Chapter 10
Vectors and Matrices

This chapter describes Mathcad arrays. While ordinary variables (scalars) hold a single value, arrays hold many values. As is customary in linear algebra, arrays having only one column will often be referred to as vectors. All others are matrices. The following sections make up this chapter.

Creating a vector or matrix

How to create or edit vectors and matrices

Computing with arrays

Defining variables as arrays and using them in expressions.

Subscripts and superscripts

Referring to individual array elements and columns.

Displaying vectors and matrices

How Mathcad displays answers involving matrices and vectors.

Limits on array sizes

Limits on the sizes of arrays to be stored, displayed, or entered.

Vector and matrix operators

Operators designed for use with vectors and matrices.

Vector and matrix functions

Built-in functions designed for use with vectors and matrices.

Doing calculations in parallel

Using Mathcad's “vectorize” operator to speed calculations.

Simultaneous definitions

Using vectors to define several variables simultaneously.

Arrays and user-defined functions

Using arrays as arguments to user defined functions.

Nested arrays

Arrays in which the elements are themselves arrays.

Creating a vector or matrix

A single number in Mathcad is called a *scalar*. A column of numbers is a *vector*, and a rectangular array of numbers is called a *matrix*. The general term for a vector or matrix is an *array*.

There are three ways to create an array:

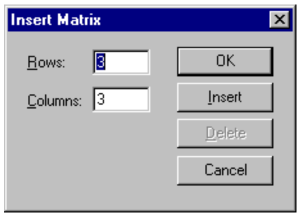
- By filling in an array of empty placeholders as discussed in this section. This technique is useful for arrays that are not too large.
- By using range variables to fill in the elements as discussed in Chapter 11, "Range Variables." This technique is useful when you have some explicit formula for the elements in terms of their indices.
- By reading data in from external files or applications as discussed in Chapter 19, "Data Management," and Chapter 28, "Importing and Exporting Graphics."

You may wish to distinguish between the names of matrices, vectors, and scalars by font. For example, in many math and engineering books, names of vectors are set in bold while those of scalars are set in italic. See the section "Math styles" in Chapter 6 for a description of how to do this.

Creating a vector

A vector is an array or matrix containing one column. To create a vector in Mathcad, follow these steps:

- Click in either a blank space or on a placeholder.
- Choose **Matrix** from the **Insert** menu, or click on the Vector or Matrix button on the Vectors and Matrices palette. A dialog box appears, as shown on the right.

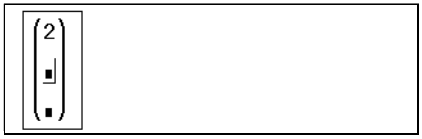


- Enter the number of elements in the text box beside "Rows." For example, to create a three-element vector, type **3**.
- Enter **1** in the text box beside "Columns." Then click "Create." Mathcad inserts a vector of placeholders.

The next step is to fill in these placeholders with scalar expressions. To do so, follow these steps:

■ Click on the top placeholder and type **2**.

■ Move the insertion point to the next placeholder. You can do this by clicking directly on the second placeholder.

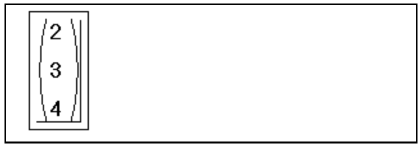


■ Type **3** on the second placeholder. Then move the insertion point to the third placeholder and type **4**.

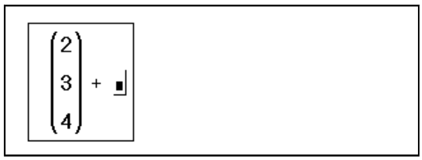
If you're going to need several vectors in your calculation, you can leave the **Insert Matrix** dialog box up for later use.

Once you have created a vector, you can use it in calculations just as you would a number. For example, to add another vector to this vector, follow these steps:

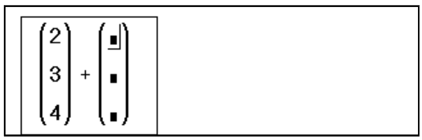
■ Press [**Space**] to enclose the entire vector is now between the editing lines. This ensures that the plus sign you type next will apply to the whole vector rather than to one of its elements.



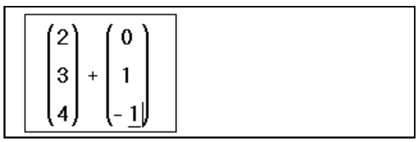
■ Type the plus key (**+**). Mathcad shows a placeholder for the second vector.



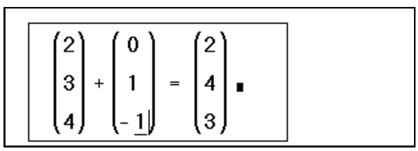
■ Use the **Insert Matrix** dialog box to create another three-element vector.



■ Fill in this vector by clicking in each placeholder and typing in the numbers shown on the right.



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

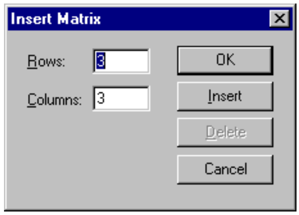


Addition is just one of Mathcad's vector and matrix operations. Mathcad also includes matrix subtraction, matrix multiplication, dot product, integer powers, determinants, and many other operators and functions for vectors and matrices. Complete lists appear in the sections “Vector and matrix operators” on page 199 and “Vector and matrix functions” on page 202.

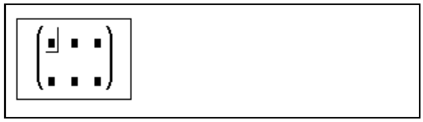
Creating a matrix

To create a matrix, first click in a blank space or on a placeholder. Then:

- Choose **Matrix** from the **Insert** menu. The dialog box shown on the right appears.



- Enter a number of rows and a number of columns in the appropriate boxes. In this example, there are two rows and three columns. Then click on “Create.” Mathcad inserts a matrix of placeholders.



- Fill in the placeholders to complete the matrix as described in the previous section for vectors.

$$\begin{pmatrix} 2 & 5 & 17 \\ 3.5 & 3.9 & -12.9 \end{pmatrix}$$

You can use this matrix in equations, just as you would a number or vector.

Throughout this *User's Guide*, the term “vector” refers to a column vector. A column vector is identical to a matrix with one column. You can also create a *row vector* by creating a matrix with one row and many columns. Operators and functions which expect vectors always expect column vectors. They do not apply to row vectors. To change a row vector into a column vector, use the transpose operator [**Ctrl**]**1**.

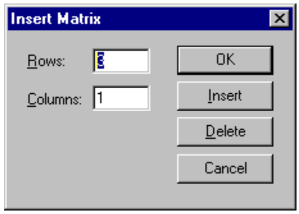
Changing the size of a matrix

You can change the size of a matrix by inserting and deleting rows and columns. To do so, follow these steps:

- Click on one of the matrix elements to place it between the editing lines. Mathcad will begin inserting or deleting with this element.

$$\begin{pmatrix} 2 & 5 & 17 \\ 3.5 & 3.9 & -12.9 \end{pmatrix}$$

- Choose **Matrix** from the **Insert** menu. The dialog box as shown on the right appears.



- Type the number of rows and/or columns you want to insert or delete. Then click on either “Insert” or “Delete.” For example, to delete the column that currently holds the selected element, type **`1`** in the box next to “Columns,” **`0`** in the box next to “Rows,” and click on “Delete.”

$$\begin{pmatrix} 5 & 17 \\ 3.9 & -12.9 \end{pmatrix}$$

Here's how Mathcad inserts or deletes rows or columns based on what you type in the dialog box:

- If you insert rows, Mathcad creates rows of empty placeholders below the selected element. If you insert columns, Mathcad creates columns of empty placeholders to the right of the selected element.
- To insert a row above the top row or a column to the left of the first column, first place the whole matrix between the editing lines. To do so, click in the matrix and press [**`Space`**]. Then choose **Matrix** and proceed as you would normally.
- If you delete rows or columns, Mathcad begins with the row or column occupied by the selected element. Mathcad deletes rows from that element downward and columns from that element rightward.
- If you type 0 as the number for “Rows,” Mathcad neither inserts nor deletes rows. If you type 0 as the number for “Columns,” Mathcad neither inserts nor deletes columns.

Note that when you delete rows or columns, Mathcad discards the information in the rows or columns you eliminate.

To delete an entire matrix or vector, place the entire matrix or vector between the editing lines and choose **Cut** from the **Edit** menu.

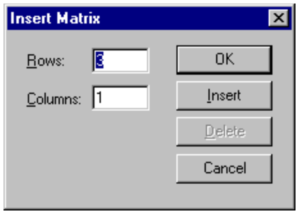
Computing with arrays

Variables can represent arrays as well as scalars. Defining a variable as an array is very much like defining a scalar. First type a variable name and a colon as you would with any other definition. Then create an array (vector or matrix) on the other side of the equation.

For example, to define a vector **v**, follow these steps:

- Click in empty space and type **v**, followed by the colon key (**:**).

- Choose **Matrix** from the **Insert** menu to bring up a dialog box. Type **3** in the box next to “Rows” and **1** in the box next to “Columns.”



- Press “Create” and fill in the elements.

You can now use the name **v** in place of the actual vector in any equation. Figure 10-1 demonstrates that the variable name **v** and the vector itself are interchangeable. Once you have defined a vector, you can of course define other vectors in terms of that vector, just as if you were doing mathematics on paper.

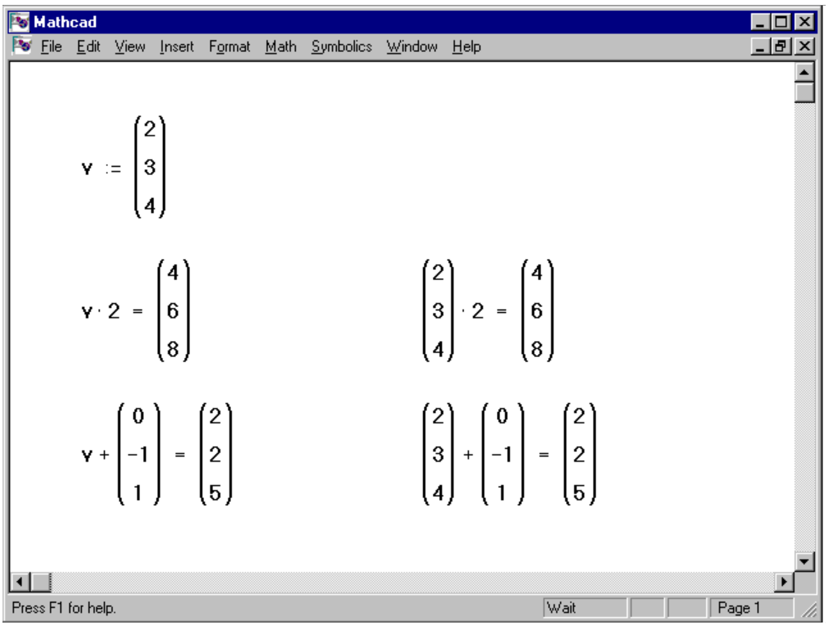


Figure 10-1: Defining and using a vector variable.

Do not use the same name for a scalar variable and a vector variable. This will simply redefine the variable.

Subscripts and superscripts

You can refer to individual array elements by using subscripts. You can also refer to an entire column of an array by using a superscript. To type a subscript, use the left bracket key "**[** " and put an integer or a pair of integers in the placeholder. To insert a superscript operator, press [**Ctrl**]**6** and place an integer in the placeholder.

Vector and matrix elements are ordinarily numbered starting with row zero and column zero. To change this, change the value of the built-in variable ORIGIN. See "Changing the array origin" on page 194.

Subscripts and vector elements

The top equation in Figure 10-1 defines the vector **v**. To see the zeroth (top) element of the vector **v**:

■ Type **v[0=**

$$\boxed{v_0 = 2}$$

You can also define individual vector elements by using a subscript on the left side of a definition. To change v_2 to 6:

■ Type **v[2:6**

$$v_2 := 6$$

Figure 10-2 shows how this changes the value of **v**.

When you define vector elements, you may leave gaps in the vector. For example, if **v** is undefined and you define v_3 as 10, v_0, v_1, and v_2 are all undefined. Mathcad fills these gaps with zeros until you enter specific values for them, as shown in Figure 10-3. Be careful of inadvertently creating very large vectors and matrices by doing this.

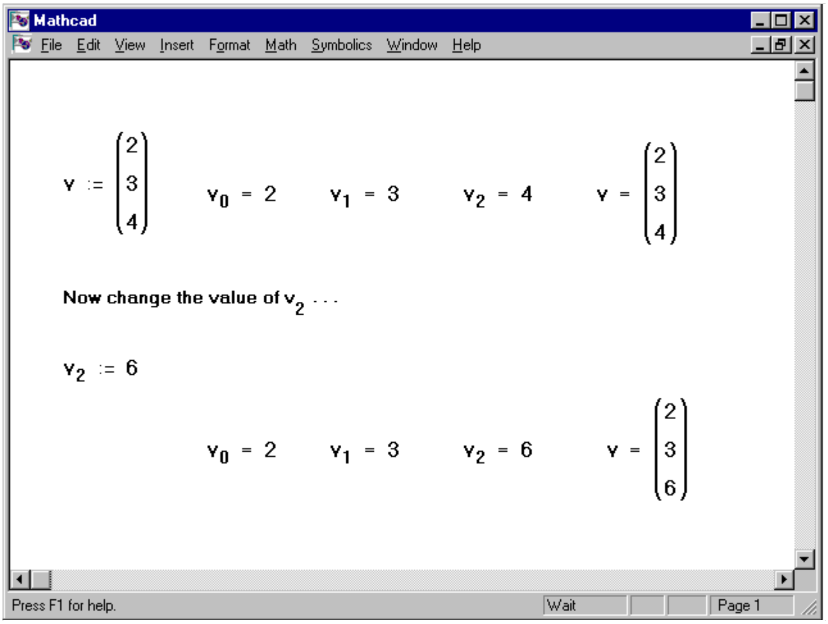


Figure 10-2: Defining a vector element.

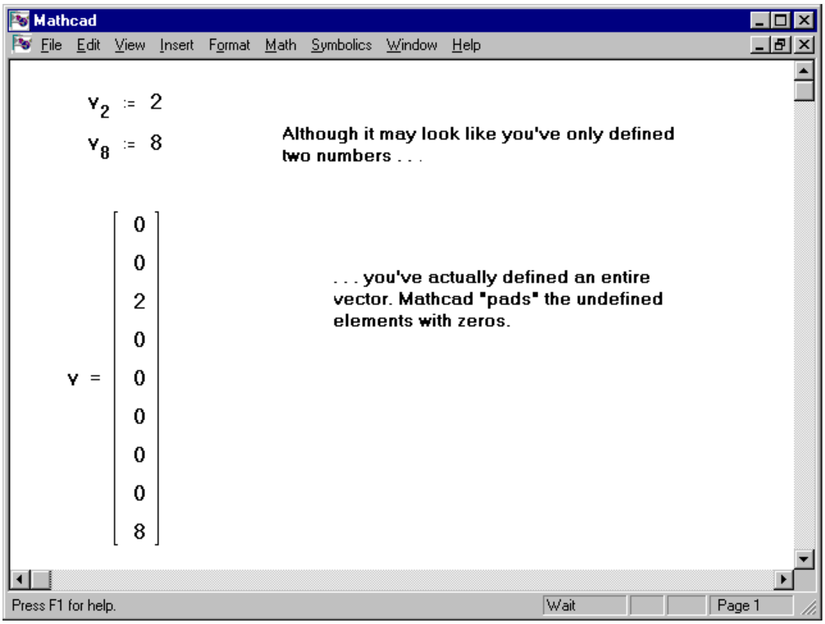


Figure 10-3: Mathcad places zeros into all elements you don't explicitly define.

Subscripts and matrix elements

To view or define a matrix element, use two subscripts separated by a comma. In general, to refer to the element in the *i*th row, *j*th column of matrix **M**, type:

`M[i,j`

Note that the subscripts, like division and exponentiation, are "sticky." Whatever you type after **`[`** remains in the subscript until you press [**`Space`**] to leave.

If you want to add more to the equation, press [**`Space`**] to place the entire matrix element name, $M_{i,\,j}$, between the editing lines.

Figure 10-4 shows some examples of how to define individual matrix elements and how to view them. Notice that, as with vectors, Mathcad fills unspecified matrix elements with zeros.

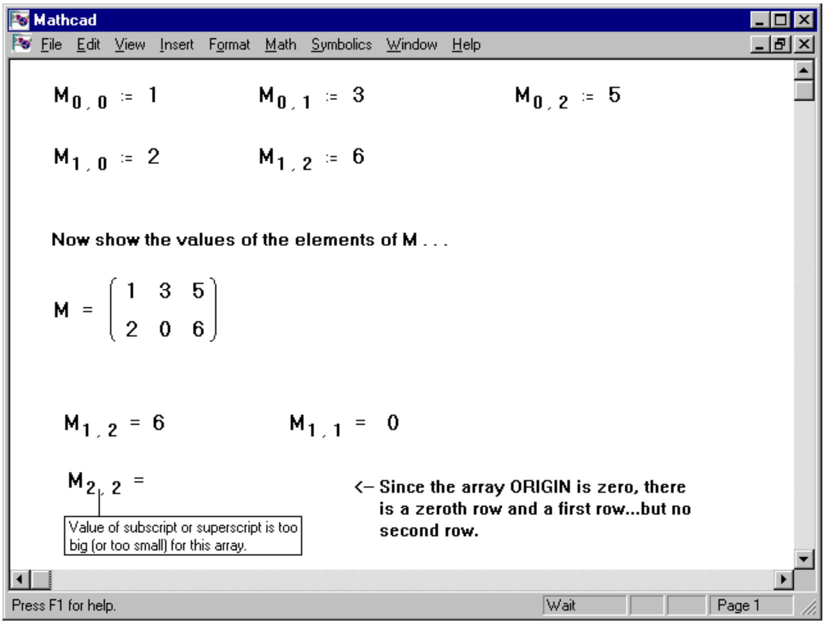


Figure 10-4: Defining and viewing matrix elements.

You can also define the elements of a vector or matrix with a definition like $v_i := i$, where *i* is a *range variable*. See Chapter 11, "Range Variables."

Superscripts with matrix columns

To refer to an entire column of an array, press [**`Ctrl`**]**`6`** and place the column number in the resultant placeholder. Figure 10-5 shows how to place the third column of the matrix **M** in the vector **v**.

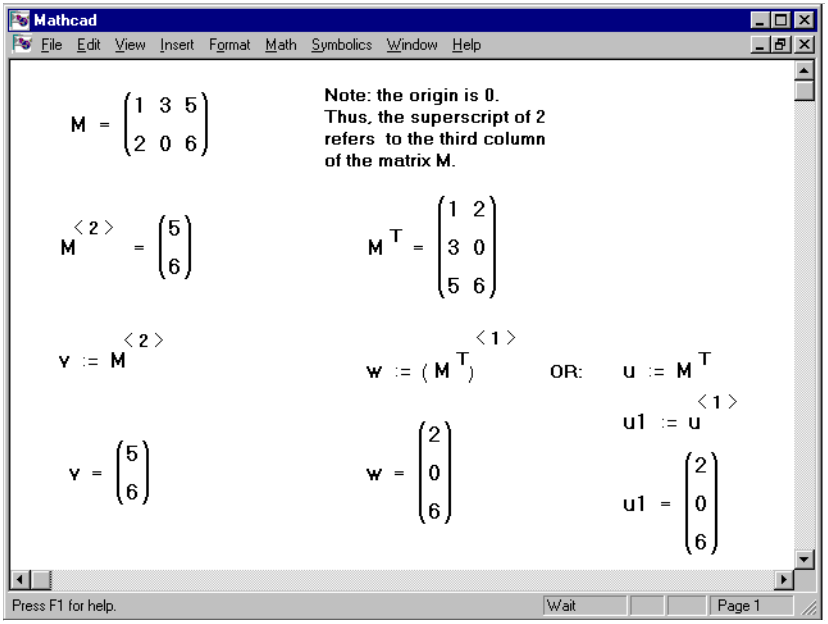


Figure 10-5: Using the superscript operator to extract a column from a matrix.

You can also extract a single row from a matrix by extracting a column from the transposed matrix. This is shown on the right-hand side of Figure 10-5.

Changing the array origin

By default, Mathcad arrays begin at element zero. To change this, change the value of the built-in variable ORIGIN. When you use subscripts to refer to array elements, Mathcad assumes the arrays begin at the current value of ORIGIN.

For example, suppose you want all your arrays to begin with element one. There are two ways to change the value of ORIGIN for the whole worksheet:

- Choose the **Options** command from the **Math** menu, click on the Built-In Variables tab, and change the value of ORIGIN.
- Enter a global definition for ORIGIN anywhere in your worksheet. For example, to change the ORIGIN to one, type: **`ORIGIN~1`**.

If you change ORIGIN to one, Mathcad no longer maintains an element zero for vectors or a zeroth row and column for matrices. Figure 10-6 shows a worksheet with the ORIGIN set to 1. Note that when you try to refer to v_0, Mathcad displays an appropriate error message.

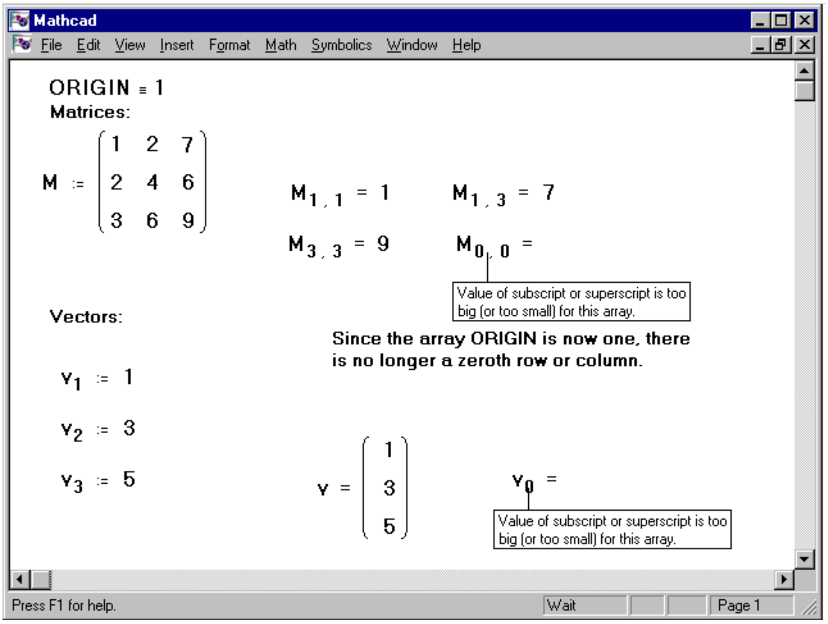


Figure 10-6: Arrays beginning at element one instead of at element zero.

When you redefine ORIGIN in a worksheet, keep in mind the following suggestions:

- If you define ORIGIN with a definition in the worksheet rather than using the **Options** command on the **Math** menu, use a single global definition. Although you can redefine ORIGIN with a "**:=**" this will invariably lead to confusion. Changing ORIGIN in the middle of a worksheet can cause confusing effects. Array elements will seem to have shifted *n* positions, where *n* is the difference between the old ORIGIN and the new ORIGIN.

- Don't forget to type ORIGIN in capital letters. Mathcad variable names are case-sensitive. Because ORIGIN is a built-in variable, its name is not font sensitive. It is however, still case-sensitive.

- When you define an array, Mathcad assigns zero to any undefined elements. See Figure 10-3 for an example.

- If you inadvertently define an array starting with element one when ORIGIN is set to its default value of zero, you will get unexpected answers with array functions like *mean* and *fft*. This is because Mathcad will automatically define $x_0 = 0$ for all these arrays. This extra element distorts the values returned by array functions. To avoid this problem, choose **Options** from the **Math** menu, click on the Built-In Variables tab, and set ORIGIN to 1.

- When you set ORIGIN in the Built-In Variable dialog box, its value applies to all array variables. It is not possible to have some variables use one ORIGIN and others use a different ORIGIN.

- You can use ORIGIN to define variables with negative subscripts. If you set ORIGIN to –10, all arrays will begin with element –10.
- If you reference an array element with a subscript less than ORIGIN, Mathcad marks the array reference with an error message indicating that the array index goes beyond the ends of the array.

Displaying vectors and matrices

After computing with arrays in Mathcad, your resulting arrays may be large and unwieldy when displayed. Mathcad therefore displays matrices and vectors having more than nine rows or columns as scrolling output tables rather than as matrices or vectors. Figure 10-7 shows an example.

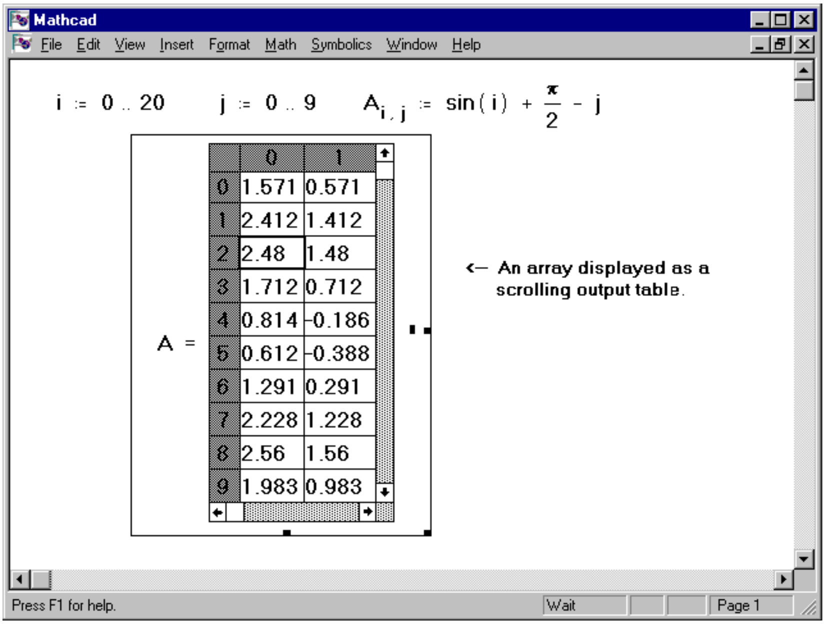


Figure 10-7: Displaying results in a scrolling output table.

A scrolling output table displays a portion of an array. To the left of each row and at the top of each column, there is a number indicating the index of the row or column. Use these row and column headers to determine the index of a particular value in the table.

If your results extend beyond the table, a scroll bar will appear along the appropriate edge of the table. You can scroll through the table using these scroll bars just as you would scroll through any window.

Another way to view more of a resulting array is to enlarge the table. To resize a scrolling output table:

- Click the mouse just outside the equation region in which the scrolling output table appears. This anchors one corner of the selection rectangle.
- Press and hold down the mouse button. With the button still held, drag the mouse across the scrolling output table. A selection rectangle emerges from the anchor point.
- When the selection rectangle just encloses the equation region, release the mouse button.
- Move the mouse pointer to the right or bottom edge of the selection rectangle. It will change to a double headed arrow.
- Press and hold down the mouse button. With the mouse button still pressed, move the mouse. The scrolling output table will be stretched in the direction of the motion.
- Once the scrolling output table is the right size, release the mouse button. Click outside the selection rectangle to deselect the equation region.

In addition to being able to resize and scroll through a scrolling output table, you can copy one or more values from it and paste them into another part of your worksheet or into another Windows application. For information on copying results from a scrolling output table, see the section "Copying numerical results" in Chapter 7.

Changing the display of arrays

Although matrices and vectors having more than nine rows or columns are automatically displayed as scrolling output tables, you can have Mathcad display them as matrices. To do so:

- Click on the scrolling output table.
- Choose **Number** from the **Format** menu.
- Click on the box beside "Display as Matrix." The box should now be checked.
- Click the "OK" button.

To display all the matrices and vectors of results in your worksheet as matrices regardless of their size:

- Click on an empty part of your worksheet.
- Choose **Number** from the **Format** menu.
- Click on the box beside "Display as Matrix."
- Make sure the "Set as worksheet default" radio button is filled and click "OK".

Graphical display of matrices

In addition to looking at the actual numbers making up an array, you can also see a graphical representation of those same numbers. There are three ways to do this:

- For an arbitrary array, you can use the various three dimensional plot types discussed starting at Chapter 22, “Surface Plots.”
- For an array of integers between 0 and 255, you can look at a grayscale image by choosing **Picture** from the **Insert** menu and entering the array's name in the placeholder.
- For three arrays of integers between 0 and 255 representing the red, green, and blue components of an image, by choosing **Picture** from the **Insert** menu and entering the arrays’ names, separated by commas, in the placeholder.

An example of viewing a matrix as a grayscale image is shown in Figure 18-19 of Chapter 18, “Programming.” See Chapter 28, “Importing and Exporting Graphics,” for more on viewing a matrix (or three matrices, in the case of a color image) in the picture operator.

Limits on array sizes

Mathcad has the following limits on the sizes of arrays to be defined, entered, or displayed:

Limit on input arrays

You cannot use the **Matrix** command on the **Insert** menu to create an array having more than 100 elements. This limitation applies whether you attempt to create a new array or add to an existing array. You can however, create larger arrays by either using the *augment* or *stack* functions to join arrays together, by using range variables, or by reading the numbers in directly from a disk file. An example of how to use the *augment* function is shown in Figure 10-8. The use of range variables to create arrays is discussed in Chapter 11, “Range Variables.” Reading data files directly from a local or network drive, the clipboard, or another application is discussed in Chapter 19, “Data Management.”

Limit on displayed arrays

If an array has more than nine rows or columns, Mathcad automatically displays it as a scrolling output table. You can enlarge the table or use the scroll bars provided in order to view all of the array. If, however, you change the local result format such that Mathcad displays it as an array rather than as a scrolling output table, Mathcad displays only the first two hundred rows or columns. Mathcad uses an ellipsis to indicate that rows and columns are present but not displayed. Although Mathcad does not display these rows or columns, it does continue to keep track of them internally.

Limit on array size

The effective array size limit depends on the memory available on your system. For most systems, it will usually be at least 1 million elements. In no system will it be higher than 8 million elements. If you try to define an array larger than your system will accommodate, you'll see an error message indicating that you have insufficient memory to do so. The elements can be distributed among any combination of rows and columns. When only limited memory is available and you define several very large arrays, the array size limit may decrease.

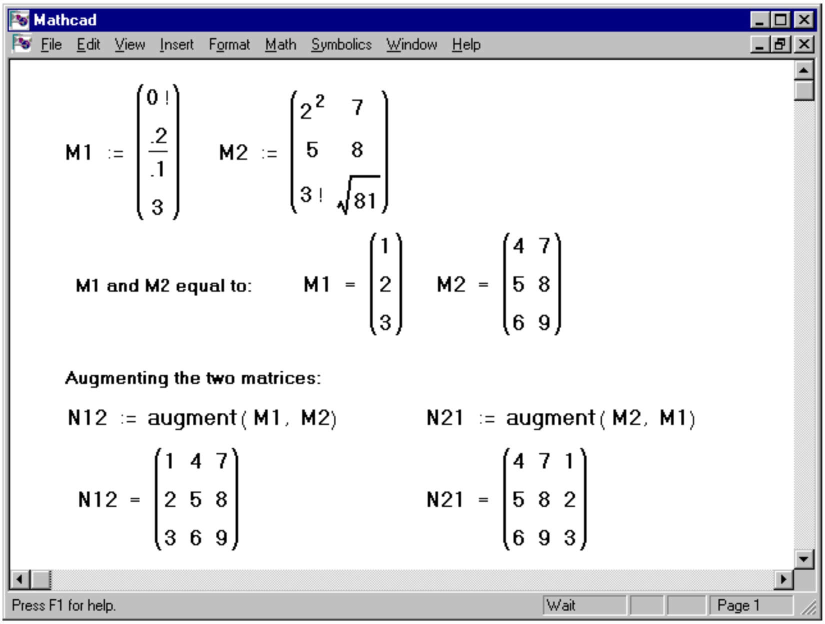


Figure 10-8: Using the augment function to combine two matrices.

Vector and matrix operators

Some of Mathcad's operators have special meanings for vectors and matrices. For example, the multiplication symbol means multiplication when applied to two numbers, but it means dot product when applied to vectors, and matrix multiplication when applied to matrices.

The table below describes Mathcad's vector and matrix operations. Many of these operators are available from the Vector and Matrices palette, available off the Math Palette. Note that operators which expect vectors always expect column vectors rather than row vectors. To change a row vector into a column vector, use the transpose operator [**`Ctrl`**]**`1`**.

Operators not listed in this table will not work for vectors and matrices. You can, however, use the "vectorize" operator to perform any scalar operation or function element by element on a vector or matrix. See "Doing calculations in parallel" on page 210. Figure 10-9 shows some ways to use vector and matrix operations.

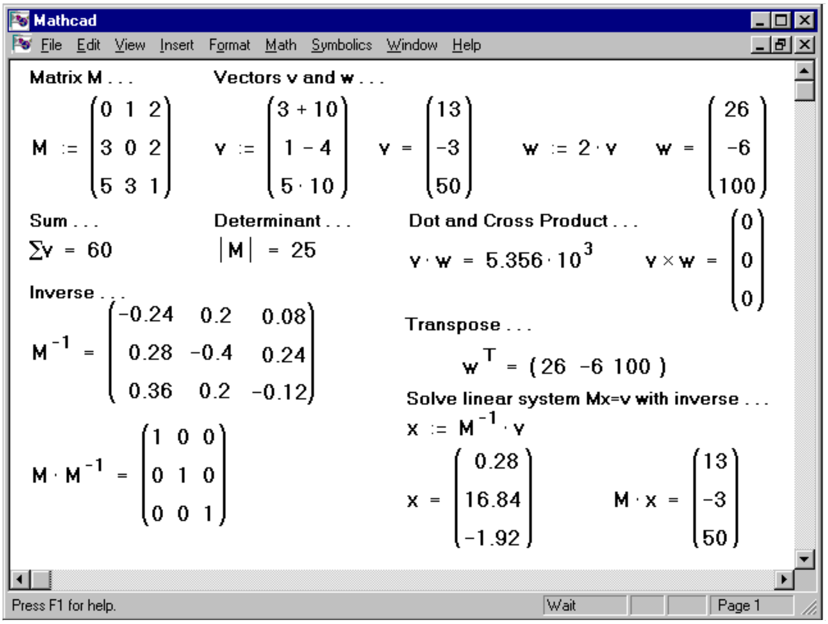


Figure 10-9: Vector and matrix operations.

In the following table,

- **A** and **B** represent arrays, either vector or matrix.
- **u** and **v** represent vectors.
- **M** represents a square matrix.
- u_i and v_i represent the individual elements of vectors **u** and **v**.
- z represents a scalar.
- m and n represent integers.

Operation	Appearance	Keystroke	Description
Scalar multiplication	$\mathbf{A} \cdot z$	*	Multiplies each element of **A** by the scalar z.
Dot product	$\mathbf{u} \cdot \mathbf{v}$	*	Returns a scalar: $\Sigma u_i \cdot v_i$. The vectors must have the same number of elements.
Matrix multiplication	$\mathbf{A} \cdot \mathbf{B}$	*	Returns the matrix product of **A** and **B**. The number of columns in **A** must match the number of rows in **B**.
Vector/Matrix multiplication	$\mathbf{A} \cdot \mathbf{v}$	*	Returns the product of **A** and **v**. The number of columns in A must match the number of rows in **v**.
Scalar division	$\frac{\mathbf{A}}{z}$	/	Divides each element of the array **A** by the scalar z.

Operation	Appearance	Keystroke	Description
Vector and matrix addition	$\mathbf{A} + \mathbf{B}$	`+`	Adds corresponding elements of **A** and **B**. The arrays **A** and **B** must have the same number of rows and columns.
Scalar addition	$\mathbf{A} + z$	`+`	Adds z to each element of **A**.
Vector and matrix subtraction	$\mathbf{A} - \mathbf{B}$	`-`	Subtracts corresponding elements of **A** and **B**. The arrays **A** and **B** must have the same number of rows and columns.
Scalar subtraction	$\mathbf{A} - z$	`-`	Subtracts z from each element of **A**.
Negative of vector or matrix	$-\mathbf{A}$	`-`	Returns an array whose elements are the negatives of the elements of **A**.
Powers of matrix, matrix inverse	$\mathbf{M}^n$	`^`	nth power of square matrix **M** (using matrix multiplication). n must be an integer. $\mathbf{M}^{-1}$ represents the inverse of **M**. Other negative powers are powers of the inverse. Returns a matrix.
Magnitude of vector	$\lvert\mathbf{v}\rvert$	`|`	Returns $\sqrt{\mathbf{v} \cdot \bar{\mathbf{v}}}$ where $\bar{\mathbf{v}}$ is the complex conjugate of **v**.
Determinant	$\lvert\mathbf{M}\rvert$	`|`	**M** must be square matrix. Returns a scalar.
Transpose	$\mathbf{A}^{\mathsf{T}}$	`[Ctrl]1`	Interchanges row and columns of **A**.
Cross product	$\mathbf{u} \times \mathbf{v}$	`[Ctrl]8`	**u** and **v** must be three-element vectors; result is another three-element vector.
Complex conjugate	$\overline{\mathbf{A}}$	`"`	Takes complex conjugate of each element of **A**.
Sum	$\Sigma\mathbf{v}$	`[Ctrl]4`	Sum elements in **v**.
Vectorize	$\overrightarrow{\mathbf{A}}$	`[Ctrl]-`	Treat all operations in **A** element by element. See the section "Doing calculations in parallel" on page 210 for a complete description.
Superscript	$\mathbf{A}^{\langle n \rangle}$	`[Ctrl]6`	nth column of array **A**. Returns a vector.
Vector subscript	v_n	`[`	nth element of a vector.
Matrix subscript	$A_{m,\,n}$	`[`	(m, n)th element of a matrix.

Vector and matrix functions

Mathcad includes functions for manipulating arrays in ways that are common in linear algebra. These functions are intended for use with vectors and matrices. If a function is not explicitly set up to take a vector or matrix argument, it is inappropriate to supply one to it as an argument. Note that functions which expect vectors always expect column vectors rather than row vectors. To change a row vector into a column vector, use the transpose operator [**`Ctrl`**]**`1`**.

The following tables list Mathcad's vector and matrix functions. In these tables,

- **A** and **B** are arrays, either vector or matrix.
- **v** is a vector.
- **M** and **N** are square matrices.
- *z* is a scalar expression.
- Names beginning with *m*, *n*, *i* or *j* are integers.

Size and scope of an array

Mathcad provides several functions that return information about the size of an array and its elements. Figure 10-10 shows how these functions are used.

Function Name	Returns...
rows(**A**)	Number of rows in array **A**. If **A** is a scalar, returns 0.
cols(**A**)	Number of columns in array **A**. If **A** is a scalar, returns 0.
length(**v**)	Number of elements in vector **v**.
last(**v**)	Index of last element in vector **v**.
max(**A**)	Largest element in array **A**. If **A** has complex elements, returns the largest real part plus *i* times the largest imaginary part.
min(**A**)	Smallest element in array **A**. If **A** has complex elements, returns the smallest real part plus *i* times the smallest imaginary part.

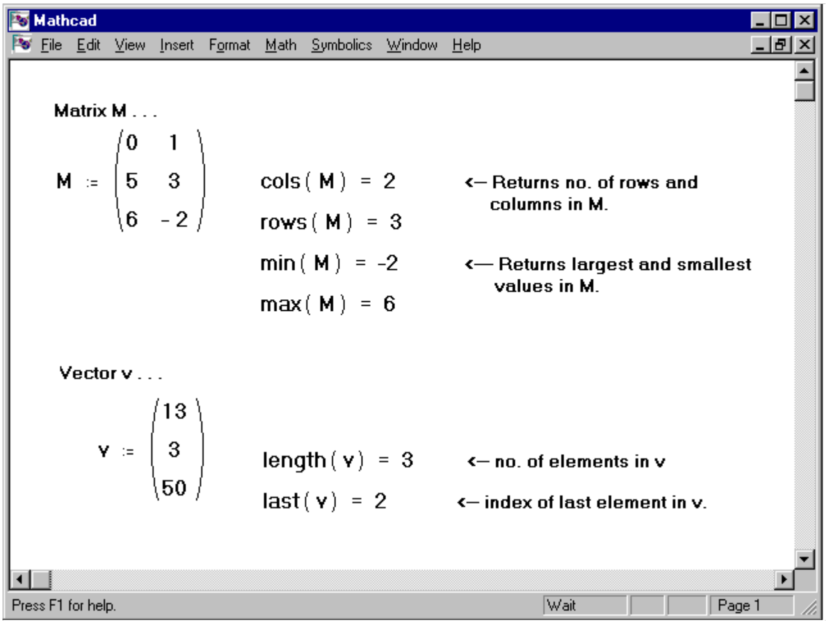


Figure 10-10: Vector and matrix functions for finding the size of an array and information about its elements.

Special types of matrices

You can use the following functions to derive from an array or scalar a special type or form of a matrix. Some of these functions are available only in Mathcad Professional.

	Function Name	Returns...
	identity(n)	An $n \times n$ matrix of 0's with 1's on the diagonal.
	Re(**A**)	An array of the same size as **A** but with the imaginary parts of each element set to 0.
	Im(**A**)	An array of the same size as **A** but with the real parts of each element set to 0.
Pro	diag(v)	A diagonal matrix containing on its diagonal the elements of **v**.
Pro	geninv(**A**)	The left inverse matrix **L** of **A**, such that $\mathbf{L} \cdot \mathbf{A} = \mathbf{I}$, where **I** is the identity matrix having the same number of columns as **A**. Matrix **A** is an $m \times n$ real-valued matrix, where $m \geq n$.
	rref(**A**)	The reduced-row echelon form of **A**.

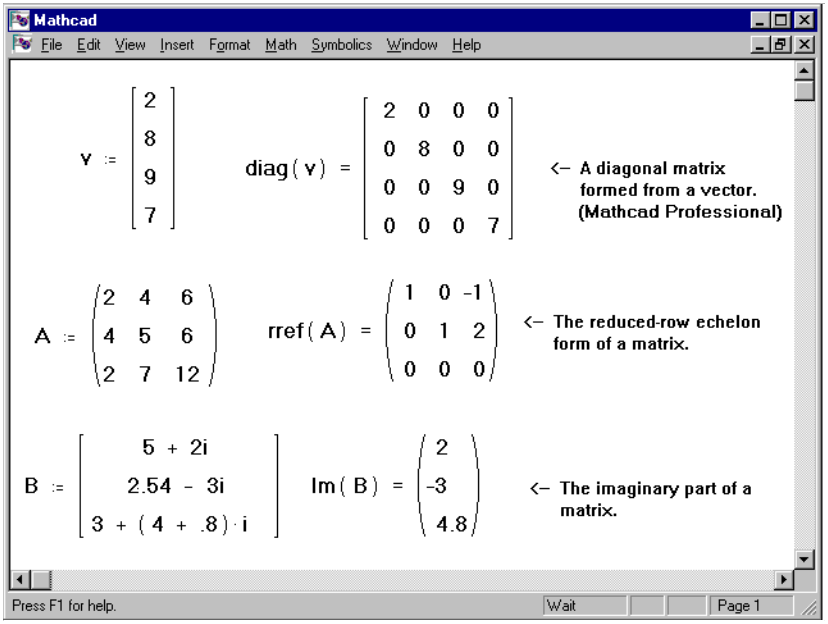


Figure 10-11: Functions for transforming arrays.

Special characteristics of a matrix

You can use the functions in the following table to find the trace, rank, norms, and condition numbers of a matrix. Most of these functions are available only in Mathcad Professional.

	Function Name	Returns . . .
	tr(**M**)	The sum of the diagonal elements, otherwise known as the *trace*, of **M**.
	rank(**A**)	The rank of the real-valued matrix **A**.
Pro	norm1(**M**)	The L_1 norm of the matrix **M**.
Pro	norm2(**M**)	The L_2 norm of the matrix **M**.
Pro	norme(**M**)	The Euclidean norm of the matrix **M**.
Pro	normi(**M**)	The infinity norm of the matrix **M**.
Pro	cond1(**M**)	The condition number of the matrix **M** based on the L_1 norm.
Pro	cond2(**M**)	The condition number of the matrix **M** based on the L_2 norm.
Pro	conde(**M**)	The condition number of the matrix **M** based on the Euclidean norm.
Pro	condi(**M**)	The condition number of the matrix **M** based on the infinity norm.

Forming new matrices

Mathcad provides two functions for joining matrices together, either side by side, or one on top of the other. Mathcad also provides a function for filling in a matrix with values of a predefined function, and a function for extracting a smaller matrix from a larger one. Figure 10-12 and Figure 10-13 show some examples.

Function Name	Returns . . .
augment(**A**, **B**)	An array formed by placing **A** and **B** side by side. The arrays **A** and **B** must have the same number of rows.
stack(**A**, **B**)	An array formed by placing **A** above **B**. The arrays **A** and **B** must have the same number of columns.
matrix(*m*, *n*, *f*)	Creates a matrix in which the *ij*th element contains $f(i, j)$ where $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$.
submatrix(**A**, *ir*, *jr*, *ic*, *jc*)	A submatrix of **A** consisting of all elements contained in rows *ir* through *jr* and columns *ic* through *jc*. To maintain order of rows and/or columns, make sure $ir \le jr$ and $ic \le jc$, otherwise order of rows and/or columns will be reversed.

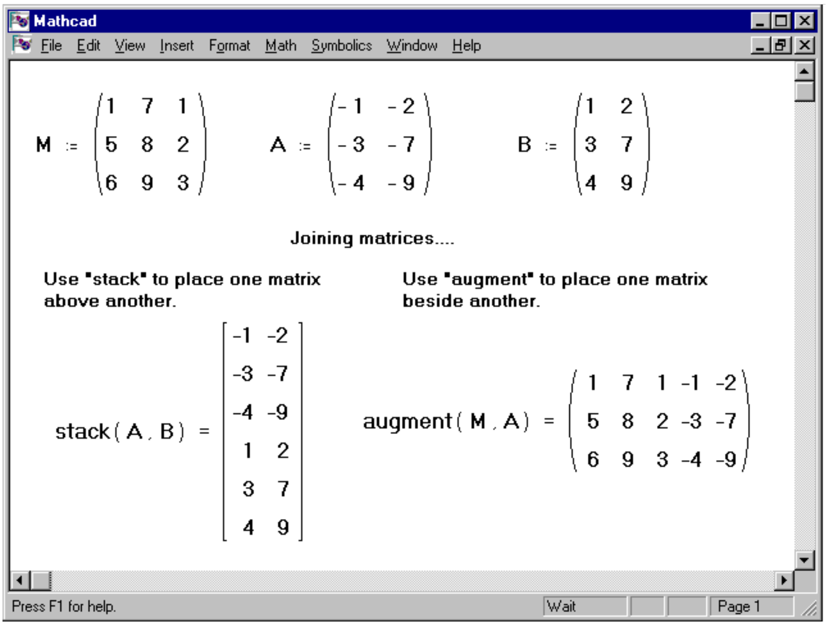


Figure 10-12: Joining matrices together with the stack and augment functions.

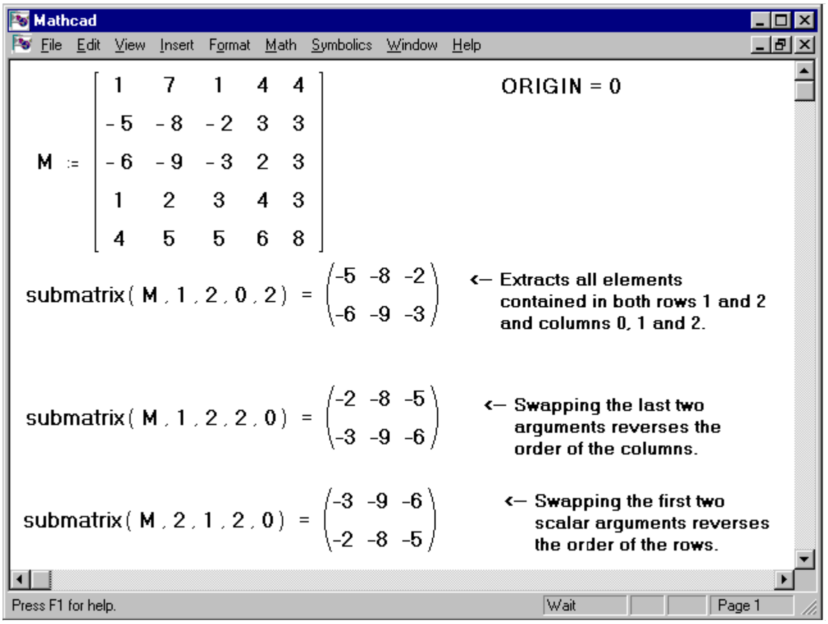


Figure 10-13: Extracting a submatrix from a matrix using the submatrix function.

Eigenvalues and eigenvectors

Mathcad provides functions for working with eigenvalues and eigenvectors of a matrix. The *eigenvecs* function, available in Mathcad Professional, obtains all the eigenvectors at once. If you're using Mathcad Professional, you'll also have access to *genvals* and *genvecs* for finding the generalized eigenvalues and eigenvectors. Figure 10-14 shows how some of these functions are used.

	Function Name	**Returns . . .**
	eigenvals(**M**)	A vector containing the eigenvalues of the matrix **M**.
	eigenvec(**M**, *z*)	A matrix containing the normalized eigenvector corresponding to the eigenvalue *z* of the square matrix **M**.
Pro	eigenvecs(**M**)	A matrix containing normalized eigenvectors corresponding to the eigenvalues of the square matrix **M**. The *n*th column of the matrix returned is an eigenvector corresponding to the *n*th eigenvalue returned by *eigenvals*.
Pro	genvals(**M**, **N**)	A vector **v** of computed eigenvalues each of which satisfies the generalized eigenvalue problem $\mathbf{M} \cdot \mathbf{x} = v_i \cdot \mathbf{N} \cdot \mathbf{x}$. Matrices **M** and **N** contain real values. Vector **x** is the corresponding eigenvector. **M** and **N** are square matrices having the same number of columns.

Pro	genvecs(**M**, **N**)	A matrix containing the normalized eigenvectors corresponding to the eigenvalues in **v**, the vector returned by *genvals*. The *n*th column of this matrix is the eigenvector **x** satisfying the generalized eigenvalue problem $\mathbf{M} \cdot \mathbf{x} = v_n \cdot \mathbf{N} \cdot \mathbf{x}$. Matrices **M** and **N** are real valued square matrices having the same number of columns.

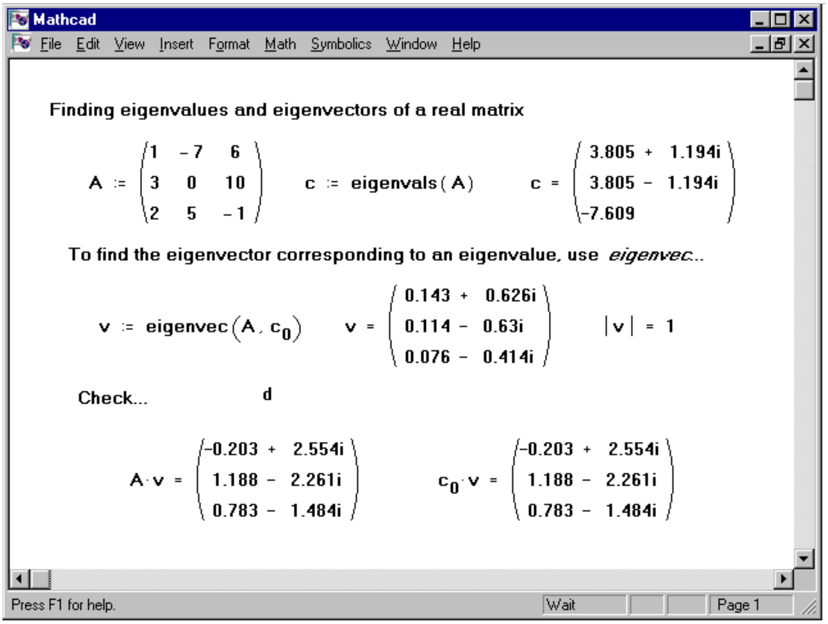


Figure 10-14: Finding eigenvalues and eigenvectors.

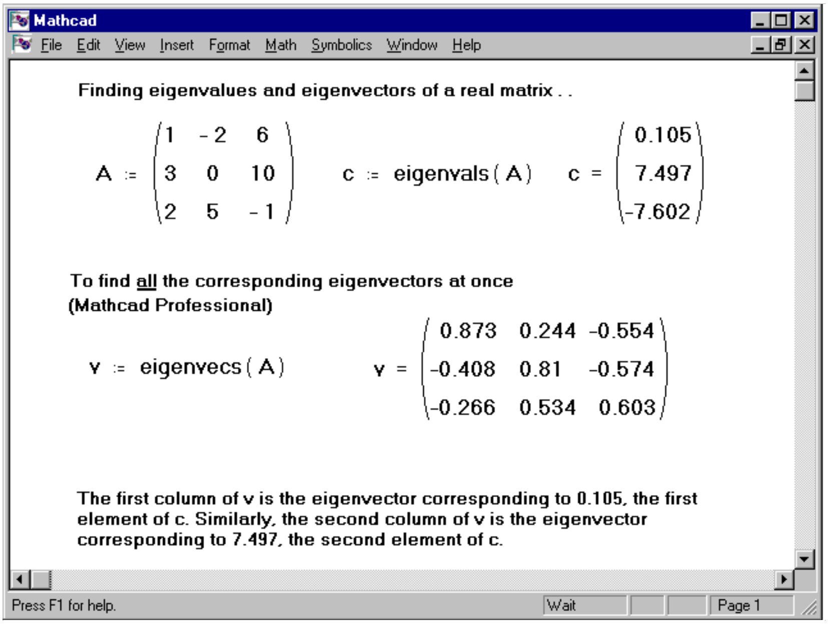


Figure 10-15: Using eigenvecs to find all the eigenvectors at once.

Decomposition

Mathcad Professional offers some additional functions for performing the cholesky decomposition, the QR decomposition, the LU decomposition, and the singular value decomposition of a matrix. Some of these functions return two or three matrices joined together as one large matrix. Use *submatrix* to extract these two or three smaller matrices. Figure 10-16 shows an example.

	Function Name	**Returns . . .**
Pro	cholesky(**M**)	A lower triangular matrix **L** such that $\mathbf{L} \cdot \mathbf{L}^{\mathbf{T}} = \mathbf{M}$. This uses only the upper triangular part of **M**. The upper triangular of **M**, when reflected about the diagonal, must form a positive definite matrix.
Pro	qr(**A**)	A matrix whose first *n* columns contain the square, orthonormal matrix **Q**, and whose remaining columns contain the upper triangular matrix, **R**. Matrices **Q** and **R** satisfy the equation $\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}$, where **A** is a real-valued array.
Pro	lu(**M**)	One matrix containing the three square matrices **P**, **L**, and **U**, all having the same size as **M** and joined together side by side, in that order. These three matrices satisfy the equation $\mathbf{P} \cdot \mathbf{M} = \mathbf{L} \cdot \mathbf{U}$, where **L** and **U** are lower and upper triangular respectively.
Pro	svd(**A**)	One matrix containing two stacked matrices **U** and **V**, where **U** is the upper $m \times n$ submatrix and **V** is the lower $n \times n$ submatrix. Matrices **U** and **V** satisfy the equation $\mathbf{A} = \mathbf{U} \cdot \mathrm{diag}(\mathbf{s}) \cdot \mathbf{V}^{\mathbf{T}}$, where **s** is a vector returned by svds(**A**). **A** is an $m \times n$ array of real values, where $m \geq n$.

Pro	svds(**A**)	A vector containing the singular values of the $m \times n$ real-valued array **A**, where $m \geq n$.

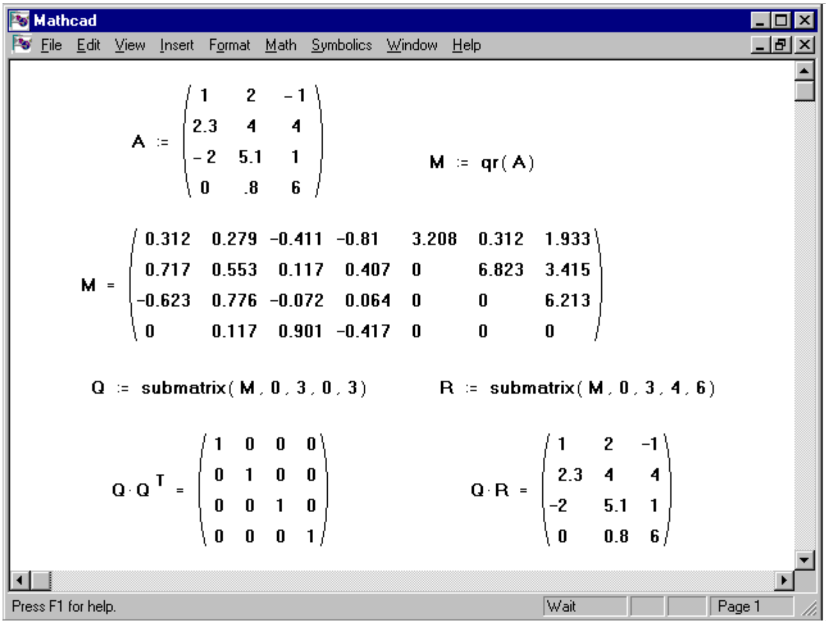


Figure 10-16: Using the submatrix function to extract the results from the qr function. Use submatrix in a similar way to extract results from the lu and svd functions.

Solving a linear system of equations

With Mathcad Professional, you'll be able to use the *lsolve* function to solve a linear system of equations. Figure 10-17 shows an example. Note that the argument **M** for *lsolve* must be a matrix that is neither singular nor nearly singular. A matrix is singular if its determinant is equal to zero. A matrix is nearly singular if it has a high condition number. You may want to use one of the functions described in“Special characteristics of a matrix” on page 204 to find the condition number of a matrix.

	Function Name	**Returns . . .**
Pro	lsolve(**M**, **v**)	A solution vector **x** such that $\mathbf{M} \cdot \mathbf{x} = \mathbf{v}$.

Alternatively, you can solve a system of linear equations by using matrix inversion as shown in the lower right corner of Figure 10-9. For other numerical solving techniques in Mathcad, see Chapter 15, “Solving Equations.” For symbolic solutions of systems of equations, see Chapter 17, “Symbolic Calculation.”

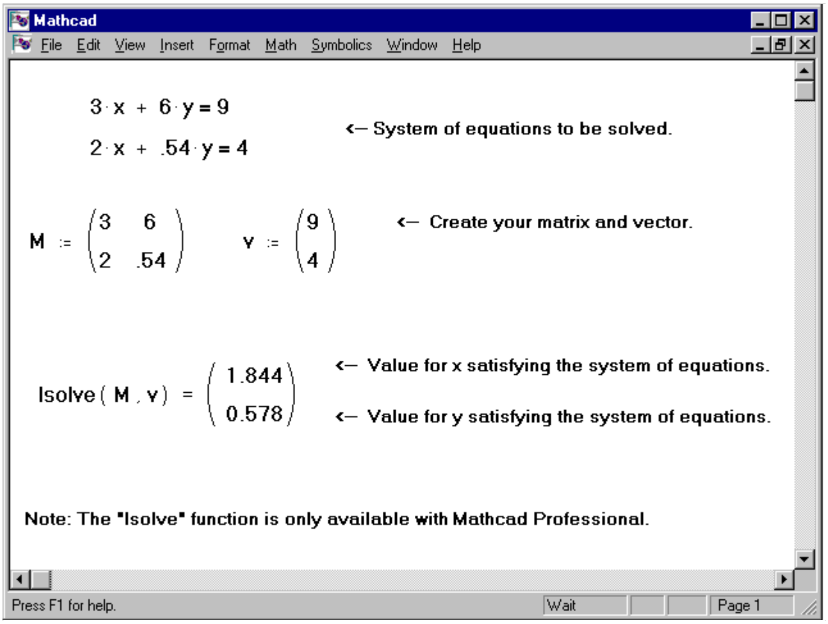


Figure 10-17: Using lsolve to solve two equations in two unknowns.

Doing calculations in parallel

Any calculation Mathcad can perform with single values, it can also perform with vectors or matrices of values. There are two ways to do this:

- By iterating over each element using range variables as described in Chapter 11, “Range Variables.”
- By using the “vectorize” operator described in this chapter.

Mathcad's vectorize operator allows it to perform the same operation efficiently on each *element* of a vector or matrix.

Mathematical notation often shows repeated operations with subscripts. For example, to define a matrix **P** by multiplying corresponding elements of the matrices **M** and **N**, you would write:

$$\mathbf{P}_{i,j} = \mathbf{M}_{i,j} \cdot \mathbf{N}_{i,j}$$

Note that this is not matrix multiplication, but multiplication element by element. It *is* possible to perform this operation in Mathcad using subscripts, as described in Chapter 11, “Range Variables,” but it is much faster to perform exactly the same operation with a vectorized equation.

How to apply the vectorize operator to an expression

Here's how to apply the vectorize operator to an expression like $\mathbf{M} \cdot \mathbf{N}$:

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

$$P := \mathbf{M} \cdot \mathbf{N}$$

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

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

How the vectorize operator changes the meaning of an expression

The vectorize operator changes the meaning of the operators and functions to which it applies. The vectorize operator tells Mathcad to apply the operators and functions with their scalar meanings, element by element.

Here are some examples of how the vectorize operator changes the meaning of expressions with vectors and matrices:

- If **v** is a vector, sin(**v**) is an illegal expression. But if you apply the vectorize operator, Mathcad applies the sine function to every element in **v**. The result is a new vector whose elements are the sines of the elements in **v**.
- If **M** is a matrix, $\sqrt{\mathbf{M}}$ is an illegal expression. But if you apply the vectorize operator, Mathcad takes the square root of every element of **M** and places the results in a new matrix.
- If **v** and **w** are vectors, then $\mathbf{v} \cdot \mathbf{w}$ means the dot product of **v** and **w**. But if you apply the vectorize operator, the result is a new vector whose *i*th element is obtained by multiplying v_i and w_i. This is *not* the same as the dot product.

These properties of the vectorize operator let you use scalar operators and functions with array operands and arguments. In this *User's Guide*, this is referred to as "vectorizing" an expression. For example, suppose you want to apply the quadratic formula to three vectors containing coefficients *a*, *b*, and *c*. Figure 10-18 shows how to do this when *a*, *b*, and *c* are just scalars. Figure 10-19 shows how to do the same thing when **a**, **b**, and **c** are vectors.

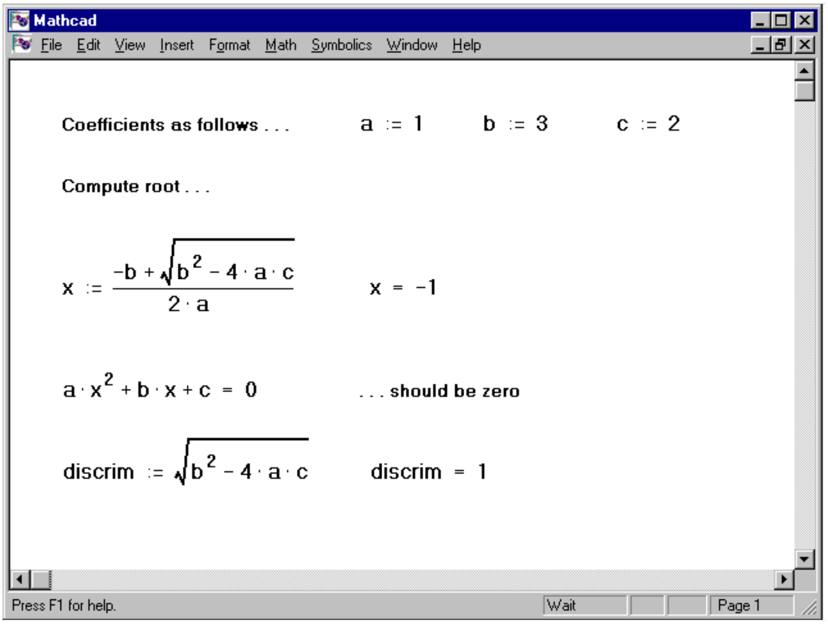


Figure 10-18: The quadratic formula.

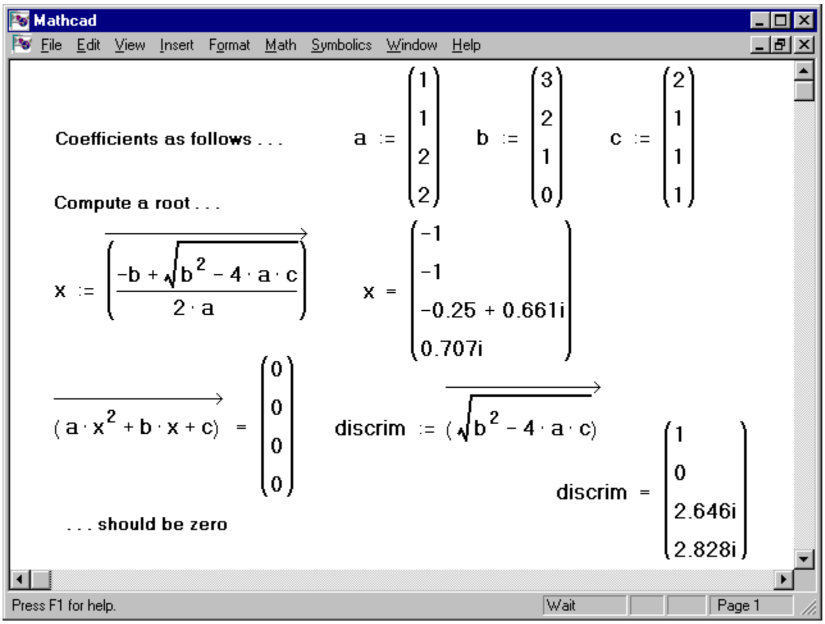


Figure 10-19: Quadratic formula with vectors and the vectorize operator.

The vectorize operator appears as an arrow above the quadratic formula in Figure 10-19. Its use is essential in this calculation. Without it, Mathcad would interpret $\mathbf{a} \cdot \mathbf{c}$ as a vector dot product and also flag the square root of a vector as illegal. But with the vectorize operator, both $\mathbf{a} \cdot \mathbf{c}$ and the square root are performed element by element.

Here are the properties of the vectorize operator:

- The vectorize operator changes the meaning of the other *operators* and *functions* to which it applies. It does not change the meaning of the actual names and numbers. If you apply the vectorize operator to a single name, it simply draws an arrow over the name. You can use this arrow just for cosmetic purposes.
- Since operations between two arrays are performed element by element, all arrays under a vectorize operator must be the same size. 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.
- The vectorize operator applies only to the final, scalar arguments of *interp* and *linterp*. The other arguments are unaffected. See “Interpolation functions” in Chapter 14, “Statistical Functions.”

Simultaneous definitions

You can use vectors and matrices to define several variables at once. You do this by placing an array of variable names on the left side of a **:=**, and a corresponding array of values to the right. Mathcad assigns the values on the right to the corresponding names on the left. Figure 10-20 shows two such definitions.

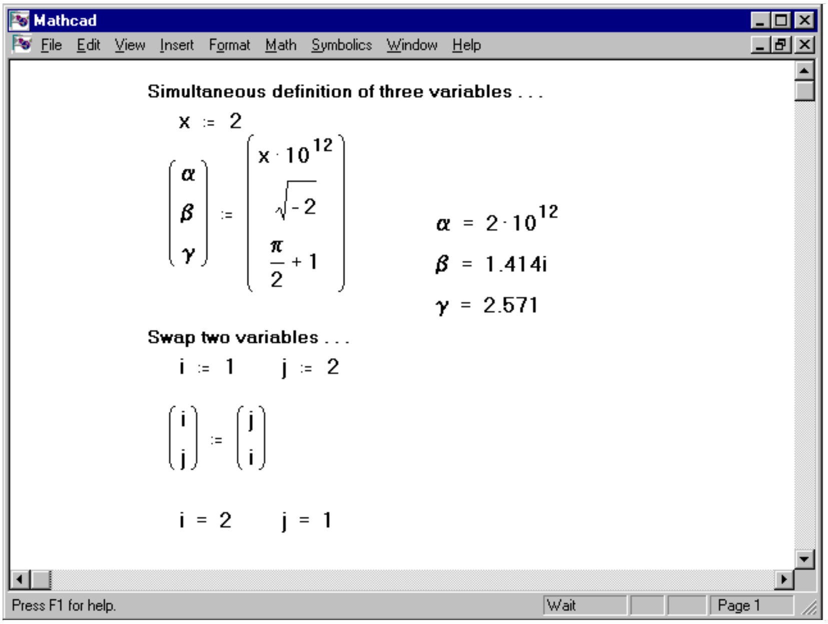


Figure 10-20: Simultaneous definitions.

The left side of a simultaneous definition is a vector or matrix whose elements are either names or subscripted variable names. The right side must be a vector or matrix expression 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 expression in the corresponding position on the right side.

Mathcad evaluates all elements on the right-hand side before assigning any of them to the left hand side. Because of this, nothing on the right hand side of an expression can depend on what is on the left hand side. You also cannot have a variable appear more than once on the left hand side.

Simultaneous definitions are useful for iterating several equations simultaneously. Several examples are described in Chapter 11, “Range Variables.”

Arrays and user-defined functions

The arguments in a function definition need not be scalar variables. They can also be vectors or matrices. Functions can return values that are scalars, vectors, or matrices.

Figure 10-21 shows some examples of functions with vector and matrix arguments and results.

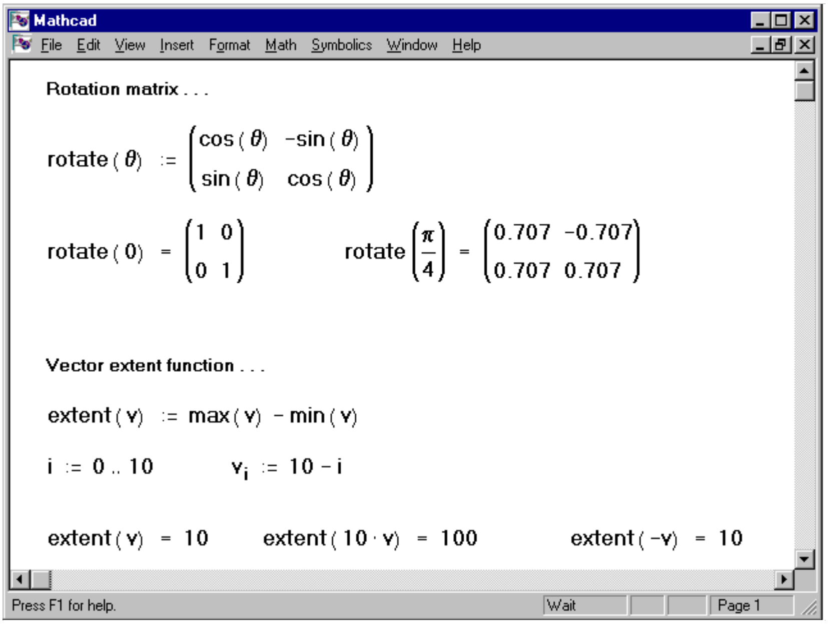


Figure 10-21: User functions used with vectors and matrices.

Note that if a function expects a vector or a matrix for an argument, it will not work on a scalar argument. In the example in Figure 10-21, trying to evaluate *extent*(3) will flag the equation with the an error message indicating that the argument must be an array.

If a function returns a vector or matrix as a result, you use the subscript and superscript operators to extract specific numbers. For example, in Figure 10-21, you could evaluate:

$$rotate(0)_{1,\,0} = 0$$

$$rotate(0)^{\langle 1 \rangle} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Nested arrays

Pro

An array element need not be a scalar. In Mathcad Professional it's possible to make an array element itself be another array. This allows you to create arrays within arrays.

These arrays behave very much like arrays whose elements are all scalars. However, there are some distinctions:

- You cannot use the **Matrix** command from the **Insert** menu to insert an array into a placeholder that's already inside an array.

- You cannot display the entire nested array. You will instead see a notation like "{3,2}" to indicate that a 3×2 array is present in a particular array location.
- Most math operators and functions do not make sense in the context of nested arrays.

The following sections explore these differences in some detail.

Defining a nested array

You define a nested array in much the same way you would define any array. The only difference is that you cannot use the **Matrix** command from the **Insert** menu when you've selected a placeholder within an existing array. You can, however, click on a placeholder in an array and type the *name* of another array as shown in Figure 10-22.

Figure 10-22 shows three ways to define a matrix of matrices: using range variables, element by element, and with the **Matrix** command from the **Insert** menu.

In addition to those methods shown in Figure 10-22, you can also use the *READPRN* function in the array of empty placeholders created using the **Matrix** command. Keep in mind, however, that you can't use *READPRN* on the same file more than once in a given matrix. The *READPRN* function is discussed more fully in Chapter 19, "Data Management."

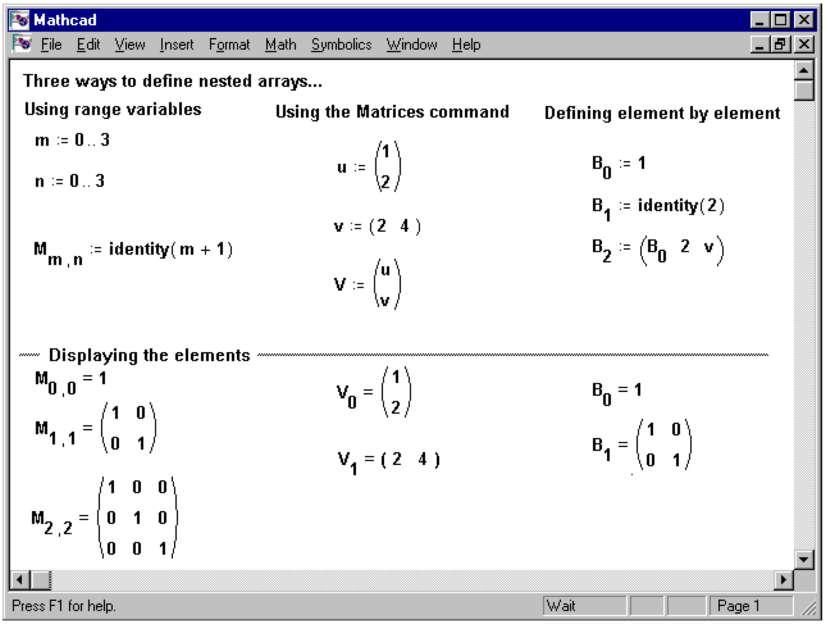


Figure 10-22: Defining nested arrays.

Displaying nested arrays

When you display a nested array using the equal sign, you won't actually see every element in every nested array. Such a display would be very cumbersome, especially when you consider that an array inside an array may itself contain arrays within it.

Instead, whenever an array element is itself an array, Mathcad indicates this by showing the number of rows and columns rather than the array itself. Figure 10-23 shows how the arrays created in Figure 10-22 would appear when displayed. Each array element is displayed either:

- As a number when the array element is simply a number, or
- As an ordered pair m, n where m and n are the number of rows and columns in the array which occupying that array element.

Note that the **B** array contains an element, B_2, which is itself a nested array. To view this array, you would simply nest your subscripts as shown in the lower-right corner of Figure 10-22.

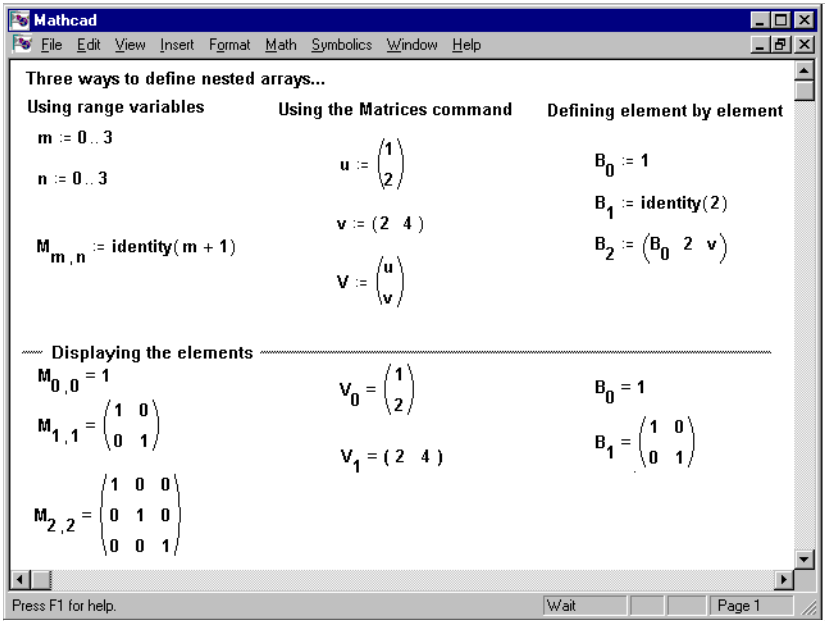


Figure 10-23: Displaying nested arrays.

Operators and functions for nested arrays

Most operators and functions do not work with nested arrays. This is because there is no universally accepted definition of what the correct behavior should be in this context. For example, there is no clear definition of what it means to “invert” such an array. When you attempt to perform the usual arithmetic operations on nested arrays, you will get either an error message or a meaningless result. For the most part, nested arrays are designed only for storing and accessing data in a convenient way.

Certain operators and functions are nevertheless useful and appropriate for nested arrays. For example, transpose does something meaningful as shown at the bottom of Figure 10-23. Operators which make sense in the context of nested arrays are:

Operation	Appearance	Keystroke	Description
Transpose	$\mathbf{A}^{\mathsf{T}}$	`[Ctrl]1`	Interchanges row and columns of **A**.
Superscript	$\mathbf{A}^{\langle n \rangle}$	`[Ctrl]6`	*n*th column of array A. Returns a vector.
Vector subscript	v_n	`[`	*n*th element of a vector.
Matrix subscript	$A_{m,\,n}$	`[`	(*m, n*)th element of a matrix.
Boolean equals	$w = z$	`[Ctrl]=`	Boolean equals. Returns 1 if the two nested arrays, along with all nested arrays contained within them, are identical; otherwise returns 0.

Useful functions for nested arrays tend to be those having to do with the number of rows and columns in an array or those used for joining or dividing arrays. In particular, you can use the *rows* and *cols* functions to distinguish between scalar array elements and array elements which are themselves arrays. Both these functions return a zero in the former case and the appropriate number in the latter. The functions you'll find useful when working with nested arrays are:

Function Name	Returns . . .
rows(**A**)	Number of rows in matrix **A**.
cols(**A**)	Number of columns in matrix **A**.
length(**v**)	Number of elements in vector **v**.
last(**v**)	Index of last element in vector **v**.
augment(**A**, **B**)	An array formed by placing **A** and **B** side by side. The arrays **A** and **B** must have the same number of rows.
stack(**A**, **B**)	An array formed by placing **A** above **B**. The arrays **A** and **B** must have the same number of columns.
submatrix(**A**, *ir, jr, ic, jc*)	A submatrix of **A** consisting of all elements contained in rows *ir* through *jr* and columns *ic* through *jc*. To maintain order of rows and/or columns, make sure $ir \le jr$ and $ic \le jc$, otherwise order of rows and/or columns will be reversed.