MathTools

Math routines for the TI-89/92+/V200

Linear Algebra, Polynomials, Calculus, Statistics, Special Functions, Vector Analysis, and more

Version: 2.4.3

Author: Bhuvanesh Bhatt (bbhatt1@towson.edu)

Introduction

A Computer Algebra System (CAS) is a kind of mathematical software that enables you to use a computer to manipulate mathematical expressions. A CAS has symbolics, numerics, graphics, programming, and typesetting capabilities, and usually also data import/export. CAS's are used in many fields such as mathematics, physics, chemistry, engineering, computer science, computational biology, economics, and education. The TI-68k devices (TI-89, TI-92 Plus, and Voyage 200) are the most advanced calculators available today. Once you have learned a topic, you can use the TI-68k CAS to automate some or all of your work, so that you can concentrate on more advanced and/or abstract topics without worrying about calculational errors. The TI-68k are also invaluable when learning a topic, since they allow you to explore the topic in many ways.

One problem with the TI-68k is that a lot of commonly used functions are not built-in. MathTools aims to bridge that gap, mainly in the area of mathematics. One of my goals in maintaining MathTools and its extensive documentation is to introduce people to the beauty of mathematics. MathTools mainly consists of TI-Basic functions, but there are also TI-Basic programs, math programs and utilities written in C, and a Flash application.

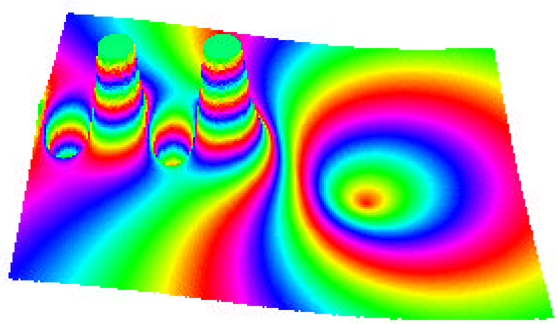
MathTools is far from complete. Some categories that need more work are factorization, Gröbner bases, GMP, differential equations, and logic. Some of these have performance problems, while others have missing functionality. However, MathTools has good organization and is quite powerful, with well over 340 routines.

Installation

- Without GMP and the C utilities, MathTools takes up approximately 230 KB space.
- I recommend using TI Connect 1.3. Extract the appropriate group file (.89g or .9xg), right-click on it, and in the context menu click Send To → Connected TI Device → Archive. If you get a garbage collection message on the calculator, reply "OK" to it, and when it is done, click "Retry" in TI Connect to send the rest of MathTools.
- You can choose the functions you want to install, as long as you also install all subroutines used by those functions. The dependencies are not given recursively in this documentation, for example, the Eigenval help entry shows that it needs mZeros and Sort, but Sort also needs ListSwap. Everything except the Flash app should be installed in the `mathtool` folder.
- To install the MathTools Flash application, which requires AMS 2.04 or higher:
  - If you are using TI Connect, connect the cable to the calculator, right click on the Flash app file and from the context menu select Send To → Connected TI Device → Archive.
  - If you are using the Graph-Link software, connect the cable to your calculator, click on Link → Send Flash Software → Applications and Certificates, select the Flash app in the dialog, click on Add, then click on OK.

Functionality

- Linear Algebra
  - Eigenvalues/Eigenvectors, Decompositions, and PseudoInverse
  - Special Matrices
  - Other functions
- Polynomials
  - Coefficients and terms
  - Polynomial division[†]
  - Resultants
  - Factorization
  - Gröbner bases
  - Other functions
- Miscellaneous
  - Arbitrary-precision arithmetic (GMP)
  - Calculus
  - Combinatorics
  - Continued Fractions
  - Data Analysis
  - Differential and Recurrence Equations
  - Equation/Inequality Solving
  - Lists
  - Logic
  - Number theory
  - Optimization
  - Pattern-matching
  - Special Functions
  - Statistics
  - Strings
  - Summation and Series
  - Vector/Tensor Analysis
  - Other functions
  - C Utilities



Links to other sections

Known issues

Thanks

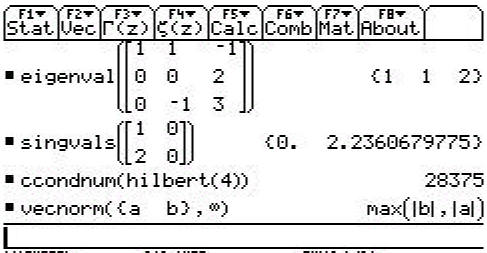
History

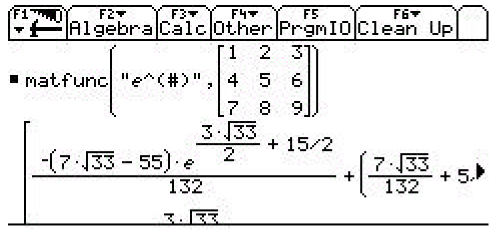
Future plans

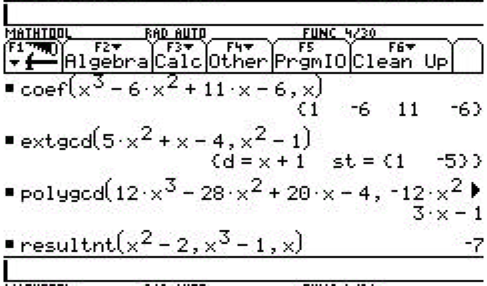
Computer Algebra links

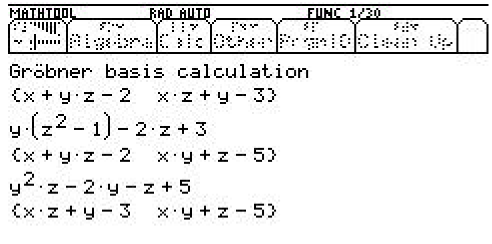
[†] Including polynomial GCD and LCM

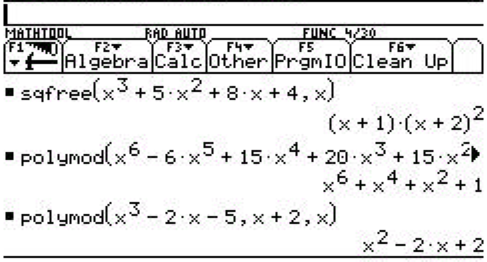
Screenshots

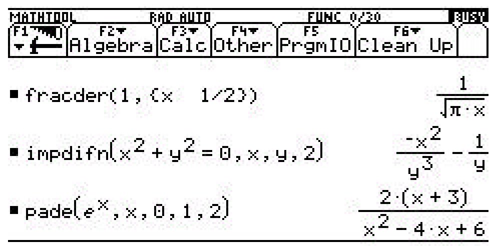


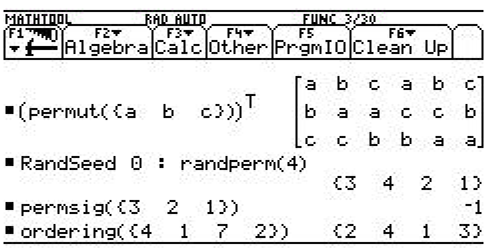


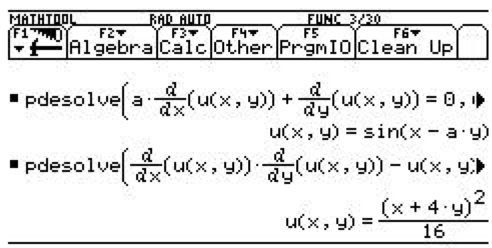


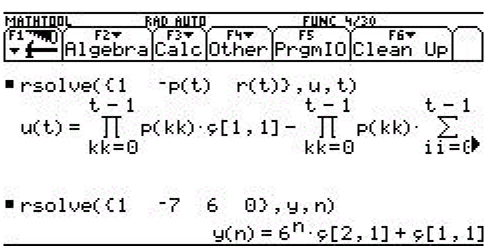


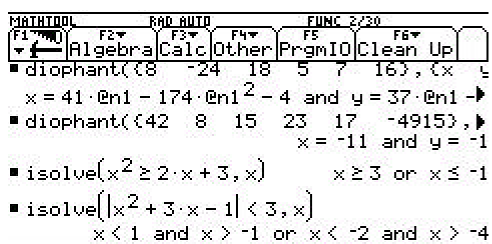


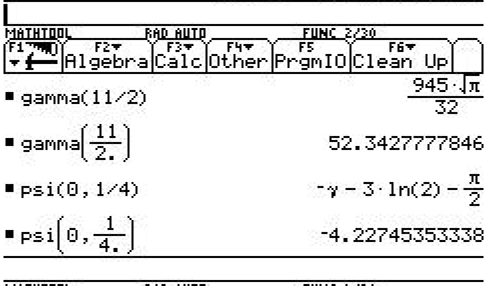


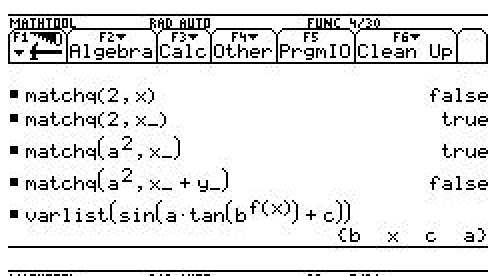












MathTools routines, with dependencies, examples, and notes

- **Linear algebra**
  - Eigenvalues/Eigenvectors, Decompositions, and PseudoInverse
    - Cholesky(mat) computes the Cholesky decomposition of a square symmetric positive-definite matrix mat and returns a lower-triangular matrix L such that mat=L·L^T
      Example: Cholesky(Hilbert(3)) ⇒ [[1,0,0][1/2,√(3)/6,0][1/3,√(3)/6,√(5)/30]]
      Notes: Cholesky returns a lower-triangular matrix instead of an upper-triangular one, because trying to transpose the lower-triangular matrix results in conj() being applied to all symbolic elements, which looks ugly. (PowerExp can be used to remove most but not all of the conj()'s.) Complex elements are not allowed, since a symmetric positive-definite matrix must be real.
    - Eigenval(matrix) returns the eigenvalues of matrix, including repeated ones
      Needs: mZeros, Sort
      Example: Eigenval([[1,1,-1][0,0,2][0,-1,3]]) ⇒ {1,1,2}
      Note: Eigenval tries to return exact eigenvalues whenever possible, but it calls the built-in eigVl() function for approximate matrices. Non-symmetric matrices can have eigenvalues with nonzero imaginary parts.
    - Eigenvec(matrix) returns the generalized eigenvectors of matrix
      Needs: DelElem, Eigenval, list2eqn, MemberQ, NullVecs
      Example: Eigenvec([[1,1,-1][0,0,2][0,-1,3]]) ⇒ [[1,@1,0][0,2,1][0,1,1]]
    - GenEigVl(mat1,mat2) returns the generalized eigenvalues of the matrix pencil mat1–λ·mat2
      Needs: Coef, Cubic, Degree, NullVecs
      Examples: GenEigVl([[3,6][1,-4]],[[-10,6][0,10]]) ⇒
      {-8/25–√(194)/50·***i***, -8/25+√(194)/50·***i***},
      GenEigVl([[1,0,0,0][0,2,0,0][0,0,3,0][0,0,0,4]],
      [[1,0,0,0][0,1,0,0][0,0,0,0][0,0,0,0]]) ⇒ {∞, ∞, 1, 2},
      GenEigVl(diag(seq(i,i,1,3)), identity(3)) ⇒ {1, 3, 2},
      GenEigVl([[0,0][0,0]],[[0,0][0,0]]) ⇒ {undef, undef},
      GenEigVl([[0,0][0,1]],[[0,0][0,0]]) ⇒
      "Error: Singular intermediate matrix P = [[1,0][0,0]]"
      Notes: Generalized eigenvalues can be infinite. The generalized eigenvalue problem is called regular if the determinant det(mat1–λ·mat2) does not vanish identically for all values of λ, otherwise it is called singular. There are n generalized eigenvalues iff the matrix rank of mat2 is n. Application areas for generalized eigenproblems include optimization, statistics, signal processing, control theory, spectral imaging, and solid mechanics.
    - GSchmidt(mat) performs Gram-Schmidt orthonormalization on mat
      Example: GSchmidt([[1,2][3,4]]) ⇒ [[√(10)/10,3·√(10)/10]
      [3·√(10)/10,-√(10)/10]]
      Note: GSchmidt returns a matrix with orthogonal columns that span the same subspace as the columns of mat, provided the columns of mat are linearly independent. Gram-Schmidt orthonormalization is known to be numerically unstable (see "Matrix Computations" by Golub and van Loan), but should be fine for symbolic manipulation.

- Jordan(matrix) returns the Jordan decomposition {J,S} of matrix such that $J=S^{-1}\cdot matrix\cdot S$
  Needs: Eigenvec
  Example: Jordan([[1,2][3,4]]) ⇒ {j=[[5/2–√(33)/2,0][0,√(33)/2+5/2]],
  s=[[-(√(33)+3)/6,(√(33)–3)/6][1,1]]}
  Note: Jordan decomposition is often used for evaluating symbolic functions of matrices, such as matrix exponentials.
- PInverse(mat,tol) returns the pseudoinverse (also called the generalized or Moore-Penrose inverse) of the matrix mat with tolerance tol
  Needs: SVD
  Example: PInverse([[1,2][3,4][5,6]],1E-10) ⇒ [[-1.33,-0.333,0.667]
  [1.08,0.333,-0.417]]
  (to Float-3 precision)
  Note: mat can be non-square. The pseudoinverse equals the inverse when mat is a non-singular (all singular values > 0) square matrix. The sum of squares of the elements of mat·PInverse(mat)–I is minimized, where I is the identity matrix, and so the pseudoinverse is useful for data fitting.
- SingVals(mat) returns the sorted singular values of the numeric matrix mat
  Needs: IsCmplxN, Sort
  Examples: SingVals([[***i***,7][-2,1–***i***]]) ⇒ {2.09236143579,7.18484680575},
  SingVals([[1,0][2,0]]) ⇒ {0.,2.2360679775}
- SVD(mat) computes the singular value decomposition of matrix mat
  Needs: Eigenval, Eigenvec, GSchmidt, Reverse, Sort
  Example: SVD([[1,2][3,4][5,6.0]]) ⇒ {uu=[[-.883,-.23][-.241,-.525][.402,-.82]],
  ΣΣ=[[.514,0][0,9.53][0,0]], vv=[[.785,-.62][-.62,-.785]]}
  (with Float-3 precision)
  Note: SVD returns a list containing three matrices, U, Σ, and V, where U·Σ·V is the decomposition. If mat is an exact numeric matrix or a symbolic matrix, SVD tries to return an exact singular value decomposition. Putting a decimal point into a numeric matrix usually speeds up the calculation. The input matrix can be non-square. The Σ matrix contains the diagonalized singular values of mat. Singular values with absolute value less than a default tolerance of 5E-13 will be considered to be zero. If you store a value such as 1E-10 to the variable 'tol', then that will be used as the tolerance instead.

- Special Matrices
  - Circulnt(col) returns the circulant matrix defined by the column col
    Example: Circulnt({1,2,3,4}) ⇒ [[1,4,3,2][2,1,4,3][3,2,1,4][4,3,2,1]]
    Note: A circulant matrix is a special case of the Toeplitz matrix.
  - Frank(n) returns the Frank matrix of order n
    Example: Frank(4) ⇒ [[4,3,2,1][3,3,2,1][0,2,2,1][0,0,1,1]]
  - Hadamard(n) returns a Hadamard matrix of order n
    Needs: KProduct
    Example: Hadamard(4) ⇒ [[1,1,1,1][1,-1,1,-1][1,1,-1,-1][1,-1,-1,1]]
    Notes: Hadamard matrices exist only for n = 2 or n a multiple of 4. For an $n\times n$ Hadamard matrix H, $H^T\cdot H = n\cdot I$, where I is the identity matrix, and $\det(H) = n^{n/2}$. Hadamard matrices are used in various combinatorics problems such as the Reed-Muller error-correcting codes.

- Hankel(list) returns the corresponding Hankel matrix
  Example: Hankel({1,1,1,6,21}) ⇒ [[1,1,1][1,1,6][1,6,21]]
  Note: Hankel matrices are symmetric.
- Hilbert(n) returns the Hilbert matrix of order n
  Example: Hilbert(3) ⇒ [[1,1/2,1/3][1/2,1/3,1/4][1/3,1/4,1/5]]
  Note: The inverse of a Hilbert matrix has integer elements. A Hilbert matrix is a special case of the Hankel matrix.
- IHilbert(n) returns the inverse of the order-n Hilbert matrix
  Needs: Hilbert
  Example: IHilbert(3) ⇒ [[9,-36,30][-36,192,-180][30,-180,180]]
  Note: IHilbert(n) is usually faster than Hilbert(n)$^{-1}$. For example, Hilbert(10)$^{-1}$ takes about 17.2 seconds while IHilbert(10) takes about 4.0 seconds.
- Pascal(n) returns the Pascal S-matrix of order n
  Example: Pascal(4) ⇒ [[1,1,1,1][1,2,3,4][1,3,6,10][1,4,10,20]]
- Toeplitz(row,col) returns the corresponding Toeplitz matrix
  Needs: Reverse
  Example: Toeplitz({a,b,c},{a,d,e}) ⇒ [[a,b,c][d,a,b][e,d,a]]
  Note: row[1] must equal col[1].
- Vanderm(list) returns the corresponding Vandermonde matrix
  Example: Vanderm({v1,v2,v3}) ⇒ [[1,1,1][v1,v2,v3][v1^2,v2^2,v3^2]]
  Note: Vandermonde matrices appear in association with fitting polynomials to data. In particular, you can create a Vandermonde matrix from the x-values, augment the y-values to the bottom, perform an rref of the transpose, and obtain an interpolating polynomial by doing the sum x^{k-1}·mat[k,colDim(mat)], k=1, …, rowDim(mat), where mat is the row-reduced matrix.

- Other functions
  - cCondNum(matrix) returns the column-norm condition number of matrix
    Example: cCondNum(Hilbert(4)) ⇒ 28375
    Note: Condition numbers can be useful for studying the propagation of errors. Higher values for the condition number indicate that the matrix is close to singular, which means that approximate numerical operations like inversion or linear solution will return inaccurate or unreliable results. The base-b logarithm of the matrix condition number is a worst-case estimate of how many base-b digits are lost in solving a linear system with that matrix (see MathWorld entry).
  - Cofactor(mat) returns the matrix of cofactors of mat
    Needs: Minor
    Example: Cofactor([[a,b][c,d]]) ⇒ [[d,-c][-b,a]]
    Notes: The adjoint of the matrix is simply the transpose of the matrix of cofactors. The determinant of the matrix $mat_{i\,j}$ can be computed from its matrix of cofactors $C_{i\,j}$ by multiplying one of the rows or columns of mat by the corresponding row or column of C and summing the elements.
  - CompMat(poly,x) returns the companion matrix of poly
    Needs: Degree, Reverse
    Example: CompMat(x^2+1,x) ⇒ [[0,-1][1,0]], eigVl([[0,-1][1,0]]) ⇒ {***i***, -***i***}
    Note: The eigenvalues of the companion matrix are the roots of poly, as demonstrated in the example above.
  - CondNum(matrix) returns the Turing condition number estimate
    Example: CondNum(Hilbert(4)) ⇒ 25920
    Note: Condition numbers can be useful for studying the propagation of errors.

Higher values for the condition number indicate that the matrix is close to singular, which means that approximate numerical operations like inversion or linear solution will return inaccurate or unreliable results. The base-b logarithm of the matrix condition number is a worst-case estimate of how many base-b digits are lost in solving a linear system with that matrix (see MathWorld entry).

- FastDet(mat) returns the determinant of the matrix mat
  Example: det([[a,b,c][d,sin(e),f][g,√(h),i]]) ⇒ a·(sin(e)·i–f·√(h))–b·(d·i–f·g)+ c·(d·√(h)-sin(e)·g) [takes about 5.5 seconds],
  FastDet([[a,b,c][d,sin(e),f][g,√(h),i]]) ⇒ a·(sin(e)·i–f·√(h))– b·(d·i–f·g)+c·(d·√(h)-sin(e)·g) [takes about 0.6 seconds]
  Note: FastDet is usually faster than the built-in det() function, especially for symbolic and exact numeric matrices. The determinant of an orthogonal matrix is ±1. One can compute the absolute value of the determinant of a matrix M from its QR factorization as product(diag(R)), and from its SVD as product(SingVals(M)).
- Givens(matrix,i,j) does a Givens rotation to zero out the [i,j] element of matrix
  Example: Givens([[1][1][1][1]],2,1) ⇒ [[√(2)][0][1][1]] (zero out 2nd element),
  Givens([[√(2)][0][1][1]],3,1) ⇒ [[√(3)][0][0][1]] (zero out 3rd elem),
  Givens([[√(3)][0][0][1]],4,1) ⇒ [[2][0][0][0]] (zero out 4th element).
  Note: A Givens rotation is similar to a Householder reflection, but a Givens rotation is useful when you want to zero out a particular element (or small set of elements), whereas a Householder reflection is more effective when you want to zero out many elements. Givens rotations are skew-symmetric and orthogonal.
- KProduct(a,b) returns the Kronecker/matrix/tensor product of matrices a and b
  Example: KProduct([[a11,a12][a21,a22]],[[b11,b12]]) ⇒
  [[a11·b11,a11·b12][a12·b11,a12·b12][a21·b11,a21·b12][a22·b11,a22·b12]]
  Note: KProduct can be used to create Dirac matrices from quantum field theory. The Kronecker product is distributive with respect to addition as in $(A+B)\otimes C = A\otimes C+B\otimes C$, and is associative as in $A\otimes(B\otimes C) = (A\otimes B)\otimes C$. For the transpose and inverse of a Kronecker product, $(A\otimes B)^T = A^T\otimes B^T$ and $(A\otimes B)^{-1} = A^{-1}\otimes B^{-1}$. For the determinant and trace, if A is n×n and B is m×m, $\det(A\otimes B) = \det(A)^m\cdot\det(B)^n$, and $\mathrm{trace}(A\otimes B) = \mathrm{trace}(A)\cdot\mathrm{trace}(B)$.
- KSum(a,b) returns the Kronecker sum of square matrices a and b
  Needs: KProduct
  Example: KSum([[a,b][c,d]],[[e,f][g,h]]) ⇒ [[a+e,f,b,0][g,a+h,0,b] [c,0,d+e,f][0,c,g,d+h]]
- MatFunc(str,mat) applies the function given in the string str to the matrix mat
  Needs: Jordan
  Example: MatFunc("*e*^#",[[1,2][3,4]]) ⇒ [[(1/2–√(33)/22)·*e*^(…)+…][…]]
  Notes: Use "#" in the string to denote the input matrix. MatFunc can handle functions of matrices (*e*^A, sin(A), √(A), etc.) on a diagonalizable square matrix.
- MatRank(mat) returns the matrix rank of mat
  Needs: NullVecs
  Examples: MatRank([a,b;c,d;e,f]) ⇒ 2, MatRank([[a,0,b][c,d,0][0,0,e]]) ⇒ 3,
  MatRank([[a,0,b][c,d,0][0,0,0]]) ⇒ 2
- Minor(mat) returns the matrix of minors for mat
  Needs: DelCol, DelRow
  Example: Minor([[a,b][c,d]]) ⇒ [[d,c][b,a]]

- mMinPoly(mat,var) returns the minimal polynomial of mat in terms of var
  Needs: NullVecs
  Examples: mMinPoly([[0,0][0,0]],x) ⇒ x, mMinPoly([[0,1][0,0]],x) ⇒ x^2,
  mMinPoly([[1,0][0,1]],x) ⇒ x–1
  Note: The minimal polynomial of a matrix M is the lowest degree polynomial such that $\sum_{i=0}^{n} c_i \cdot M^i = 0$, where n is the polynomial degree. It divides the characteristic polynomial of M.
- NullVecs(mat) returns a matrix whose columns form a basis for the nullspace of mat
  Examples: NullVecs([[4,2,-4][2,1,-2][-4,-2,4]]) ⇒ [[-1/2,1][1,0][0,1]],
  NullVecs({{1,1},{1,0}}) ⇒ "nullspace={0}"
  Note: As shown in the second example above, the null space of a nonsingular matrix mat is trivial, so that there is no nonzero vector v such that mat·v = 0.
- SparsRnd({m,n},nz) returns an m×n random sparse array with nz nonzero values
  Needs: ToSparse
  Example: RandSeed 0:SparsRnd({2,3},2) ⇒
  sparse(2,{2,3},{0,1,2},{2,3},{0.733812311248,0.405809641769},0)
- ToDense(arr) converts a sparse matrix arr to a matrix in the usual form
  Examples: ToDense(ToSparse([[4,-2][0,-7][8,0]],a)) ⇒ [[4,-2][0,-7][8,0]],
  ToDense(sparse(4,{3,2},{0,2,3,4},{1,2,2,1},{4,-2,-7,8},a))
  ⇒ [[4,-2][a,-7][8,a]]
- ToSparse(mat,pat) converts a matrix in the usual form to a sparse matrix with default element pat
  Needs: MatchQ
  Example: ToSparse([[4,-2][0,-7][8,0]],0) ⇒
  sparse(4,{3,2},{0,2,3,4},{1,2,2,1},{4,-2,-7,8},0)
  Note: The format used is the CSR (compressed sparse row) format.
- VecNorm(vec,p) returns the p-norm of the vector vec
  Examples: VecNorm({a,b},2) ⇒ $\sqrt{(a^2+b^2)}$,
  VecNorm({a,b},3) ⇒ ($|a^3|+|b^3|$)^(1/3),
  VecNorm({a,b},∞) ⇒ max(|b|,|a|)

- **Polynomial algebra**
  - Coefficients and terms
    - Coef(poly,x) returns the coefficients of poly in the variable x
      Example: Coef(x^3–6·x^2+11·x–6,x) ⇒ {1,-6,11,-6}
      Note: The exponents must be numeric (for example, x^2 is valid but x^a is not).
    - CoefList(poly,x) is an alternative function for returning the coefficients of poly
      Needs: FreeQ, IsCmplxN, Pad, PolyDeg
      Examples: CoefList((a+b·***i***)·x^3+(c+d·***i***)·x+(e+f·***i***),x) ⇒ {e+f·***i***,c+d·***i***,0,a+b·***i***}
      Note: This function is experimental and slow, and may return incorrect results.
    - Degree(poly,x) returns the degree of poly in x
      Needs: Coef
      Example: Degree(x^2–x+1,x) ⇒ 2
    - LeadCoef(poly,x) returns the leading coefficient of poly in x
      Needs: Coef
      Example: LeadCoef((x+x)(x+2·x^2)(x+3·x^3),x) ⇒ 12

- LeadTerm(poly) returns the leading term in poly
  Example: LeadTerm($4\cdot x\cdot y\cdot z^2+4\cdot x^3-5\cdot y^4+7\cdot x\cdot y^2\cdot z$) ⇒ $4\cdot x^3$
- NthCoeff(poly,x,n) returns the coefficient of x^n in poly
  Examples: seq(NthCoeff(x^3-x+1,x,i),i,4,0,-1) ⇒ {0,1,0,-1,1},
  seq(NthCoeff($-@n1^2+\sqrt{(a)}\cdot @n1+e^b$,@n1,i),i,3,0,-1) ⇒ {0,-1,√(a),e^b},
  NthCoeff($-3\cdot x^{100}+2\cdot x^4-7$,x,100) ⇒ -3
  Note: NthCoeff can handle polynomials with degrees upto 449.
- PolyDeg(poly,x) is an alternative function that returns the degree of poly in x
  Needs: MatchQ
  Example: PolyDeg($-5\cdot x^4-x^2+4\cdot x-5$,x) ⇒ 4
- sFactors(xpr) returns a list of all symbolic factors in xpr
  Needs: nChars
  Example: sFactors((x+1)·(x–2)) ⇒ {x–2, x+1}
  Note: sFactors is very similar to Terms, except that it deals with multiplication instead of addition. If xpr is provided as a string, the factors are found without any evaluation.
- Terms(xpr) returns a list of terms in xpr
  Needs: nChars
  Example: Terms($\cos(a+b)\cdot x^2+\boldsymbol{e}^{2\cdot c}\cdot x$) ⇒ {$\cos(a+b)\cdot x^2$, $(\boldsymbol{e}^c)^2\cdot x$}
  Note: If xpr is provided as a string, the terms are found without any evaluation.

◆ Polynomial division

- ExtGCD(poly1,poly2) returns polynomials {d,{s,t}} such that
  d = s·poly1+t·poly2 = GCD(poly1,poly2)
  Needs: IsCmplxN, Quotient, VarList
  Example: ExtGCD($5\cdot x^2+x-4$, x^2-1) ⇒ {d = x+1, st = {1, -5}}
  Note: ExtGCD also works for integers. The extended gcd is useful for solving linear Diophantine equations.
- MPolyDiv(poly1,poly2,x) returns polynomial quotient and remainder for multivariate polynomials
  Needs: LeadTerm, Remaindr
  Example: MPolyDiv(x^2-y,x+y,{x,y}) ⇒ {x–y,y^2-y}
- PolyDiv(poly1,poly2,x) returns the univariate polynomial quotient and remainder
  Needs: Coef, Degree, LeadCoef
  Example: PolyDiv($3\cdot x^4+2\cdot x^3+x+5$,$x^2+2\cdot x+3$,x) ⇒ {$3\cdot x^2-4\cdot x-1$,$15\cdot x+8$}
- PolyGCD(poly1,poly2,x) returns the polynomial GCD of poly1 and poly2
  Needs: Coef, PrimPoly, Remaindr
  Example: PolyGCD($12\cdot x^3-28\cdot x^2+20\cdot x-4$,$-12\cdot x^2+10\cdot x-2$,x) ⇒ $3\cdot x-1$
  Note: Intermediate expression swell is encountered in polynomial gcd computation, especially with the classical Euclidean algorithm.
- PolyLCM(poly1,poly2,x) returns the polynomial LCM of poly1 and poly2
  Needs: PolyGCD
  Example: PolyLCM($(1-x)^2\cdot(1+x)^2$,(1–x)·(2+x),x) ⇒ $(x-1)^2\cdot(x+1)^2\cdot(x+2)$
- Quotient(poly1,poly2,x) returns PolyDiv[1] (polynomial quotient)
  Needs: PolyDiv
  Example: Quotient($3\cdot x^4+2\cdot x^3+x+5$,$x^2+2\cdot x+3$,x) ⇒ $3\cdot x^2-4\cdot x-1$
- Remaindr(poly1,poly2,x) returns PolyDiv[2] (polynomial remainder)
  Needs: PolyDiv
  Example: Remaindr($3\cdot x^4+2\cdot x^3+x+5$,$x^2+2\cdot x+3$,x) ⇒ $15\cdot x+8$

- Resultants
  - Resultnt(poly1,poly2,x) returns the polynomial resultant
    Needs: Sylvestr
    Example: Resultnt(x^2-2,x^3-1,x) ⇒ -7
  - Subres(poly1,poly2,x) returns a list of polynomial subresultants
    Needs: Coef
    Example: Subres($2\cdot x^4+x^2-4,3\cdot x^2+2$,x) ⇒ {1156,102}
  - Sylvestr(poly1,poly2,x) returns the Sylvester matrix
    Needs: Coef
    Example: Sylvestr($x^2-3\cdot x,x^3-1$,x)
    ⇒ [[1,0,0,1,0][-3,1,0,0,1][0,-3,1,0,0][0,0,-3,-1,0][0,0,0,0,-1]]
    Note: Sylvestr cannot handle constant (degree 0) polynomials.
- Factorization
  - BerlQMat(poly,x,m) returns the Berlekamp Q-matrix modulo m
    Needs: Coef, Reverse, sMod
    Example: BerlQMat($x^2+3\cdot x$,x,2) ⇒ [[1,0,0][0,1,0][0,1,0]]
    Note: Finding the null-space of the Q-matrix modulo a prime is a first step in the Berlekamp factorization algorithm.
  - SqFree(poly,x) finds the squarefree factorization for a primitive polynomial poly
    Needs: PolyGCD
    Example: SqFree($x^3+5\cdot x^2+8\cdot x+4$,x) ⇒ $(x+1)\cdot(x+2)^2$
    Note: Squarefree factorization is a first step in many factoring algorithms. There is a Flash app extension function `MathTool.SqrFree` that does the same thing as SqFree, but faster.
    Note: SqFree finds the derivative of poly and then iteratively computes GCDs.
  - SqFreeQ(poly,x) returns true if poly is squarefree, else false
    Needs: PolyGCD
    Examples: SqFreeQ($x^5+x^3+2\cdot x^2+x+2$,x) ⇒ true
    SqFreeQ($x^8-2\cdot x^6+2\cdot x^2-1$,x) ⇒ false
- Gröbner bases
  - Groebner(polylist,vars) computes a symbolic Gröbner basis for the set of multivariate polynomials in polylist
    Needs: ListPair, NormalF, Spoly
    Example: Groebner({$x-y-z,x+y-z^2,x^2+y^2-1$},{x,y,z}) [will take ≈30 minutes!]
    Notes: Groebner is a program, unlike most other components of this package. It is an experimental program and is usually very slow. Groebner computes the (unreduced) Gröbner basis via the Buchberger algorithm. At each step, it shows the combination of polynomials selected ({ p_i , p_j }) and the normal form of Spoly(p_i , p_j). Computing a symbolic Gröbner basis with a lexicographic ordering can be computationally intensive. When Groebner ends, it displays the Gröbner basis. Gröbner bases can be used to solve systems of polynomial equations, and are also useful for other applications such as integer programming and converting parametric curves and surfaces to implicit form. The reduced Gröbner basis has the same set of roots as the original polynomials. If the Gröbner basis reduces to 1, the nonlinear system polylist has no solution.
  - NormalF(poly,polylist,vars) returns the normal form of poly modulo polylist
    Needs: LeadTerm, MPolyDiv, Terms
    Example: NormalF($x+y-z^2,x-y-z$,{x,y,z}) ⇒ $2\cdot y-z^2+z$

Note: The “normal form” is guaranteed to be unique only if the polynomials in polylist form a Gröbner basis.

- Spoly(poly1,poly2,vars) returns the S-polynomial of poly1 and poly2
  Needs: LeadTerm, Multideg
  Example: Spoly(x^3–2·x·y,x^2·y–2·y^2+x,{x,y}) ⇒ -x^2

- Other functions
  - CompMat(poly,x) returns the companion matrix of poly
    Needs: Degree, Reverse
    Example: CompMat(x^2+1,x) ⇒ [[0,-1][1,0]], eigVl([[0,-1][1,0]]) ⇒ {***i***, -***i***}
    Note: The eigenvalues of the companion matrix are the roots of poly, as demonstrated in the example above.
  - Content(poly,x) returns the “content” of poly
    Needs: Coef, LGcd
    Example: Content(8·x^4–2·x^2–2·x+12,x) ⇒ 2
    Note: The “content” of a polynomial is the gcd of its coefficients.
  - Cyclotom(n,x) returns the n^{th} cyclotomic polynomial
    Needs: RelPrime
    Example: Cyclotom(6,x) ⇒ x^2–x+1
    Note: The n^{th} cyclotomic polynomial is the minimal polynomial of the primitive n^{th} root of unity and is irreducible over $\mathbb{Z}$ with degree φ(n), where φ(n) is the Euler totient function.
  - mMinPoly(mat,var) returns the minimal polynomial of mat in terms of var
    Needs: NullVecs
    Examples: mMinPoly([[0,0][0,0]],x) ⇒ x, mMinPoly([[0,1][0,0]],x) ⇒ x^2,
    mMinPoly([[1,0][0,1]],x) ⇒ x–1
    Note: The minimal polynomial of a matrix M is the lowest degree polynomial such that $\sum_{i=0}^{n} c_i \cdot M^i = 0$, where n is the polynomial degree. It divides the characteristic polynomial of M.
  - Multideg(poly,vars) returns the multidegree of poly
    Needs: Degree, LeadTerm
    Example: Multideg(4·x·y·z^2+4·x^3–5·y^4+7·x·y^2·z,{x,y,z}) ⇒ {3,0,0}
  - Normal(poly,x) returns the normal form of poly
    Needs: LeadCoef
    Example: expand(Normal(8·x^4–2·x^2–2·x+12,x)) ⇒ x^4–x^2/4–x/4+3/2
    Note: This is not the same normal form as that given by NormalF.
  - PolyMod(poly,m,x) returns poly mod m
    Needs: Coef, IsCmplxN, Quotient
    Example: PolyMod(x^6+6·x^5+15·x^4+20·x^3+15·x^2+6·x+1,2,x) ⇒ x^6+x^4+x^2+1
  - PolyQ(xpr,x) returns true if xpr is a polynomial, else false
    Example: PolyQ($(x+1)^2$, x) ⇒ true
    Note: If you have variables raised to unknown powers, PolyQ will return false. For example, PolyQ(x^2·y^b,x) ⇒ false
  - PrimPoly(poly,x) returns the corresponding primitive polynomial
    Needs: Content
    Example: PrimPoly(6·x^3+26·x^2+26·x+6,x) ⇒ 3·x^3+13·x^2+13·x+3

- SymPoly(list,k) returns the k^{th} symmetric polynomial of the variables in list
  Needs: KSubsets
  Example: seq(SymPoly({x,y,z},i),i,1,3) ⇒ {x+y+z, x·y+x·z+y·z, x·y·z}

- **Miscellaneous functions**
  - Arbitrary-precision arithmetic (GMP)
    - GMPFMul(numstr1,numstr2) does arbitrary-precision multiplication of the floating point real numbers given in numstr1 and numstr2 and returns the result as a list {string,intdigits}
      Needs: GMPMulDt
      Example: GMPFMul("32.868852111489014254382668691",
      "4.882915734918426996824267") ⇒
      {"160495835163896471004691233980227653746755621409924496999999999
      99999999999999999999999999999999999999999999999999999999999999999999999
      99999999999999999999999999999998",3}, which means 160.4958…
      Note: GMPFMul is experimental and currently has a small memory leak.
  - Calculus
    - CPVInt(f(x),x,a,b) returns the Cauchy principal value of the integral ∫(f(x),x,a,b)
      Examples: CPVInt(1/x,x,-1,2) ⇒ ln(2), CPVInt(1/(x+x^2),x,-2,1) ⇒ -2·ln(2),
      CPVInt(tan(x),x,π/4,3·π/4) ⇒ 0,
      CPVInt(x/(x^3–3),x,0,∞) ⇒ 0.302299894039
      Note: CPVInt may not be able to handle some improper integrals (i.e. integrals with one or both integration limits infinite) because it does not compute residues.
    - ExactDif(flist,vars) returns true if flist is an exact differential, else returns false or the conditions that must be satisfied
      Needs: FuncEval, list2eqn, Permut
      Examples: ExactDif({f(x),g(y),h(z)},{x,y,z}) ⇒ true,
      ExactDif({f(x,y),g(y),h(z)},{x,y,z}) ⇒ $\frac{df(x,y)}{dy} = 0$
    - FracDer(xpr,{x,ordr[,a]}) returns the fractional derivative (from Riemann-Liouville calculus) of xpr with respect to variable x to order ordr
      Needs: FreeQ, Gamma, IsCmplxN, MatchQ
      Examples: FracDer(1,{x,-1/2}) ⇒ 2·√(x)/√(π),
      FracDer(*d*(f(x),x),{x,-2/3}) ⇒ 0.664639300459·$x^{5/3}$
      Notes: The default for a is zero (a is what is called the ‘lower limit’). The fractional derivative of a constant is not zero, as the first example shows. Because the gamma function cannot be evaluated symbolically for many fractions, you may get a numeric result, as shown in the second example. Weyl calculus is a subset of Riemann-Liouville calculus.
    - GetExpLn(xpr) is a subroutine for Risch that returns exponential or logarithmic subexpressions
      Needs: mand
    - Horowitz(numerator,denominator,var) reduces an integral using the Horowitz method
      Needs: Coef, Degree, list2eqn, Pad, PolyDiv, PolyGCD, Quotient, RTInt
      Example: Horowitz(x,x^4–1,x) ⇒ ln(x^2–1)/4+(-ln(x^2+1)/4)–π/4·***i***
      Note: This is an experimental function.
    - ImpDifN(eq,x,y,n) returns the n^{th} implicit derivative of the equation eq
      Example: ImpDifN(x^4+y^4=0,x,y,2) ⇒ -3·x^6/y^7–3·x^2/y^3

- mTaylor(f(x,y,…),{x,y,…},{x0,y0,…},k) returns the multivariate Taylor expansion of f(x,y,…) to order k
  Needs: list2eqn
  Examples: mTaylor($x^2+x\cdot y^2$, x,x0,5) ⇒ $(x-x0)^2 + (x-x0)\cdot(y^2+2\cdot x0) + x0\cdot y^2 + x0^2$
  mTaylor(x^2+y^2, {x,y},{x0,y0},2)
  ⇒ $x0\cdot x^2-2\cdot x0\cdot(x0-1)\cdot x+y0\cdot y^2-2\cdot y0\cdot(y0-1)\cdot y+x0^3-x0^2+y0^2\cdot(y0-1)$
- nIntSub(f,x,a,b,n) numerically integrates f from x=a to x=b by subdividing the interval into n subintervals
  Example: nIntSub(tan(x),x,0,1.5707,5) ⇒ 9.24776403537
- Pade(f,x,x_0,n,d) returns the Padé approximant of f(x) about x=x_0 with degrees n for the numerator and d for the denominator
  Examples: Pade(e^x,x,0,1,2) ⇒ $2\cdot(x+3)/(x^2-4\cdot x+6)$, Pade($x^x$,x,1,0,2) ⇒ -1/(x–2)
  Note: Padé approximants are a kind of generalization of Taylor series and are a good way of approximating functions that have poles.
- Risch(xpr,var) does indefinite integration
  Needs: Coef, Degree, FreeQ, GetExpLn, list2eqn, mand, PolyGCD, Resultnt, Terms, VarList
  Examples: Risch(ln(x),x) ⇒ (ln(x)–1)·x,
  Risch(1/ln(x),x) ⇒ "Not integrable in terms of elementary functions"
  Notes: Risch uses a partial implementation of the Risch algorithm, for logarithmic and exponential extensions. The Risch algorithm builds a tower of logarithmic, exponential, and algebraic extensions. Liouville's principle, which dates back to the 19th century, is an important part of the Risch algorithm. Hermite reduction is applicable to arbitrary elementary functions (functions that can be obtained from the rational functions in x using a finite number of nested logarithms, exponentials, and algebraic numbers or functions) in the algorithm. At this time, this (Risch) function is experimental and should be seen as a framework for future functionality rather than a currently useful tool.
- RTInt(numerator,denominator,var) computes ∫(numerator/denominator,x) using the Rothstein-Trager method
  Needs: Degree, LeadCoef, PolyGCD, Resultnt, SqFreeQ
  Example: RTInt(1,x^3+x,x) ⇒ $\ln(x)-\ln(x^2+1)/2$
  Note: numerator and denominator should be polynomials. RTInt is an experimental function.
- TanPlane(f,vars,pt) returns the equation of the tangent plane to the curve/surface f at the point pt
  Needs: list2eqn
  Example: TanPlane(x^2+y^2,{x,y},{1,1}) ⇒ $2\cdot x+2\cdot y-4=0$

- Combinatorics
  - Cyc2Perm(clist) returns the permutation that has the cycle decomposition clist
    Needs: Sort
    Example: Cyc2Perm({cycle_={1},cycle_={3,2}}) ⇒ {1,3,2}
    Note: Perm2Cyc and Cyc2Perm use the cycle_ = list structure because the AMS does not allow arbitrary nesting of lists.
  - InvPerm(perm) returns the inverse permutation of the permutation perm
    Example: InvPerm({2,4,3,1}) ⇒ {4,1,3,2}
    Note: InvPerm does not check that the permutation is valid.
  - isPermut(list) returns true if list is a valid permutation, else false
    Needs: Sort

Examples: isPermut({1,3,2}) ⇒ true, isPermut({2,3,2}) ⇒ false
Note: For valid permutations, the elements of the list must be numbers.

- KSubsets(plist,k) returns subsets of plist with k elements
  Example: KSubsets({a,b,c},2) ⇒ [[a,b][a,c][b,c]]
- LevCivit(indexlist,g) returns the Levi-Civita symbol for the given metric
  Needs: PermSig
  Example: LevCivit({1,3,2},[[1,0,0][0,r^2,0][0,0,r^2·sin(θ)^2]]) ⇒ -|sin(θ)|·r^2
  Note: LevCivit could be used for computing cross products.
- Ordering(list) returns the positions at which the elements of list occur when the list is sorted
  Needs: Sort
  Example: Ordering({4,1,7,2}) ⇒ {2,4,1,3}
  Note: For the example above, element #2 (1) occurs at position 1 in the sorted list, element #4 (2) occurs at position 2, element #1 (4) occurs at position 3, and element #3 (7) occurs at position 4.
- Perm2Cyc(perm) returns the cyclic decomposition for the permutation perm
  Example: Perm2Cyc({1,3,2}) ⇒ {cycle_={1}, cycle_={3,2}}
  Notes: Perm2Cyc does not check that the permutation is valid. Perm2Cyc and Cyc2Perm use the cycle_ = list structure because the AMS does not allow arbitrary nesting of lists.
- PermSig(list) returns the signature of the permutation (even = 1, odd = -1, invalid = 0) given by list
  Needs: isPermut, Position
  Example: PermSig({3,2,1}) ⇒ -1 (this is an odd permutation)
  Notes: For valid permutations, the elements of the corresponding list must be numbers. The permutation is even(odd) if an even(odd) number of element transpositions will change the permutation to {1,2,3,…}; in the example, 321 → 231 → 213 → 123 ⇒ 3 transpositions ⇒ odd permutation.
- Permut(list) returns all possible permutations of list
  Needs: ListSwap
  Example: Permut({a,b,c}) ⇒ [[a,b,c][b,a,c][c,a,b][a,c,b][b,c,a][c,b,a]]
- RandPerm(n) returns a random permutation of {1,2,…,n}
  Needs: Sort
  Example: RandSeed 0:RandPerm(4) ⇒ {3,4,2,1}

- Continued Fractions
  - CFracExp(num,ordr) returns the continued fraction expansion of the number num to order ordr
    Examples: CFracExp(π,5) ⇒ {3,7,15,1,292}, CFracExp(√(2),4) ⇒ {1,2,2,2},
    CFracExp(5,11) ⇒ {5}, CFracExp(5,∞) ⇒ {5},
    CFracExp(5/2,∞) ⇒ {2,2},
    CFracExp(√(28),∞) ⇒ {5, rep={3,2,3,10}}
    Note: Rational numbers have terminating continued fraction expansions, while quadratic irrational numbers have eventually repeating continued fraction representations. The continued fraction of √(n) is represented as {a, rep={b_1,b_2, b_3,…}}, where the b_i are cyclically repeated. Continued fractions are used in calendars and musical theory, among other applications.
  - ContFrac({num(k),denom(k)},k,n) returns the continued fraction num(k)/(denom(k)+num(k+1)/(denom(k+1)+…))

Example: ContFrac({-(k+a)·z/((k+1)·(k+b)), 1+(k+a)·z/((k+1)·(k+b))},k,2) ⇒

$$\frac{-(a+1)\cdot z\cdot((a+2)\cdot z+3\cdot(b+2))}{(a+1)\cdot(a+2)\cdot z^2+3\cdot(a+1)\cdot(b+2)\cdot z+6\cdot(b+1)\cdot(b+2)}$$

- ContFrc1(list) returns the continued fraction list[1]+1/(list[2]+1/(list[3]+…
  Example: approx(ContFrc1({3,7,15,1,292})) ⇒ 3.14159265301

- Data Analysis
  - DaubFilt(k) returns the FIR coefficients for the Daubechies wavelets
    Needs: Map, Reverse, Select
    Example: factor(DaubFilt(4)) ⇒ {(√(3)+1)·√(2)/8, (√(3)+3)·√(2)/8, -(√(3)–3)·√(2)/8, -(√(3)–1)·√(2)/8}
  - DCT(list/mat) returns the discrete cosine transform of list/mat
    Example: DCT({2+3·***i***, 2–3·***i***}) ⇒ {2·√(2), 3·√(2)·***i***}
  - DST(list/mat) returns the discrete sine transform of list/mat
    Example: DCT({2+3·***i***, 2–3·***i***}) ⇒ {0, 2·√(2)}
  - Fit(mat,f(par,vars),par,vars) returns a least-squares fit of the model f with parameter par to the data in mat
    Needs: FuncEval, VarList
    Example: {{0.3,1.28},{-0.61,1.19},{1.2,-0.5},{0.89,-0.98},{-1,-0.945}}→mat
    Fit(mat,$x^2+y^2-a^2$,a,{x,y}) ⇒ a = 1.33057 or a = -1.33057
    Note: The data in mat is given in the form {{x_1,y_1,…},{x_2,y_2,…},...}. The number of columns of mat should be equal to the number of variables vars.
  - Lagrange(mat,x) returns the Lagrange interpolating polynomial for the data in mat
    Example: Lagrange({{2,9},{4,833},{6,7129},{8,31233},{10,97001}, {12,243649}},x) ⇒ $x^5-3\cdot x^3+1$
    Note: The data matrix mat is in the form {{x1,y1},{x2,y2},…}.
  - LinFilt(list,θ) uses the linear filter given by θ to filter the data list
    Examples: LinFilt({a,b,c},{d,e}) ⇒ {a·e+b·d, b·e+c·d},
    LinFilt({a,b,c},[[d,e][f,g]]) ⇒ [[a·f+b·d, b·f+c·d][a·g+b·e, b·g+c·e]]
  - Residual(mat,poly,x) returns the residual of a polynomial fit of the data in mat
    Examples: [[1,3][2,6][3,11]]→mat:Residual(mat,Lagrange(mat,x),x) ⇒ 0,
    [[0.9][3.82][3,11.1]]→mat:Residual(mat,Lagrange(mat,x),x) ⇒ 2.E-26
    Note: 'poly' is the polynomial from the polynomial fit.

- Differential and Recurrence Equations
  - Casorati({$f_1(t)$,$f_2(t)$,…,$f_n(t)$},t) returns the Casorati matrix for the functions $f_i(t)$
    Example: det(Casorati({2^t,3^t,1},t)) ⇒ $2\cdot 6^t$
    Note: The Casorati matrix is useful in the study of linear difference equations, just as the Wronskian is useful with linear differential equations.
  - ChebyPts(x_0,L,n) returns n points with the Chebyshev-Gauss-Lobatto spacing with leftmost point x_0 and interval length L
    Example: ChebyPts(x0,len,4) ⇒ {x0, len/4+x0, 3·len/4+x0, len+x0}
  - FDApprox(m,n,s,x,i) returns an n^{th} order finite-difference approximation of $\frac{d^m f(x)}{dx^m}$ on a uniform grid
    Needs: Weights
    Example: FDApprox(1,2,1,x,i) ⇒ (f(x[i+1])–f(x[i–1]))/(2·h),

which corresponds to $\frac{f(x_{i+1}) - f(x_{i-1})}{2 \cdot h}$ = nDeriv(f(x),x,h)

Note: Please see the notes for Weights for information on the arguments n and s.

- Lin1ODEs(mat,x) returns solutions of the system of ODEs represented by the matrix mat
  Needs: FreeQ, MatchQ, MatFunc, NthCoeff, Reverse
  Examples: Lin1ODEs([[0,t][-t,0]],t) ⇒ [[cos(t^2/2),sin(t^2/2)][-sin(t^2/2),cos(t^2/2)]],
  Lin1ODEs([[0,3][5,x]],x) ⇒ {},
  factor(Lin1ODEs([[7,3][5,1]],x))|x≥0 ⇒
  [[…,√(6)·($e^{4\cdot\sqrt{(6)}\cdot x}$–1)·($e^{(4-2\cdot\sqrt{(6)})\cdot x}$)/8][5·√(6)·($e^{4\cdot\sqrt{(6)}\cdot x}$–1)·($e^{(4-2\cdot\sqrt{(6)})\cdot x}$)/24,…]]
  Note: The matrix represents the coefficients of the right-hand sides of the ODEs. If no solution can be found, Lin1ODEs returns {}.
- ODE2Sys(ode,x,y) breaks up the ordinary differential equation ode into a system of first-order ODEs
  Needs: RepChars
  Example: ODE2Sys(y'' = -y,x,y) ⇒ {ÿ1'(x) = ÿ2, ÿ2'(x) =-ÿ1}
- ODEExact(plist, r) solves linear nth-order exact ordinary differential equations (ODEs) of the form: p[1](x)·y + p[2](x)·y' + p[3](x)·y'' + … = r(x)
  Needs: RepChars
  Example: Solve the ODE (1+x+x^2)·y'''(x) + (3+6·x)·y''(x) + 6·y'(x) = 6·x
  ODEExact({0,6,3+6·x,1+x+x^2},6·x)
  ⇒ (x^2+x+1)·y = x^4/4 + @3·x^2/2 + @4·x + @5
  Notes: The independent and dependent variables must be x and y, respectively. If the ODE is not exact, the function will say so. The input syntax for this function may be changed in the future.
- ODEOrder(ode) returns the order of the ordinary differential equation ode
  Needs: nChars, Terms
  Examples: ODEOrder(y'+y) ⇒ 1, ODEOrder(-y''''+2·x·y''–x^2 = 0) ⇒ 4
  Note: 'ode' must be given in the prime notation (e.g. y''+y = 0).
- PDESolve(pde, u, x, y, u(var=0) = u_0) solves the PDE in u(x,y)
  $$F(x,y,u,\frac{\partial u}{\partial x},\frac{\partial u}{\partial y}) = 0$$
  Needs: FreeQ, ReplcStr, Terms
  Example: PDESolve(*d*(u(x,y),x)^2·*d*(u(x,y),y)–1=0,u,x,y,u(y=0)=x) ⇒ u(x,y)=x+y
  For more examples, see the included calculator text file PDEDemo.
  Notes: Please do not use the variables p, q, s and t, as they are used internally. PDESolve uses the Lagrange-Charpit and Exact Equation methods. Sometimes, integral transforms can also be used to solve linear differential equations. Use Laplace transforms for initial value problems and Fourier/Hankel transforms for boundary value problems. The Fourier/Hankel transform removes the spatial dependence, while the Laplace transform removes the temporal dependence.
  PDEs usually have the following types of boundary conditions:
  Dirichlet condition: $u(x,y)$ specified on the boundary
  Neumann condition: $\frac{\partial u}{\partial n}$ specified on the boundary
  Cauchy condition: u and $\frac{\partial u}{\partial n}$ specified on the boundary.
- PDEType(pde,f(x,y)) returns the type of a 2nd-order partial differential equation (hyperbolic, elliptic, parabolic, or mixed), with the function given as e.g. u(x,y)

Needs: FreeQ, Terms

Examples: PDEType(*d*(u(x,y),x,2)+y^2, u(x,y)) ⇒ "parabolic"

PDEType(*d*(u(x,y),x,2)+*d*(*d*(u(x,y),x),y)+y^2, u(x,y)) ⇒ "hyperbolic"

PDEType(*d*(u(x,y),x,2)+*d*(u(x,y),y,2)+f(x,y), u(x,y)) ⇒ "elliptic"

PDEType(x·*d*(u(x,y),x,2)–*d*(u(x,y),y,2), u(x,y)) ⇒ "mixed"

- RSolve(list,u,t) solves a linear recurrence/difference equation list[1]·u(t+k)+list[2]·u(t+k–1)+…+list[dim(list)–1]·u(t)+list[dim(list)]=0 for u(t)
  Needs: FreeQ, Freq, mZeros
  Examples: RSolve({1,-p,r},u,t) ⇒ u(t) = ç[1,1]·p^t–r·(p^t–1)/(p–1),
  RSolve({1,-7,6,0},y,n) ⇒ y(n) = 6^n·ç[2,1]+ç[1,1]
  Note: At this time, initial conditions must be substituted manually.
- Weights(d,n,s) returns the weights for an n^{th} order finite-difference approximation for the d^{th}-order derivative on a uniform grid
  Needs: Coef, Reverse
  Examples: Weights(1,2,1) ⇒ {-1/(2·h), 0, 1/(2·h)},
  Weights(1,3,3/2) ⇒ {1/(24·h), -9/(8·h), 9/(8·h), -1/(24·h)}
  Notes: n is the total number of grid intervals enclosed in the finite difference stencil and s is the number of grid intervals between the left edge of the stencil and the point at which the derivative is approximated. s = n/2 for centered differences. When s is not an integer, the grid is called a staggered grid, because the derivative is requested between grid points.
- Wronsk({f_1(x),f_2(x),…,f_n(x)},x) returns the Wronskian of the functions f_i(x)
  Example: Wronsk({x^2,x+5,ln(x)},x) ⇒ [[x^2,x+5,ln(x)][2·x,1,1/x][2,0,-1/x^2]]
  Note: The functions are linearly independent if det(Wronskian)≠0.

◆ Equation/Inequality Solving

- Cubic({a,b,c,d}) returns the exact roots of the cubic a·x^3+b·x^2+c·x+d = 0
  Example: Cubic({35,-41,77,3})
  ⇒ {(315·√(4948777)–477919)$^{1/3}$/105–…+41/105,…,…}
- Diophant({a,b,c,d,e,f},{x,y}) solves the quadratic bivariate diophantine equation a·x^2+b·x·y+c·y^2+d·x+e·y+f = 0, {a,b,c,d,e,f}∈**Z**, for integers {x,y}
  Needs: CFracExp, ContFrc1, Divisors, ExtGCD, ListSwap, mor
  Examples: Diophant({42,8,15,23,17,-4915},{x,y}) ⇒ x=-11 and y=-1,
  exp list(Diophant({0,2,0,5,56,7},{x,y}),{x,y})
  ⇒ [[105,-2][-9,1][-21,7][-27,64][-29,-69][-35,-12][-47,-6][-161,-3]],
  Diophant({0,0,0,5,22,18},{x,y}) ⇒ x=22·@n1+8 and y=-5·@n1–1,
  Diophant({8,-24,18,5,7,16},{x,y})
  ⇒ x=41·@n2–174·@n2^2–4 and y=37·@n2–116·@n2^2–4 or
  x=17·@n2–174·@n2^2–2 and y=21·@n2–116·@n2^2–2,
  Diophant({1,0,4,0,0,-1},{x,y}) ⇒ x=1 and y=0 or x=-1 and y=0,
  Diophant({1,0,3,0,0,1},{x,y}) ⇒ false,
  Diophant({1,0,-2,0,0,-1},{x,y}) ⇒ x=cosh(@n3·ln(…)) and
  y=sinh(@n3·ln(…))·√(2)/2 and @n3≥0 or …
  Note: Diophant returns false if there are no solutions for {x,y} in the integers. Diophant can currently solve any solvable linear, elliptic, or parabolic equation, as well as many hyperbolic (including some Pell-type) equations.
- iSolve(xpr,var) solves an inequality given by xpr for the variable var
  Needs: IsCmplxN, mor, Sort, VarList
  Examples: iSolve(x^3–2·x^2+x<0,x) ⇒ x<0, iSolve(x^2≥2·x+3,x) ⇒ x≥3 or x≤-1,
  iSolve(e^x+x≤1,x) ⇒ x≤0., iSolve(abs(x–3)≥0,x) ⇒ true,

iSolve($x^2 \cdot e^x \geq 1/2$,x) ⇒ x≤-1.48796 and x≥-2.61787 or x≥0.539835
Note: Wrapping approx() around the iSolve() or evaluating in Approx mode gives faster results, especially for complicated results.

- ModRoots(poly,p) returns roots of poly mod prime p
  Needs: Degree, PolyMod, Select, sFactors
  Examples: ModRoots(x^2=16,x,7) ⇒ {4,3}, ModRoots(x^2=16,x,11) ⇒ {4,7},
  ModRoots($(x^2+1)^2-25$,x,97) ⇒ {2,95},
  ModRoots(x^3+1,x,137) ⇒ {136},
  ModRoots($x^3+11\cdot x^2-25\cdot x+13$,x,5) ⇒ {2}
  Note: ModRoots often does not return all the roots of poly mod p.
- mZeros(poly,x) returns the zeros of the polynomial poly, with multiplicities
  Needs: Degree, Sort
  Example: mZeros($x^2+2\cdot x+1$,x) ⇒ {-1,-1}
- NewtRaph(f,vars,start,tol,intermresults) returns a solution for f = 0
  Needs: Jacobian, list2eqn
  Example: nSolve($x\cdot e^x$ = 2, x=0) ⇒ 0.852605502014,
  NewtRaph($x\cdot e^x$ = 2, x, 0, 1E-10, false) ⇒ {x=0.852605502014},
  NewtRaph($x\cdot e^x$ = 2, x, 0, 1E-10, true) ⇒ {solution={x=…}, …}
  Notes: NewtRaph uses Newton-Raphson iteration. The 'intermresults' argument should be true or false; it specifies whether or not intermediate results are given.
- NRoots(coeflist) finds approximate roots of the polynomial with coefficients in coeflist in descending order
  Needs: Reverse
  Example: NRoots({1,0,0,0,-1,-1}) ⇒ {1.1673, 0.18123+1.084·***i***,
  0.18123–1.084·***i***, -0.76488+0.35247·***i***,
  -0.76488–0.35247·***i***}
  Note: NRoots uses the eigenvalue method, which can be computationally expensive but is quite robust at finding close and multiple roots. The example polynomial above is x^5-x-1.
- NumRoots(poly,var,istart,iend) returns the number of real roots of the polynomial poly in the open interval (istart,iend)
  Needs: Sturm
  Examples: NumRoots(x^2+1,x,-2,2) ⇒ 0, NumRoots(Π(x–i,i,-3,2),x,-2,0) ⇒ 2
  Note: The computation may take some time due to a slowness in Sturm.
- Quartic({a,b,c,d,e}) gives the exact roots of the quartic $a\cdot x^4+b\cdot x^3+c\cdot x^2+d\cdot x+e = 0$
  Needs: Cubic
  Example: Quartic({1,0,0,1,4})
  ⇒ {-√(2·cos($\tan^{-1}$(√(49071)/9)/3))·$3^{3/4}$/3–…,…,…,…}
  Note: Abel and Galois proved that general equations of fifth and higher order cannot be solved in terms of n^{th} roots of numbers (see [MathWorld entry]).
- Sturm(poly,var) returns the Sturm sequence for the polynomial poly
  Needs: Degree, Remaindr
  Example: Sturm($x^3+2\cdot x+1$,x) ⇒ {$x^3+2\cdot x+1$, $3\cdot x^2+2$, -4·x–3, -59}
  Note: Sturm may be slow.

- Lists
  - DelCol(mat,c) returns the matrix mat with column c deleted
    Example: DelCol([[a,b][c,d]],2) ⇒ [[a][c]]
  - DelElem(list,i) returns list with the ith element removed
    Example: DelElem({a,b,c,d},3) ⇒ {a,b,d}

- DelRow(mat,r) returns the matrix mat with row r deleted
  Example: DelRow([[a,b][c,d]],1) ⇒ [[c,d]]
- LGcd(list) returns the GCD of the elements of list
  Example: LGcd({2,16,4,0,2}) ⇒ 2
- list2eqn(list) inserts ‘and’ between the elements of list
  Example: list2eqn({a≠b,c=d}) ⇒ a–b≠0 and c=d
  Note: list2eqn is now deprecated, since the MathTools Flash app provides a faster function `MathTool.ListAnd`.
- ListPair(list) returns possible two-element combinations from list as rows in a matrix
  Example: ListPair({a,b,c,d}) ⇒ [[a,b][a,c][b,c][a,d][b,d][c,d]]
- ListPlot(ylist,joined) plots the points given in the list ylist versus x, with ‘joined’ specifying whether the points should be connected by lines
  Example: ListPlot(seq(2^i,i,1,6),true)
- ListRpt(list,n) returns the list repeated n times
  Example: ListRpt({a,b,c},3) ⇒ {a,b,c,a,b,c,a,b,c}
- ListSubt(list1,list2) returns elements from list1 that are not in list2
  Needs: FreeQ
  Example: ListSubt({a,b,c,d,e},{d,b,e}) ⇒ {a,c}
- ListSwap(list,i,j) returns list with the ith and jth elements swapped
  Example: ListSwap({a,b,c,d,e,f},2,5) ⇒ {a,e,c,d,b,f}
- LLcm(list) returns the LCM of the elements of list
  Example: LLcm({3,11,9,1,15}) ⇒ 495
- mand(list) is an auxiliary function that returns true if all elements of list are true, else returns false
  Examples: mand({a=a,2>3,1<2}) ⇒ false, mand({x+x=2·x,2·2=4}) ⇒ true
- MemberQ(list/matrix,e) returns true if e is an element of list/matrix, else false
  Needs: MatchQ
  Examples: MemberQ({a, b, c},b) ⇒ true, MemberQ({x, y^2, z},y) ⇒ false,
  MemberQ({2, e^x, x^2}, e^z) ⇒ false, MemberQ({2, e^x, x^2}, $e^{z\_}$) ⇒ true
  Note: MemberQ works with patterns, as shown in the last example above.
- mor(list) is an auxiliary function that inserts an “or” between the elements of list
  Needs: ReplcStr
  Example: mor({a>b,a>c}) ⇒ a–b>0 or a–c>0
  Note: mor is now deprecated, since the MathTools Flash app provides a faster function `MathTool.ListOr`.
- Pad(list,n) simply pads list with zeros so that the total number of elements is |n|
  Examples: Pad({a,b,c},5) ⇒ {a,b,c,0,0}, Pad({a,b,c},-4) ⇒ {0,a,b,c}
  Note: If n is negative, the zeros are prepended to the list; if n is positive, they are appended.
- Position(list/mat,elem/list) returns the position of element/list in the list/matrix
  Needs: mand
  Examples: Position({a,b,c,d},c) ⇒ 3, Position({a,b,c,d},e) ⇒ 0,
  Position([[a,b][c,d]],c) ⇒ {2,1} (row 2, column 1),
  Position([[1,2][3,4]],{3,4}) ⇒ 2
  Note: Searching for a symbolic list is not currently recommended, because it can give incorrect answers (e.g. Position([[a,b][c,d]],{c,d}) ⇒ 1 while it is 2).

- Reverse(list) simply reverses the elements of the list or matrix
  Examples: Reverse({a,b,c,d}) ⇒ {d,c,b,a}
  Reverse([[a,b][c,d]]) ⇒ [[c,d][a,b]]
- RmDup(list) returns list with duplicate elements removed
  Needs: DelElem, MemberQ
  Example: RmDup({a,b,c,d^2,c}) ⇒ {a,b,c,d^2}
- Select(list,str) returns elements of list that satisfy the criteria given in str
  Needs: ReplcStr
  Example: Select({-4.5,-3,2,0,7.2,∞},"not isApprox(#) and #≥0") ⇒ {2,0, ∞}
- Sequency(list) returns the number of sign changes in list
  Examples: Sequency({3,-2,1}) ⇒ 2, Sequency({-3,-2,1}) ⇒ 1
- Sort(list) sorts the elements of list in ascending order
  Needs: ListSwap
  Example: Sort({4,1,9,3,7,1}) ⇒ {1,1,3,4,7,9}
  Note: Sort works only for real numeric elements. If a matrix is provided, it sorts the rows of the matrix, treating columns past the first column as dependent.
- Union(list1,list2) returns the union of list1 and list2
  Needs: RmDup
  Example: Union({c,a,b},{d,c,a}) ⇒ {c,a,b,d}
  Note: The union is a joined list with duplicate elements removed.

- Logic
  - IfThen(boolxpr1,boolxpr2) gives the conditional boolxpr1⇒boolxpr2, which means 'if boolxpr1 then boolxpr2'
    Example: IfThen(a, b or c) ⇒ ~a or b or c
  - IfOnlyIf(boolxpr1,boolxpr2) gives the biconditional boolxpr1⇔boolxpr2, which means 'boolxpr1 iff boolxpr2'
    Example: IfOnlyIf(a and b, c) ⇒ ~a and ~c or a and b and c or ~b and ~c
  - Nand(xpr1,xpr2) returns the logical nand function
    Example: Nand(x or y, z) ⇒ ~x and ~y or ~z
  - Nor(xpr1,xpr2) returns the logical nor function
    Example: Nor(a, b xor c) ⇒ ~a and ~b and ~c or ~a and b and c
  - TruthTbl(boolxpr) returns a truth table for the expression/list boolxpr
    Needs: BaseConv, Str2Chrs, VarList
    Example: TruthTbl(a xor b) ⇒ $\begin{bmatrix} \text{a} & \text{b} & \text{~ a and b or a and ~ b} \\ \text{true} & \text{true} & \text{false} \\ \text{true} & \text{false} & \text{true} \\ \text{false} & \text{true} & \text{true} \\ \text{false} & \text{false} & \text{false} \end{bmatrix}$

- Number theory
  - ANumNorm(a) returns the norm of the algebraic number a
    Needs: Coef, MinPoly, RatNumQ
    Example: ANumNorm(AlgNum($x^4-10\cdot x^2+1$,{1,-9/2,0,1/2})) ⇒ -1
    Note: The norm of a is the product of all roots of its minimal polynomial. An algebraic number field is a finite extension field of the field of rational numbers. Within an algebraic number field is a ring of algebraic integers, which plays a role similar to the usual integers in the rational numbers. Algebraic integers and the usual "rational" integers have their similarities and differences. For example,

one may lose the unique factorization of elements into products of prime powers (the Fundamental Theorem of Arithmetic). Algebraic numbers are roots of polynomial equations. A non-algebraic number is called transcendental. A good reference on algebraic number theory is Henri Cohen's book "A Course in Computational Algebraic Number Theory."

- ANumTrac(a) returns the trace of the algebraic number a
  Needs: Coef, MinPoly, RatNumQ
  Example: ANumTrac(AlgNum(x^4–10·x^2+1,{1,-9/2,0,1/2})) ⇒ 2
  Note: The trace of a is the sum of all roots of its minimal polynomial.
- CarLamb(n) returns the Carmichael lambda λ(n), the least universal exponent
  Needs: FactList, LLcm, Totient
  Example: CarLamb(5040) ⇒ 12
  Note: A universal exponent for an integer n is an integer m such that for all k relatively prime to n, $k^m \equiv 1$ (mod n).
- CarNumQ(n) returns true if n is a Carmichael number, else false
  Needs: FactList, mand
  Examples: CarNumQ(561) ⇒ true, CarNumQ(563) ⇒ false
  Note: A Carmichael number n is a composite integer such that for every integer b relatively prime to n, $b^{n-1} \equiv 1$ (mod n). For any Carmichael number n, n–1 is divisible by p–1, where p represents any of the prime factors of n.
- CentrRem(i,j,x) returns the central remainder
  Example: CentrRem(5,8,11) ⇒ 2
  Note: The central remainder, as its name suggests, is the middle remainder in the remainder sequence (from the Euclidean algorithm) for finding a representation of a prime p as a sum of two squares.
- cGCD(a,b) returns the GCD of complex numbers a and b
  Example: cGCD(70+6·***i***, 150+10·***i***) ⇒ -2+2·***i***
- CharPoly(a,x) gives the characteristic polynomial of the algebraic number a relative to its number field
  Needs: Degree, LLcm, Resultnt, VarList
  Example: CharPoly(AlgNum(x^3–x+1,{2,7/3,-1}),x) ⇒ x^3–4·x^2–67·x/9+925/27
  Note: CharPoly currently works only for algebraic numbers. It will be extended to work for matrices in the future.
- ChinRem({{a_1, m_1},{a_2, m_2},…}) solves the system of congruences
  $k \equiv a_i$ (mod m_i) for the integer k
  Needs: ExtGCD, LLcm
  Example: ChinRem({{1,3},{3,5},{5,8}}) ⇒ 120·@n1 + 13
  Note: ChinRem returns the general solution. For a particular solution, set the constant @n1 to zero.
- Divisors(n) returns the divisors of n
  Needs: FactList, ListPair
  Example: Divisors(147) ⇒ {1,2,19,38}, Divisors(541) ⇒ {1,541},
  Divisors(3^7) ⇒ {1,3,9,27,81,243,729,2187}
  Note: The divisor list returned by Divisors occasionally has duplicates, and may not be sorted. A perfect number equals the sum of its proper divisors, i.e. divisors excluding itself.
- ExtGCD(a,b) returns integers {d,{s,t}} such that d=s·a+t·b=GCD(a,b)
  Needs: IsCmplxN, Quotient, VarList
  Example: ExtGCD(541,547) ⇒ {d = 1, st = {91,-90}}

Notes: ExtGCD can be used to find the multiplicative inverse of a (mod b). For example, to solve the congruence 3·k ≡ 4 (mod 5), multiply both sides by the mod-5 inverse (the s returned from ExtGCD(3,5)), so k = 2 · 4 = 8 is a solution. ExtGCD also works for univariate polynomials.

- FactList(n) returns a matrix of the factors of n, with multiplicities
  Needs: ReplcStr
  Example: FactList(123456) ⇒ [[2,6][3,1][643,1]],
  FactList(x^2+1) ⇒ [[x–***i***,1][x+***i***,1]],
  FactList($(x+\boldsymbol{i})^2$) ⇒ [[x+***i***, 2]],
  FactList($(x+\boldsymbol{i})^2$/2) ⇒ "Error: cannot handle division at present"
  Note: If xpr is an integer, FactList returns its prime factors with multiplicities. If xpr is provided as a string, the factors are found without any evaluation.
- FundDisc(n) returns true if n is a fundamental discriminant of a quadratic number field, else returns false
  Needs: Moebius
  Examples: seq(FundDisc(i),i,-5,5) ⇒
  {false,true,true,false,false,false,false,false,false,false,true}
- gFactor(n) returns the Gaussian factors of n
  Needs: FactList, IsGPrime, Sum2Sqr
  Examples: gFactor(2) ⇒ [[1+***i***,1][1–***i***,1]], gFactor(3) ⇒ [[3, 1]],
  gFactor(2) ⇒ [[1+***i***,2][1–***i***,2]], gFactor(6) ⇒ [[1+***i***,1][1–***i***,1][3,1]]
  Note: The returned factors may not be coprime. The result of this function may be changed in the future.
- IsGPrime(z) returns true if z is a Gaussian prime, else false
  Examples: IsGPrime(2) ⇒ false, (1+***i***)·(1–***i***) ⇒ 2, IsGPrime(1+***i***) ⇒ true
- IsMPrime(n) returns true if n is a Mersenne prime, else returns false
  Needs: LogBase, Nest
  Example: seq(IsMPrime(i),i,1,8) ⇒ {false,false,true,false,false,false,true,false}
- JacSymb(k,n) returns the Jacobi symbol
  Needs: FactList, LegSymb
  Examples: JacSymb(2,8) ⇒ 0, JacSymb(3,8) ⇒ -1
  Note: JacSymb(k,n) equals LegSymb(k,n) for prime n.
- KronSymb(m,n) returns the Kronecker symbol (m/n)
  Needs: JacSymb
  Example: KronSymb(2,3) ⇒ 1
- LegSymb(k,n) returns the Legendre symbol
  Needs: QuadResQ
  Examples: LegSymb(2,3) ⇒ -1, LegSymb(2,7) ⇒ 1
- LucasNum(n) returns the n^{th} Lucas number
  Needs: LucasSeq
  Example: LucasNum(21) ⇒ 24476
- LucasSeq(p,q,L_0,L_1,n) returns the n^{th} term L_n of the generalized Lucas sequence with $L_{n+2} = p\cdot L_{n+1} - q\cdot L_n$
  Example: LucasSeq(1,-1,0,1,30) ⇒ 832040
  Note: L_0 and L_1 are the initial values for the recurrence.
- MinPoly(a,x) returns the minimal polynomial of the algebraic number a
  Needs: CharPoly, ConstQ, Degree, FactList, NthCoeff, Select, SqrFree, VarList
  Example: MinPoly(AlgNum(x^3–x+1,{2,7/3,-1}),x) ⇒ $27\cdot x^3-108\cdot x^2-201\cdot x+925$
  Note: The minimal polynomial of an algebraic number a is the unique

irreducible monic polynomial p of smallest degree such that p(a) = 0. MinPoly currently works only for algebraic numbers. It will be extended to work for matrices in the future.

- Moebius(n) returns the Möbius function μ(n)
  Needs: ReplcStr
  Examples: Moebius(12) ⇒ 0, Moebius(12345) ⇒ -1, Moebius(1234567) ⇒ 1
- MultMod(a,b,p) returns (a·b) (mod p) for large positive integers a, b, and p
  Example: MultMod(123456789,987654321,7) ⇒ 3, MultMod(2^5,3^4,5) ⇒ 2
  Note: MultMod is part of the MathTools Flash app; for installation instructions see the "Installation" section. Of course, you can use this function for small integers too. The full path for the MultMod function is `MathTool.MultMod`.
- NFMahler(a) returns the Mahler measure of the algebraic number a
  Needs: Coef, MinPoly
  Example: NFMahler(AlgNum($x^4-10\cdot x^2+1$,{1,-9/2,0,1/2})) ⇒ √(2)+1
  Note: The Mahler measure is widely used in the study of polynomials.
- NFNorm(a) gives the norm of the algebraic number a relative to its number field
  Needs: Degree, LLcm, Resultnt, VarList
  Example: NFNorm(AlgNum(x^3-x+1,{2,7/3,-1})) ⇒ -925/27
  Note: The relative norm of a is the product of all roots of its characteristic polynomial.
- NFTrace(a) gives the trace of the algebraic number a relative to its number field
  Needs: CharPoly, Degree, VarList
  Example: NFTrace(AlgNum(x^3-x+1,{2,7/3,-1})) ⇒ 4
  Note: The relative trace of a is the sum of all roots of its characteristic polynomial.
- NxtPrime(n[,dir]) returns the next/previous prime
  Examples: NxtPrime(10621) ⇒ 10627, NxtPrime(10621,-1) ⇒ 10613
  Note: NxtPrime returns the next prime if dir is positive, and the previous prime if dir is negative.
- OrderMod(a,m) returns the multiplicative (mod-m) order $\mathrm{ord}_m(a)$
  Needs: PowerMod, RelPrime
  Examples: OrderMod(2,3) ⇒ 2, OrderMod(2,5) ⇒ 4, OrderMod(2,137) ⇒ 68
- PowerMod(a,b,p) returns (a^b) (mod p) for large positive integers a, b, and p
  Example: PowerMod(129140163,488281255,7) ⇒ 5, PowerMod(2,4^7,5) ⇒ 1
  Note: The modular power is perhaps the most important function in elementary number theory. PowerMod is part of the MathTools Flash app; for installation instructions see the "Installation" section. Of course, you can use this function for small integers as well. The full path for the PowerMod function is `MathTool.PowerMod`.
- PowTower(x) returns the value of the infinite power tower x^x^x^… = x↑↑∞
  Needs: LambertW
  Example: PowTower(1.1) ⇒ 1.11178201104,
  1.1^1.1^1.1^1.1^1.1 ⇒ 1.11178052653
- Prime(n) returns the n^{th} prime number
  Examples: Prime(10001) ⇒ 104743, Prime(0) ⇒ undef, Prime(-7) ⇒ undef
- PrimePi(n) returns the number of primes less than or equal to n
  Needs: LogInt, Prime
  Example: PrimePi(179) ⇒ 41

- Primes(n) returns the first n prime numbers
  Example: Primes(11) ⇒ {2,3,5,7,11,13,17,19,23,29,31}
- QuadResQ(k,p) returns true if k is a quadratic residue mod p, else false
  Needs: PowerMod
  Examples: QuadResQ(7,11) ⇒ false, QuadResQ(7,31) ⇒ true
  Note: A number k is a quadratic residue mod p if it is relatively prime to p and there is an integer j such that $k \equiv j^2$ (mod p).
- RelPrime(list) returns true when all elements of list are relatively prime, else returns false
  Examples: RelPrime({5,7}) ⇒ true, RelPrime({2,4}) ⇒ false,
  RelPrime({x–2,x^3+2·x^2–x–2}) ⇒ true
  Notes: RelPrime works for numbers and univariate polynomials. The probability that n integers chosen at random are relatively prime is 1/ζ(n), where ζ(n) is the zeta function. The Fermat numbers 2^2^n+1 are relatively prime.
- RndPrime(n) returns a random prime number in [1,n]
  Example: RandSeed 0:RndPrime(10^9) ⇒ 951898333
- SqrtModP(a,p) returns the smallest nonnegative square-root of a (mod p)
  Needs: JacSymb, PowerMod
  Examples: SqrtModP(123456,987) ⇒ 360, SqrtModP(1234567,987) ⇒ {}
  Note: SqrtModP uses Shanks' method. It returns {} if there is no square-root.
- SqrtNOMP(p) returns a square-root of -1 mod p
  Needs: JacSymb, PowerMod, Sort
  Example: SqrtNOMP(1741) ⇒ 59
- Sum2Sqr(p) returns {a,b} where $p = a^2+b^2$
  Needs: CentrRem, SqrtNOMP
  Examples: Sum2Sqr(5) ⇒ {1,2}, Sum2Sqr(103841) ⇒ {104,305}
  Note: Sum2Sqr uses Smith's algorithm to find {a,b}.
- SumOfSqN(d,n) returns the number of representations $r_d(n)$ of n as a sum of d squares of integers
  Needs: Divisors, DivSigma, FactList, IntExp, JacSymb, Moebius
  Examples: SumOfSqN(2,5) ⇒ 8, SumOfSqN(3,4) ⇒ 6,
  SumOfSqN(7,3) ⇒ 280, SumOfSqN(5,217) ⇒ 27840
- Totient(n) returns the Euler φ(n) function, which is the number of positive integers less than n that are relatively prime to n
  Needs: FactList
  Examples: Totient(143055667) ⇒ 142782138, Totient(17^{100}) ⇒ 1043669997…
  Note: The Euler totient function is a multiplicative function, which is a function f: $\mathbb{Z}^+ \to \mathbb{Z}^+$ such that whenever gcd(i,j) = 1, f(i·j) = f(i)·f(j), so that one can obtain f(n) by multiplying $f(p^k)$, where p^k is a prime power from the factorization of n. For all n greater than 1, the value of φ(n) lies between 1 and n–1, inclusive. φ(n) = n–1 iff n is prime. Also, for n > 2, φ(n) is even.

◆ Optimization

- LagrMult(f(x,y,…),{cond1,cond2,…},{x,y,…}) finds extrema for f(x,y,…) subject to the conditions {cond1=0, cond2=0, …}
  Needs: list2eqn
  Example: LagrMult(x^2+y^2,{x^2+y–1},{x,y}) ⇒ x = √(2)/2 and y = 1/2 and …
  Note: If you want LagrMult to return the maximum/minimum values of f(x,y,…) in addition to the values of x,y,…, open the function in the Program

Editor and delete the "Return hh" line. The extrema returned may be local or global extrema. You can use the Hessian to determine the type of extremum.

- Minimize({f, constraints},{vars, start},{{opt1,val1},…}) finds a minimum of the objective function f subject to the given constraints
  Needs: FuncEval, mand
  Examples: RandSeed 0: Minimize({x^2+(y–1/2)^2, y>=0, y>=x+1},
  {{x,y},{5,5}},{{"MaxIterations",2}})
  Note: Minimize is a program (instead of a function) so that the user can get an overview of the progress. It currently uses the method of simulated annealing. Minimize is experimental and is generally slow, especially with constraints.
- Simplex(obj,cmat) returns the minimum for the linear programming problem with the objective function given by obj and the constraints given by cmat
  Needs: ListSubt, Select, SimpStep, Sort
  Example: Minimize $f = 5 \cdot x - 3 \cdot y - 8 \cdot z$ subject to the constraints
  $\{2 \cdot x+5 \cdot y-z \le 1, -2 \cdot x-12 \cdot y+3 \cdot z \le 9, -3 \cdot x-8 \cdot y+2 \cdot z \le 4, x \ge 0, y \ge 0, z \ge 0\}$
  Simplex({5,-3,-8,0,0,0,-1,0},
  [[2,5,-1,1,0,0,0,1][-2,-12,3,0,1,0,0,9][-3,-8,2,0,0,1,0,4]]) ⇒
  {-121, vars = {0,3,14,0,3,0}}
  Note: There are two widely used solution techniques for linear programming: simplex methods and interior point methods. While simplex methods move from vertex to vertex along the edges of the boundary of the feasible region, interior point methods visit points within the interior of the feasible region. The Simplex function currently uses the tableau form of the simplex method. The input list is obtained in the above example by setting $5 \cdot x-3 \cdot y-8 \cdot z-f = 0$, so that f has a coefficient of -1. You can also enter a shortened form of the list as {5,-3,-8}. The rows in cmat give the coefficients of the original variables and the "slack" variables in the constraints. There is a slack variable for each inequality constraint. A slack variable allows you to turn an inequality constraint into an equality constraint. So, for example, the first constraint in the above example would become $2 \cdot x+5 \cdot y-z+s_1 = 1$. The final result displays the minimum of f and the values of the original and slack variables at the minimum. So, for the above example, f has a minimum at -121, where $\{x=0, y=3, z=14, s_1=0, s_2=3, s_3=0\}$. To find the maximum rather than the minimum, simply find the minimum of $-f$ and flip the sign of f in the result.
- SimpStep(obj,mat,bvars) does pivoting for a single step of the simplex method and gives the tableau for the next basic feasible solution with the specified basic variables bvars
  Needs: ListSubt
  Example: SimpStep({5,-3,-8,0,0,0,-1,0}, [[2,5,-1,1,0,0,0,1][-2,-12,3,0,1,0,0,9]
  [-3,-8,2,0,0,1,0,4]], {3,4,5}) ⇒
  [[-3/2,-4,1,0,0,1/2,0,2][1/2,1,0,1,0,1/2,0,3]
  [5/2,0,0,0,1,-3/2,0,3][-7,-35,0,0,0,4,1,16]]
  Note: The rows represent the results for {bvars[1],bvars[2],…,bvars[n],*f*}, where *f* is the objective function. Non-basic variables (the variables not in bvars, in this case {1,2,6}) are assumed to be zero, so that one can read off the result for each basic variable by looking at the number in the last column.

- Pattern-matching
  - FreeQ(expression1,expression2) returns true if expression1 is free of expression2, else returns false
    Needs: mand

Examples: FreeQ(y^(a·x^2+b),x) ⇒ false, FreeQ(ln(cos(x)),y) ⇒ true,
FreeQ({a,$b^{2·x}$,c,d^2},2·x) ⇒ false, FreeQ(x^2,"cos") ⇒ true,
FreeQ($cos(x)^2$,"cos") ⇒ false
Note: Please see MatchQ entry for a note on auto-simplification.

- MatchQ(expression,form) returns true if the structure of expression matches that of form, else returns false
  Needs: mand
  Examples: MatchQ(2,x) ⇒ false, MatchQ(2,x_) ⇒ true, MatchQ(a^2,x_) ⇒ true,
  MatchQ(a^2,x_+y_) ⇒ false, MatchQ(a^x·f(x),2^x·f(x)) ⇒ false,
  MatchQ(2^x·f(x), a^x·f(x)) ⇒ false, MatchQ(2^x·f(x), $a\_^x$·f(x)) ⇒ true,
  MatchQ(a(b,c),d(e,f,g)) ⇒ false, MatchQ(a(b,c),d(e,f)) ⇒ false,
  MatchQ(a_(b_,c_),d_(e_,f_)) ⇒ true
  Note: The AMS performs some auto-simplification before MatchQ can use the expression, so for example MatchQ(x+x,a_+b_) ⇒ false, because x+x is simplified to 2·x. This can affect the form as well. Here is another example: MatchQ(x^2·y(n),a_·f_(n)) ⇒ false, while MatchQ(x^2·c(n),a_·f_(n)) ⇒ true. This auto-simplification can be good or bad, depending on the context. In general, I would suggest trying expressions and forms out on the home screen to see how they are auto-simplified before using them in MatchQ. Also, until I implement default values for MatchQ, x will not match $a\_^{n\_}$; this is because the structure does not match exactly unless you use a default value of 1 for powers.

- Special Functions
  - AiryAi(x) returns the Airy function Ai(x) for real x
    Needs: Gamma
    Examples: AiryAi(0.1) ⇒ 0.329203129944, AiryAi(7.3) ⇒ 0.000000332514
  - AiryBi(x) returns the Airy function Bi(x) for real x
    Needs: Gamma
    Examples: AiryBi(-5.2) ⇒ -0.275027044184, AiryBi(9.5) ⇒ 96892265.5804
  - BellNum(n) returns the n^{th} Bell number
    Needs: BellPoly
    Example: seq(BellNum(i),i,0,6) ⇒ {1,1,2,5,15,52,203}
  - BellPoly(n,x) returns the n^{th} Bell polynomial
    Needs: BellNum
    Example: BellPoly(5,x) ⇒ x^5+10·x^4+25·x^3+15·x^2+x
    Note: BellPoly(n,1) returns the n^{th} Bell number.
  - BernNum(n) returns the n^{th} Bernoulli number B_n
    Needs: Zeta
    Example: seq(BernNum(i),i,1,6) ⇒ {-1/2,1/6,0,-1/30,0,1/42}
  - BernPoly(n,x) returns the n^{th} Bernoulli polynomial B_n(x)
    Needs: IsCmplxN, Zeta
    Example: BernPoly(3,x) ⇒ x^3–3·x^2/2+x/2
  - BesselI(ν,z) returns the $ν^{th}$ modified Bessel function of the first kind I_n(z)
    Needs: BesselJ
    Examples: BesselI(3/2,x) ⇒ √(2)·((x–1)·$(e^x)^2$+x+1)/(2·√(π)·$x^{3/2}$·e^x),
    BesselI(0.92,0.35) ⇒ 0.210999194087–2.2083340901E-16·***i***
    Note: Please see the note in the help entry for BesselJ.
  - BesselJ(ν,z) returns the $ν^{th}$ Bessel function of the first kind J_n(z)
    Needs: Gamma, IsCmplxN
    Examples: BesselJ(3/2,x) ⇒ √(2)·sin(x)/(√(π)·$x^{3/2}$)–√(2)·cos(x)/(√(π)·√(x)),

BesselJ(2,1/2) ⇒ 0.030604023459, BesselJ(51/2,1.·π) ⇒ 0.

Note: Symbolic results from BesselJ for half-integer ν may give incorrect results for subsequent floating point evaluation because of catastrophic cancellation. Do comDenom(symbresult|x=«value», q) and then approximate it, or calculate the numeric result directly (see the last example above).

- BesselK(ν,z) returns the ν^{th} modified Bessel function of the second kind $K_n(z)$
  Needs: BesselI
  Example: BesselK(5/2,x) ⇒
  $\sqrt{(\pi)}\cdot\sqrt{(2)}/(2\cdot\sqrt{(x)}\cdot e^x)+3\cdot\sqrt{(\pi)}\cdot\sqrt{(2)}/(2\cdot x^{3/2}\cdot e^x)+3\cdot\sqrt{(\pi)}\cdot\sqrt{(2)}/(2\cdot x^{5/2}\cdot e^x)$
  Note: Please see the note in the help entry for BesselJ.
- BesselY(ν,z) returns the ν^{th} Bessel function of the second kind $Y_n(z)$
  Needs: BesselJ
  Example: BesselY(3/2,x) ⇒ $-\sqrt{(2)}\cdot\cos(x)/(\sqrt{(\pi)}\cdot x^{3/2})-\sqrt{(2)}\cdot\sin(x)/(\sqrt{(\pi)}\cdot\sqrt{(x)})$,
  BesselY(1,10.3) ⇒ 0.24706994
  Note: Please see the note in the help entry for BesselJ.
- Beta(p,q) returns the beta function *B*(p,q)
  Needs: Gamma
  Example: Beta(1/2,2) ⇒ 4/3
  Note: The (complete) beta function is symmetric in its arguments, that is, *B*(p,q) = *B*(q,p).
- Catalan(n) returns the n^{th} Catalan number
  Example: seq(Catalan(i),i,1,6) ⇒ {1,2,5,14,42,132}
- ChebyT(n,x) returns the n^{th} Chebyshev polynomial of the first kind $T_n(x)$
  Needs: Gamma
  Example: ChebyT(5,x) ⇒ $16\cdot x^5-20\cdot x^3+5\cdot x$, ChebyT(n,0) ⇒ cos(n·π)/2
- ChebyU(n,x) returns the n^{th} Chebyshev polynomial of the second kind $U_n(x)$
  Needs: Gamma
  Example: ChebyU(6,x) ⇒ $64\cdot x^6-80\cdot x^4+24\cdot x^2-1$, ChebyU(n,1) ⇒ n+1
- Clebsch(j_1,m_1,j_2,m_2,j,m) returns the Clebsch-Gordan coefficient
  Needs: ReplcFac
  Examples: Clebsch(1/2,1/2,1/2,-1/2,1,0) ⇒ √(2)/2,
  Clebsch(1,1,1,1/2,2,3/2) ⇒ √(105)/12
- Cyclotom(n,x) returns the n^{th} cyclotomic polynomial
  Needs: RelPrime
  Example: Cyclotom(6,x) ⇒ x^2-x+1
  Note: The n^{th} cyclotomic polynomial is the minimal polynomial of the primitive n^{th} root of unity and is irreducible over $\mathbb{Z}$ with degree φ(n), where φ(n) is the Euler totient function.
- DblFact(n) returns the double factorial n!!
  Needs: IsCmplxN
  Example: DblFact(14) ⇒ 645120, DblFact(2.1) ⇒ 2.10479089216
- DivSigma(n,k) returns the sum $\sigma_k(n)$ of k^{th} powers of the divisors of n
  Needs: FactList
  Examples: DivSigma(111,0) ⇒ 4, DivSigma(111,1) ⇒ 152,
  DivSigma(111,2) ⇒ 13700
  Note: As special cases, DivSigma(n,0) returns the number of divisors and DivSigma(n,1) returns the sum of the divisors. The divisors in the example above are {1,3,37,111}. A perfect number equals the sum of its proper divisors, i.e. divisors excluding itself.

- Erf(x) returns the error function
  Needs: Erfi
  Examples: Erf(0.5) ⇒ 0.520499877813, Erf(3.7) ⇒ 0.999999832845,
  Erf(***i***) ⇒ 1.6504257588·***i***
  Note: The n^{th} derivative of erf(x) involves the $n-1^{th}$ Hermite polynomial.
- Erfc(x) returns the complementary error function
  Needs: Erf
  Example: Erfc(0.5) ⇒ 0.479500122187
- Erfi(z) returns the imaginary error function erf(***i***·z)/***i***
  Needs: Erf, Hypg1F1
  Examples: Erfi(1) ⇒ 1.6504257588, Erfi(-***i***) ⇒ -0.84270079295·***i***
- Euler(n,x) returns the n^{th} Euler polynomial
  Needs: BernPoly, IsCmplxN
  Example: Euler(4,x) ⇒ $x^4-2\cdot x^3+x$
  Note: The n^{th} Euler number is given by $2^n\cdot$Euler(n,1/2).
- ExpIntE(n,x) returns the exponential integral $E_n(x)$
  Needs: IncGamma
  Examples: ExpIntE(-1,z) ⇒ $((z+1)\cdot e^{-z})/z^2$, ExpIntE(8,0) ⇒ 1/7
- ExpIntEi(x) returns the exponential integral Ei(x)
  Examples: ExpIntEi(2.3) ⇒ 6.15438079133,
  ExpIntEi(98.1) ⇒ 4.14110745294E40,
  ExpIntEi(-5) ⇒ -0.001148295591,
  ExpIntEi(-6) ⇒ -0.000360082452
- FibNum(n) returns the n^{th} Fibonacci number F_n
  Examples: seq(FibNum(i),i,1,6) ⇒ {1,1,2,3,5,8},
  FibNum(2000) ⇒ <418-digit integer> (takes about 29 seconds),
  FibNum(2000.) ⇒ 4.22469633342E417 (takes about 0.21 seconds)
  Note: The ratios of alternate Fibonacci numbers are said to measure the fraction of a turn between successive leaves on the stalk of a plant. Also, the number of petals on flowers are said to be Fibonacci numbers. (See MathWorld entry.)
  FibNum uses a combination of the Binet formula and the Q-matrix method.
- FibPoly(n,x) returns the n^{th} Fibonacci polynomial $F_n(x)$
  Example: FibPoly(7,x) ⇒ $x^6+5\cdot x^4+6\cdot x^2+1$
  Note: Fibonacci polynomials are used in a variety of fields, such as cryptography and number theory.
- FresnelC(x) returns the Fresnel cosine integral C(x) for real x
  Examples: FresnelC(0.78) ⇒ 0.748357308436,
  FresnelC(6.4) ⇒ 0.549604555704, FresnelC(0) ⇒ 0
- FresnelS(x) returns the Fresnel sine integral S(x) for real x
  Needs: FresnelC
  Examples: FresnelS(1.1) ⇒ 0.625062823934,
  FresnelS(13.7) ⇒ 0.479484940352, FresnelS(0) ⇒ 0
- Gamma(z) returns the gamma function Γ(z)
  Needs: DblFact, IsCmplxN
  Examples: Gamma(11/2) ⇒ 945·√(π)/32, Gamma(1/3) ⇒ 2.67893853471,
  Gamma(1+5·***i***) ⇒ -0.001699664494–0.001358519418·***i***
  Note: The gamma function generalizes the factorial function from the natural numbers to the complex plane. However, it is not a unique solution to the

problem of extending x! into complex values of x. Other solutions are of the form f(x) = Γ(x)·g(x), where g(x) is an arbitrary periodic function with period 1.

- Gegenbau(n,m,x) returns the n^{th} Gegenbauer polynomial for the parameter m
  Needs: Gamma, IsCmplxN
  Example: Gegenbau(4,2,x) ⇒ $80\cdot x^4-48\cdot x^2+3$
- Hermite(n,x) returns the n^{th} Hermite polynomial $H_n(x)$
  Example: Hermite(4,x) ⇒ $16\cdot x^4-48\cdot x^2+12$, Hermite(n,0) ⇒ $2^n\cdot\sqrt{(\pi)}/(-n/2-1/2)!$,
  Hermite(100,1) ⇒ -1448706729…, Hermite(10,-3) ⇒ -3093984,
  Hermite(-1,3) ⇒ 0.15863565
  Note: For any given interval (a,b) there is always a Hermite polynomial that has a zero in the interval.
- Hypg1F1(a,c,z) returns Kummer's confluent hypergeometric function $_1F_1(a;c;z)$
  Needs: Gamma
  Examples: $1/\sqrt{(\pi)}$·Hypg1F1(1/2,3/2,-$(1/2)^2$) ⇒ 0.520499877813,
  Hypg1F1(1,2,1) ⇒ $e-1$, Hypg1F1(3,1,x) ⇒ $(x^2/2+2\cdot x+1)\cdot e^x$,
  Hypg1F1(-2,3/2,x) ⇒ $((4\cdot x^2-20\cdot x+15)/15)$, Hypg1F1(a,a,x) ⇒ e^x,
  expand(Hypg1F1(-2,b,x),x) ⇒ $x^2/(b\cdot(b+1))-2\cdot x/b+1$,
  Note: $1/\sqrt{(\pi)}$·Hypg1F1(1/2,3/2,-x^2) gives the error function erf(x).
- Hypg2F1(a,b,c,z) returns Gauss' hypergeometric function $_2F_1(a;b;c;z)$
  Needs: ChebyT, Gamma, Gegenbau, Jacobi, Legendre
  Examples: Hypg2F1(1/2,1/2,3/2,x^2) ⇒ $\sin^{-1}(x)/x$, Hypg2F1(1,1,1,-x) ⇒ 1/(x+1),
  Hypg2F1(1,1,2,1–x) ⇒ ln(x)/(x–1),
  Hypg2F1(-3,3,-5,z) ⇒ $z^3+9\cdot z^2/5+9\cdot z/5+1$,
  Hypg2F1(1/3,1/3,4/3,1/2) ⇒ 1.05285157425,
- HypgPFQ({$a_1,a_2,\ldots,a_p$},{$b_1,b_2,\ldots,b_q$},z) returns the generalized hypergeometric function $_pF_q(a;b;z)$
  Needs: Gamma, Hypg1F1, Hypg2F1
  Examples: HypgPFQ({},{},x) ⇒ e^x, HypgPFQ({},{1/2},-x^2/4) ⇒ cos(x)
- HZeta(n,z) returns the Hurwitz zeta function $\zeta(n,z)=\sum_{k=0}^{\infty}\frac{1}{(k+z)^n}$
  Needs: Psi
  Examples: HZeta(4,1/2) ⇒ $\pi^4/6$, HZeta(0,-x/2) ⇒ (x+1)/2,
  HZeta(2,1/4) ⇒ 8·çatalan+π^2, HZeta(n,3) ⇒ $-2^{-n}+\zeta(n)-1$,
  HZeta(-3,1/4) ⇒ -7/15360, HZeta(4.2,5) ⇒ 0.002471293458
- IncBeta(z,a,b) returns the incomplete beta function $B_z(a,b)$
  Needs: Beta, Pochhamm
  Examples: IncBeta(1,-3/2,2) ⇒ 4/3, IncBeta(5,4,9) ⇒ 1142738125/396
- IncGamma(n,x) returns the incomplete gamma function Γ(n,x)
  Needs: ExpIntEi, Gamma, IsCmplxN, Pochhamm
  Examples: IncGamma(2,z) ⇒ $(z+1)\cdot e^{-z}$, IncGamma(11,1) ⇒ $9864101\cdot e^{-1}$,
  IncGamma(1.2,5.8) ⇒ 0.004435405791, IncGamma(n,1) ⇒ Γ(n,1)
- InvErf(x) returns the inverse $\mathrm{erf}^{-1}(x)$ of the error function
  Example: InvErf(0.013) ⇒ 0.011521459812, Erf(0.011521459812) ⇒ 0.013
- InvErfc(x) returns the inverse $\mathrm{erfc}^{-1}(x)$ of the complementary error function
  Needs: InvErf
  Example: InvErfc(1/2) ⇒ 0.476936276204, Erfc(0.476936276204) ⇒ 0.5

- Jacobi(n,α,β,x) returns the Jacobi polynomial $P_n^{(a,b)}(x)$
Needs: Gamma, IsCmplxN
Examples: Jacobi(3,2,2,1) ⇒ 10, Jacobi(3,2,1,x) ⇒ $(21\cdot x^3+7\cdot x^2-7\cdot x-1)/2$
Note: Jacobi polynomials are the most general orthogonal polynomials for x in the interval [-1, 1].
- KDelta({n_1,n_2,…}) returns 1 if n_1=n_2=…, else returns 0
Examples: KDelta({0,0,1,0}) ⇒ 0, KDelta({0,0,0,0}) ⇒ 1,
KDelta({1,1,1}) ⇒ 1, KDelta({a,1}) ⇒ 0, KDelta({a,a}) ⇒ 1
- Laguerre({n,k},x) returns the associated Laguerre polynomial $L_n^k(x)$
Examples: Laguerre(3,x) ⇒ $(-x^3+9\cdot x^2-18\cdot x+6)/6$, Laguerre(n,0) ⇒ 1,
Laguerre({3,1},x) ⇒ $(-x^3+12\cdot x^2-36\cdot x+24)/6$
Note: If you give an integer n as the first argument, you get the ordinary Laguerre polynomial $L_n(x)$. Laguerre polynomials appear as eigenfunctions of the hydrogen atom in quantum mechanics.
- LambertW(x) returns the principal branch of the Lambert W function
Example: LambertW(2/3·$\boldsymbol{e}^{2/3}$) ⇒ 2/3,
LambertW(0.12) ⇒ 0.10774299816228 [approximately 0.68 seconds]
(solving the equation without an initial guess takes about 8 seconds)
Note: The Lambert W function is defined by x = W(x)·$e^{W(x)}$
- Legendre({n,m},x) returns the associated Legendre function of the first kind $P_n^m(x)$
Needs: Gamma, ReplcStr
Examples: Legendre(3,x) ⇒ $(5\cdot x^3)/2-(3\cdot x)/2$, Legendre(10,0) ⇒ -63/256,
Legendre({3,1},x) ⇒ $(15\cdot x^2\cdot\sqrt{}(1-x^2))/2-(3\cdot\sqrt{}(1-x^2))/2$
Note: The returned function may have an extra factor of $(-1)^m$. If you give an integer n as the first argument, you get the ordinary Legendre polynomial $P_n(x)$. Because of the rather low value of the maximum representable number on these devices, a bug can crop up when trying to evaluate the Legendre polynomials for large n, e.g. Legendre(1000,0) incorrectly returns zero.
- Legendr2({n,m},x) returns the associated Legendre function of the second kind $Q_n^m(x)$
Needs: Gamma, Legendre
Examples: Legendr2(70,0) ⇒ 0, Legendr2(1/2,0) ⇒ -0.847213084794,
Legendr2(0,x) ⇒ ln(x+1)/2–ln(-(x–1))/2
Note: If you give an integer n as the first argument, you get the ordinary Legendre polynomial $Q_n(x)$. Please also see the notes for Legendre.
- LogInt(x) returns the logarithmic integral li(x)
Needs: ExpIntEi
Example: LogInt(0.1) ⇒ -0.032389789593
- LucasNum(n) returns the n^{th} Lucas number
Needs: LucasSeq
Example: LucasNum(21) ⇒ 24476
- LucasSeq(p,q,L_0,L_1,n) returns the n^{th} term L_n of the generalized Lucas sequence with $L_{n+2} = p\cdot L_{n+1}-q\cdot L_n$
Example: LucasSeq(1,-1,0,1,30) ⇒ 832040
Note: L_0 and L_1 are the initial values for the recurrence.

- Moebius(n) returns the Möbius function μ(n)
  Needs: ReplcStr
  Examples: Moebius(12) ⇒ 0, Moebius(12345) ⇒ -1, Moebius(1234567) ⇒ 1
- Multinom(list) returns the number of ways of partitioning N=sum(list) objects into m=dim(list) sets of sizes list[i]
  Example: Multinom({a,b,c}) ⇒ (a+b+c)!/(a!·b!·c!)
- PellNum(n) returns the n^{th} Pell number
  Example: seq(PellNum(i),i,1,6) ⇒ {1,2,5,12,29,70}
  Note: The Pell numbers are the U_n's in the Lucas sequence.
- Pochhamm(a,n) returns the Pochhammer symbol $(a)_n$
  Needs: Gamma
  Example: Pochhamm(x,3) ⇒ x·(x+1)·(x+2)
- Polylog(n,z) returns the polylogarithm $Li_n(z)$
  Needs: FuncEval, Gamma, HZeta, IsCmplxN, Psi, Zeta
  Examples: Polylog(-1,x) ⇒ $x/(x-1)^2$, Polylog(2,-1) ⇒ $-\pi^2/12$,
  Polylog(2,-2) ⇒ -1.43674636688
- Psi(n,z) returns the polygamma function

  $$\psi^{(n)}(z) = \frac{\partial^n \psi(z)}{\partial z^n} = (-1)^{n+1}\, n! \sum_{k=0}^{\infty} \frac{1}{(k+z)^{n+1}}$$

  Needs: BernNum, Zeta
  Examples: Psi(3,1/2) ⇒ π^4,
  Psi(0,***i***) ⇒ 0.094650320623+2.07667404747·***i***,
  Psi(2,6.38) ⇒ -0.028717418078,
  Psi(0,11/3) ⇒ -γ – (3·ln(3))/2 + (π·√(3))/6 + 99/40,
  Psi(13,2) ⇒ $(2048\cdot\pi^{14})/3 - 6227020800$
  Note: Psi(0,z) returns the digamma function $\psi(z) = \psi^{(0)}(z) = \Gamma'(z)/\Gamma(z)$. Psi(n,z) is the $(n+1)^{th}$ logarithmic derivative of the gamma function Γ(z). The γ returned in results is the Euler gamma constant γ ≈ 0.57721566490153, sometimes also called the Euler-Mascheroni constant.
- RegGamma(a,x) returns the regularized incomplete gamma function *Q*(a,x)
  Needs: Erfc, Gamma, IncGamma, Pochhamm
  Examples: RegGamma(0,x) ⇒ 0, RegGamma(-91,x) ⇒ 0,
  RegGamma(3,x) ⇒ $(x^2/2+x+1)\cdot e^{-x}$
- SBesselH(k,n,x) returns the spherical Bessel function $h_n^{(k)}(x)$ for k=1 or k=2
  Examples: SBesselH(1,1,x) ⇒ $-((x+\boldsymbol{i})\cdot e^{\boldsymbol{i}\cdot x})/x^2$,
  SBesselH(2,1,x) ⇒ $-((x+-\boldsymbol{i})\cdot e^{-\boldsymbol{i}\cdot x})/x^2$
- SBesselI(n,x) returns the modified spherical Bessel function $i_n(x)$
  Example: SBesselI(n,x) ⇒ $(e^{-x}\cdot((x-1)\cdot e^{2\cdot x}+x+1))/(2\cdot x^2)$
- SBesselJ(n,x) returns the spherical Bessel function $j_n(x)$
  Example: SBesselJ(1,x) ⇒ $-(x\cdot\cos(x)-\sin(x))/x^2$
- SBesselK(n,x) returns the modified spherical Bessel function $k_n(x)$
  Example: SBesselK(1,x) ⇒ $((x+1)\cdot e^{-x})/x^2$
- SBesselN(n,x) returns the spherical Bessel function $n_n(x)$
  Example: SBesselN(1,x) ⇒ $-\cos(x)/x^2-\sin(x)/x$

- SinInt(x) returns the sine integral function Si(x) for real x
  Examples: SinInt(0.007) ⇒ 0.00699998, SinInt(0.1) ⇒ 0.0999445,
  SinInt(-0.1) ⇒ -0.0999445, SinInt(18.2) ⇒ 1.529091
- SphrHarm(l,m,θ,φ) returns the spherical harmonic $Y_l^m(\theta,\varphi)$
  Needs: Legendre
  Example: SphrHarm(2,0,θ,ϕ) ⇒ $((3\cdot(\cos(\theta))^2-1)\cdot\sqrt{(5)})/(4\cdot\sqrt{(\pi)})$
- StirNum1(n,k) returns the Stirling number of the first kind
  Needs: StirNum2
  Example: StirNum1(5,3) ⇒ 35
  Note: $(-1)^{n-k}$·StirNum1(n,k) gives the number of permutations of n items that have exactly k cycles.
- StirNum2(n,k) returns the Stirling cycle number
  Example: StirNum2(11,4) ⇒ 145750
  Note: The Stirling cycle number gives the number of ways one can partition an n-element set into k non-empty subsets. It also occurs in summing powers of integers.
- StruveH(n,z) returns the Struve function $H_n(z)$
  Needs: BesselJ, BesselY, Pochhamm
  Examples: StruveH(1/2,0) ⇒ 0,
  StruveH(1/2,x) ⇒ $\sqrt{(2)}\cdot(\sqrt{(\pi)}\cdot\sqrt{(x)})-\sqrt{(2)}\cdot\cos(x)/(\sqrt{(\pi)}\cdot\sqrt{(x)})$
  Note: This function currently works mainly only for half-integer n.
- Subfact(n) returns the number of permutations of n objects which leave none of them in the same place
  Needs: IsCmplxN
  Example: Subfact(20) ⇒ 895014631192902121
- Totient(n) returns the Euler φ(n) function, which is the number of positive integers less than n that are relatively prime to n
  Needs: FactList
  Examples: Totient(143055667) ⇒ 142782138, Totient(17^{100}) ⇒ 1043669997…
- Wigner3j(j_1,m_1,j_2,m_2,j_3,m_3) returns the Wigner 3-j symbol $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$
  Needs: Clebsch
  Example: Wigner3j(2,1,5/2,-3/2,1/2,1/2) ⇒ $\sqrt{(30)}/15$
- Zeta(z) returns the Riemann zeta function ζ(z)
  Examples: Zeta(2) ⇒ $\pi^2/6$, Zeta(1–***i***) ⇒ 0.00330022–0.418155·***i***,
  Zeta(1/2+7·***i***) ⇒ 1.02143 + 0.396189·***i***, Zeta(k) ⇒ ζ(k)
  Notes: The Riemann zeta function has a central role in number theory. It also appears in integration and summation. For small approximate values, Zeta seems to have an error of 10^{-9} or less. For large negative values, the error may be significant due to the limited precision on these calculators, even though most of the digits returned are correct. For example, the error for Zeta(-98.4) is approximately $4.17\cdot10^{62}$, and the error for Zeta(-251.189) is approximately $3.99\cdot10^{281}$. Zeta seems quite accurate for large positive values, where the zeta function approaches 1. Two other commonly used sums of reciprocal powers of integers can be related to the zeta function:

$$\eta(s)=\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k^s}=(1-2^{1-s})\cdot\zeta(s), s>1, \eta(1)=\ln(2)\ ,$$

$$\lambda(s)=\sum_{k=0}^{\infty}\frac{1}{(2\cdot k+1)^s}=(1-2^{-s})\cdot\zeta(s), s>1$$

- ZetaPrim(z) returns the first derivative of the zeta function ζ'(z)
  Needs: BernNum
  Examples: ZetaPrim(2) ⇒ -0.937548, ZetaPrim(***i***) ⇒ 0.0834062–0.506847·***i***,
  ZetaPrim(-1) ⇒ 1/12–ln(glaisher),
  ZetaPrim(-1.0) ⇒ -0.165421143696,
  ZetaPrim(k) ⇒ ζ'(k)
  Note: The symbol ‘glaisher’ returned in results represents the Glaisher-Kinkelin constant $A \approx 1.2824271291006$, and satisfies the equation $\ln(A) = 1/12 - \zeta'(-1)$. It appears in sums and integrals of some special functions.

- Statistics
  - CDF(StatDist("DistributionName",parameters), x) returns the distribution probability between -∞ and x for the given distribution and parameters
    Needs: Erf, Erfc, Gamma, IncGamma
    Examples: CDF(statdist("Cauchy",{1,2}),0.7) ⇒ 0.452606857723,
    CDF(statdist("Cauchy",{1,2}),x) ⇒ $\tan^{-1}$((x–1)/2)/π+1/2,
    CDF(statdist("Chi^2",{0.3}),0.11) ⇒ 0.688767224365,
    CDF(statdist("Exponential",{0.83}),0.04) ⇒ 0.032654928773,
    CDF(statdist("Gamma",{1/2,1/4}),2) ⇒ 0.999936657516,
    CDF(statdist("Gamma",{1/2,1/4}),x) ⇒ 1–Γ(1/2,4·x)/√(π),
    CDF(statdist("Hypergeometric",{2,3,7}),1/3) ⇒ 2/7,
    CDF(statdist("Laplace",{0.96,0.32}),0.5) ⇒ 0.118760409548,
    CDF(statdist("Logistic",{2.5,1}),4) ⇒ 0.817574476194,
    CDF(statdist("Normal",{0,1}),0.2) ⇒ 0.579259709439,
    CDF(statdist("Poisson",{4/3}),9/4) ⇒ $29 \cdot e^{-4/3}/9$
  - CentrMom(list,r) returns the r^{th} central moment of list
    Example: CentrMom({a,b},2) ⇒ $(a–b)^2/4$
  - Freq(list) returns the distinct elements of list with their frequencies
    Needs: Sort
    Example: Freq({a,b,b,c,a}) ⇒ [[a,b,c][2,2,1]]
  - GeomMean(list) returns the geometric mean of list
    Example: GeomMean({a,b}) ⇒ √(a·b)
  - HarmMean(list) returns the harmonic mean of list
    Example: HarmMean({a,b}) ⇒ 2·a·b/(a+b)
  - IntQuant(list,q) returns the q^{th} interpolated quantile of a probability distribution
    Needs: Sort
    Example: IntQuant({a,b,c,d},0.75) ⇒ c
    Note: The quantile is obtained from a linear interpolation of list. IntQuant can be used to find quartiles, deciles, etc.
  - Kurtosis(list) returns the Pearson kurtosis coefficient for list
    Needs: CentrMom, PStdDev
    Example: Kurtosis({a,b,c}) ⇒ 3/2

- MeanDev(list) returns the mean absolute deviation about the mean for list
  Example: MeanDev({a,b}) ⇒ |a–b|/2
- MedDev(list) returns the median absolute deviation about the median for list
  Example: MedDev({2,3}) ⇒ 1/2
- Mode(list) returns the most frequent element(s) of list
  Needs: Freq
  Example: Mode({a,b,b,c,a}) ⇒ {a,b}
- MovAvg(list,n) smooths list using an n-point moving average
  Example: MovAvg({a,b,c},2) ⇒ {(a+b)/2,(b+c)/2}
- MovMed(list,n) smooths list using a span-n moving median
  Example: MovMed({3,7,1,0,5},3) ⇒ {3,1,1}
- PDF(StatDist("DistributionName",parameters), x) returns the probability density function at x for the given distribution and parameters
  Needs: Gamma
  Examples: PDF(statdist("Cauchy",{0.22,0.81}),0.75) ⇒ 0.275166497128,
  PDF(statdist("Chi^2",{1/2}),0.4) ⇒ 0.377535275007,
  PDF(statdist("Exponential",{5}),1/4) ⇒ $5\cdot e^{-5/4}$,
  PDF(statdist("Exponential",{a}),x) ⇒ $a\cdot e^{-a\cdot x}$,
  PDF(statdist("FRatio",{0.03,0.14}),0.9) ⇒ 0.013213046749,
  PDF(statdist("Gamma",{0.3,0.72}),0.45) ⇒ 0.178995053492,
  PDF(statdist("Hypergeometric",{4,5,8}),3) ⇒ 3/7,
  PDF(statdist("Laplace",{0.61,0.05}),0.37) ⇒ 0.08229747049,
  PDF(statdist("Logistic",{0.14,0.8}),5.7) ⇒ 0.001195999788,
  PDF(statdist("Normal",{0,1}),x) ⇒ $\sqrt{(2)}\cdot e^{-x^2/2}/(2\cdot\sqrt{(\pi)})$,
  PDF(statdist("Normal",{0.2,3}),1/3) ⇒ 0.132849485949,
  PDF(statdist("Poisson",{9}),2) ⇒ $81\cdot e^{-9}/2$,
  PDF(statdist("StudentT",{0.66}),0.5) ⇒ 0.222461859264,
  PDF(statdist("StudentT",{3}),x) ⇒ $6\cdot\sqrt{(3)}/(\pi\cdot(x^2+3)^2)$
- PStdDev(list) returns the most likely estimate for the population standard deviation of list
  Example: stdDev({a,b}) ⇒ |a–b|·√(2)/2, PStdDev({a,b}) ⇒ |a–b|/2
- Quantile(list,q) returns the q^{th} quantile
  Needs: Sort
  Example: Quantile({a,b},0.6) ⇒ b
- Random(StatDist("DistributionName",parameters)) returns a pseudorandom number using the given statistical distribution
  Examples: RandSeed 0:Random(statdist("Bernoulli",{14})) ⇒ 1,
  RandSeed 0:Random(statdist("Beta",{2.4,-0.11})) ⇒ 0.304489,
  RandSeed 0:Random(statdist("Binomial",{4,2/3})) ⇒ 2,
  RandSeed 0:Random(statdist("Cauchy",{-3/2,4})) ⇒ 20.8374,
  RandSeed 0:Random(statdist("Chi^2",{1})) ⇒ 0.0603222,
  RandSeed 0:Random(statdist("DiscreteUniform",{-21,37})) ⇒ 34,
  RandSeed 0:Random(statdist("Gamma",{0.5,0.5})) ⇒ 0.0150806,
  RandSeed 0:Random(statdist("Geometric",{0.027})) ⇒ 2,
  RandSeed 0:Random(statdist("Hypergeometric",{5,3,10})) ⇒ 3,
  RandSeed 0:Random(statdist("Logistic",{0.5,0.5})) ⇒ 1.90859,
  RandSeed 0:Random(statdist("LogNormal",{0.5,0.5})) ⇒ 1.90195,

RandSeed 0:Random(statdist("Normal",{1,0.4})) ⇒ 2.01958,
RandSeed 0:Random(statdist("Poisson",{3.2})) ⇒ 6,
RandSeed 0:Random(statdist("Uniform",{})) ⇒ 0.943597,
RandSeed 0:Random(statdist("Uniform",{-13,21.2})) ⇒ 19.2710

- RMS(list) returns the root-mean-square of list
  Example: RMS({a,b,c}) ⇒ $\sqrt{}(a^2+b^2+c^2)/3$
- Skewness(list) returns the skewness coefficient for list
  Needs: CentrMom, PStdDev
  Examples: Skewness({a,b}) ⇒ 0, Skewness({1,2,2.1}) ⇒ -0.685667537489
- TrimMean(list,α) returns the α-trimmed mean
  Needs: Sort
  Examples: TrimMean({a,b,c},1/10) ⇒ (7·a+10·b+7·c)/24,
  TrimMean({2,8,4},1/10) ⇒ 55/12
  Note: The trimmed mean is equal to the arithmetic mean when α=0.

- Strings
  - nChars(str,chr) returns the number of occurrences of chr in the string str
    Examples: nChars("mathtools","o") ⇒ 2, nChars("mathematics","ma") ⇒ 2
  - RepChars(chars,n) returns a string with chars repeated n times
    Example: RepChars("math",3) ⇒ "mathmathmath"
  - ReplcStr(str1,str2,str3) replaces occurrences of the string str2 with str3 in str1
    Example: ReplcStr("aububu","ub","qq") ⇒ "aqqqqu"
  - Str2Chrs(str) returns a list of the characters in the string str
    Example: Str2Chrs("a bc") ⇒ {"a"," ","b","c"}
  - StrCount(str,substr) is an alternative function that returns the number of occurrences of substr in str
    Example: StrCount("more & more","or") ⇒ 2
- Summation and Series
  - Aitken(list) accelerates series with Aitken's δ^2 method and returns {sn, delta}
    Example: Aitken(seq($(-1)^k$/(2·k+1),k,0,10)) ⇒ {0.785459904732, 0.00014619883}
    Notes: The first few terms of the series are given in 'list'. For example, to accelerate the series ∑(x(i),i,a,b), one could use seq(x(i),i,a,a+10). 'sn' is the last accelerated sum, and 'delta' is the difference between the last two sums.
  - EMSum(x(i),i,a,expstart,b,exporder) returns an asymptotic (Euler-Maclaurin) expansion for the sum ∑(x(i),i,a,b)
    Examples: ∑(1/(2^k),k,0,∞) ⇒ 2,
    EMSum(1/(2^k),k,0,1,∞,5.) ⇒ 2.0000000000047,
    ∑($e^{1/x\wedge 2}$–1,x,1,∞) ⇒ ∑($e^{1/x\wedge 2}$–1,x,1,∞) [unevaluated],
    EMSum($e^{1/x\wedge 2}$–1,x,1,5,∞,4.) ⇒ 2.40744653864
    Note: expstart is the start of the expansion and exporder is the expansion order. The formula was taken from the "Handbook of Mathematical Functions", by Abramowitz and Stegun.
  - Gosper(xpr,var,low,up) does hypergeometric summation ∑(xpr,var,low,up)
    Examples: Gosper(k·k!,k,0,n–1) ⇒ n!–1,
    Gosper(4^k/nCr(2·k,k),k,0,n–1) ⇒ (2·n–1)·$(n!)^2 \cdot 4^n$/(3·(2·n)!)+1/3
    Notes: Gosper returns "No solution found" if the requested sum cannot be summed as a hypergeometric series, and gives an "Argument error" if the series is not hypergeometric or xpr(n+1)/xpr(n) contains variable powers of n. It will display a "Questionable accuracy" warning if computations are done using

approximations, because due to rounding errors, you can end up with a solution where there is none. Gosper returns an "Overflow" if one of the polynomials used in the computations has an excessive degree (256 or higher), and gives an error "Too many undefined variables" if the difference between a root of the numerator and a root of the denominator is not constant.

- PsiSum(xpr,x) does the sum ∑(xpr,x,1,∞) in terms of polygamma functions
  Needs: Coef, Degree, FreeQ, IsApprox, IsCmplxN, Psi, Terms
  Example: PsiSum(1/(n^2·(8n+1)^2),n) ⇒ ψ(1,9/8)+ψ(1,1)–16·ψ(0,9/8)+16·ψ(0,1),
  Psi(1,9/8)+Psi(1,1)–16·Psi(0,9/8)+16·Psi(0,1) ⇒ 0.013499486145
  Note: PsiSum(xpr,{x,lowlimit}) does the sum ∑(xpr,x,lowlimit,∞).
- Sum2Rec(xpr,var) does indefinite summation (from var=1 to var=N)
  Needs: Coef, Degree, PolyGCD, Resultnt
  Examples: Sum2Rec(k·k!,k) ⇒
  {n=f(n)·(n+1)–f(n–1), f(1)=1/2, s(n)=n!·f(n)·(n+1)},
  Sum2Rec(2^k/k!,k) ⇒ false (the result involves special functions)
  Note: Sum2Rec implements the Gosper algorithm and returns a recurrence equation to be solved. The result is given in terms of the variable n. Many recurrence equations can be solved using z-transforms (using Glenn Fisher's z-transform routines, for example). The solution for f(n) needs to be substituted into the last element of the list to obtain the final result. If the indefinite sum cannot be computed in terms of elementary functions, Sum2Rec returns false.
- Wynn(x[i],i,k,n) returns an nth-order Wynn convergence acceleration formula
  Example: factor(Wynn(x[i],i,k,2)) ⇒
  (x[k]·x[k+2]–x[k+1]^2)/(x[k+2]–2·x[k+1]+x[k])

- Vector/Tensor Analysis
  - Curl({E^1,E^2,E^3},{x^1,x^2,x^3},coordstr|{g^{11},g^{22},g^{33}}) returns the curl of the vector field E for arbitrary 3-coordinates x^a with the coordinate system given as a string or by the metric g^{ab}
    Needs: Metric
    Example: Curl({x^2,x+y,x–y+z},{x,y,z},"Rectangular") ⇒ {-1,-1,1}
    Notes: The third argument specifies the coordinate system. Any orthogonal coordinate system can be used. The name of a common coordinate system can be used ("Rectangular", "Cylindrical"|"CircularCylinder", "EllipticCylinder", "ParabolicCylinder", "Spherical", "ProlateSpheroidal", "OblateSpheroidal", "Parabolic", "Conical"|"Conic", "Ellipsoidal", "Paraboloidal") or the diagonal elements of the metric can be explicitly provided.
  - DirDeriv(f,{x^1,x^2,x^3},coordstr|{g^{11},g^{22},g^{33}},u) returns the directional derivative of f in the direction of the vector u
    Needs: Gradient
    Example: factor(DirDeriv(x^2·e^y·sin(z),{x,y,z},"Rectangular",{2,3,-4}),z)
    ⇒ √(29)/29·(sin(z)·(3·x+4)–4·cos(z)·x)·x·e^y
    Note: The vector u is given as a list {u^1,u^2,u^3}. The third argument specifies the coordinate system. Any orthogonal coordinate system can be used. The name of a common coordinate system can be used (see Curl) or the diagonal elements of the metric can be explicitly provided.
  - Div({E^1,E^2,E^3},{x^1,x^2,x^3},coordstr|{g^{11},g^{22},g^{33}}) returns the divergence of the vector field E for arbitrary 3-coordinates x^a with the coordinate system given as a string or by the metric g^{ab}
    Needs: Metric

Example: Div({x^2,x+y,x–y+z},{x,y,z},"Rectangular") ⇒ 2·x+2
Note: The third argument specifies the coordinate system. Any orthogonal coordinate system can be used. The name of a common coordinate system can be used (see Curl) or the diagonal elements of the metric can be explicitly provided.

- Gradient(ϕ,{x^1,x^2,x^3},coordstr|{g^{11},g^{22},g^{33}}) returns the gradient of the scalar field ϕ for arbitrary 3-coordinates x^a with the coordinate system given as a string or by the metric g^{ab}
Needs: Metric
Example: Gradient($e^{x\cdot y+z}$,{x,y,z},"Rectangular") ⇒ {$e^{x\cdot y+z}$·y, x·$e^{x\cdot y+z}$, $e^{x\cdot y+z}$}
Note: The third argument specifies the coordinate system. Any orthogonal coordinate system can be used. The name of a common coordinate system can be used (see Curl) or the diagonal elements of the metric can be explicitly provided.
- Hessian(f(a[i]),a) returns the Hessian matrix
Example: Hessian(x·y^2 + x^3,{x,y}) ⇒ [[6·x,2·y][2·y,2·x]]
Note: 'a' must be a list of variables.
- Jacobian(funcs,vars) returns the Jacobian matrix of functions funcs w.r.t. vars
Example: Jacobian({x+y^2,x–z,y·e^z},{x,y,z}) ⇒ [[1,2·y,0][1,0,-1][0,e^z,y·e^z]]
- Laplacn(ϕ,{x^1,x^2,x^3},coordstr|{g^{11},g^{22},g^{33}}) returns the Laplacian of the scalar or vector field ϕ for arbitrary 3-coordinates x^a with the coordinate system given as a string or by the metric g^{ab}
Needs: Metric
Example: factor(Laplacn($e^{x\cdot y+z}$,{x,y,z},"Rectangular")) ⇒ (x^2+y^2+1)·$e^{x\cdot y+z}$
Notes: For a vector field, input ϕ as a list. The third argument specifies the coordinate system. Any orthogonal coordinate system can be used. The name of a common coordinate system can be used (see Curl) or the diagonal elements of the metric can be explicitly provided.
- Metric(coordstr,vars) returns the diagonal elements of the metric for the coordinate system specified by coordstr, with the variable list given in vars
Example: Metric("Spherical",{r,θ,ϕ}) ⇒ {1, r^2, $\sin(\theta)^2\cdot r^2$}
Note: The coordinate system names are given in the help entry for Curl. They are case-insensitive, so that you can use "Cylindrical" or "cyLinDRical".

- Other functions
  - AGM(a,b) returns the arithmetic-geometric mean of a and b, alongwith the number of iterations
  Needs: IsCmplxN
  Example: AGM(1,1.1) ⇒ {1.0494043395823,3}
  Note: The AGM can be used to compute some special functions. It can also be used to compute π via the Brent-Salamin algorithm.
  - Apply(funcstr,xpr) applies expr(funcstr) to xpr
  Needs: ReplcStr
  Example: Apply("ln(#)+1",2·a) ⇒ 1+ln(2·a)
  - BaseConv(numstr,b1,b2) converts a number string from base b1 to base b2
  Examples: BaseConv("11",10,2) ⇒ "1011", BaseConv("1011",2,10) ⇒ "11",
  BaseConv("0.1",10,2) ⇒ "0.0001100110011001100110011…"
  - BSOptVal(sprice,eprice,vol,intrate,time) computes the Black-Scholes value of a call option
  Needs: Erf

Example: BSOptVal(20,25,3.1,0.027,3/4) ⇒ 16.0335379358
Note: 'sprice' is the stock trading price, 'eprice' is the exercise price, 'vol' is the stock volatility (the standard deviation of the annualized continuously compounded rate of return), 'intrate' is the risk-free interest rate (the annualized continuously compounded rate on a safe asset with the same maturity as the stock option), and 'time' is the time to maturity of the call option (in years). A stock option is a contract that gives you the right to buy/sell a stock at the pre-specified exercise price (this is called "exercising the option") until the option expires. There are two kinds of stock options: American (you can exercise the option any time until the option expires) and European (you can exercise the option only at the expiration date). Instead of exercising the option, you can sell it to someone else before the option expires. Note that the Black-Scholes model makes several assumptions about the underlying securities and their behavior.

- Conic(f(x,y)) returns a description of the conic given by f(x,y)
  Needs: FreeQ
  Examples: Conic(x^2-y^2=0) ⇒ "Intersecting lines", Conic(x^2+y^2=4) ⇒ "Ellipse"
  Note: The function must be in terms of x and y.
- Curvatur(f,x) returns the extrinsic curvature κ of the function f(x)
  Examples: Curvatur(x^2-y^2=1,{x,y}) ⇒ $-(x^2-y^2)/(x^2+y^2)^{3/2}$,
  Curvatur(y=x^2,{x,y}) ⇒ $-2/(4\cdot x^2+1)^{3/2}$,
  Curvatur({t,t^2},t) ⇒ $2/(4\cdot t^2+1)^{3/2}$,
  Curvatur({t,t^2,t},t) ⇒ $1/(2\cdot t^2+1)^{3/2}$
  Note: f can be a 2D scalar function or a 2D/3D vector-valued function.
- Date() returns the current date and time on Hardware 2 units
  Example: Date() ⇒ "Thu 03/06/03 10:45 PM"
  Note: The date and time are shown using the formats specified in the Clock app.
- FactList(xpr) returns a matrix of the factors of xpr, with multiplicities
  Needs: ReplcStr
  Examples: FactList(123456) ⇒ [[2,6][3,1][643,1]],
  FactList(x^2+1) ⇒ [[x–***i***,1][x+***i***,1]],
  FactList($(x+\boldsymbol{i})^2$) ⇒ [[x+***i***, 2]],
  FactList($(x+\boldsymbol{i})^2/2$) ⇒ "Error: cannot handle division at present"
  Note: If xpr is an integer, FactList returns its prime factors with multiplicities. If xpr is provided as a string, the factors are found without any evaluation.
- FixPoint(funcstr,xpr) applies expr(funcstr) repeatedly to xpr until the result doesn't change
  Needs: Apply
  Example: FixPoint("cos(#)",0.) ⇒ 0.739085133215
- FnPeriod(f(x),x) returns the period of a univariate function f(x)
  Needs: IsCmplxN, LLcm, RmDup, Select
  Examples: FnPeriod(sin(3·x+2),x) ⇒ 2·π/3, FnPeriod(sin(a·x+b),x) ⇒ ∞,
  FnPeriod(x^2,x) ⇒ ∞, FnPeriod(sin(sin(x)),x) ⇒ 2·π,
  FnPeriod(sin(x^2),x) ⇒ ∞, FnPeriod(ln(x),x) ⇒ ∞, FnPeriod(a,x) ⇒ 0
- FuncEval(xpr,vals) evaluates xpr with constraints given by the list vals
  Needs: list2eqn
  Examples: FuncEval(x,{x=y,y=z}) ⇒ z, FuncEval(x^2, x=3) ⇒ 9
  Note: FuncEval(xpr,{constr1,constr2,…}) is equivalent to 'xpr|constr1 and constr2 and …'

- GetArg(xpr,head) returns the argument of the function with the given head
  Examples: GetArg(ln(a·tan(x+b)),"tan(") ⇒ b+x,
  GetArg(sin(x+a),"sin(") ⇒ a+x, GetArg(cos(x),"ln(") ⇒ cos(x)
  Note: head = part(expression,0). If head is not present, xpr is returned.
- GrayCode(m) returns the reflected Gray code ordering of the integers $\{1,\ldots,2^m\}$
  Needs: Nest, Reverse
  Examples: GrayCode(2) ⇒ {0,1,3,2}, GrayCode(2) ⇒ {0,1,3,2,6,7,5,4}
  Note: Two integers are considered adjacent if they differ by a power of two.
- IntExp(n,b) returns the largest integer k such that b^k divides n
  Examples: IntExp(2,2) ⇒ 1, IntExp(2,7) ⇒ 0, IntExp(5,2) ⇒ 0,
  IntExp(100,2) ⇒ 2, IntExp(71!,3) ⇒ 32
  Note: IntExp(n,b) gives the number of trailing zeros in the base-b representation of n.
- IsApprox(x) returns true if x is an approximate number, else returns false
  Examples: IsApprox(π) ⇒ false, IsApprox(approx(π)) ⇒ true,
  IsApprox(1–√(2)·***i***) ⇒ false, IsApprox(1–√(2.0)·***i***) ⇒ true
- IsCmplxN(xpr) returns true if xpr is real or complex, else returns false
  Examples: IsCmplxN(π) ⇒ true, IsCmplxN(1–***i***) ⇒ true, IsCmplxN(x) ⇒ false,
  IsCmplxN({1,***i***,7.3}) ⇒ true, IsCmplxN({*e*,2^k}) ⇒ false
  Note: xpr can be a list or matrix, besides a scalar expression.
- IsMarkov(mat) returns true if mat is a valid Markov matrix, else false
  Examples: IsMarkov([[a,b][c,d]]) ⇒ false, IsMarkov([[1,2][3,4]]) ⇒ true
- Laplace(xpr,var1,var2) returns the Laplace transform of xpr with the result given in terms of var2
  Needs: FreeQ, RepChars
  Examples: Laplace(e^t,t,s) ⇒ 1/(s–1), Laplace(sin(t),t,s) ⇒ 1/(s^2+1),
  Laplace(*d*(f(t),t,2),t,s) ⇒ f(s)·s^2–f(0)·s–f′(0), which means
  F(s)·s^2–F(0)·s–F′(0)
- LeafCnt(xpr) returns the number of leaves in xpr
  Examples: LeafCnt(x^2) ⇒ 2, LeafCnt(ln(a·x+b)) ⇒ 3, LeafCnt(1/4) ⇒ 1
  Note: The leaves of an expression are the lowest nodes in its tree structure.
- LnFrcLim(f1,f2,x) is a subroutine for the MRV limit algorithm
  Needs: MatchQ
  Note: This function is experimental.
- LogBase(b,xpr) returns the base-b logarithm of xpr
  Example: LogBase(2,1.1) ⇒ 0.137504
  Note: 'b' and 'xpr' need not be numbers.
- Map(funcstr,xpr) maps expr(funcstr) over the head of the expression xpr
  Needs: ReplcStr
  Example: Map("sin(#)",f(1,2,3)) ⇒ {sin(1),sin(2),sin(3)}
  Note: Use "#" in the string to denote the parts of xpr.
- MarkovEP(mat) returns the stationary probabilities from a Markov matrix
  Example: MarkovEP([[1/2,1/4,1/4][1/4,1/2,1/4][1/4,1/4,1/2]]) ⇒ {1/3,1/3,1/3}
  Note: MarkovEP returns a list representing a vector p such that mat·p = p.
- MRVList(xpr,x) returns a list of the most rapidly varying subexpressions of xpr
  Needs: FreeQ, LnFrcLim, MemberQ, Union
  Note: MRVList is an experimental subroutine for the MRV limit algorithm.

- NaturalQ(x) returns true if x is a natural number, else returns false
  Examples: NaturalQ(72581) ⇒ true, NaturalQ(0) ⇒ true, NaturalQ(-1) ⇒ false,
  NaturalQ(z) ⇒ false
- Nest(funcstr,xpr,n) applies expr(funcstr) to xpr n times
  Needs: Apply
  Example: Nest("#^#",x,2) ⇒ x^x^x^x
- NestList(funcstr,xpr,n) applies expr(funcstr) to xpr n times and returns a list showing each application
  Needs: Apply
  Example: NestList("mod(2·#,1)",approx(1/3),10) ⇒ {0.33333333333333,
  0.666666666666,0.333333333332,0.666666666664,0.333333333328,
  0.666666666656,0.333333333312,0.666666666624,0.333333333248,
  0.666666666496,.333333332992}
- NIrrPoly(q,d) returns the number of degree d irreducible polynomials over the finite field GF(q)
  Needs: Moebius
  Example: seq(NIrrPoly(2,i),i,1,10) ⇒ {2,1,2,3,6,9,18,30,56,99}
- nResidue(f(x),x,x_0) returns a numeric value for the residue of f(x) at x = x_0
  Needs: Chop
  Examples: nResidue(1/x,x,0) ⇒ 1., nResidue(1/sin(x),x,0) ⇒ 1.000008333
  Note: nResidue assumes that there are no poles within 0.01 of x_0.
- ParCurve(f,x,y) returns a parametrization (x(t),y(t)) of a curve f(x,y)=0
  Example: ParCurve($x^3+x^2-y^2$,x,y) ⇒ x = t^2-1 and y = $(t^2-1)\cdot t$
  Note: The parametrization may not be unique. For parametric graphing, you may need to modify window variables such as tmin and tmax.
- PolyArea(vertxmat) returns the area of the simple polygon
  Example: PolyArea([[a,b][c,d][e,f]]) ⇒ (a·(d–f)–b·(c–e)+c·f–d·e)/2
  This is the area of a triangle (on a plane) with the given vertices.
  Note: The vertices are given in rectangular coordinates. The vertex matrix is given as {{x_1,y_1},{x_2,y_2},...}.
- PowerExp(xpr) expands products and powers of products in xpr
  Needs: VarList
  Example: expand(ln($x^m\cdot y^n$)) ⇒ ln($x^m\cdot y^n$) [not completely expanded],
  PowerExp(ln($x^m\cdot y^n$)) ⇒ m·ln(x)+n·ln(y)
- PtInPgon(vertxmat,{x,y}) returns true if point {x,y} is in the polygon, else false
  Examples: PtInPgon({{0,0},{0,1},{2,1},{2,0},{0,0}},{1/2,1/2}) ⇒ true,
  PtInPgon({{0,0},{0,1},{2,1},{2,0},{0,0}},{0,0}) ⇒ true,
  PtInPgon({{0,0},{0,1},{2,1},{2,0},{0,0}},{1,2}) ⇒ false
  Note: The vertices are given in rectangular coordinates. The vertex matrix is given as {{x_1,y_1},{x_2,y_2},...}. The last vertex in vertxmat must be the same as the first (closed polygon).
- RatNumQ(xpr) returns true if xpr is a rational number, else returns false
  Examples: RatNumQ(13) ⇒ true, RatNumQ(1/7) ⇒ true,
  RatNumQ(1/*e*) ⇒ false, RatNumQ(√(2)) ⇒ false,
  RatNumQ(x) ⇒ false
- ReplcFac(xpr) returns xpr with all occurrences of _! replaced by gamma(_+1)
  Needs: Gamma
  Example: ReplcFac(√(3·(3/2)!)/√((5/2)!)) ⇒ √(30)/5
  Note: xpr should be a number.

- RotMat(axis,θ) returns the rotation matrix for a rotation by angle θ around the given axis
  Example: RotMat({1,0,1},π/3) ⇒
  [[3/4,√(6)/4,1/4][-√(6)/4,1/2,√(6)/4][1/4,-√(6)/4,3/4]]
- sMod(num,n) returns mod(num,n) where num is a signed number
  Example: mod(-7,3) ⇒ 2, sMod(-7,3) ⇒ -1
- Torsion(v,t) returns the torsion of a parametrized vector function v(t)
  Example: Torsion({sin(s),ln(s),s},s) ⇒ $-s\cdot(s\cdot\cos(s)+2\cdot\sin(s))/(s^4\cdot(\sin(s))^2+1)$
- UNormVec(v,t) returns the unit normal vector of a parametrized curve v(t)
  Needs: Curvatur, UTanVec
  Examples: UNormVec($\{t, t^2, t^3\}$,t) ⇒ $\{-t\cdot(9\cdot t^2+2)/\sqrt{(9\cdot t^4+9\cdot t^2+1)}$,
  $-(9\cdot t^4-1)/\sqrt{(9\cdot t^4+9\cdot t^2+1)}, 3\cdot t\cdot(2\cdot t^2+1)/\sqrt{(9\cdot t^4+9\cdot t^2+1)}\}$
- UTanVec(v,t) returns the unit tangent vector for a parametrized curve v(t)
  Example: UTanVec($\{t, t^2, e^t\}$,t) ⇒
  $\{1/\sqrt{(e^{2\cdot t}+4\cdot t^2+1)}, 2\cdot t/\sqrt{(e^{2\cdot t}+4\cdot t^2+1)}, e^t/\sqrt{(e^{2\cdot t}+4\cdot t^2+1)}\}$
- VarList(xpr) returns a list of all variables in the expression xpr
  Needs: IsCmplxN, RmDup
  Example: VarList(sin(a·tan(b^f(x))+c·_m)) ⇒ {b,x,c,a}
  (f is considered to be an operator, and units are not returned)

◆ C Utilities[‡]

- FindHead(expr|exprstring,headexpr) returns a list of subexpressions with the specified head
- HelpMsg(str) displays the message given in the string str in the status bar
- IntDig(integer[,base]) returns the digits of the integer in the given base
- RealDig(realnum[,base]) returns the digits of the real number in the given base
- VarList(xpr) returns a list of variables in the expression xpr, just like the TI-Basic function VarList under Miscellaneous→OtherFunctions

◆ MathTools Flash app

- AllApprx(xpr) returns true if every elements of xpr is an approximate number (real or complex), else returns false
  Examples: AllApprx(2/3) ⇒ false, AllApprx(2/3.) ⇒ true,
  AllApprx({3.4,1}) ⇒ false, AllApprx({3.4,1+***i***}) ⇒ true,
  AllApprx([[2.74,8.613][x,1.1]]) ⇒ false
- AppsInfo() returns information about installed applications

  Example: AppsInfo() ⇒ $$\begin{pmatrix} \text{"App name"} & \text{"Token name"} & \text{"Size"} & \ldots \\ \text{"Desk Top"} & \text{""} & 4 & \ldots \\ \text{"Home"} & \text{""} & 4 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

  Note: AppsInfo returns a matrix with one row per application. Of course, the result depends on what apps you have installed and your AMS version (e.g. the DeskTop app was introduced in AMS 2.07). Some apps may have blank token names because they are not intended to be called from TI-Basic. The size of built-in apps is currently shown as 4 bytes.

[‡] To make non-Flash C programs behave like TI-Basic functions, you will need to follow these steps:
1. Download and install Kevin Kofler's h220xtsr (*if* you have a Hardware 2 or 3 calculator), then
2. Download and install IPR (this disables the "invalid program reference" error you get otherwise).

- ByteCnt(xpr) returns the number of bytes needed for xpr on the estack
  Examples: ByteCnt(2+2) ⇒ 7, ByteCnt(4) ⇒ 3, ByteCnt(x+x) ⇒ 3,
  ByteCnt(2·x) ⇒ 5, ByteCnt(x^2) ⇒ 5, ByteCnt(a·x^2) ⇒ 7,
  ByteCnt(sin(x)) ⇒ 2
  Note: As shown above, ByteCnt does not evaluate xpr.
- ConstQ(xpr) returns true if xpr is a constant expression, free of any variables, else returns false
  Examples: ConstQ(π) ⇒ true, ConstQ(x^3) ⇒ false, ConstQ(@n1) ⇒ false,
  ConstQ("str") ⇒ true
  Note: Arbitrary real (@*k*) and integer (@n*k*) constants are considered variables.
- FFT(list[,direction[,convention]]) computes the radix-2 FFT (or inverse FFT) of a list or matrix, using the signal-processing convention for the FFT by default
  Examples: FFT({1+***i***,1–***i***}) ⇒ {2, 2·***i***}, FFT({2, 2·***i***},-1) ⇒ {1+***i***,1–***i***}
  Notes: The list can have complex numbers, but it must be numeric, and it also cannot have numbers like *e*. Its length (or, if it is a matrix, both dimensions) must be a power of 2. The convention can be:
  "s" or "signal" – signal processing (default)
  "p" or "physics" – physical sciences
  "d" or "data" – data analysis
- IsUnivar(xpr,var) returns true if xpr has no variables other than var, else false
  Examples: IsUnivar(ln(x),x) ⇒ true, IsUnivar(1,x) ⇒ true,
  IsUnivar(x,x) ⇒ true, IsUnivar(y,x) ⇒ false,
  IsUnivar(@5,x) ⇒ false
  Note: Arbitrary real (@*k*) and integer (@n*k*) constants are considered variables.
- ListAnd(list) “ands” together the elements of list
  Examples: ListAnd({a, b or c}) ⇒ a and b or a and c,
  ListAnd({x>0,x<0}) ⇒ false
- ListOr(list) “ors” together the elements of list
  Example: ListOr({a,b}) ⇒ a or b
- MultMod(a,b,p) returns (a·b) (mod p) for large positive integers a, b, and p
  Example: MultMod(123456789,987654321,7) ⇒ 3, MultMod(2^5,3^4,5) ⇒ 2
  Note: MultMod is part of the MathTools Flash app; for installation instructions see the “Installation” section. Of course, you can use this function for small integers too. The full path for the MultMod function is `MathTool.MultMod`.
- PowerMod(a,b,p) returns (a^b) (mod p) for large positive integers a, b, and p
  Example: PowerMod(129140163,488281255,7) ⇒ 5, PowerMod(2,4^7,5) ⇒ 1
  Note: The modular power is perhaps the most important function in elementary number theory. PowerMod is part of the MathTools Flash app; for installation instructions see the “Installation” section. Of course, you can use this function for small integers as well. The full path for the PowerMod function is `MathTool.PowerMod`.
- ResetArb() resets the CAS’ arbitrary real (@*k*) and integer (@n*k*) constants
  Example: ResetArb() ⇒ "Arbitrary constant counters set to 1",
  zeros(sin(x),x) ⇒ {@n1·π}, deSolve(y'=0,x,y) ⇒ y=@1
- SqrFree(xpr [,var]) returns the squarefree factorization of xpr
  Examples: SqrFree(x^3+5·x^2+8·x+4,x) ⇒ (x+1)·$(x+2)^2$,
  SqrFree(x^6·z+2·x^3·y^3·z+2·x^5·z^2+x^2·y^3·z^2) ⇒
  x^2·(x^4+2·x^3·z+2·x·y^3+y^3·z)·z

Note: The factorization is done with respect to variable 'var' if it is specified, or with respect to all variables otherwise.

- Timing(cmd [,returntiming]) gives the execution time of a command cmd
  Examples: Timing(expand($(x+y)^{15}$)) ⇒ <Degree-15 polynomial>
  [Status bar shows 0.32 sec],
  Timing(expand($(x+y)^{15}$),true) ⇒ 0.315
  Note: If 'returntiming' is specified, the execution time is returned instead of the result of the calculation.

Important notes

These mode settings are recommended: Angle → Radian, Complex → Rectangular, Vector → Rectangular, Exact/Approx → Auto, Base → Dec, Language → English (these are what I use). All TI-Basic and FlashApp routines have Catalog help. Some of the non-Flash C programs are not binary-compatible between the TI-89 and the TI-92+/V200.

Known issues

- GMPFMul has a small memory leak (2)
- The Position function returns incorrect results when searching for a symbolic list (2)
- Cubic(Coef($(x-\boldsymbol{i})^3$,x)) returns a list of undef's (3)
- IncGamma(n,x) often returns incorrect results for complex x
- NumRoots($(x+1)^n$,x,-2,0) incorrectly returns 1 for $n \in \mathbb{Z}$ (3)
- Cannot use the variables p, q, s and t in PDESolve, as they are reserved for use by PDESolve. Please use other variable names until this is fixed. (3)
- Sylvestr cannot handle constant (degree 0) polynomials (3)
- Laguerre({2,-3},x) returns undef (it should return $x^2/2+x+1$) (3)
- SubRes currently does not return the last subresultant (4)
- Legendre gives incorrect results for large n, e.g. Legendre(1000,0) returns 0. This is because the calculation hits the maximum representable number (note the warning). (4)
- Hermite(n,num) and Laguerre(n,num) give Domain Error for most values of num
- FibNum(32768) gives a Domain Error because of an issue with matrix powers in the AMS
- PsiSum gives an incorrect result with a non-default lower limit for the sum
- mZeros returns incorrect results for multiple complex roots when the variable is not specified to be complex, e.g. mZeros($(x+\boldsymbol{i})^2$,x)
- Groebner is currently in beta and is very slow. Groebner seems to spend most of its time in NormalF, so if you can optimize NormalF and/or Groebner, please let me know.
- RTInt and Horowitz are also in beta, and may give Domain Errors. They currently work best for polynomials and rational functions.
- Risch sometimes cannot solve transformed integrands when the resultant is constant in θ. Risch is a "framework" for future functionality and may not be particularly useful (in the sense of being able to handle integrals that the built-in symbolic integrator cannot do). Risch also sometimes incorrectly claims that the integral cannot be done in terms of elementary functions (e.g. for $\ln(x)^2$).

Thanks to…

- My dear cockatiel Pi, who doesn't hurt me the way humans do
- My friends, especially Heidi, Doug, E.W., Ray, Brett, Dave, Craig, Roberto, Chris, Patrick, and Matt, for being great friends
- My family as well as my friends at Wolfram Research, for their support during a personal crisis
- Alex Astashyn, for his PolyQ function
- Doug Burkett, for his ListSwap function and for offering to let me reuse some routines from the tip list (including a couple of random number generation functions)
- Glenn Fisher, for his Coef function and the idea for PowerExp
- S.L. Hollis, for his NullVecs and GSchmidt functions
- Glenn Fisher, Lars Frederiksen, and Brett Sauerwein, for trying to optimize MathTools
- Brett Sauerwein, for his Grab function (renamed to GetArg)
- Everyone who has reported bugs (cj, Moshe Mesieles, George Dorner, OAA, Samuel Stearley, and Steve Zalek)
- Artraze, Nevin McChesney, and Brett Sauerwein, for helping to test iSolve
- David Stoutemyer, for his base conversion routines (integrated into BaseConv), for ImpDifN, and for the reference to the method for quickly computing the inverse Hilbert matrix
- Kevin Kofler, for his help in compiling GMP and for the amazing Gosper program
- Zeljko Juric, for his PowerMod/MultMod functions
- Scott Noveck, for his help in getting PowerMod and MultMod signed as a Flash app
- Cybernesto and Doug, who wrote routines similar to NRoots
- Timité Hassan, for his FastDet function
- Gosia Brothers and Brian Dean from TI, for their help with TI Flash Studio and the TI-68k platform in general

History

- Version 2.4.3:
  - Updated: ParCurve, SingVals
  - Bugfixes: Gegenbau, MatFunc, mZeros, SVD (SVD was completely rewritten)
  - Removed: CoefList, Minimize
- Version 2.4.2:
  - Added around 40 routines
  - Added MathTools Flash app, with several TI-Basic extensions
  - Updated around 15 routines
  - Bugfixes: AiryAi, AiryBi, FactList, FresnelC, FresnelS
  - Renamed: Interpol → Lagrange
  - Removed: Install (and InstData)
- Version 2.4.1:
  - Added around 10 routines
  - Updated around 8 routines
  - Renamed: Gosper → Sum2Rec
  - Bugfixes: SVD, demo text files (they were mysteriously corrupted earlier)
- Version 2.4.0:
  - Added around 50 routines

  - Updated over 30 routines
  - Renamed: ContFrac → ContFrc1, Grab → GetArg
  - Removed: Interval (if it is present in your MathTool folder, please delete it)
  - Modified all routines so that they can be used from any folder
- Versions 2.3 to 2.331:
  - Added around 165 routines
  - Updated around 35 routines
  - Renamed: GenEigVc → Eigenvec, RepEigVl → Eigenval, PInv → PInverse, PGcd → PolyGCD, PLcm → PolyLCM, PMod → PolyMod
  - Removed: FNorm (it was equivalent to the built-in norm() function)
  - Bugfixes: ExtGCD, FreeQ, GenEigVc, Install (and InstData), iSolve, list2eqn, mZeros, ReplcFac, RTInt, Terms, VarList, ZetaPrim
- Version 2.2:
  - Added around 15 routines
  - Bugfixes: Resultnt, Sort, Sylvestr, VarList
- Version 2.1:
  - Added around 35 routines
  - Updated a couple of routines
  - Improved documentation
- Version 2.0:
  - First major update (many functions added)

Future plans

- Bessel K and Y functions for integer ν (high priority)
- Update GMP stuff for AMS 2.07/2.08
- More experimental GMP functions
- Nonlinear multivariate local and global optimization
- Merge nSeries and CmplxMap into MathTools
- Include E.W.'s UniWrite in MathTools, and add feature requested by user
- Fix known bugs and limitations
- Support for trig functions in iSolve
- Hermite, Smith, and Frobenius normal form for matrices
- Matrix tests (HermMatQ, UnitaryQ, PosDefQ, etc.)
- Clement and Wilkinson matrices
- Improved speed for BesselJ for complex arguments
- Return symbolic result for Psi(1,n/4) in terms of çatalan
- IncGamma(n/2,x) and RegGamma(n/2,x), in terms of the error function
- Cosine integral Ci(z), and hyperbolic counterparts for Ci(z) and Si(z)
- Elliptic integrals
- Polylog({n,p},z) should return the Nielsen polylogarithm $S_{n,p}(z)$
- More statistical distributions, such as the Erlang and Weibull distributions
- Some inferential and non-parametric statistics
- Goodness-of-fit tests
- Exponential smoothing for data
- Higher-order non-homogeneous constant-coefficient recurrence equations

- Systems of recurrence equations
- Most-rapidly-varying (MRV) algorithm for computing limits
- Asymptotic expansions
- Tensor analysis functions for multidimensional arrays
- Series convergence tests
- Roots of polynomials modulo a composite integer using Hensel lifting and the CRT
- Improved support for congruence systems and nonlinear Diophantine equations (as a solution to Hilbert's 10^{th} problem (MathWorld entry), Matiyasevich proved that there is no way to determine solvability of a general Diophantine equation; however, many special classes of Diophantine equations can be solved)
- Quantifier elimination using cylindrical algebraic decomposition (CAD)
- Improve RTInt and Horowitz so that they can integrate all rational functions
- Complete Risch and Minimize functions
- Allow ChinRem to take univariate polynomials as arguments
- Primitive roots (from number theory)
- Generalize PolyQ for multivariate polynomials
- Implement PSLQ with multiple precision to find integer relations (I implemented it in TI-Basic, but the 14-digit precision is too low)
- Sugarcube selection strategy and basis reduction for Groebner
- Series solution of ordinary differential equations
- Multivariate limits
- Inflection points of a function
- LQ, Schur (using QR iteration) and Hessenberg matrix decompositions
- Matrix scaling
- More list operations, such as list convolution and correlation
- Linear algebra functions MatDet(), MatInv() and LinSolve() with modulus argument
- Derivatives of special functions
- Multidimensional interpolation
- More filters and discrete transforms for signal processing
- Trigonometric fits for data with regular spacing (using Fourier transform)
- Spline fits
- 3D parametric plots
- 3D vector plots
- 3D matrix plots
- Support for error bars in ListPlot
- Computational geometry
- Lattice reduction to give a reduced basis for a set of vectors (LLL algorithm)
- Improve Hadamard matrix function
- Default values for MatchQ; function Replace for pattern-based substitution
- Complexity analysis for various types of simplification (this would work only if the auto-simplification of the AMS were disabled)
- Second-order or higher PDEs and PDE systems, using Lie symmetry analysis, integral transforms, and separation of variables
- Support for integral and integro-differential equations
- Multidimensional finite differences
- More functions for numerical solution of PDEs

- Solvers for stochastic and delay differential equations
- More integral transforms such as the Mellin transform
- Symbolic Laurent series expansions and symbolic residues
- Closed-form n^{th} derivative, e.g. *d*(sin(x),x,n) is

$$\begin{cases} \sin(x) & n \bmod 4 = 0 \\ \cos(x) & n \bmod 4 = 1 \\ -\sin(x) & n \bmod 4 = 2 \\ -\cos(x) & n \bmod 4 = 3 \end{cases}$$

- Graph theory
- Differential geometry
- Coulomb wave functions
- StruveL and Weber functions
- Euler numbers
- Lucas polynomials
- Derivatives of the Airy functions
- Improve speed for PermSig
- PowerSet function (combinatorics)
- Number partitions
- Cartesian product
- Sum-of-products and product-of-sums forms (logic)
- Surface, volume and contour integrals
- Implicit representations of parametric curves and surfaces (transform to a polynomial form, do trig expansion, and use trig identities)
- Wigner 6-j and 9-j symbols for couplings of quantum momentum states
- PolyGCD(poly1,poly2,prime) [multivariate polynomial GCD modulo a prime]
- Heuristic polynomial GCD
- Arithmetic and squarefree factorization over finite fields, such as Galois fields
- Berlekamp factorization (implement nullspace computation mod a prime)
- More finance functions (cash flows, bonds, interest rates, etc.)
- Add some basics on computer algebra to the documentation (ongoing process)
- Optimization of methods used (for speed and size)
- Any suggestions?

Computer Algebra links

- A = B, an online book on hypergeometric identities by Petkovsek/Wilf/Zeilberger (foreword by Knuth)
- ATLAS (automatically tuned linear algebra software)
- Combinatorics electronic journal
- Data Analysis BriefBook
- Derivation of Numerical Methods using Computer Algebra (article)
- Discovering Number Theory
- Factoring Polynomials over Finite Fields of Small Characteristic (article)
- Finite Element Methods universal resource
- GMP, the GNU multiple-precision library
- Gröbner Bases Algorithm with a tutorial

- Gröbner Bases and Elimination of Variables
- Heuristic Algorithms
- Lexicographic ordering
- Linear Algebra (thanks to E.W. for this link)
- Mathematical Functions from Wolfram Research
- Mathematical Programming Glossary
- Mathematics textbooks online
- MathWorld, the online math encyclopedia
- NEOS optimization server, to solve linear programming problems online
- NIST Journal of Research
- Nonlinear Estimation
- Numerical Evaluation of Special Functions from NIST
- Numerical Methods for Partial Differential Equations
- On Computing Limits in a Symbolic Manipulation System (article)
- Online Encyclopedia of Integer Sequences
- Optimization Technology Center from Argonne Nat'l Lab and Northwestern Univ.
- Papers by Manuel Bronstein
- Risch algorithm overview (Lecture notes in computer algebra)
- Symbolic integration tutorial (article)
- UMFPACK (sparse linear algebra; uses Level-3 BLAS)
- Weights for finite-difference formulas (article)
- Wilf-Zeilberger theory quick review (slides)

Copyright notice and disclaimer

Thank you for using MathTools! I hope you like it and it is useful to you.

Contact information

Questions/comments/suggestions/bug-reports are welcomed at bbhatt1@towson.edu
For updates and other packages/resources, visit: http://triton.towson.edu/users/bbhatt1/ti/