {kk, 0, 20}],
 Flatten[Table[s2[n + nn, k + kk], {nn, 0, 1}, {kk, 0, 1}]], {n, k}, 1]
```

Out[33]=
$$\{-s2[n, k] - (1 + k)\, s2[n, 1 + k] + s2[1 + n, 1 + k]\}$$

In[35]:=
```
? Annihilator
```

Annihilator[expr, ops] computes annihilating relations for expr w.r.t. the given operator(s). It returns the Groebner basis of an annihilating ideal (with monomial order DegreeLexicographic). If expr is ∂–finite, the result will be a ∂–finite ideal. If expr is not recognized to be ∂–finite, there is still a chance to find at least some relations (in this case the ideal is not zero–dimensional which is indicated by a warning). Annihilator[expr] automatically determines for which operators relations exist. The relations are computed by executing the ∂–finite closure properties DFinitePlus, DFiniteTimes, and DFiniteSubstitute.

The expression expr can contain hypergeometric expressions, hyperexponential expressions, and algebraic expressions.

Additionally the following functions are recognized: AiryAi, AiryAiPrime, AiryBi, AiryBiPrime, AngerJ, AppellF1, ArcCos, ArcCosh, ArcCot, ArcCoth, ArcCsc, ArcCsch, ArcSec, ArcSech, ArcSin, ArcSinh, ArcTan, ArcTanh, ArithmeticGeometricMean, BellB, BernoulliB, BesselI, BesselJ, BesselK, BesselY, Beta, BetaRegularized, Binomial, CatalanNumber, ChebyshevT, ChebyshevU, Cos, Cosh, CoshIntegral, CosIntegral, EllipticE, EllipticF, EllipticK, EllipticPi, EllipticTheta, EllipticThetaPrime, Erf, Erfc, Erfi, EulerE, Exp, ExpIntegralE, ExpIntegralEi, Factorial, Factorial2, Fibonacci, FresnelC, FresnelS, Gamma, GammaRegularized, GegenbauerC, HankelH1, HankelH2, HarmonicNumber, HermiteH, Hypergeometric0F1, Hypergeometric0F1Regularized, Hypergeometric1F1, Hypergeometric1F1Regularized, Hypergeometric2F1, Hypergeometric2F1Regularized, HypergeometricPFQ, HypergeometricPFQRegularized, HypergeometricU, JacobiP, KelvinBei, KelvinBer, KelvinKei, KelvinKer, LaguerreL, LegendreP, LegendreQ, LerchPhi, Log, LogGamma, LucasL, Multinomial, NevilleThetaC, ParabolicCylinderD, Pochhammer, PolyGamma, PolyLog, qBinomial, QBinomial, qBrackets, qFactorial, QFactorial, qPochhammer, QPochhammer, Root, Sin, Sinc, Sinh, SinhIntegral, SinIntegral, SphericalBesselJ, SphericalBesselY, SphericalHankelH1, SphericalHankelH2, Sqrt, StirlingS1, StirlingS2, StruveH, StruveL, Subfactorial, WeberE, WhittakerM, WhittakerW, Zeta.

If expr contains the commands D and ApplyOreOperator then the closure property DFiniteOreAction is performed: Note the difference between
Annihilator[D[LegendreP[n, x], x], {S[n], Der[x]}] and
expr = D[LegendreP[n, x], x]; Annihilator[expr, {S[n], Der[x]}].

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 is used; if it fails (or if you don't trust it), you can use the option Inhomogeneous –> True, in order to obtain an inhomogeneous recurrence (resp. differential equation).

S[n] ... forward shift in n

In[36]:=
```
ann = Annihilator[Sum[LucasL[3 k] Fibonacci[n - k], {k, 0, n}], S[n]]
```
Out[36]= $\left\{S_n^4 - 5\,S_n^3 + 2\,S_n^2 + 5\,S_n + 1\right\}$

In[39]:=
```
ApplyOreOperator[ann, a[n]]
```
Out[39]= $\{a[n] + 5\,a[1+n] + 2\,a[2+n] - 5\,a[3+n] + a[4+n]\}$

In[40]:=
```
%[[1]] == 0
```
Out[40]= $a[n] + 5\,a[1+n] + 2\,a[2+n] - 5\,a[3+n] + a[4+n] == 0$

In[42]:=
```
Annihilator[LegendreP[n, x], {Der[x]}]
```
Out[42]= $\left\{\left(-1+x^2\right) D_x^2 + 2\,x\,D_x + \left(-n-n^2\right)\right\}$

In[43]:=
```
Annihilator[LegendreP[n, x], {Der[x], S[n]}]
```
Out[43]= $\left\{\left(1-x^2\right) D_x + \left(1+n\right) S_n + (-x - n\,x),\ \left(2+n\right) S_n^2 + \left(-3\,x - 2\,n\,x\right) S_n + (1+n)\right\}$

In[44]:=
```
Sum[LucasL[3 k] Fibonacci[n - k], {k, 0, n}]
```
Out[44]=
$$-\frac{1}{3\sqrt{5}\left(3+\sqrt{5}\right)}2^{2-3n}\left(-1-\sqrt{5}\right)^{-1-3n}\left((-4)^n\left(1+\sqrt{5}\right)^{1+4n} + 64^n\left(5+2\sqrt{5}\right) + \left(i\left(1+\sqrt{5}\right)\right)^{6n}\left(5+2\sqrt{5}\right) - 16^n\left(1+\sqrt{5}\right)^{2n}\left(11+5\sqrt{5}\right)\right)$$

We know that Lucas and Fibonacci numbers both satisfy a recurrence of order 2.
By closure properties, the subsequence L[3n] satisfies a recurrence of order 2.
The Cauchy product of the two sequences corresponds to the multiplication of the generating functions: each has a denominator of degree 2, so the product a denominator of degree at most 4, which translates into a recurrence of order at most 4.
Hence, if we compute the coefficients of a C-finite recurrence of order 4 from the first 25 values of the sum, we know that it holds for all n.

In[45]:=
```
Clear[s];
s[n_] := Sum[LucasL[3 k] Fibonacci[n - k], {k, 0, n}];
```

In[47]:=
```
Solve[Table[Sum[c[i] s[5 j + i], {i, 0, 4}] == 0, {j, 0, 4}]]
```
Out[47]= $\{\{c[1] \to 5\,c[0],\ c[2] \to 2\,c[0],\ c[3] \to -5\,c[0],\ c[4] \to c[0]\}\}$

In[48]:=
```
NullSpace[Table[s[5 j + i], {j, 0, 4}, {i, 0, 4}]]
```
Out[48]= $\{\{1, 5, 2, -5, 1\}\}$

In[49]:=
```
deCatalan = AE2DE[{x y[x]^2 - y[x] + 1 == 0, y[0] == 1}, y[x]]
```
Out[49]= $\left\{-1 - \left(-1+2\,x\right) y[x] - \left(-x+4\,x^2\right) y'[x] == 0,\ y[0] == 1\right\}$

In[50]:=
```
DE2RE[deCatalan, y[x], c[n]]
```
Out[50]= $\left\{2\,(1+n)\left(1+2\,n\right) c[n] - (1+n)\left(2+n\right) c[1+n] == 0,\ c[0] == 1\right\}$

In[51]:=
```
Annihilator[CatalanNumber[n], S[n]]
```

Out[51]=
$$\left\{\left(2+n\right)\,S_n+\left(-2-4\,n\right)\right\}$$