Chapter 2
Operators

This chapter lists and describes Mathcad's built-in operators. The operators are listed according to the toolbar (Arithmetic, Matrix, Calculus, Evaluation and Boolean, or Programming) on which they appear.

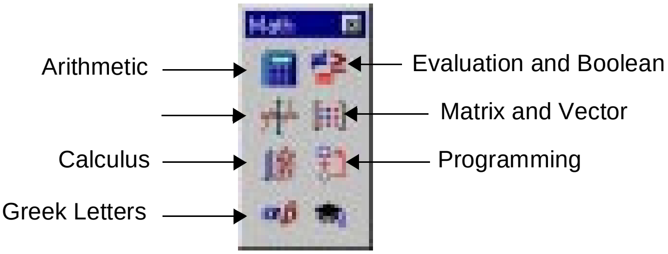
Operators labeled *Professional* are available only in Mathcad Professional.

Accessing Operators

You can access operators in two ways:

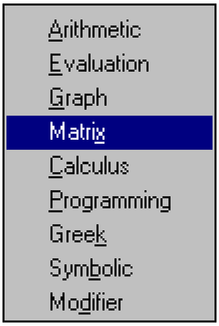
- Simply type in the keystroke shown for that operator, or
- Select the operator from a toolbar.

1. First, select **View** ⇒ **Toolbars** ⇒ **Math**.
   The Math toolbar appears showing the various operator toolbar buttons.

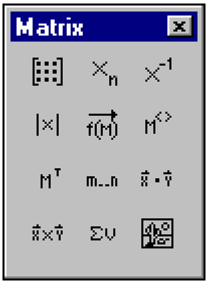


2. Click a button for a specific toolbar. The corresponding toolbar appears.

   You can alternatively go directly to an operator toolbar from the View menu by selecting **View** ⇒ **Toolbars**, and then selecting one of the operator toolbars listed; for example **Matrix**:



The Matrix operator toolbar appears, showing the various matrix operator buttons:



3. Click the button of the operator that you want to use.

Finding More Information

Refer to the Resource Center QuickSheets for examples involving operators. Select **Resource Center** from the **Help** menu. Then click on the QuickSheets icon and select a specific topic.

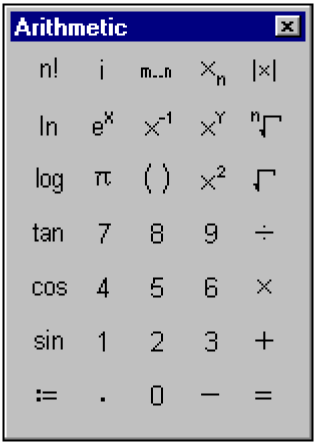
About the References

References are provided in Appendix B for you to learn more about the numerical algorithm underlying a given Mathcad function or operator. References are not intended to give a description of the actual underlying source code. Some references (such as *Numerical Recipes*) do contain actual C code for the algorithms discussed therein, but the use of the reference does not necessarily imply that the code is what is implemented in Mathcad. The references are cited for their textbook nature only.

Arithmetic Operators

To access an arithmetic operator:

- type its keystroke, or
- choose the operator from the Arithmetic toolbar (if it has a button):



Refer to “Accessing Operators” on page 135 for more information on how to access a toolbar.

Parentheses (X)

Keystroke **'** Toolbar Button ( )

Description Groups parts of an expression.

Addition $X + Y$

Keystroke **+** Toolbar Button +

Description If X and Y are real or complex numbers, adds X and Y.
If X and Y are real or complex vectors or matrices of the same size, adds elements of X to corresponding elements of Y.
If X is a real or complex array and Y is a real or complex number, adds Y to each element of X.

Addition with line break $X\ldots$ $+ Y$

Keystroke **[Ctrl][↵]**

Description Adds in the same manner as Addition, but inserts a line break for cosmetic formatting reasons.

Comments This formatting feature cannot be used for multiplication or division. It can be used with subtraction if $X - Y$ is written instead as $X + (-Y)$.

Subtraction and Negation $X - Y,\ -X$

Keystroke **-** Toolbar Button −

Subtraction

Description If X and Y are real or complex numbers, subtracts Y from X.
If X and Y are real or complex vectors or matrices of the same size, subtracts elements of Y from corresponding elements of X.
If X is an real or complex array and Y is a real or complex number, subtracts Y from each element of X.

Negation

Description If X is a real or complex number, reverses the sign of X.
If X is an real or complex array, reverses the sign of each element of X.

Multiplication $X \cdot Y$

Keystroke ***** Toolbar Button ×

Description If X and Y are real or complex numbers, multiplies Y by X.
If Y is a real or complex array and X is a real or complex number, multiplies each element of Y by X.
If X and Y are real or complex vectors of the same size, returns the dot product (inner product).
If X and Y are real or complex conformable matrices, performs matrix multiplication.

Division $\frac{X}{z}$

Keystroke **/** Toolbar Button ÷

Description If X and z are real or complex numbers and z is nonzero, divides X by z.
If X is an real or complex array and z is a nonzero real or complex number, divides each element of X by z.

Range variable

Keystroke **;** Toolbar Button m..n

Description Specifies that a variable assume a range of values (for the sake of repeated or iterative calculations).

Factorial $n!$

Keystroke **!** Toolbar Button n!

Description Returns $n \cdot (n-1) \cdot (n-2) \ldots 2 \cdot 1$ if n is an integer and $n \geq 1$; 1 if $n = 0$.

Vector and matrix subscript $\mathbf{v}_n, \mathbf{M}_{i,j}$

Keystroke **[** Toolbar Button x_n

Description If **v** is a vector, $\mathbf{v}_n$ returns the *n*th element of **v**.
If **M** is a matrix, $\mathbf{M}_{i,j}$ returns the element in row *i* and column *j* of **M**.

Complex conjugate $\overline{X}$

Keystroke **"**

Description If *X* is a complex number, reverses the sign of the imaginary part of *X*.

Absolute value $|x|$

Keystroke **|** Toolbar Button |x|

Description If *z* is a real or complex number, |*z*| returns the absolute value (or modulus or magnitude) $\sqrt{\mathrm{Re}(z)^2 + \mathrm{Im}(z)^2}$ of *z*.

If **v** is real or complex vector, |**v**| returns the magnitude (or Euclidean norm or length) $\sqrt{\mathbf{v} \cdot \overline{\mathbf{v}}}$ of **v**. If all elements in **v** are real, this definition is equivalent to $\sqrt{\mathbf{v} \cdot \mathbf{v}}$.

If **M** is a real or complex square matrix, |**M**| returns the determinant of **M**.

Square root $\sqrt{z}$

Keystroke \ Toolbar Button $\sqrt{\ }$

Description Returns the positive square root for positive *z*; principal value for negative or complex *z*.

*n*th root $\sqrt[n]{z}$

Keystroke **[Ctrl]\** Toolbar Button $\sqrt[n]{\ }$

Description Returns the positive *n*th root for positive *z*; negative *n*th root for negative *z* and odd *n*; principal value otherwise. *n* must be an integer, $n \geq 1$.

See also Exponentiation, Square root

Comments This operator gives the same values as the Exponentiation operator except when $z < 0$ and *n* is an odd integer and $n \geq 3$ (by special convention).

Exponentiation z^w

Keystroke **^** Toolbar Button x^y

Scalar case

Description Returns the principal value of *z* raised to the power *w*, where *z* and *w* are real or complex numbers.

See also nth root

Comments The principal value is given by the formula $|z|^w \cdot \exp(\pi \cdot i \cdot w)$. In the special case $z < 0$ and $w = 1/n$, where *n* is an odd integer and $n \geq 3$, the principal value has a nonzero imaginary part. Hence, in this special case, Exponentiation does not give the same value as the *n*th root operator (by convention).

Matrix case

Description If **M** is a real or complex square matrix and $n \geq 0$ is an integer, $\mathbf{M}^n$ returns the *n*th power of **M** (using iterated matrix multiplication). Under the same conditions, $\mathbf{M}^{-n}$ is the inverse of $\mathbf{M}^n$ (assuming additionally that **M** is nonsingular).

Algorithm LU decomposition used for matrix inversion (Press *et al.*, 1992)

Equals *c =*

Keystroke **=** Toolbar Button =

Description Returns numerical value of *c* if *c* is: a variable previously defined in the worksheet; a built-in variable; a globally-defined variable; or a function of several such variables. Appears as an ordinary = on the screen. Not used for symbolic evaluation.

Definition *z : = c , f(x,y,z,...) : = expr*

Keystroke **:** Toolbar Button :=

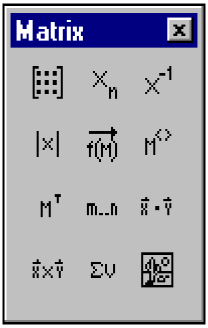
Description Gives *z* the numerical value *c* from that point onward throughout the worksheet. Gives a function *f(x,y,z,...)* the meaning prescribed by the expression *expr* from that point onward throughout the worksheet. *expr* need not involve *x, y, z, ...* but it usually does; it may involve other built-in or user-defined functions.

See also Definition (under Evaluation and Boolean Operators) for example.

Matrix Operators

To access a matrix operator:

- type its keystroke, or
- choose the operator from the Matrix toolbar:



Refer to “Accessing Operators” on page 135 for more information on how to access a toolbar.

Insert matrix

Keystroke **`[Ctrl]M`** Toolbar Button

Description Creates a vector or matrix of specified dimensions.

Vector and matrix subscript $\mathbf{v}_n$, $\mathbf{M}_{i,\,j}$

Keystroke **`[`** Toolbar Button

Description If **v** is a vector, $\mathbf{v}_n$ returns the *n*th element of **v**.
If **M** is a matrix, $\mathbf{M}_{i,\,j}$ returns the element in row *i* and column *j* of **M**.

Dot product $\mathbf{u} \cdot \mathbf{v}$

Keystroke **`*`** Toolbar Button

Description Returns the dot product (scalar or inner product) of two *n*-dimensional real or complex vectors **u** and **v**.

Cross product $\mathbf{u} \times \mathbf{v}$

Keystroke **`[Ctrl]8`** Toolbar Button

Description Returns the cross product (vector product) of two 3-dimensional real or complex vectors **u** and **v**.

Vector sum $\Sigma \mathbf{v}$

Keystroke **[Ctrl]4** Toolbar Button Σv

Description Returns the sum (a scalar) of all elements of a real or complex vector **v**. (No range variable or vector subscripts are needed.)

Matrix Inverse

Keystroke **^-1** Toolbar Button X^{-1}

Description Returns the multiplicative inverse of a real or complex nonsingular square matrix **M**.

Magnitude and Determinant $|x|$

Keystroke **|** Toolbar Button $|x|$

Description If z is a real or complex number, $|z|$ returns the absolute value (or modulus or magnitude) $\sqrt{\mathrm{Re}(z)^2 + \mathrm{Im}(z)^2}$ of z.

If **v** is real or complex vector, returns the magnitude (or Euclidean norm or length) $\sqrt{\mathbf{v} \cdot \overline{\mathbf{v}}}$ of **v**. If all elements in **v** are real, this definition is equivalent to $\sqrt{\mathbf{v} \cdot \mathbf{v}}$.

If **M** is a real or complex square matrix, returns the determinant of **M**.

Algorithm LU decomposition (Press *et al.*, 1992)

Matrix superscript $\mathbf{M}^{\langle n \rangle}$

Keystroke **[Ctrl]6** Toolbar Button $M^{<>}$

Description Extracts column n (a vector) from matrix **M**.

Matrix transpose $\mathbf{M}^{\mathrm{T}}$

Keystroke **[Ctrl]1** Toolbar Button M^T

Description If **M** is a vector or matrix, returns a matrix whose rows are the columns of **M** and whose columns are the rows of **M**.

Vectorize $\overrightarrow{X}$

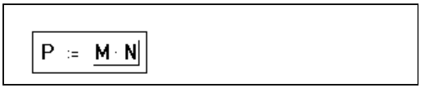
Keystroke **[Ctrl]-** Toolbar Button $\overrightarrow{f(M)}$

Description Forces operations in expression X to take place element by element. All vectors or matrices in X must be the same size.

Comments

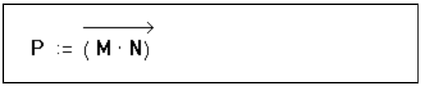
Mathcad's vectorize operator allows parallel operations to be performed efficiently on each element of a vector or matrix. For example, to define a matrix **P** by multiplying corresponding elements of the matrices **M** and **N**, you could write $\mathbf{P}_{i,\,j} = \mathbf{M}_{i,\,j} \cdot \mathbf{N}_{i,\,j}$ where *i* and *j* are range variables. (This is not matrix multiplication, but rather multiplication element by element.) It's faster, however, to define **P** using vectorize:

- Select the whole expression by clicking inside and pressing [`Space`] until the right-hand side is held between the editing lines.



$P := M \cdot N$

- Press [`Ctrl`]`–` to apply the vectorize operator. Mathcad puts an arrow over the top of the selected expression.



$P := \overrightarrow{(M \cdot N)}$

Extending ordinary scalar multiplication to matrices in this fashion, element by element, is referred to as "vectorizing" an expression.

Here are some properties of the vectorize operator:

- The vectorize operator changes the meaning of functions and operators but not constants or variables.
- Operations between an array and a scalar are performed by applying the scalar to each element of the array. For example, if **v** is a vector and *n* is a scalar, applying the vectorize operator to $\mathbf{v}^n$ returns a vector whose elements are the *n*th powers of the elements of **v**.
- You cannot use any of the following matrix operations under a vectorize operator: dot product, matrix multiplication, matrix powers, matrix inverse, determinant, or magnitude of a vector. The vectorize operator will transform these operations into element-by-element scalar multiplication, exponentiation, or absolute value, as appropriate.
- The vectorize operator has no effect on operators and functions that *require* vectors or matrices: transpose, cross product, sum of vector elements, and functions like mean. These operators and functions have no scalar meaning.

Picture

Keystroke **`[Ctrl]T`** Toolbar Button

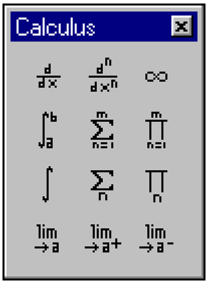
Description

Displays a matrix **M** as a grayscale image. Each element of **M** corresponds to a pixel. The value of an element determines the shade of gray associated with the corresponding pixel. Each element of **M** is an integer between 0 (black) and 255 (white).

Calculus Operators

To access a calculus operator:

- type its keystroke, or
- choose the operator from the Calculus toolbar:



Refer to “Accessing Operators” on page 135 for more information on how to access a toolbar.

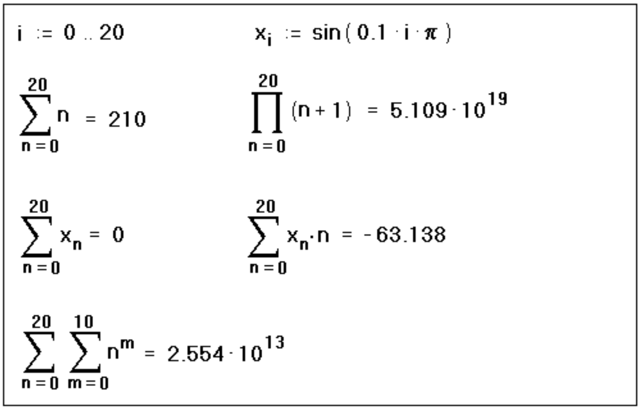
Summation

$$\sum_{i=m}^{n} X$$

Keystroke **[Ctrl][Shift]4** Toolbar Button $\sum_{n=1}^{m}$

Description Performs iterated addition of X over $i = m, m + 1, \ldots, n$. X can be any expression; it need not involve i but it usually does. m and n must be integers. If $m = -\infty$ or $n = \infty$, the evaluation must be performed symbolically.

Example



$i := 0 .. 20$ $x_i := \sin(0.1 \cdot i \cdot \pi)$

$$\sum_{n=0}^{20} n = 210 \qquad \prod_{n=0}^{20} (n+1) = 5.109 \cdot 10^{19}$$

$$\sum_{n=0}^{20} x_n = 0 \qquad \sum_{n=0}^{20} x_n \cdot n = -63.138$$

$$\sum_{n=0}^{20} \sum_{m=0}^{10} n^m = 2.554 \cdot 10^{13}$$

See also Range sum

Comments To evaluate multiple summations, place another summation in the final placeholder of the first summation, as illustrated in the example.

Product $\prod_{i=m}^{n} X$

Keystroke **[Ctrl][Shift]3** Toolbar Button

Description Performs iterated multiplication of X over $i = m, m+1, \ldots, n$. X can be any expression; it need not involve i but it usually does. If $m = -\infty$ or $n = \infty$, the evaluation must be performed symbolically. Works similar to Summation.

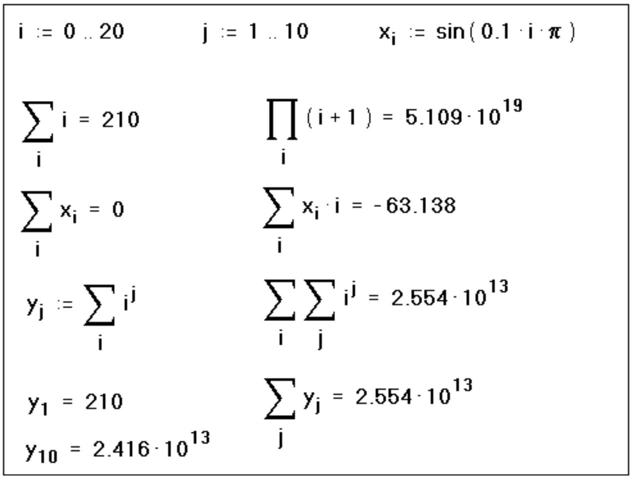
See also Range product. See Summation for an example.

Range sum $\sum_{i} X$

Keystroke **$** Toolbar Button

Description Performs iterated addition of X over the range variable i. X can be any expression; it need not involve i but it usually does.

Example



$i := 0..20$ $j := 1..10$ $x_i := \sin(0.1 \cdot i \cdot \pi)$

$\sum_{i} i = 210$ $\prod_{i} (i+1) = 5.109 \cdot 10^{19}$

$\sum_{i} x_i = 0$ $\sum_{i} x_i \cdot i = -63.138$

$y_j := \sum_{i} i^j$ $\sum_{i} \sum_{j} i^j = 2.554 \cdot 10^{13}$

$y_1 = 210$ $\sum_{j} y_j = 2.554 \cdot 10^{13}$

$y_{10} = 2.416 \cdot 10^{13}$

See also Summation

Comments When you use the Summation operator described earlier, the summation must be carried out over integers and in steps of one. Mathcad provides a more general version of this operator that can use any range variable you define as an index of summation.

The Range sum operator, unlike the Summation operator, cannot stand alone. It requires the existence of a range variable. However, a single range variable can be used with any number of these operators.

To evaluate multiple summations, place another summation in the final placeholder of the first summation and use two range variables, as illustrated in the example.

Range product $\prod_i X$

Keystroke **#** Toolbar Button $\prod_n$

Description

Performs iterated multplication of X over the range variable i. X can be any expression; it need not involve i but it usually does. Works similar to Range sum.

See also

Product. See Range sum for an example.

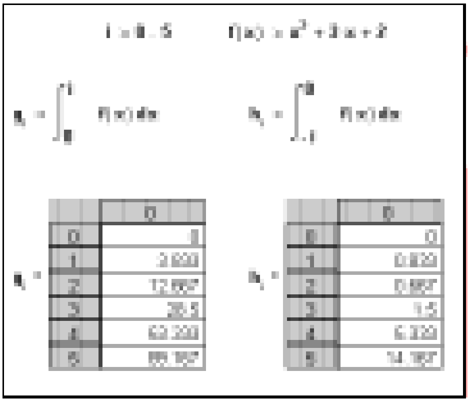
Definite integral $\int_a^b f(t)\,dt$

Keystroke **&** Toolbar Button $\int_a^b$

Description

Returns the definite integral of $f(t)$ over the interval $[a, b]$. a and b must be real scalars. All variables in the expression $f(t)$, except the variable of integration t, must be defined. The integrand, $f(t)$, cannot return an array. $a = -\infty$, $b = \infty$, or both are permitted.

Examples



$i := 0 .. 5$ $f(x) := x^2 + 3x + 2$

$a_i := \int_0^i f(x)\,dx$ $b_i := \int_{-i}^0 f(x)\,dx$

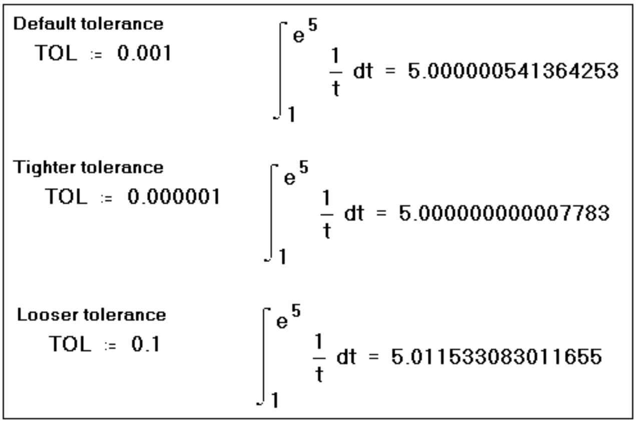
$a_i =$

	0
0	0
1	3.833
2	12.667
3	28.5
4	53.333
5	89.167

$b_i =$

	0
0	0
1	0.833
2	0.667
3	1.5
4	5.333
5	14.167

Example 1



Default tolerance

TOL := 0.001

$$\int_{1}^{e^5} \frac{1}{t}\,dt = 5.000000541364253$$

Tighter tolerance

TOL := 0.000001

$$\int_{1}^{e^5} \frac{1}{t}\,dt = 5.000000000007783$$

Looser tolerance

TOL := 0.1

$$\int_{1}^{e^5} \frac{1}{t}\,dt = 5.011533083011655$$

Example 2

Comments

There are some important things to remember about integration in Mathcad:

- The limits of integration must be real, but the expression to be integrated can be either real or complex.
- Except for the integrating variable, all variables in the integrand must have been defined elsewhere in the worksheet.
- The integrating variable must be a single variable name.
- If the integrating variable involves units, the upper and lower limits of integration must have the same units.

Like all numerical methods, Mathcad's integration algorithm can have difficulty with ill-behaved integrands. If the expression to be integrated has singularities, discontinuities, or large and rapid fluctuations, Mathcad's solution may be inaccurate.

In some cases, you may be able to find an exact expression for your definite integral or even the indefinite integral (antiderivative) by using Mathcad’s symbolics.

Although the result of an integration is a single number, you can always use an integral with a range variable to obtain results for many numbers at once (as illustrated in Example 1). Such repeated evaluations may take considerable time depending on the complexity of the integrals, the length of the interval, and the value of TOL.

Mathcad's numerical integration algorithm makes successive estimates of the value of the integral and returns a value when the two most recent estimates differ by less than the value of the built-in variable TOL. Example 2 above shows how changing TOL affects the accuracy of integral calculations. (This is not to be confused with the mere formatting issue of how many digits to display.)

You can change the value of the tolerance by including definitions for TOL directly in your worksheet as shown. You can also change the tolerance by using the Built-In Variables tab when you choose **Options** from the **Math** menu. To see the effect of changing the tolerance, choose **Calculate Document** from the **Math** menu to recalculate all the equations in the worksheet.

If Mathcad's approximations to an integral fail to converge to an answer, Mathcad marks the integral with an appropriate error message.

When you change the tolerance, keep in mind the trade-off between accuracy and computation time. If you decrease (tighten) the tolerance, Mathcad will compute integrals more accurately. However, because this requires more work, Mathcad will take longer to return a result. Conversely, if you increase (loosen) the tolerance, Mathcad will compute more quickly, but the answers will be less accurate.

You can also use Mathcad to evaluate double or multiple integrals. To set up a double integral, press the ampersand key, **[&]**, twice. Fill in the integrand, the limits, and the integrating variable for each integral. Keep in mind that double integrals take much longer to converge to an answer than single integrals. Wherever possible, use an equivalent single integral in place of a double integral.

Because certain numerical integration methods work best on certain kinds of integrals, Mathcad has an AutoSelect feature for integration. Depending on the kind of integral you are evaluating, Mathcad automatically chooses the most appropriate integration method to use. Using AutoSelect, Mathcad examines the integral and evaluates it using one of the following methods:

- Romberg (Romberg trapezoidal approximation with Richard extrapolation – equal intervals)
- Adaptive (if the values of *f(x)* vary significantly over the interval – unequal intervals)
- Infinite Limit (if $a = -\infty$, $b = \infty$ or both)
- Singular Endpoint (if *f(a)* and/or *f(b)* is undefined)

In Mathcad Professional, if you want to evaluate an integral using a method other than the one chosen during the AutoSelect process, turn off AutoSelect and choose another method. To do so:

1. Type the integral and allow AutoSelect to return a result.
2. Click on the integral with the right mouse button.
3. Click on the method you want to use.

The integral is automatically re-evaluated using the method you clicked and is re-evaluated that way from then on, unless you specify another method or AutoSelect later.

Algorithm

Romberg, Kahan transform, QAGS, Clenshaw-Curtis, Gauss-Kronrod formulas (Piessens 1983, Lorczak)

Indefinite integral $\int f(t)\,dt$

Keystroke **[Ctrl]i** Toolbar Button $\int$

Description

Returns the indefinite integral (that is, an antiderivative) of *f*(*t*). Must be performed symbolically. The integrand, *f*(*t*), cannot return an array.

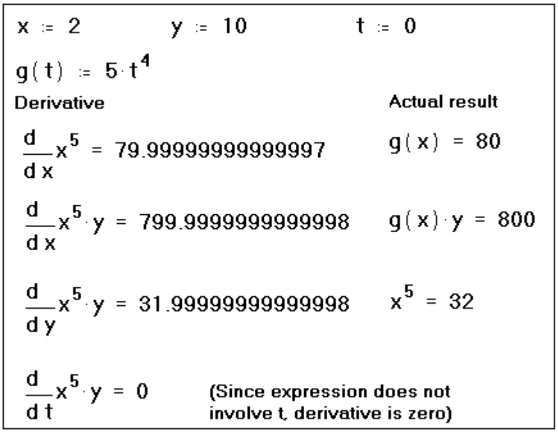
Derivative $\frac{d}{dt}f(t)$

Keystroke **?** Toolbar Button $\frac{d}{dx}$

Description

Returns the derivative of *f*(*t*) evaluated at *t*. All variables in the expression *f*(*t*) must be defined. The variable *t* must be a scalar value. The function *f*(*t*) must return a scalar.

Example



$x := 2$ $y := 10$ $t := 0$

$g(t) := 5 \cdot t^4$

Derivative **Actual result**

$\frac{d}{dx}x^5 = 79.99999999999997$ $g(x) = 80$

$\frac{d}{dx}x^5 \cdot y = 799.9999999999998$ $g(x) \cdot y = 800$

$\frac{d}{dy}x^5 \cdot y = 31.99999999999998$ $x^5 = 32$

$\frac{d}{dt}x^5 \cdot y = 0$ (Since expression does not involve t, derivative is zero)

Comments

With Mathcad's derivative algorithm, you can expect the first derivative to be accurate to within 7 or 8 significant digits, provided that the value at which you evaluate the derivative is not too close to a singularity of the function. The accuracy of this algorithm tends to decrease by one significant digit for each increase in the order of the derivative (see nth derivative operator).

The result of differentiating is not a function, but a single number: the computed derivative at the indicated value of the differentiation variable. In the previous example, the derivative of x^3 is not the expression $3x^2$ but $3x^2$ evaluated at $x = 2$. If you want the expression $3x^2$, you will need to use either live or menu symbolics.

Although differentiation returns just one number, you can still define one function as the derivative of another. For example: $f(x) := \frac{d}{dx}g(x)$. Evaluating *f*(*x*) will return the numerically computed derivative of *g*(*x*) at *x*. You can use this technique to evaluate the derivative of a function at many points via range variables.

Algorithm

Modified Ridder’s method (Press *et al.*, 1992; Lorczak)

*n*th derivative $\frac{d^n}{dt^n}f(t)$

Keystroke **`[Ctrl]?`** Toolbar Button $\frac{d^n}{dx^n}$

Description

Returns the *n*th derivative of *f*(*t*) evaluated at *t*. All variables in *f*(*t*) must be defined. The variable *t* must be a scalar value. The function *f*(*t*) must return a scalar. *n* must be an integer between 0 and 5 for numerical evaluation or a positive integer for symbolic evaluation.

Comments

For $n = 1$, this operator gives the same answer as the Derivative operator. For $n = 0$, it simply returns the value of the function itself.

Algorithm

Modified Ridder’s method (Press *et al.*, 1992; Lorczak)

Limit $\lim_{t \to a} f(t)$

Keystroke **`[Ctrl]L`** Toolbar Button lim →a

Description Returns the two-sided limit of *f*(*t*). Must be evaluated symbolically.

Algorithm Series expansion (Geddes and Gonnet, 1989)

Right-Hand Limit $\lim_{t \to a^+} f(t)$

Keystroke **`[Ctrl][Shift]A`** Toolbar Button lim →a+

Description Returns the right-hand limit of *f*(*t*). Must be evaluated symbolically.

Algorithm Series expansion (Geddes and Gonnet, 1989)

Left-Hand Limit $\lim_{t \to a^-} f(t)$

Keystroke **`[Ctrl][Shift]B`** Toolbar Button lim →a-

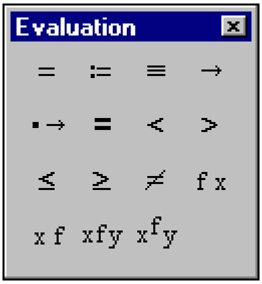
Description Returns the left-hand limit of *f*(*t*). Must be evaluated symbolically.

Algorithm Series expansion (Geddes and Gonnet, 1989)

Evaluation and Boolean Operators

To access an Evaluation or Boolean operator:

- type its keystroke, or
- choose the operator from the Evaluation and Boolean toolbar:



Refer to “Accessing Operators” on page 135 for more information on how to access a toolbar.

Greater than $x > y$, $S1 > S2$

Keystroke **>** Toolbar Button >

Description For real scalars *x* and *y*, returns 1 if $x > y$, 0 otherwise.
For string expressions *S*1 and *S*2, returns 1 if *S*1 strictly follows *S*2 in ASCII order, 0 otherwise.

Less than $x < y$, $S1 < S2$

Keystroke **<** Toolbar Button <

Description For real scalars *x* and *y*, returns 1 if $x < y$, 0 otherwise.
For string expressions *S*1 and *S*2, returns 1 if *S*1 strictly precedes *S*2 in ASCII order, 0 otherwise.

Greater than or equal to $x \geq y$, $S1 \geq S2$

Keystroke **[Ctrl])** Toolbar Button ≥

Description For real scalars *x* and *y*, returns 1 if $x \geq y$, 0 otherwise.
For string expressions *S*1 and *S*2, returns 1 if *S*1 follows *S*2 in ASCII order, 0 otherwise.

Less than or equal to $x \leq y$, $S1 \leq S2$

Keystroke **[Ctrl](** Toolbar Button ≤

Description For real scalars *x* and *y*, returns 1 if $x \leq y$, 0 otherwise.
For string expressions *S*1 and *S*2, returns 1 if *S*1 precedes *S*2 in ASCII order, 0 otherwise.

Not equal to $z \neq w$, $S1 \neq S2$

Keystroke `[Ctrl]3` Toolbar Button ≠

Description

For scalars z and w, returns 1 if $z \neq w$, 0 otherwise.
For string expressions $S1$ and $S2$, returns 1 if $S1$ is not character by character identical to $S2$, 0 otherwise.

Bold Equals $z = w$

Keystroke `[Ctrl]=` Toolbar Button **=**

Description

Returns 1 if $z = w$, 0 otherwise (also known as Boolean equals). Appears as a bold = on the screen. Also used when typing constraint equations within solve blocks or when typing equations to be solved symbolically.

Equals $c =$

Keystroke **=** Toolbar Button =

Description

Returns numerical value of c if c is: a variable previously defined in the worksheet; a built-in variable; a globally-defined variable; or a function of several such variables. Appears as an ordinary = on the screen. Not used for symbolic evaluation.

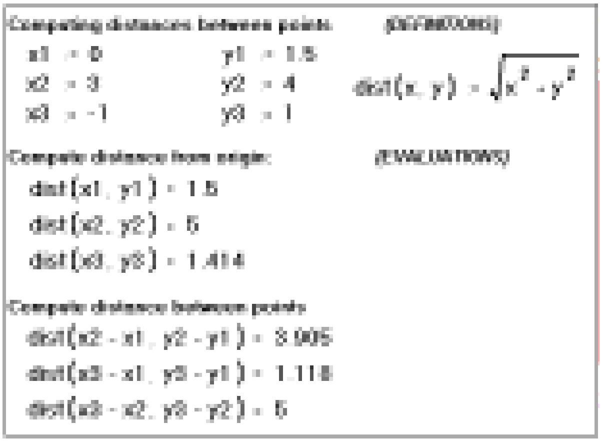
Definition $z := c$, $f(x,y,z,...) := expr$

Keystroke **:** Toolbar Button :=

Description

Gives z the numerical value c from that point onward throughout the worksheet. Gives a function $f(x,y,z,...)$ the meaning prescribed by the expression *expr* from that point onward throughout the worksheet. *expr* need not involve $x, y, z, ...$ but it usually does; it may involve other built-in or user-defined functions.

Examples



Computing distances between points (DEFINITIONS)

$x1 := 0$ $y1 := 1.5$

$x2 := 3$ $y2 := 4$ $\mathrm{dist}(x, y) := \sqrt{x^2 + y^2}$

$x3 := -1$ $y3 := 1$

Compute distance from origin: (EVALUATIONS)

$\mathrm{dist}(x1, y1) = 1.5$

$\mathrm{dist}(x2, y2) = 5$

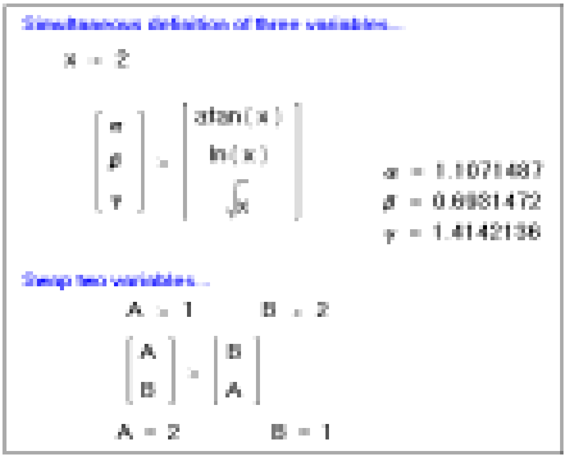
$\mathrm{dist}(x3, y3) = 1.414$

Compute distance between points

$\mathrm{dist}(x2 - x1, y2 - y1) = 3.905$

$\mathrm{dist}(x3 - x1, y3 - y1) = 1.118$

$\mathrm{dist}(x3 - x2, y3 - y2) = 5$



Simultaneous definition of three variables...

$x := 2$

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} := \begin{vmatrix}\operatorname{atan}(x)\\ \ln(x)\\ \sqrt{x}\end{vmatrix}$$

$\alpha = 1.1071487$
$\beta = 0.6931472$
$\gamma = 1.4142136$

Swap two variables...

$A := 1 \qquad B := 2$

$$\begin{bmatrix}A\\ B\end{bmatrix} := \begin{vmatrix}B\\ A\end{vmatrix}$$

$A = 2 \qquad B = 1$

Comments

You can define arrays in the same way as scalars, with the array name **A** on the left side of a **:=**, and a corresponding array of values to the right.

You can likewise use arrays to define several variables at once, as the previous example shows. The left side of a simultaneous definition is an array whose elements are either names or subscripted variable names. The right side must be an array of values having the same number of rows and columns as the left side. Mathcad defines each variable on the left side with the value of the array in the corresponding position on the right side. Elements on the right side are all evaluated before assigning any of them to the left side. Because of this, nothing on the right side of an expression can depend on what is on the left side. You also cannot have a variable appear more than once on the left side.

When you define a function, Mathcad does not try to evaluate it until you use it later on in the worksheet. If there is an error, the use of the function is marked in error, even though the real problem may be in the definition of the function itself. For example, if *f(x)* **:=** 1/*x* and you attempt to evaluate *f*(0), the error flag occurs not at the definition of *f*(x) but when Mathcad encounters *f*(0) for the first time.

Global Definition $z \equiv c$, $f(x, y, z, \dots) \equiv \text{expr}$

Keystroke **~** Toolbar Button ≡

Description

Gives *z* the numerical value *c* and this holds throughout the worksheet (regardless of where the global definition is positioned). Likewise, gives a function *f(x,y,z,...)* the meaning prescribed by the expression *expr* throughout the worksheet. *expr* need not involve *x*, *y*, *z*, ... but it usually does; it may involve other built-in or user-defined functions.

Comments

You can globally define arrays in the same way as scalars, with the array name **A** on the left side of a ≡, and a corresponding array of values to the right.

This is the algorithm that Mathcad uses to evaluate all definitions, global and otherwise:

- First, Mathcad takes one pass through the entire worksheet from top to bottom. During this first pass, Mathcad evaluates global definitions only.

- Mathcad then makes a second pass through the worksheet from top to bottom. This time, Mathcad evaluates all definitions made with **:=** as well as all equations containing ≡.

Although global definitions are evaluated before any local definitions, Mathcad evaluates global definitions the same way it evaluates local definitions: top to bottom and left to right. This means that whenever you use a variable to the right of a ≡:

- that variable must also have been defined with a ≡, *and*
- the variable must have been defined *above* the place where you are trying to use it.

Otherwise, the variable is marked in red to indicate that it is undefined.

It is good practice to allow only one definition for each global variable. Although you can define a variable with two different global definitions or with one global and one local definition, this is never necessary and usually makes your worksheet difficult to understand.

Symbolic Equals $c \rightarrow$

Keystroke **`[Ctrl].`** Toolbar Button →

Description Returns live symbolic "value" of *c* if *c* is a variable previously defined in the worksheet, is a built-in variable, is a globally-defined variable, or is a function of several such variables.

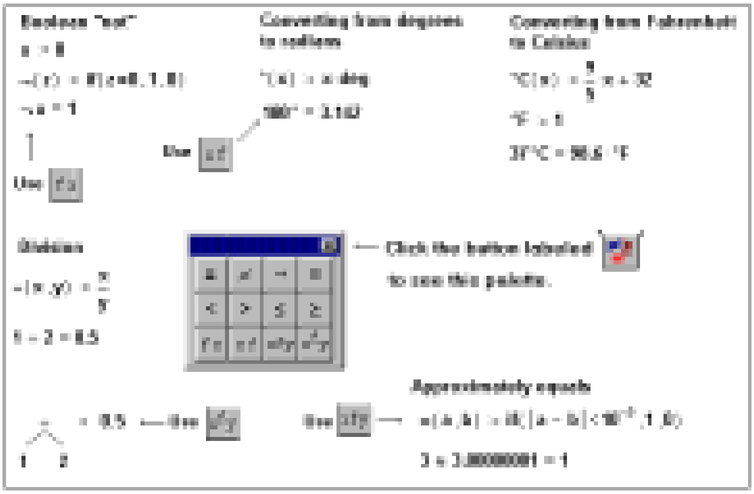
Comments The live symbolic equals sign is analogous to the numerical equals sign "=" and is capable of giving expressions. You can use it, for example, to symbolically simplify or factor algebraic expressions, or to symbolically evaluate derivatives, integrals and limits. Note that "→" applies only to an entire expression (unlike menu symbolics).

Prefix *f x* *(Professional)*

Keystroke **`NONE`** Toolbar Button f x

Description Using the prefix custom operator, *f x* returns the value *f(x)*, where *f* is either a built-in or user-defined function and *x* is a real or complex number.

Examples



Boolean "not"

$x := 0$

$\neg(x) := \mathrm{if}(x=0, 1, 0)$

$\neg x = 1$

Use f x

Converting from degrees to radians

$°(x) := x \cdot \mathrm{deg}$

$180° = 3.142$

Use x f

Converting from Fahrenheit to Celsius

$°C(x) := \frac{9}{5} \cdot x + 32$

$°F := 1$

$37°C = 98.6 \cdot °F$

Division

$\div(x,y) := \frac{x}{y}$

$1 \div 2 = 0.5$

← Click the button labeled to see this palette.

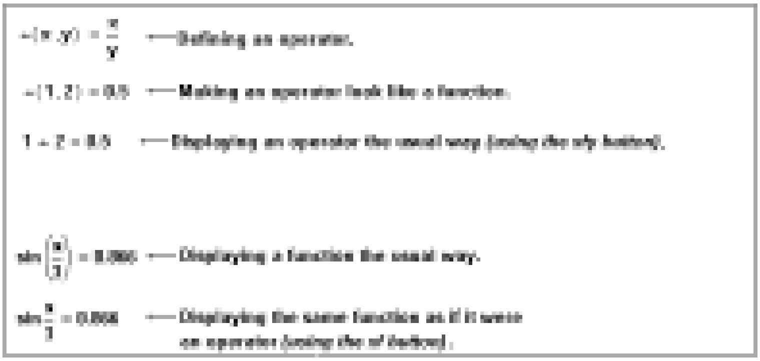
$\div$ 1 2 = 0.5 ← Use

Approximately equals

Use x f y → $\approx(a,b) := \mathrm{if}(|a-b| < 10^{-5}, 1, 0)$

$3 \approx 3.00000001 = 1$

Example 1: Defining your own operators



$\div(x,y) := \frac{x}{y}$ ← Defining an operator.

$\div(1,2) = 0.5$ ← Making an operator look like a function.

$1 \div 2 = 0.5$ ← Displaying an operator the usual way (using the xfy button).

$\sin\left(\frac{\pi}{3}\right) = 0.866$ ← Displaying a function the usual way.

$\sin\frac{\pi}{3} = 0.866$ ← Displaying the same function as if it were an operator (using the xf button).

Example 2: Displaying an operator as a function and a function as an operator

Comments

In Example 1, the symbol “¬” comes from the Symbol font. First define a function “¬(*x*)” as illustrated, then click the prefix button on the Evaluation toolbar to use prefix notation. For prefix notation, type the name of the operator in the left placeholder and the operand in the right placeholder.

Many publishers prefer to omit parentheses around the arguments to certain functions (*sin* x rather than $\sin(x)$). You can do the same thing by treating the *sin* function as an operator with one operand, as in Example 2.

Postfix

x f *(Professional)*

Keystroke **NONE** Toolbar Button x f

Description

Using the postfix custom operator, *x f* returns the value *f(x)*, where *f* is either a built-in or user-defined function and *x* is a real or complex number.

Comments

In Example 1, on page 155, the symbol “°” comes from the Symbol font. First define a function “°(*x*)” as illustrated, then click the postfix button on the Evaluation toolbar to use postfix notation. For postfix notation, type the name of the operator in the right placeholder and the operand in the left placeholder.

Infix

x f y *(Professional)*

Keystroke **NONE** Toolbar Button xfy

Description

Using the infix custom operator, *x f y* returns the value *f(x,y)*, where *f* is either a built-in or user-defined function and *x, y* are real or complex numbers.

Comments

In Example 1, on page 155, the symbol “≈” comes from the Symbol font. First define a binary function “≈(*x,y*)” as illustrated, then click the infix button on the Evaluation toolbar to use infix notation. For infix notation, type the name of the operator in the middle placeholder and the operands in the left and right placeholders.

Likewise, in Example 2, on page 155, the binary function “÷(*x,y*)” is defined and then displayed in the more conventional manner: “*x*÷*y*”. Functions and operators are fundamentally the same. Although notation like “÷(*x,y*)” is unconventional, use it if you prefer.

Treefix

$_{x}f_{y}$ *(Professional)*

Keystroke **NONE** Toolbar Button $_{x}f_{y}$

Description

Using the treefix custom operator, $_{x}f_{y}$ returns the value *f(x,y)*, where *f* is either a built-in or user-defined function and *x* and *y* are real or complex numbers.

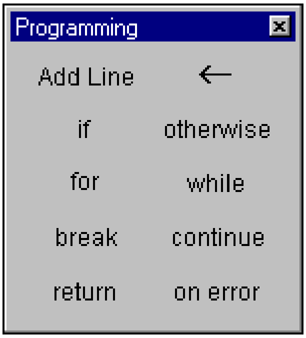
Comments

In Example 1, on page 155, the symbol “÷” comes from the Symbol font. First define a binary function “÷(*x,y*)” as illustrated, then click the treefix button on the Evaluation toolbar to use treefix notation. For treefix notation, type the name of the operator in the middle placeholder and the operands in the left and right placeholders.

Programming Operators

To access a Programming operator:

- type its keystroke, or
- choose the operator from the Programming toolbar:



Refer to “Accessing Operators” on page 135 for more information on how to access a toolbar

Special Note: these operators are valid only within a Mathcad programming structure.

Local Definition $w \leftarrow f(a, b, c, \ldots)$ *(Professional)*

Keystroke **{** Toolbar Button ←

Description Gives *w* the numerical value of the function *f(a,b,c,...)* within a program. Outside the program, *w* remains undefined.

Add Line *(Professional)*

Keystroke **]** Toolbar Button Add Line

Description Inserts a line in a program. When you insert the Add Line operator the first time, a program is created (a vertical bar with two placeholders). If you select either of these placeholders and insert the Add Line operator again, more placeholders are created.

Conditional Statement ▮ if ▮ *(Professional)*

Keystroke **}** Toolbar Button if

Description Within a program, permits evaluation of a statement only when a specified condition is met. You must insert this operator using its keystroke or toolbar button. (Conditional if is not the same as the built-in if function. Do not just type the word “if”.)

Otherwise Statement ■ otherwise

(Professional)

Keystroke **NONE** Toolbar Button [otherwise]

Description Within a program, used in conjunction with the if statement to exhaust possibilities not yet covered. You must insert this operator using its toolbar button. (Do not just type the word “otherwise”.)

For Loop for ■ ∈ ■

■

(Professional)

Keystroke **NONE** Toolbar Button [for]

Description Within a program, permits evaluation of a sequence of statements a specified number of times. The right hand placeholder usually contains a range variable. You must insert this operator using its toolbar button. (Do not just type the word “for”.)

While Loop while ■

■

(Professional)

Keystroke **NONE** Toolbar Button [while]

Description Within a program, permits evaluation of a sequence of statements until a specified condition is met. The right hand placeholder usually contains a Boolean expression. You must insert this operator using its toolbar button. (Do not just type the word “while”.)

Break Statement break

(Professional)

Keystroke **NONE** Toolbar Button [break]

Description Within a for or while loop, halts loop execution. Usually used in conjunction with an if statement, that is, halting occurs if a specified condition occurs. Execution moves to the next statement outside the loop. You must insert this operator using its toolbar button. (Do not just type the word “break”.)

See also continue and return

Continue Statement continue

(Professional)

Keystroke **NONE** Toolbar Button [continue]

Description Within a for or while loop, halts loop execution. Usually used in conjunction with an if statement, that is, halting occurs if a specified condition occurs. Execution moves to the beginning of the next iteration of the current loop. You must insert this operator using its toolbar button. (Do not just type the word “continue”.)

See also break and return

Return Statement return ▮ *(Professional)*

Keystroke **NONE** Toolbar Button return

Description

Within a program, halts program execution. Usually used in conjunction with an if statement, that is, halting occurs if a specified condition occurs. Also, within a for or while loop, halts loop execution; execution in this case moves to the next statement outside the loop. You must insert this operator using its toolbar button. (Do not just type the word "return".)

See also

break and continue

Comments

The break function does the same as return, but gives the value only of the last expression evaluated before break is called. Therefore return is more flexible, because it allows you to interrupt the program and return particular values other than the default value last computed.

On Error Statement ▮ on error ▮ *(Professional)*

Keystroke **NONE** Toolbar Button on error

Description

Within a program, permits computation of an alternative expression when an arbitrary numerical error flag is raised. You must insert this operator using its toolbar button. (Do not just type the phrase "on error".)

Comments

on error executes the right-hand argument first. If no error occurs, it returns the result of the right argument. If an error occurs, then the error is cleared and the left argument is returned.

on error is a general purpose error trap. It is more powerful than using the return statement, coupled with some specific test, to deal with inputs that give rise to numerical error.