10

Using mathematical functions

Math functions

The HP 39G/40G contains many math functions. The functions are grouped in categories. For example, the Matrix category contains functions for manipulating matrices. The Probability category (shown as `Prob.` on the MATH menu) contains functions for working with probability.

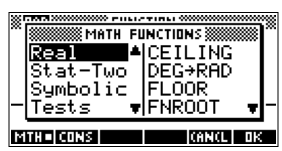
To use a math function, you enter the function onto the command line, and include the argument in parentheses after the function.

To enter a function onto the command line, either type it in *or* select its name from the MATH menu. Press MATH to access the MATH menu.

The MATH menu

The MATH menu provides access to *Math Functions* and *Programming Constants*.

The MATH menu is organized by *category*. For each category of functions on the left, there is a list of function names on the right. The highlighted category is the *current category*



- When you press MATH, you see the menu list of Math functions. The menu key **MTH** indicates that the MATH FUNCTIONS menu list is active. Press the **CONS** menu key to display a list of Program Constants.
- To display the menu list of Program Commands, press SHIFT *CMDS*. See “Programming commands” on page 15-210 for further information.

The programming commands and programming constants are discussed in Chapter 15, Programming.

To select a function

1. Press MATH to display the MATH menu. The categories appear in alphabetical order. Use ▼ and ▲ to scroll through the categories. To skip directly to the initial letter, press a letter key. *Note: You do not need to press ALPHA first*.
2. The list of functions (on the right) applies to the currently highlighted category (on the left). Use ► and ◄ to switch between the category list and the function list.
3. Highlight the name of the function you want and press OK. This copies the function name (and an initial parenthesis, if appropriate) to the edit line.

Function categories

Calculus	Loop	Statistics-Two Variable
Complex numbers	Matrices	Symbolic
Constant	Polynomial	Tests
Hyperbolic trig	Probability	Trigonometry
Lists	Real numbers	

Math functions by category

Following are definitions for all categories of functions *except* List, Matrix, and Statistics, each of which appears in its own chapter. Except for the keyboard operations, which do not appear in the MATH menu, all other functions are listed by their category in the MATH menu.

Syntax

Each function's definition includes its *syntax*, that is, the exact order and spelling of a function's name, its delimiters (punctuation), and its arguments. Note that the syntax does *not* include spaces.

Keyboard functions

The most frequently used functions appear on the keyboard. Many of the keyboard functions also accept complex numbers as arguments.

[+], [-], [*], [/]

Add, Subtract, Multiply, Divide. Also accept complex numbers, lists and matrices.

value1 + *value2*, etc.

[SHIFT] e^x

Natural exponential. Also accepts complex numbers.

`e^`*value*

Example

`e^5` returns `148.413159103`

[ln]

Natural logarithm. Also accepts complex numbers.

`LN(`*value*`)`

Example

`LN(1)` returns `0`

[SHIFT] 10^x

Exponential (antilogarithm). Also accepts complex numbers.

`10^`*value*

Example

`10^3` returns `1000`

[log]

Common logarithm. Also accepts complex numbers.

`LOG(`*value*`)`

Example

`LOG(100)` returns `2`

[SIN], [COS], [TAN]

Sine, cosine, tangent. Inputs and outputs depend on the current angle format (Degrees, Radians, or Grads).

`SIN(`*value*`)`
`COS(`*value*`)`
`TAN(`*value*`)`

Example

`TAN(45)` returns `1` (Degrees mode).

SHIFT *ASIN*	Arc sine : $\sin^{-1}x$. Output ranges from -90° to 90°, –π/2 to π/2, or –100 to 100 grads. Inputs and outputs depend on the current angle format. Also accepts complex numbers. `ASIN`(*value*) **Example** `ASIN(1)` returns `90` (Degrees mode).
SHIFT *ACOS*	Arc cosine: $\cos^{-1}x$. Output ranges from 0° to 180°, 0 to π, or 0 to 200 grads. Inputs and outputs depend on the current angle format. Also accepts complex numbers. Output will be complex for values outside the normal COS domain of -1<=x<=1 `ACOS`(*value*)
SHIFT *ATAN*	Arc tangent : $\tan^{-1}x$. Output ranges from -90° to 90°, 2π/2 to π/2, or –100 to 100 grads. Inputs and outputs depend on the current angle format. Also accepts complex numbers. `ATAN`(*value*)
x^2	Square. Also accepts complex numbers. $value^2$ **Example** `18`^{2} returns `324`
SHIFT √	Square root. Also accepts complex numbers. √*value* **Example** √`324` returns `18`
(-)	Negation. Also accepts complex numbers. –*value* **Example** `-(1,2)` returns `(-1,-2)`
X^y	Power (*x* raised to *y*). Also accepts complex numbers. *value*^*power* **Example** `2^8` returns `256` (that is, 2 x^y 8)

[SHIFT] *ABS* — Absolute value. For a complex number, this is $\sqrt{x^2 + y^2}$.

ABS(*value*)
ABS((*x*, *y*))

Example

`ABS(-1)` returns `1`
`ABS((1,2))` returns `2.2360679775`

[SHIFT] $\sqrt[n]{x}$ — Takes the *n*th root of *x*.

root `NTHROOT` *value*

Example

`3 NTHROOOT 8` returns `2` (that is, 3 [SHIFT] $\sqrt[n]{x}$ 8)

These functions are common to the keyboard and menus.

[SHIFT] π	For a description, see “p” on page 10-145.
[SHIFT] *ARG*	For a description, see “ARG” on page 10-144.
[d/dx]	For a description, see “D” on page 10-144.
[SHIFT] *AND*	For a description, see “AND” on page 10-155.
[SHIFT] !	For a description, see “!” on page 10-149.
[SHIFT] Σ	For a description, see “S” on page 10-147.
[SHIFT] *EEX*	For a description, see
[SHIFT] ∫	For a description, see “S” on page 10-144.
[SHIFT] x^{-1}	For a description, see

Symbolic calculations

The HP 39G/40G has the ability to perform symbolic calculations, for example, symbolic integration and differentiation. You can perform symbolic calculations in the HOME view and in the Function aplet.

In HOME

When you perform calculations that contain normal variables, the calculator substitutes values for any variables. For example, if you enter A+B on the command line and press ENTER, the calculator retrieves the values for A and B from memory and substitutes them in the calculation.

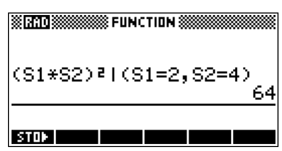
Using formal variables

To perform symbolic calculations, for example symbolic differentiations and integrations, you need to use formal names. The HP 39G/40G has six formal names available for use in symbolic calculations. These are S0 to S5. When you perform a calculation that contains a formal name, the HP 39G/40G does not carry out any substitutions.

You can mix formal names and real variables. Evaluating $(A+B+S1)^2$ will evaluate A+B, but not S1.

If you need to evaluate an expression that contains formal names numerically, you use the | (*where*) command, listed in the Math menu under the Symbolic category.

For example to evaluate (S1*S2)2 when S1= 2 and S2 = 4, you would enter the calculation as follows. (The | symbol is in the CHARS menu: press SHIFT *CHARS*. The = sign is listed in the MATH menu under Symbolic functions.)



Note: You can also use the | (where) command with a home variable.

Symbolic calculations in the Function aplet

You can perform symbolic operations in the Function aplet's Symbolic view. For example, to find the derivative of a function in the Function aplet's Symbolic view, you define two functions and define the second function as a derivative of the first function. You then evaluate the second function. See "To find derivatives in the Function aplet's Symbolic view" on page 10-141 for an example.

Finding derivatives

The HP 39G/40G can perform symbolic differentiation on some functions. There are two ways of using the HP 39G/40G to find derivatives.

- You can perform differentiations in HOME by using the formal variables, S1 to S5.
- You can perform differentiations of functions of X in the Function aplet.

To find derivatives in HOME

To find the derivative of the function in HOME, use a formal variable in place of X. If you use X, the differentiation function substitutes the value that X holds, and returns a numeric result.

For example, consider the function:

$dx(\sin(x^2) + 2\cos(x))$

1. Enter the differentiation function onto the command line, substituting S1 in place of X.

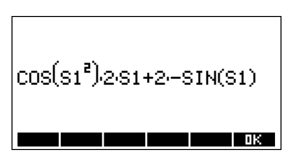
   [d/dx] [ALPHA] S 1
   [(] [SIN] [ALPHA] S 1
   [x²] [)] [+] 2 [x]
   [COS] [ALPHA] S 1
   [)] [)]

2. Evaluate the function.

   [ENTER]

3. Show the result.

   [▲] **SHOW**



To find derivatives in the Function aplet's Symbolic view

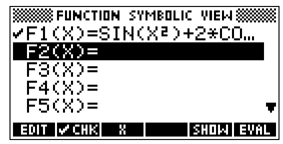
To find the derivative of the function in the Function aplet's Symbolic view, you define two functions and define the second function as a derivative of the first function. For example, to differentiate the function used above, that is

$\sin(x^2) + 2\cos x$

1. Access the Function aplet's Symbolic view and define F1.

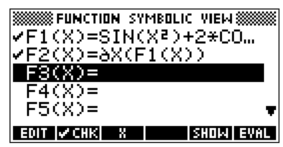
   SYMB SIN **X** x² + 2 * COS **X**) **OK**

   Note: You can use the **X** *menu key to assist in the entry of equations.*



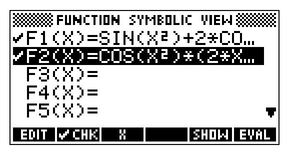
2. Define F2(X) as the derivative of F(1).

   d/dx **X** (ALPHA F1 (**X**)) **OK**



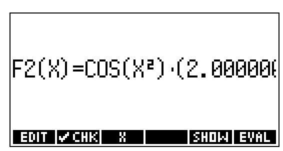
3. Select F2(X) and evaluate it.

   ▲ **EVAL**



4. Press show to display the result. (Use the arrow keys to view the entire function.)

   SHOW



You could also just define $F1(x) = dx(\sin(x^2) + 2\cos(x))$ without using 2 function definitions.

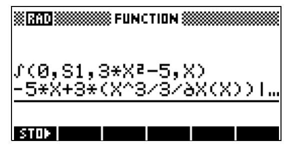
To find the indefinite integral using symbolic variables

For example, to find the indefinite integral of $\int 3x^2 - 5dx$ use:

$\int(0, S1, 3X^2 - 5, X)$

1. Enter the function.

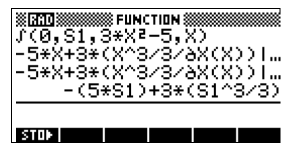
   SHIFT d/dx 0 ,
   ALPHA S 1 , 3 x
   ALPHA **X** x² (–) 5 ,
   ALPHA **X**) ENTER



   The format of the result shown on the screen can be seen more clearly by using the **SHOW** menu key and scrolling.

2. Highlight and copy the result from step 1 to the Editline and evaluate.

   ▲ **COPY** ENTER



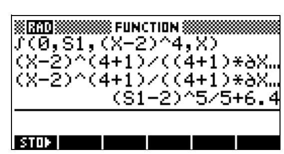
   Thus, substituting X for S1, it can be seen that

$$\int 3x^2 - 5dx = -5x + 3\left(\frac{\frac{x^3}{3}}{\frac{\partial}{\partial X}(X)}\right)$$

This result derives from substituting X=S1 and X=0 into the original expression found in step 1. However, substituting X=0 will not always evaluate to zero and may result in an unwanted constant.

For example: $\int (x-2)^4 dx = \frac{(x-2)^5}{5}$

The ‘extra’ constant of 6.4 results from the substitution of x=0 into $\frac{(x-2)^5}{5}$, and should be disregarded if an *indefinite* integral is required.



Calculus functions

You find the symbols for the calculus functions derivative and integral in the CHARS menu (SHIFT *CHARS*) as well as the MATH menu.

∂

Differentiates *expression* with respect to the *variable* of differentiation. From the command line, use a formal name (S1, etc.) for a non-numeric result. See “Finding derivatives” on page 10-141.

∂*variable*(*expression*)

Example

`∂s1(s1`^{2}`+3*s1)` returns `2*s1+3`

∫

Integrates *expression* from *lower* to *upper* limits with respect to the *variable* of integration. To find the definite (numeric) integral, both limits must have numeric values (that is, be numbers or real variables). To find the indefinite integral, one of the limits must be a formal name (s1, etc.).

∫(*lower*,*upper*,*expression*,*variable*)

Example

`∫(0,s1,2*X+3,X)` ENTER ▲ **COPY** ENTER finds the indefinite result `3*s1+2*(s1^2/2)`

See “To find the indefinite integral using symbolic variables” on page 10-142 for more information on finding indefinite integrals.

TAYLOR

Calculates the *n*th order Taylor’s polynomial of *expression* at the point where the given *variable* = 0.

TAYLOR(*expression*,*variable*,*n*)

Example

`TAYLOR(1+ sin(s1)`^{2}`,s1,5)` with Radians angle measure and Fraction number format (set in MODES) returns `1+s1^-21/3*s1^4 **need to sort this.`

Complex-number functions

These functions are for complex numbers only. You can also use complex numbers with all trigonometric and hyperbolic functions, and with some real-number and keyboard functions. Enter complex numbers in the form (*x*,*y*), where *x* is the real part and *y* is the imaginary part.

ARG

Argument. Finds the angle defined by a complex number. Inputs and outputs use the current angle format set in Modes.

ARG((*x*,*y*))

Example

`ARG((3,3))` returns `45 in DEG mode`

CONJ

Complex conjugate. Conjugation is the negation (sign reversal) of the imaginary part of a complex number.

`CONJ`((*x*,*y*))

Example

`CONJ((3,4))` returns `(3,-4)`

IM

Imaginary part, *y*, of a complex number, (*x*,*y*).

`IM` ((*x*,*y*))

Example

`IM((3,4))` returns `4`

RE

Real part *x*, of a complex number, (*x*,*y*).

`RE`((*x*,*y*))

Example

`RE((3,4))` returns `3`

Constants

The HP 39G/40G has an internal numeric representation for these constants.

e

Natural logarithm base. Internally represented as 2.71828182846.

e

i

Imaginary value for $\sqrt{-1}$, the complex number (0,1).

i

MAXREAL

Maximum real number. Internally represented as $9.99999999999 \times 10^{499}$.

MAXREAL

MINREAL

Minimum real number. Internally represented as 1×10^{-499}.

MINREAL

π

The constant: diameter. Internally represented as 3.14159265359.

π

Hyperbolic trigonometry

The hyperbolic trigonometry functions can also take complex numbers as arguments.

ACOSH

Inverse hyperbolic cosine : $\cosh^{-1}x$.

`ACOSH`(*value*)

ALOG

Antilogarithm (exponential). This is more accurate than `10^x` due to limitations of the power function.

`ALOG`(*value*)

ASINH

Inverse hyperbolic sine : $\sinh^{-1}x$.

`ASINH`(*value*)

ATANH

Inverse hyperbolic tangent : $\tanh^{-1}x$. If the input is 61, an `Infinite Result` occurs.

`ATANH`(*value*)

COSH

Hyperbolic cosine : $(e^x+e^{2x})/2$.

`COSH`(*value*)

SINH

Hyperbolic sine.

`SINH`(*value*)

TANH

Hyperbolic tangent.

`TANH`(*value*)

EXP

Natural exponential. This is more accurate than `e^x` due to limitations of the power function.

`EXP`(*value*)

EXPM1

Exponent minus 1 : e^x–1. This is more accurate than EXP when x is close to zero.

`EXPM1`(*value*)

LNP1

Natural log plus 1 : $\ln(x+1)$. This is more accurate than LN when x is close to zero.

`LNP1`(*value*)

List functions

These functions are for list data stored in list variables. See “List functions” on page 10-146.

Loop functions

The loop functions display a result after evaluating an expression a given number of times.

ITERATE

Repeatedly (the specified *#times*) evaluates an *expression* in terms of *variable*. The value for *variable* is updated each time, starting with *initialvalue*.

`ITERATE(`*expression*`,`*variable*`,`*initialvalue*`,` *#times*`)`

Example

`ITERATE(X`^{2}`,X,2,3)` returns `256`

RECURSE

Provides a method of defining a sequence without using the Symbolic view of the Sequence aplet. If used with | (“where”), RECURSE will step through the evaluation.

`RECURSE(`*sequencename*`,`*term-n*`,`*term1*`,`*term2*`)`

Example

`RECURSE(U,U(N-1)*N,1,2)STO›U1(N)`
Produces the factorial
`U1(1)=1`
`U1(2)=2`
`U1(N)=U1(N-1)*N`

Σ

Summation. Finds the sum of *expression* with respect to *variable* from *initialvalue* to *finalvalue.*

`Σ(`*variable*=*initialvalue*`,`*finalvalue*`,`*expression*`)`

Example

`Σ(C=1,5,C`^{2}`)` returns 55.

Matrix functions

These functions are for matrix data stored in matrix variables. See “Matrix functions and commands” on page 12-176.

Polynomial functions

Polynomials are products of constants (*coefficients*) and variables raised to powers (*terms*).

POLYCOEF

Polynomial coefficients. Returns the coefficients for the polynomial with the specified *roots*.

POLYCOEF ([*roots*])

Example

To find the polynomial with roots 2, -3, 4, -5:
POLYCOEF([2,-3,4,-5]) returns[1,2,-25,-26,120], representing $x^4+2x^3-25x^2-26x+120$.

POLYEVAL

Polynomial evaluation. Evaluates a polynomial with the specified *coefficients* for the *value* of *x*.

POLYEVAL([*coefficients*],*value*)

Example

For $x^4+2x^3-25x^2-26x+120$:
POLYEVAL([1,2,-25,-26,120],8) returns 3432.

POLYFORM

Polynomial form. Creates a polynomial in *variable1* from *expression.* Can express the coefficients as a polynomial in *variable2.* (The coefficients of the coefficients can be expressed as polynomials in *variable3,* etc.)

POLYFORM(*expression*,*variable1*,...,*variable-n*)

Example

POLYFORM((X+1)^2+1,X) returns X^2+2*X+2.

POLYROOT

Polynomial roots. Returns the roots for the *n*th-order polynomial with the specified *n*+1 *coefficients*.

POLYROOT([*coefficients*])

Example

For $x^4+2x^3-25x^2-26x+120$:
POLYROOT([1,2,-25,-26,120]) returns [2,-3,4,-5].

The results of POLYROOT will often not be easily seen in HOME due to the number of decimal places, especially if they are complex numbers. It is better to store the results of POLYROOT to a matrix e.g. POLYROOT([1,0,0,-8] STO►M1 will store the three complex cube roots of 8 to matrix M1 as a complex vector.
Then you can:
1. See them easily by going to the Matrix Catalogue.
2. Access them individually in calculations by referring to M1(1), M1(2) etc.

Probability functions

Probability functions are often used in statistical analyses.

COMB

Number of combinations (without regard to order) of *n* things taken *r* at a time, *n*!/(*r*!(*n*–*r*)).

`COMB(`*n*`,`*r*`)`

Example

`COMB(5,2)` returns `10`. That is, there are ten different ways that five things can be combined two at a time.

!

Factorial of a positive integer. For non-integers, $! = \Gamma(x + 1)$.

value!

PERM

Number of permutations (with regard to order) of *n* things taken *r* at a time, *n*!/ (*n*-*r*)!.

`PERM (`*n*`,`*r*`)`

Example

`PERM(5,2)` returns `20`. That is, there are 20 different permutations of five things taken two at a time.

RANDOM

Random number (between zero and 1). Produced by a pseudo-random number sequence. The algorithm used in the RANDOM function uses a “seed” number to begin its sequence. To ensure that two calculators must produce different results for the RANDOM function, use the RANDSEED function to seed different starting values before using RANDOM to produce the numbers.

The setting of Time will be different for each calculator, so using RANDSEED(Time) is guaranteed to produce a set of numbers which are as close to random as possible. You can set the seed using the command RANDSEED.

RANDOM

RANDSEED

Sets random number seed to *value*. The random number seed is used in the calculation of random numbers by the `RANDOM` function.

`RANDSEED` *value*

UTPC

Upper-Tail Chi-Squared Probability given *degrees* of freedom, evaluated at *value*. Returns the probability that a x^2 random variable is greater than *value.*

`UTPC`(*degrees*,*value*)

UTPF

Upper-Tail Snedecor's F Probability given *numerator* degrees of freedom and *denominator* degrees of freedom (of the F distribution), evaluated at *value*. Returns the probability that a Snedecor's F random variable is greater than *value.*

`UTPF`(*numerator*,*denominator*,*value*)

UTPN

Upper-Tail Normal Probability given *mean* and *variance*, evaluated at *value*. Returns the probability that a normal random variable is greater than *value* for a normal distribution. *Note: The variance is the square of the standard deviation*.

`UTPN`(*mean*,*variance*,*value*)

UTPT

Upper-Tail Student's t-Probability given *degrees* of freedom, evaluated at *value*. Returns the probability that the Student's t-random variable is greater than *value*.

`UTPT`(*degrees*,*value*)

Real-number functions

Some real-number functions can also take complex arguments.

CEILING

Smallest integer greater than, equal to *value*.

`CEILING`(*value*)

Examples

`CEILING(3.2)` returns `4`
`CEILING(-3.2)` returns `-3`

DEG→RAD

Degrees to radians. Converts *value* from Degrees angle format to Radians angle format.

`DEG→RAD`(*value*)

Example

`DEG→RAD(180)` returns `3.14159265359`, the value of π.

FLOOR

Greatest integer less than or equal to *value*.

`FLOOR`(*value*)

Example

`FLOOR(-3.2)` returns `-4`

FNROOT

Function root-finder (like the Solve aplet). Finds the value for the given *variable* at which *expression* most nearly evaluates to zero. Uses *guess* as initial estimate.

FNROOT(*expression, variable, guess*)

Example

FNROOT(M*9.8/600-1,M,1) returns 61.2244897959.

FRAC

Fractional part.

FRAC(*value*)

Example

FRAC (23.2) returns .2

HMS→

Hours-minutes-seconds to decimal. Converts a number or expression in *H.MMSSs* format (time or angle that can include fractions of a second) to *x.x* format (number of hours or degrees with a decimal fraction).

HMS→(*H.MMSSs*)

Example

HMS→(8.30) returns 8.5

→HMS

Decimal to hours-minutes-seconds. Converts a number or expression in *x.x* format (number of hours or degrees with a decimal fraction) to *H.MMSSs* format (time or angle up to fractions of a second).

→HMS(*x.x*)

Example

→HMS(8.5) returns 8.3

INT

Integer part.

INT(*value*)

Example

INT(23.2) returns 23

MANT

Mantissa (significant digits) of *value*.

MANT(*value*)

Example

MANT(21.2E34) returns 2.12

MAX

Maximum. The greater of two values.

`MAX`(*value1*`,`*value2*)

Example

`MAX(210,25)` returns `210`

MIN

Minimum. The lesser of two values.

`MIN`(*value1*`,`*value2*)

MOD

Modulo. The remainder of *value1*/*value2*.

value1 `MOD` *value2*

Example

`9 MOD 4` returns `1`

%

x percent of *y*; that is, *x/100*y*.

`%(`*x*`,`*y*`)`

Example

`%(20,50)` returns `10`

%CHANGE

Percent change from *x* to *y*, that is, 100(*y*–*x*)/*x*.

`%CHANGE`(*x*`,`*y*)

Example

`%CHANGE(20,50)` returns `150`

%TOTAL

Percent total : (100)*y/x*. What percentage of *x* is *y*.

`%TOTAL`(*x*`,`*y*)

Example

`%TOTAL(20,50)` returns `250`

RAD→DEG

Radians to degrees. Converts *value* from radians to degrees.

`RAD→DEG` (*value*)

Example

`RAD→DEG(π)` returns `180`

ROUND

Rounds *value* to decimal *places*. Accepts complex numbers.

`ROUND`(*value*`,`*places*)

Round can also round to a number of significant digits as showed in example 2.

Example

`ROUND(7.8676,2)` returns `7.68`

`ROUND (0.0036757,-3)` returns `0.00368`

SIGN

Sign of *value*. If positive, the result is +1. If negative, –1. If zero, result is zero. For a complex number, this is the unit vector in the direction of the number

`SIGN(`*value*`)`
`SIGN((`*x*,*y*`))`

Example

`SIGN((3,4))` returns `(.6,.8)`

TRUNCATE

Truncates *value* to decimal *places*. Accepts complex numbers.

`TRUNCATE(`*value*`,`*places*`)`

Example

`TRUNCATE(2.3678,2)` returns `2.36`

XPON

Exponent of *value*.

`XPON(`*value*`)`

Example

`XPON(123.4)` returns `2`

Statistics-Two

These are functions for use with two-variable statistics. See “Two-variable computed STATS” on page 8-108.

Symbolic functions

The symbolic functions are used for symbolic manipulations of expressions. The variables can be formal or numeric, but the result is usually in symbolic form (not a number). You will find the symbols for the symbolic functions = and | (*where*) in the CHARS menu (SHIFT *CHARS*) as well as the MATH menu.

= (*equals*)

Sets an equality for an equation. This does *not* store values. This is *not* a logical operator. (See “Test functions” on page 10-155.)

expression1=expression2

ISOLATE

Isolates the first occurrence of *variable* in *expression*=0 and returns a new expression, where *variable*=*newexpression*. The result is a general solution that represents multiple solutions by including the (formal) variables *s1* to represent any sign and *n1* to represent any integer.

`ISOLATE(`*expression*`,`*variable*`)`

Examples

`ISOLATE(2*X+8,X)` returns `-4`
`ISOLATE(A+B*X/C,X)` returns `-(A*C/B)`

LINEAR?

Tests whether *expression* is linear for the specified *variable*. Returns `0` (false) or `1` (true).

`LINEAR?(`*expression*`,`*variable*`)`

Example

`LINEAR?((X^2-1)/(X+1),X)` returns `0`

QUAD

Solves quadratic *expression*=0 for *variable* and returns a new expression, where *variable*=*newexpression*. The result is a general solution that represents both positive and negative solutions by including the (formal) variable *s1* to represent any sign, + or – .

`QUAD(`*expression*`,`*variable*`)`

Example

`QUAD((X-1)`^{2}`-7,X)` returns
`(2+s1*3.46410161514)/2`

QUOTE

Encloses an expression that should not be evaluated numerically.

`QUOTE(`*expression*)

Examples

`QUOTE(SIN(Z)) STO› F1(Z)` stores the expression.
`GROB G1;QUOTE(1/(1+1/X));0` makes a graphic from the expression.

Another method is to enclose the expression in single quotes.

For example, 'X^3+2*X' STO> F1(X) puts the expression X^3_2*X into F1(X) in the Function aplet. Remember, before the expression can be PLOTed it would need to be CHKed.

| (*where*)

Evaluates *expression* where each given variable is set to the given *value*. Defines numeric evaluation of a symbolic expression.

expression|(*variable1=value1,variable2=value2,*)

Example

`3*(X+1)|(X=3)` returns `12`.
`s1(SIN(s1+SIN(s1)))|(s1=5)` returns `2.798456889121`

Test functions

The test functions are *logical* operators that always return either a 1 (*true*) or a 0 (*false*).

<

Less than. Returns 1 if true, 0 if false.

value1<value2

≤

Less than or equal to. Returns 1 if true, 0 if false.

value1≤value2

= =

Equals (logical test). Returns 1 if true, 0 if false.

value1==value2

≠

Not equal to. Returns 1 if true, 0 if false.

value1≠value2

>

Greater than. Returns 1 if true, 0 if false.

value1>value2

≥

Greater than or equal to. Returns 1 if true, 0 if false.

value1≥value2

AND

Compares *value1* and *value2*. Returns 1 if they are both non-zero, otherwise returns 0.

value1 `AND` *value2*

IFTE

If *expression* is true, do the *trueclause*; if not, do the *falseclause.*

`IFTE(`*expression*`,`*trueclause*`,`*falseclause*`)`

Example

IFTE(X>0,X^2,X^3)

NOT Returns 1 if *value* is zero, otherwise returns 0.

`NOT` *value*

OR Returns 1 if either *value1* or *value2* is non-zero, otherwise returns 0.

value1 `OR` *value2*

XOR Exclusive OR. Returns 1 if either *value1* or *value2* ---but not both of them---is non-zero, otherwise returns 0.

value1 `XOR` *value2*

Trigonometry functions

The trigonometry functions can also take complex numbers as arguments. For SIN, COS, TAN, ASIN, ACOS, and ATAN, see the Keyboard category (all these functions have keys).

ACOT Arc cotangent.

`ACOT`(*value*)

ACSC Arc cosecant.

`ACSC`(*value*)

ASEC Arc secant.

`ASEC`(*value*)

COT Cotangent : $\cos x/\sin x$.

`COT`(*value*)

CSC Cosecant : $1/\sin x$

`CSC`(*value*)

SEC Secant : $1/\cos x$.

`SEC`(*value*)