Chapter 12
Operators

Mathcad includes ordinary operators like + and /, matrix operators like transpose and determinant, and special operators like iterated sum, iterated product, integrals, and derivatives.

This chapter contains a list of Mathcad operators and describes how to enter and use the special operators.

This chapter contains the following sections:

List of operators

List of Mathcad's operators in order of precedence.

Summations and products

How to use Mathcad's summation and product operators.

Derivatives

How to use Mathcad's derivative operators.

Integrals

How to use Mathcad's definite integral operator.

Boolean operators

How to use Mathcad's boolean operators such as “>” and “<.”

Pro **Customizing operators**

How to define your own operators just the way you define your own functions.

List of operators

This is a list of Mathcad operators in order of precedence. For details on vector and matrix operators, see Chapter 10, “Vectors and Matrices.” Most of the following operators are available by clicking on one of the operator palettes, or by using the keystrokes listed in the table below. To open the operator palettes, click on the buttons on the Math Palette, which you can see by choosing **Math Palette** from the **View** menu:

In this table:

- **A** and **B** represent arrays, either vector or matrix.
- **u** and **v** represent vectors with real or complex elements.
- **M** represents a square matrix.
- *z* and *w* represent real or complex numbers.
- *x* and *y* represent real numbers.
- *m* and *n* represent integers.
- *i* represents a range variable.
- *S* and any names beginning with *S* represent string expressions.
- *t* represents any variable name.
- *f* represents a function.
- *X* and *Y* represent variables or expressions of any type.

Operation	Appearance	Keystroke	Description
Parentheses	(X)	`'`	Grouping operator.
Vector Subscript	$\mathbf{v}_n$	`[`	Returns indicated element of a vector.
Matrix Subscript	$\mathbf{A}_{m,n}$	`[`	Returns indicated element of a matrix.
Superscript	$\mathbf{A}^{\langle n \rangle}$	`[Ctrl]6`	Extracts column *n* from array **A**. Returns a vector.
Vectorize	$\vec{X}$	`[Ctrl]-`	Forces operations in expression *X* to take place element by element. All vectors or matrices in *X* must be the same size.

Operation	Appearance	Keystroke	Description
Factorial	$n!$	`!`	Returns $n \cdot (n-1) \cdot (n-2)\ldots$ The integer n cannot be negative.
Complex conjugate	$\bar{X}$	`"`	Inverts the sign of the imaginary part of X.
Transpose	$\mathbf{A}^{\mathsf{T}}$	`[Ctrl]1`	Returns a matrix whose rows are the columns of **A** and whose columns are the rows of **A**. **A** can be a vector or a matrix.
Power	z^w	`^`	Raises z to the power w.
Powers of matrix, matrix inverse	$\mathbf{M}^n$	`^`	nth power of square matrix **M** (using matrix multiplication). n must be a whole number. $\mathbf{M}^{-1}$ is the inverse of **M**. Other negative powers are powers of the inverse. Returns a square matrix.
Negation	$-X$	`-`	Multiplies X by -1.
Vector sum	$\Sigma\mathbf{v}$	`[Ctrl]4`	Sums elements of vector v; returns a scalar.
Square root	$\sqrt{z}$	`\`	Returns positive square root for positive z; principal value for negative or complex z.
nth root	$\sqrt[n]{z}$	`[Ctrl]\`	Returns nth root of z; returns a real valued root whenever possible.
Magnitude, Absolute value	$\lvert z\rvert$	`|`	Returns $\sqrt{\mathrm{Re}(z)^2 + \mathrm{Im}(z)^2}$.
Magnitude of vector	$\lvert\mathbf{v}\rvert$	`|`	Returns the magnitude of the vector **v**: $\sqrt{\mathbf{v} \cdot \mathbf{v}}$ if all elements in **v** are real. Returns $\sqrt{\mathbf{v} \cdot \bar{\mathbf{v}}}$ if any element in **v** is complex.
Determinant	$\lvert\mathbf{M}\rvert$	`|`	Returns the determinant of the square matrix **M**. Result is a scalar.
Division	$\frac{X}{z}$	`/`	Divides the expression X by the non-zero scalar z. If X is an array, divides each element by z.
Multiplication	$X \cdot Y$	`*`	Returns the product of X and Y if both X and Y are scalars. Multiplies each element of Y by X if Y is an array and X is a scalar. Returns the dot product (inner product) if X and Y are vectors of the same size. Performs matrix multiplication if X and Y are conformable matrices.
Cross product	$\mathbf{u} \times \mathbf{v}$	`[Ctrl]8`	Returns cross-product (vector product) for the three-element vectors **u** and **v**.
Summation	$\sum_{i=m}^{n} X$	`[Ctrl] [Shift]4`	Performs summation of X over $= m, m+1, \ldots, r$. X can be any expression. It need not involve i but it usually does. m and n must be integers.
Product	$\prod_{i=m}^{n} X$	`[Ctrl] [Shift]3`	Performs iterated product of X for $= m, m+1, \ldots, r$. X can be any expression. It need not involve i but it usually does. m and n must be integers.

Operation	Appearance	Keystroke	Description
Range sum	$\sum_i X$	**$**	Returns a summation of *X* over the range variable *i*. *X* can be any expression. It need not involve *i* but it usually does.
Range product	$\prod_i X$	**#**	Returns the iterated product of *X* over the range variable *i*. *X* can be any expression. It need not involve *i* but it usually does.
Integral	$\int_a^b f(t)\,dt$	**&**	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.
Derivative	$\frac{d}{dt}f(t)$	**?**	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.
nth Derivative	$\frac{d^n}{dt^n}f(t)$	`[Ctrl]?`	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.
Addition	$X + Y$	**+**	Scalar addition if *X*, *Y*, or both are scalars. Element by element addition if *X* and *Y* are vectors or matrices of the same size. If *X* is an array and *Y* is a scalar, adds *Y* to each element of *X*.
Subtraction	$X - Y$	**–**	Performs scalar subtraction if *X*, *Y*, or both are scalars. Performs element by element subtraction if *X* and *Y* are vectors or matrices of the same size. If *X* is an array and *Y* is a scalar, subtracts *Y* from each element of *X*.
Addition with line break	$X\ldots$ $+ Y$	`[Ctrl][↵]`	Same as addition. Line break is purely cosmetic.
Greater than	$x > y$, $S1 > S2$	**>**	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	$x < y$, $S1 < S2$	**<**	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	$x \ge y$, $S1 \ge S2$	`[Ctrl]0`	For real scalars *x* and *y*, returns 1 if $x \ge y$, 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	$x \le y$, $S1 \le S2$	`[Ctrl]9`	For real scalars *x* and *y*, returns 1 if $x \le y$, 0 otherwise. For string expressions *S*1 and *S*2, returns 1 if *S*1 precedes *S*2 in ASCII order, 0 otherwise.

Operation	Appearance	Keystroke	Description
Not equal to	$z \neq w$, $S1 \neq S2$	**[Ctrl]3**	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 $S2$.
Equal to	$X = Y$	**[Ctrl]=**	Returns 1 if $X = Y$, 0 otherwise. Appears as a bold = on the screen.

Help with typing operators

You can avoid having to remember the keystrokes that go with each operator by using the operator palettes. To open the operator palettes, click on the buttons on the Math Palette. Each button opens a palette of operators grouped loosely by function.

The icons on the operator palette buttons indicate what operator appears when you click on that button. You can also hold the mouse pointer momentarily over a button to see a tool tip indicating what the button does.

To type any operator from the table on the previous pages, just click wherever you want to put the operator, then click on its button on the appropriate operator palette.

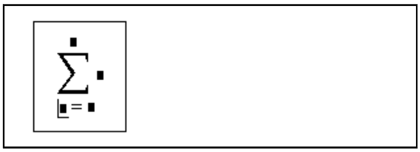
In general, operator palettes only work in math regions. To use the operator palettes in text, you must first click in the text and choose **Math Region** from the **Insert** menu. This will create a math placeholder in the text into which you can insert operators using the palettes.

Summations and products

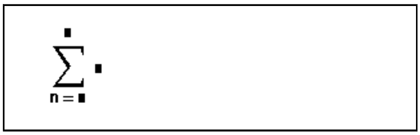
The summation operator sums an expression over all values of an index. The iterated product operator works much the same way. It takes the product of an expression over all values of an index.

To create a summation operator in your worksheet:

- Click in a blank space. Then type [**Ctrl**][**Shift**]**4**. A summation sign with four placeholders appears.



- In the placeholder to the left of the equal sign, type a variable name. This variable is the index of summation. It is defined only within the summation operator and therefore has no effect on, and is not influ-



enced by, variable definitions outside the summation operator.

- In the placeholder to the right of the equal sign, type an integer or any expression that evaluates to an integer.

$$\sum_{n=1}^{\blacksquare} \blacksquare$$

- In the single placeholder above the sigma, type an integer or any expression that evaluates to an integer.

$$\sum_{n=1}^{10} \blacksquare$$

- In the remaining placeholder, type the expression you want to sum. Usually, this expression will involve the index of summation. If this expression has several terms, type an apostrophe (`'`) to create a pair of parentheses around the placeholder.

$$\sum_{n=1}^{10} n^2$$

Iterated products are similar. Just type [**`Ctrl`**][**`Shift`**]**`3`** and fill in the placeholders as described earlier.

Figure 12-1 shows some examples of how to use the summation and product operators. You can use a summation or an iterated product just as you would any other expression.

To evaluate multiple summations, place another summation in the final placeholder of the first summation. An example of this appears at the bottom of Figure 12-1.

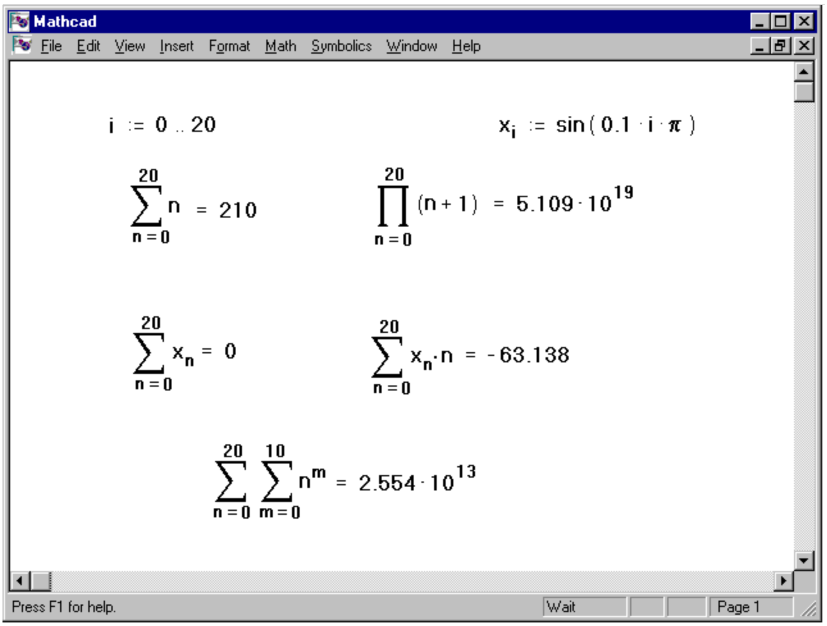
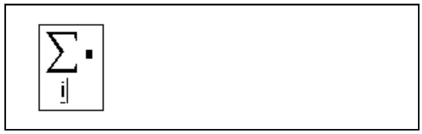


Figure 12-1: Summations and products.

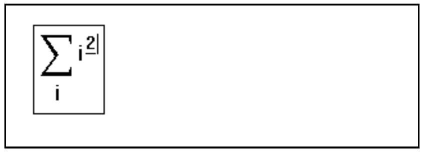
When you use the summation operator shown in Figure 12-1, the summation must be carried out over integers and in steps of one. Mathcad provides more general versions of these operators that can use any range variable you define as an index of summation. To use these operators, first define a range variable. In the following example type **`i:1,2;10`**. Then do the following:

- Click in a blank space. Then type **`$`**. A summation sign with two placeholders appears.

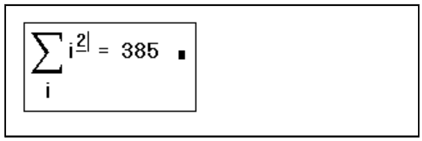
- Click on the bottom placeholder and type the name of a range variable. The range variable you use here should already have been defined earlier in the worksheet.



- Click on the placeholder to the right of the summation sign and type an expression involving the range variable. If this expression has several terms, type an apostrophe (`'`) to create a pair of parentheses around the placeholder.



- Press the equal sign (**`=`**) to see the result.



If you don't want to take the time to click in each placeholder, you can enter the previous expression by typing **`i$i^2`**.

A generalized version of the iterated product also exists. To use it, type **`#`**. Then fill in the two placeholders.

Figure 12-2 shows some examples of how to apply the range sum and range product operators. These operators, unlike the summation and product operators created with [**`Ctrl`**][**`Shift`**]**4** and [**`Ctrl`**][**`Shift`**]**3**, cannot stand alone. They require the existence of a range variable. Note however, that a single range variable can be used with any number of these operators.

You can use summations and iterated products just as you would any other expression. To evaluate multiple summations, use two range variables as shown in Figure 12-2.

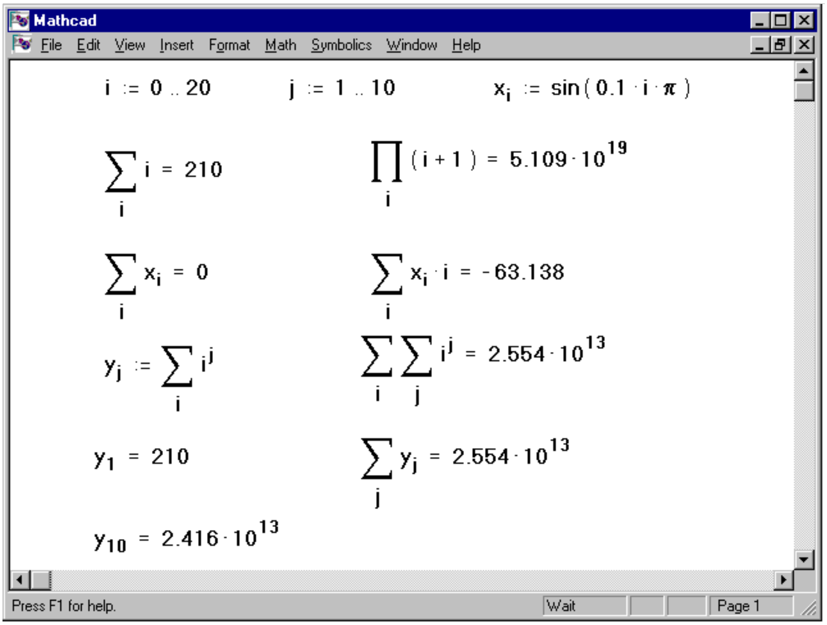


Figure 12-2: Range sums and range products.

Variable upper limit of summation

Mathcad's range summation operator runs through each value of the range variable you place in the bottom placeholder. It is possible, by judicious use of boolean expressions, to sum only up to a particular value. In Figure 12-3, the term $i \le x$ returns the value 1 whenever it is true and 0 whenever it is false. Although the summation operator still sums over each value of the index of summation, those terms for which $i > x$ are multiplied by 0 and hence do not contribute to the summation.

You can also use the four-placeholder summation and product operators to compute sums and products with a variable upper limit, but note that the upper limit in these operators must be an integer.

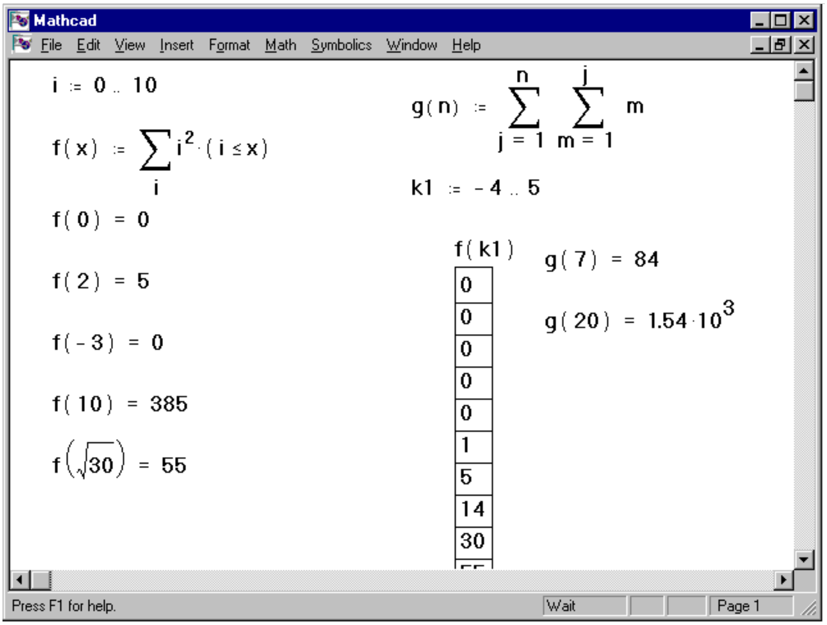


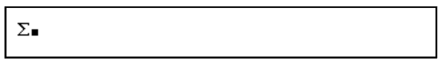
Figure 12-3: A variable upper limit of summation.

The vector-sum operator

The operation of summing the elements of a vector is so common that Mathcad provides a special operator for it. While the ordinary summation operator sums a ranged expression, the vector sum operator sums the elements of a vector without needing a range variable.

To sum all the elements of a vector **v** defined elsewhere in your worksheet, follow these steps:

- Click in blank space or on a placeholder. Then press [**Ctrl**]**4**.



- Type the name of a vector or vector-valued expression. Mathcad returns the sum of all the elements in the vector. In this example, the vector used is that shown in Figure 12-2.

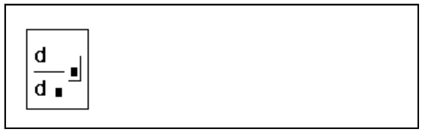
Derivatives

You can use Mathcad's derivative operator to evaluate the derivative of a function at a particular point.

As an example, here's how you would evaluate the derivative of x^3 with respect to x at the point $x = 2$:

- First define the point at which you want to evaluate the derivative. Type **x:2** .

- Click below the definition of *x*. Then type **?**. A derivative operator appears, with a placeholder in the denominator and another to the right.



- Click on the bottom placeholder and type **x**. You are differentiating with respect to this variable.

- Click on the placeholder to the right of the d/dx and type **x^3** . This is the expression to be differentiated.

- Press the equals sign **=** to see the derivative of the expression at the indicated point.

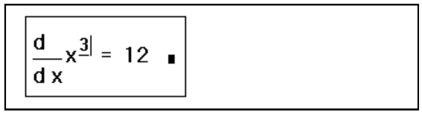


Figure 12-4 shows examples of differentiation in Mathcad.

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 the section “Derivatives of higher order” on page 251).

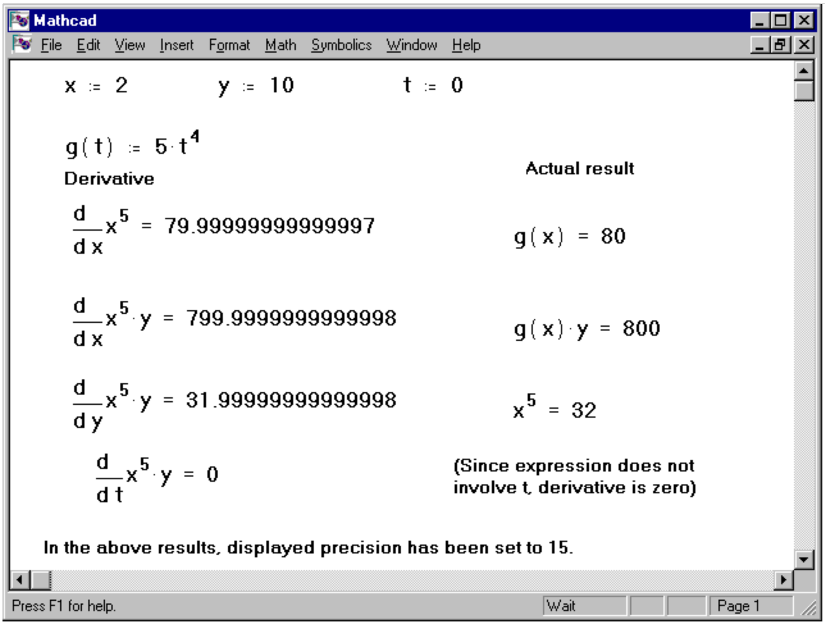


Figure 12-4: Examples of Mathcad differentiation.

Keep in mind that 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 to evaluate derivatives symbolically, see Chapter 17, "Symbolic Calculation."

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. An example of this is shown in Figure 12-5.

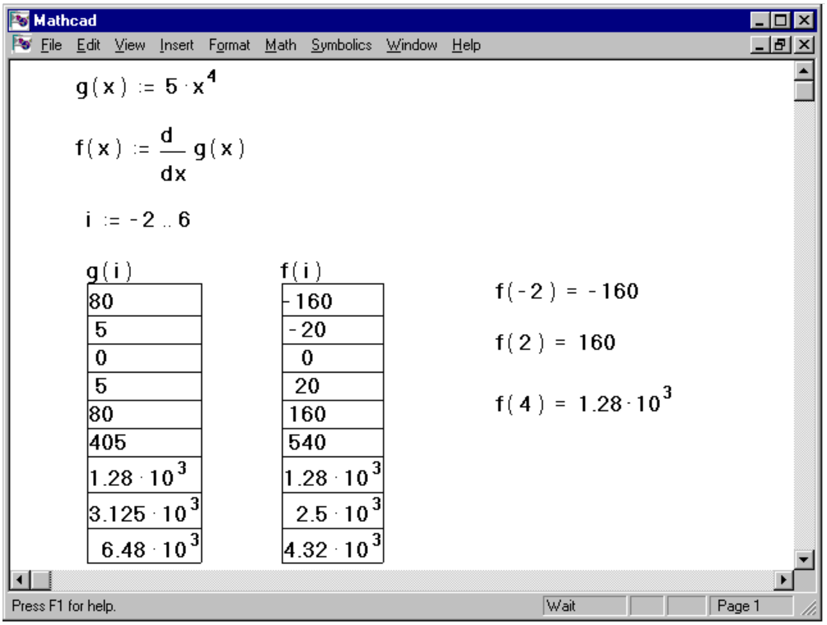


Figure 12-5: Evaluating the derivative of a function at several points.

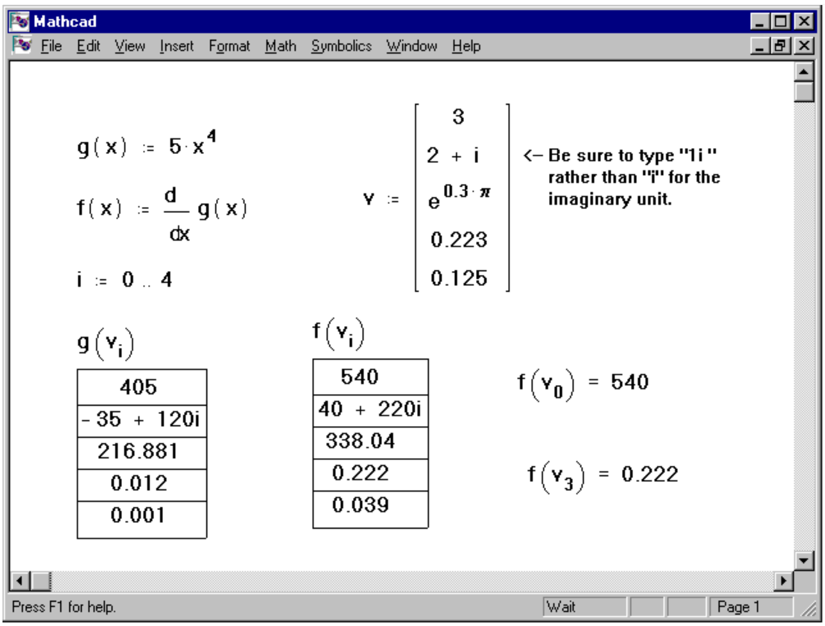


Figure 12-6: Evaluating the derivative of a function at several values stored as elements of a vector.

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

- The expression to be differentiated can be either real or complex.
- The differentiation variable must be a single variable name. If you want to evaluate the derivative at several different values stored in a vector, use the technique illustrated in Figure 12-6.

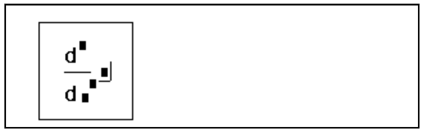
Derivatives of higher order

Mathcad has an additional derivative operator for evaluating the *n*th order derivative of a function at a particular point.

As an example, here's how you would evaluate the third derivative of x^9 with respect to x at the point $x = 2$:

- First define the point at which you want to evaluate the derivative. Type **x:2**.

- Click below the definition of *x*. Then type **[Ctrl]?**. A derivative operator appears, with two placeholders in the denominator, one in the numerator, and another to the right.

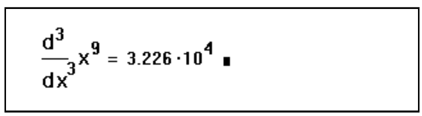


- Click on the bottom placeholder and type **x**. You are differentiating with respect to this variable.

- Click on the expression above and to the right of the previous placeholder and type **3**. This must be an integer between 0 and 5 inclusive. Note that the placeholder in the numerator automatically mirrors whatever you've typed.

- Click on the placeholder to the right of the d/dx and type **x^9**. This is the expression to be differentiated.

- Press the equal sign (**=**) to see the third derivative of the expression at the indicated point.



For $n = 1$, this operator gives the same answer as the first-derivative operator discussed above. For $n = 0$, it simply returns the value of the function itself.

Integrals

You can use Mathcad's integral operator to numerically evaluate the definite integral of a function over some interval.

As an example, here's how you would evaluate the definite integral of $\sin(x)^2$ from 0 to $\pi/2$. Follow these steps:

- Click in a blank space and type **&**. An integral appears, with placeholders for the integrand, the limits of integration, and the variable of integration.

$$\int_{\blacksquare}^{\blacksquare} \blacksquare \, d\blacksquare$$

- Click on the bottom placeholder and type **0**. Click on the top placeholder and type **[Ctrl]p/4**. These are the upper and lower limits of integration.

$$\int_{0}^{\frac{\pi}{4}} \blacksquare \, d\blacksquare$$

- Click on the placeholder between the integral sign and the “d.” Then type **sin(x)^2**. This is the expression to be integrated.

$$\int_{0}^{\frac{\pi}{4}} \sin(x)^2 \, d\blacksquare$$

- Click on the remaining placeholder and type **x**. This is the variable of integration. Then press the equal sign (**=**) to see the result.

$$\int_{0}^{\frac{\pi}{4}} \sin(x)^2 \, dx = 0.143$$

Mathcad uses a numerical algorithm called *Romberg integration* to approximate the integral of an expression over an interval of real numbers.

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

- The limits of integration must be real. The expression to be integrated can, however, 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.

Because Mathcad's integration method divides the interval into four subintervals and then successively doubles the number of points, it can return incorrect answers for periodic functions with having periods $1/2^n$ times the length of the interval. To avoid this problem, divide the interval into two uneven subintervals and integrate over each subinterval separately.

In some cases, you may be able to find an exact numerical value for your integral by using Mathcad's symbolic integration capability. You can also use this capability to evaluate indefinite integrals. See Chapter 17, “Symbolic Calculation.”

Variable limits of integration

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. You might do this, for example, when you set up a variable limit of integration. Figure 12-7 shows how to do this.

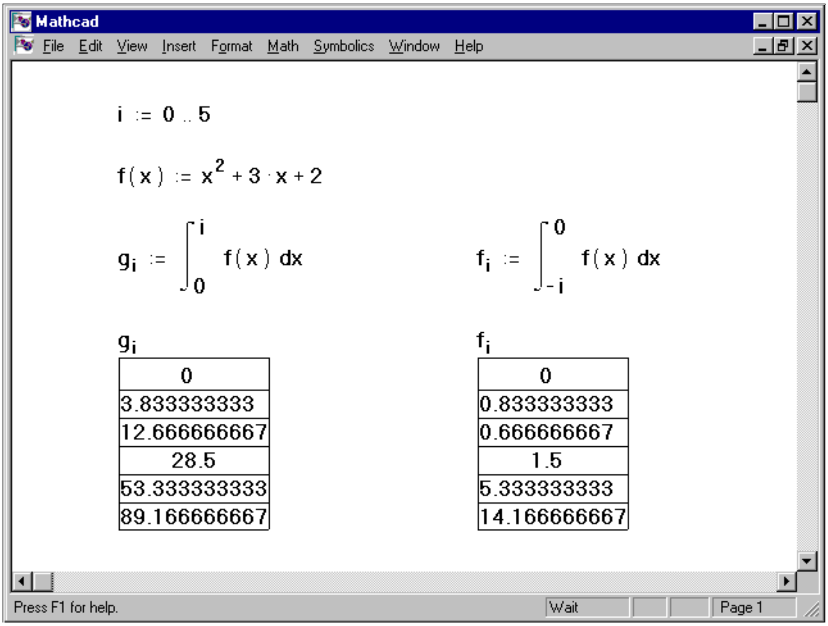


Figure 12-7: Variable limits of integration.

Keep in mind that calculations such as those shown in Figure 12-7 may require repeatedly evaluating an integral. This may take considerable time depending on the complexity of the integrals, the length of the interval, and the value of TOL (see below).

Changing the tolerance for integrals

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. Figure 12-8 shows how changing TOL affects the accuracy of integral calculations. To display many digits of precision, see Chapter 6, “Equation and Result Formatting.”

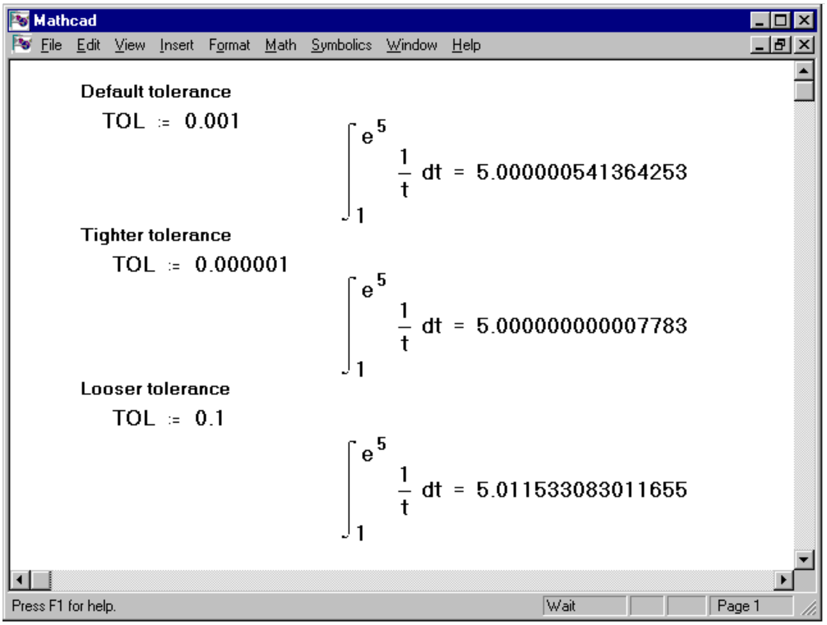


Figure 12-8: Effects of tolerance on integral calculations.

You can change the value of the tolerance by including definitions for TOL directly in your worksheet as shown on Figure 12-8. 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. Failure to converge can occur when the function has singularities or “spikes” in the interval or when the interval is extremely long.

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.

Contour integrals and double integrals

You can use Mathcad to evaluate complex contour integrals. To do so, first parametrize the contour. Then integrate over the parameter. If the parameter is something other than arc length, you must also include the derivative of the parametrization as a correction factor. Figure 12-9 shows an example. Note that the imaginary unit *i* used in specifying the path must be typed as **`1i`**.

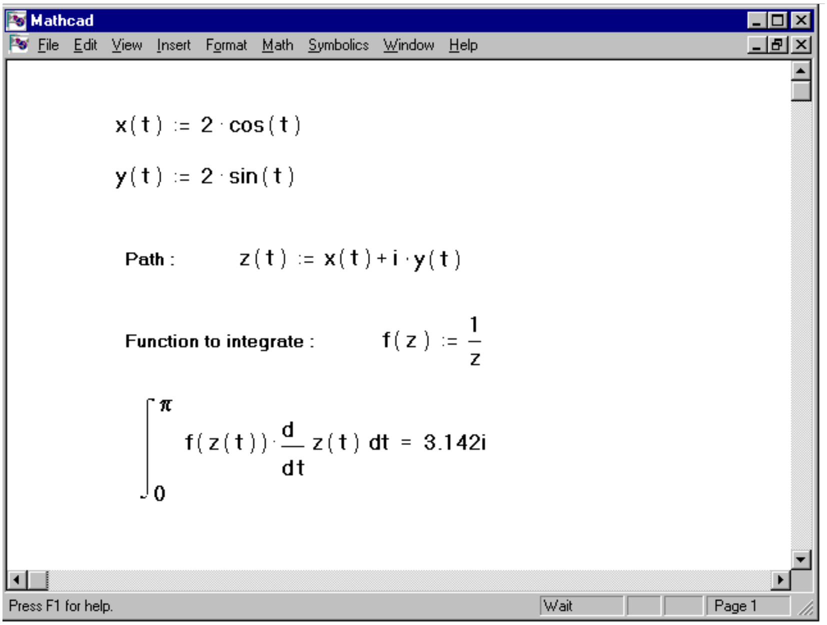


Figure 12-9: How to do a complex contour integral in Mathcad.

You can also use Mathcad to evaluate double or multiple integrals. To set up a double integral, press **&** twice. Fill in the integrand, the limits, and the integrating variable for each integral. Figure 12-10 shows an example.

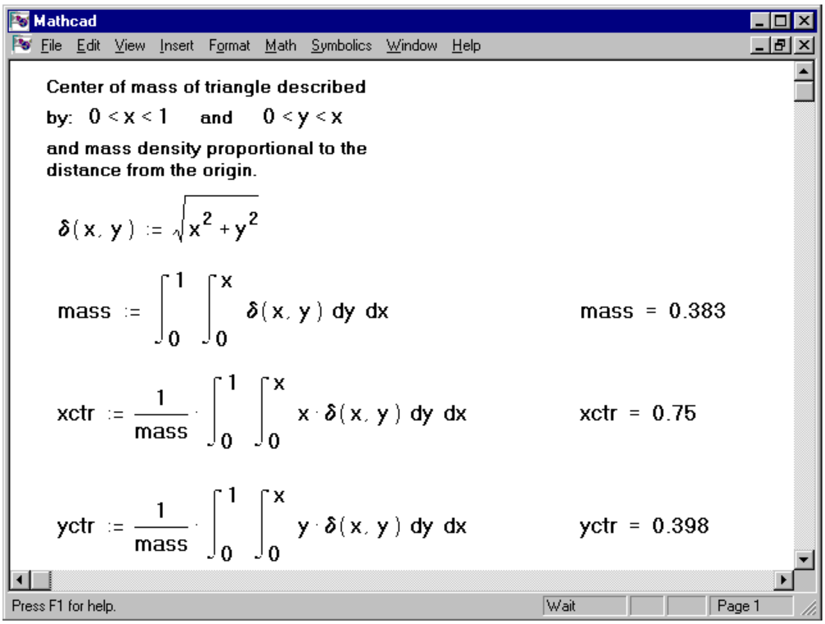


Figure 12-10: Double integrals.

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.

Boolean operators

Unlike other operators, the boolean operators can return only a zero or a one. Despite this, they can be very useful. You have already seen an example in Figure 12-3 showing how a boolean operator made a variable upper limit of summation possible. Chapter 12, "Operators," shows how a boolean operator makes it possible to determine the array index of a particular element.

The following table lists the boolean operators and their meaning with numbers:

Condition	How to type	Description
$w = z$	**[Ctrl]=**	Boolean equals. Returns 1 if expressions are equal; otherwise 0.
$x > y$	**>**	Greater than.
$x < y$	**<**	Less than.
$x \geq y$	**[Ctrl]0**	Greater than or equal to.
$x \leq y$	**[Ctrl]9**	Less than or equal to.
$w \neq z$	**[Ctrl]3**	Not equal to.

The four operators >, <, $\leq$, and $\geq$ cannot take complex numbers because the concepts of greater than and less than lose their meaning in the complex plane.

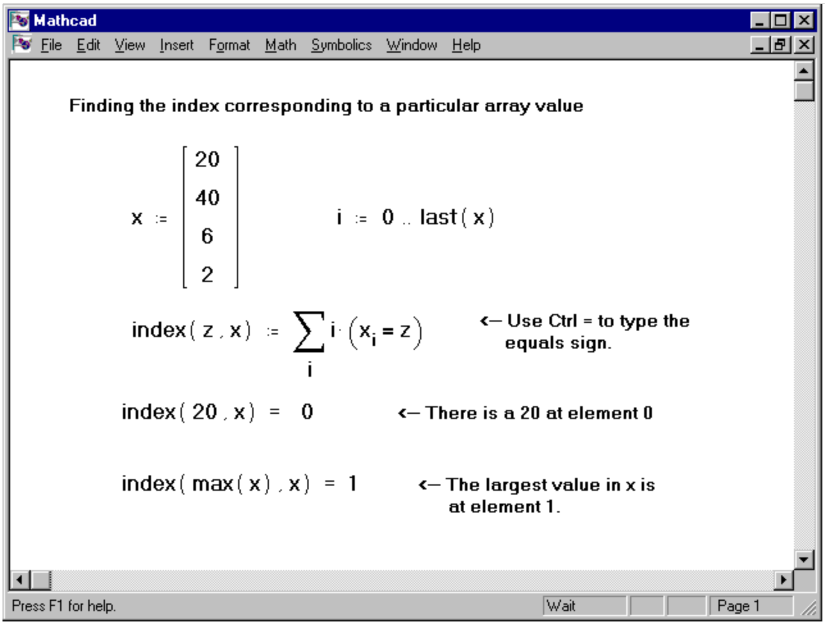


Figure 12-11: Using boolean operators.

As shown in Figure 12-12, the boolean operators can also be used to compare string expressions.

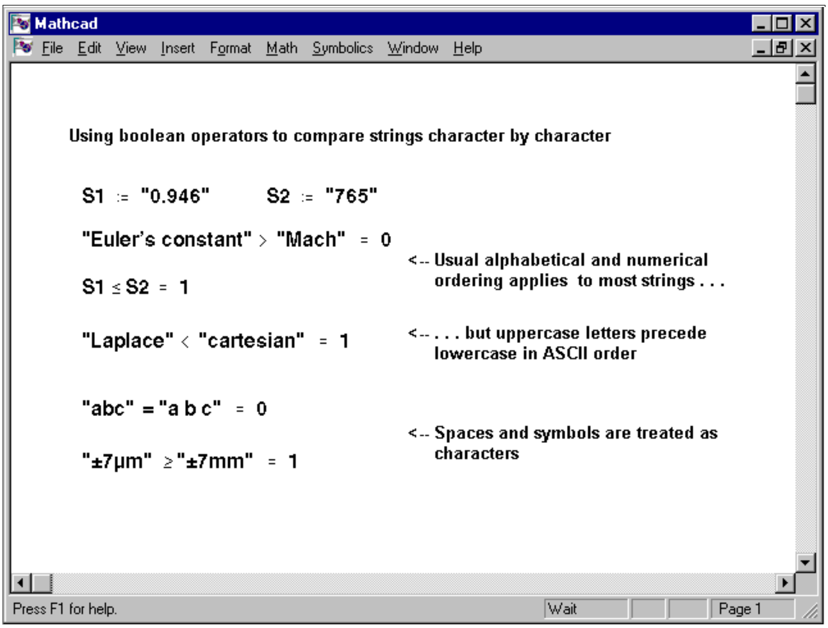


Figure 12-12: Comparing strings using boolean operators.

Mathcad compares two strings character by character by determining the ASCII codes of the component characters. For example, the string “Euler” precedes the string “Mach” in ASCII order and so the expression (“Euler”<“Mach”) evaluates to 1. See the table of ASCII codes in Appendix A, “Reference.” to determine the character ordering Mathcad uses in comparing strings. Using a boolean operator to compare a string to a number produces a type mismatch error.

Customizing operators

You can think of operators and functions as really being the same thing. A function takes “arguments” and returns a result. An operator, likewise, takes “operands” and returns a result. The differences are merely cosmetic:

- Functions have names you can spell, like *tan* or *spline*; operators are generally symbols like “+” or “×”.
- Arguments to a function are enclosed by parentheses, they come after the function's name, and they're separated by commas. Operands on the other hand, can appear elsewhere. For example, you'll often see $f(x, y)$ but you'll rarely see $x\ f\ y$. Similarly, you'll often find “$x + y$” but you'll rarely find “$+(x, y)$”.

Since operators and functions are fundamentally the same, and since you can define your own functions, there's no reason why you can't define your own customized operators as well. With Mathcad Professional, you'll be able to do just that.

The first section describes how to define a new operator. This is followed by a section on how to use the operator you've just defined. The last section brings together these ideas by showing how functions can themselves be displayed as if they were operators.

Defining a custom operator

You define an operator just as if you were defining a function. You'd type the operator name followed by a pair of parentheses. The operands (two at the most) would go between the parentheses. On the other side of the **:=** you'd type an expression describing what you want the operator to do with its operands. These steps are described in detail in the section “Defining variables and functions” in Chapter 7.

Since operators tend to have names that aren't found on a keyboard, a problem arises when you try and type the name. For example, suppose you want to define a new division operator using “ ÷ ”. You first have to know how to put a “ ÷ ” into your worksheet. The simplest way to do this is to drag the symbol from the “Math Symbols” QuickSheet.

We recommend that you save your custom operators by dragging them into a QuickSheet. Open the QuickSheets from the Resource Center as described in Chapter 2, “On-line Resources.” Then click on “Personal QuickSheets” from the topics in the table of contents. Click on “My Operators.” Then drag the definitions into the this QuickSheet.

The next time you need them, you'll be able to drag them off the same QuickSheet rather than having to redefine them.

■ When you paste the character, it will appear in the default math font as shown on the right.

■ To see the " ÷ ," you'll need to change this into the Symbol font. Press the **[Ins]** key if necessary to move the vertical arm of the insertion point directly in front of the character as shown.

■ Press **[Ctrl]G** to display the character in the Symbol font.

You can now continue as if you were defining a function of two variables that happens to have an unusual looking "name".

■ Type a left parenthesis followed by two names separated by a comma. Complete this argument list by typing a right parenthesis.

■ Press the colon (**:**) key. You see the definition symbol, "**:=**," followed by a placeholder.

■ Type the function definition in the placeholder.

At this point, you've defined a function which behaves in every way like the user-defined functions described in Chapter 7, "Equations and Computation." You could, if you wanted to, type " ÷(1, 2) = " in your worksheet and see the result "0.5" on the other side of the equal sign.

The difference between functions and operators lies not so much in the way they're defined but in the way they're displayed. This is discussed further in the next section.

Using a custom operator

Once you've defined a new operator, you can use it in your calculations just as you would use one of Mathcad's built-in operators. You can't, however, just type the name of your operator since Mathcad has no way of knowing whether you intend to use your new operator or whether you just want to define a variable having the same name.

The procedure for inserting a custom operator depends on whether the operator has one operand (like "–1 " or "5!" for example) or two (like "1 ÷ 2 "). In either case, you'll need to click on the button for the Evaluation and Boolean Palette on the Math Palette. This opens a palette that you'll need in the following procedures.

To insert an operator having two operands:

- Click on the button labelled "*xfy*" on the palette. You'll see three empty placeholders.

- In the middle placeholder, insert the name of the operator. You may find it more convenient to copy the name from the operator definition and paste it into the placeholder.

- In the remaining two placeholders, place the two operands.

- Press **=** to evaluate the expression.

Another way to display an operator having two operands is to use the other button showing the letters "x", "f" and "y" arranged like a water molecule. If you follow the preceding steps using this operator, you'll see the tree shaped display shown in the lower-left corner of Figure 12-13.

To insert an operator having only one operand, decide first whether you want the operator to appear before the operand, as in "–1 ", or after the operand as in "5!". The former is called a *prefix* operator; the latter is a *postfix* operator. The example below shows how to use a prefix operator. The steps for creating a postfix operator are almost identical.

In the following example, the symbol "¬" comes from the Symbol font. Before you can reproduce the steps in this example, you'll first have to define an operator "¬(*x*)". To do so, follow the steps for defining ÷(*x*, *y*) in the previous section, substituting the symbol "¬" for "÷" and using only one argument instead of two.

- To make a *prefix* operator click on the button labeled "*fx*" on the symbol palette. Otherwise, click on the "*xf*" button. In either case, you'll see two empty placeholders.

- If you clicked the "*xf*" button, put the operator name in the first placeholder. Otherwise put it in the second placeholder. In either case, you may find it more convenient to copy the name from the operator definition and paste it into the placeholder.

- In the remaining placeholder, place the operand.

■ Press **=** to evaluate the expression.

¬0 = 1

Be careful when you use operators this way. Since the placeholders look identical, there are no visual cues to tell you where the operands go and where the operator goes.

The most convenient way to use operators like this is create them once and then save them in a QuickSheet. To do this, open the QuickSheets from the Resource Center (choose **Resource Center** on the **Help** menu). Then click on "Math Symbols" to see a selection of common math symbols. You can drag any of these to your worksheet to help you define a new operator. Once you've defined the new operator, click on "Personal QuickSheets" and drag its definition into the QuickSheet. When you need to use this operator again, just open your Personal QuickSheet and drag it back off.

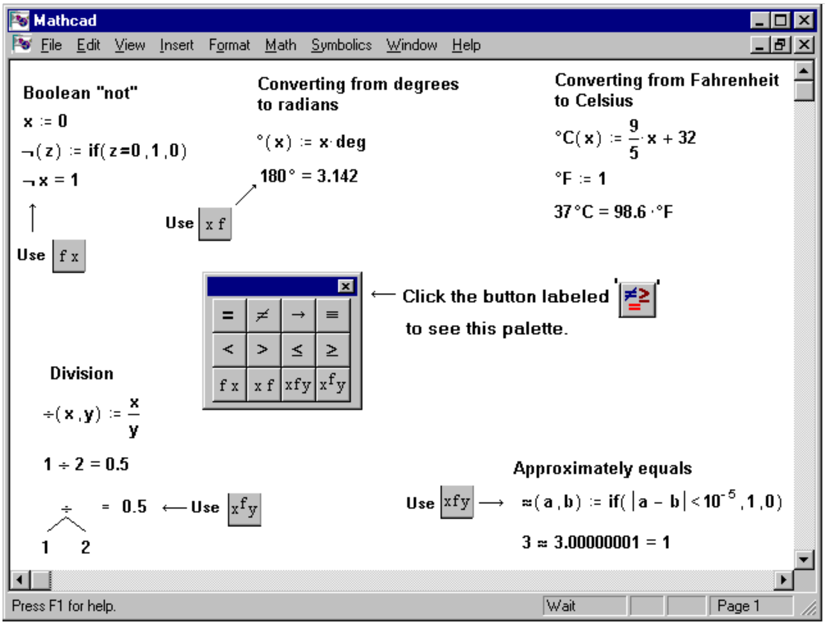


Figure 12-13: Defining your own operators.

Display of functions as operators

As noted earlier, there is really no fundamental difference between functions and operators. The steps given in the section "Defining a custom operator" on page 258 exactly parallel the steps given on page 139 for defining a function.

Since you *define* an operator just as if it were a function, you might expect to be able to *display* that operator as if it were a function as well. Figure 12-14 shows that this is indeed true. Although notation like "÷(1, 2)" is unconventional, nothing stops you from using it if you prefer it.

Conversely, you can display a function as if it were an operator. For example, 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 and following the steps in the section "Using a custom operator." The lower half of Figure 12-14 shows an example of this.

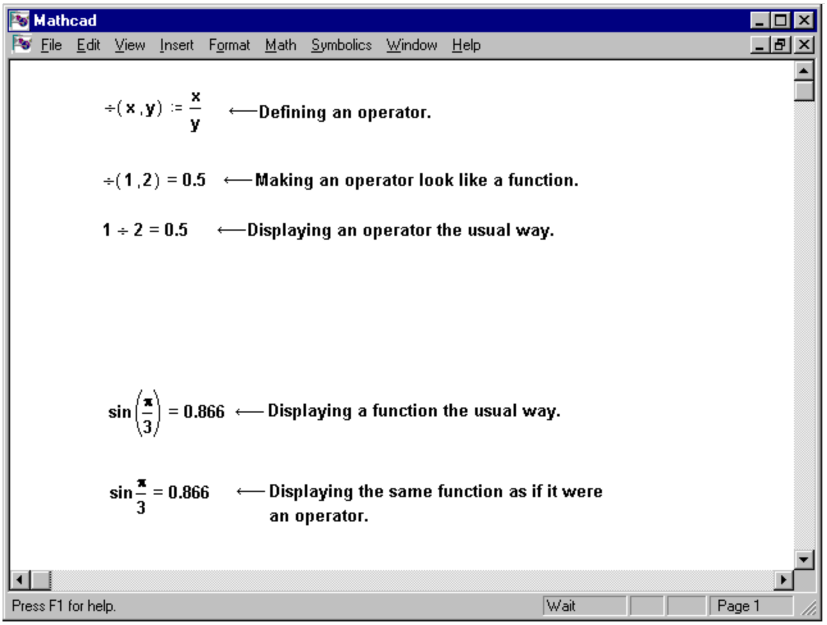


Figure 12-14: Displaying an operator as a function and a function as an operator.