

Mathematica 3.0

Mathematica 3.0 allows you to type in several lines of input before executing them. Each line of input is ordinarily followed by **Enter**, but when you are ready to execute the input lines, type **Shift** and **Enter** together after the last input line in the series. For example, in Figure D.13, you could type in all four lines of input and then execute them as a group. In Figures D.13 through D.18, we list the output from each input line next to that line, even though several input lines may, in fact, appear consecutively on the computer screen.

Mathematica 3.0 assigns a label of the form “In[k]=” (for some positive integer k) to each input line, and similarly, an “Out[k]=” label to each output line. The In[k] label will not print on the screen until you have used **Shift+Enter** instead of just **Enter**, and only the first In[k] label will appear in each group. In Figure D.13, do not type in the In[k]= labels. A line label often is a handy way to refer to the contents of a previous line in a *Mathematica 3.0* expression, as shown in line In[5] in Figure D.13. Even if a line number is not displayed, it can often be inferred for use in subsequent calculations. To save space, we only illustrate the In[] and Out[] labels in Figure D.13.

In *Mathematica 3.0*, you can refer to the most recently computed expression using the symbol “%”.

Input of Vectors and Matrices; Fundamental Operations

Vectors and matrices are entered with set notation (using braces, such as “{ }”). A matrix is expressed as a set of vectors representing successive rows of the matrix. For this reason, in *Mathematica 3.0* matrices are not typically displayed with their columns aligned vertically, but as sets of vectors. However, the command **MatrixForm**[M] will cause *Mathematica 3.0* to display the matrix M in the customary matrix format. It is often useful to assign names to vectors and matrices as they are entered. Use an equal sign (“=”) to give a symbolic name to a vector or matrix. For matrices, use the “=” sign *inside* the **MatrixForm** command, as illustrated in Figure D.14.

Matrices can also be entered using the **Create/Table/Matrix/Palette** sub-heading under the **Input** menu. A shortcut to this is **Shift+Ctrl+C** (hold down the **Shift** and **Ctrl** keys while typing **C**).¹ You choose the size of the matrix, and then are presented with a template to fill in the entries. Use the **Tab** key to move from entry to entry within the matrix. Use the arrow key to place the cursor outside the matrix before hitting **Enter** or **Shift+Enter** to finish entering the matrix.

Figures D.13 and D.14 illustrate the input of vectors/matrices as well as the fundamental operations of dot product, scalar multiplication, matrix multiplication, transpose, and inverse. Addition and subtraction are performed using “+” and “-”, respectively. You may use “*” for scalar multiplication, or merely leave a space between the scalar and the vector or matrix. A dot product of

¹There are similar alternatives, shortcuts, and/or toolbar icons for many commands, but we will not discuss all the possibilities for every command we illustrate.

two vectors or a multiplication of two matrices is performed by placing a “.” (period) between them. The function `MatrixPower[M,n]` function calculates the n^{th} power of the matrix `M`. The function `Inverse[M]` calculates the inverse of the square matrix `M`. The `Transpose[M]` function calculates the transpose of the matrix `M`. Note the use of brackets (“ [] ”) in these last three functions. Placing “//N” after a calculation causes a decimal approximation of the result to be printed out. To specify that n significant digits are to be displayed, enter `N[X, n]`, where `X` is the result to be printed.

Input	Output
<code>In[1] = v1 = {5,7,-4}</code>	<code>Out[1] = {5,7,-4}</code>
<code>In[2] = v2 = {x,y,z}</code>	<code>Out[2] = {x,y,z}</code>
<code>In[3] = 2*v1+3*v2</code>	<code>Out[3] = {10 + 3x, 14 + 3y, -8 + 3z}</code>
<code>In[4] = v1.v2</code>	<code>Out[4] = 5x + 7y - 4z</code>
<code>In[5] = In[1].Out[2]</code>	<code>Out[5] = 5x + 7y - 4z</code>

Figure D.13: *Mathematica 3.0* session: vectors; fundamental vector operations

Input	Output
$M1 = \{\{4,-1,6,-2\},\{-3,2,-3,2\},\{-6,8,1,3\}\}$	$\{\{4,-1,6,-2\},\{-3,2,-3,2\},\{-6,8,1,3\}\}$
<code>MatrixForm[M1]</code>	$\begin{pmatrix} 4 & -1 & 6 & -2 \\ -3 & 2 & -3 & 2 \\ -6 & 8 & 1 & 3 \end{pmatrix}$
<code>MatrixForm[M2 = {\{2,-3,0\},\{6,8,-1\},\{3,1,-2\},\{2,-4,-2\}}</code>	$\begin{pmatrix} 2 & -3 & 0 \\ 6 & 8 & -1 \\ 3 & 1 & -2 \\ 2 & -4 & -2 \end{pmatrix}$
<code>MatrixForm[M3 = M1-2*Transpose[M2]]</code>	$\begin{pmatrix} 0 & -13 & 0 & -6 \\ 3 & -14 & -5 & 10 \\ -6 & 10 & 5 & 7 \end{pmatrix}$
<code>MatrixForm[M4 = M1. M2]</code>	$\begin{pmatrix} 16 & -6 & -7 \\ 1 & 14 & 0 \\ 45 & 71 & -16 \end{pmatrix}$
<code>MatrixForm[M5 = MatrixPower[M4, -1]]</code>	$\begin{pmatrix} -\frac{224}{233} & -\frac{593}{233} & \frac{98}{233} \\ \frac{16}{233} & \frac{59}{233} & -\frac{7}{233} \\ -\frac{559}{233} & -\frac{1406}{233} & \frac{230}{233} \end{pmatrix}$
<code>MatrixForm[M5 = Inverse[M4]]</code>	$\begin{pmatrix} -\frac{224}{233} & -\frac{593}{233} & \frac{98}{233} \\ \frac{16}{233} & \frac{59}{233} & -\frac{7}{233} \\ -\frac{559}{233} & -\frac{1406}{233} & \frac{230}{233} \end{pmatrix}$
<code>MatrixForm[M5] // N</code>	$\begin{pmatrix} -0.961373 & -2.54506 & 0.420601 \\ 0.0686695 & 0.253219 & -0.0300429 \\ -2.39914 & -6.03433 & 0.987124 \end{pmatrix}$
<code>MatrixForm[N[M5,3]]</code>	$\begin{pmatrix} -0.961 & -2.55 & 0.421 \\ 0.0687 & 0.253 & -0.03 \\ -2.4 & -6.03 & 0.987 \end{pmatrix}$

Figure D.14: *Mathematica 3.0* session: matrices; fundamental matrix operations

Notice that both ways of calculating matrix M5 in Figure D.14 (using `MatrixPower` and `Inverse`) produce the same result.

Solving a Linear System; Gauss-Jordan Row Reduction Method

You can solve a linear system using the `Solve[{eqns},{vars}]` function. The set `{eqns}` contains each equation of the system in turn, but with “==” (double equal sign) used in place of each equal sign. The set `{vars}` contains all the variables of the system. When printing out the solution set of such a system, *Mathematica 3.0* expresses each dependent variable in terms of the independent variables. If there is no solution to the system, the output is “{ }”.

You can also solve a linear system using the `RowReduce[M]` function. This function calculates the reduced row echelon form of the (possibly augmented) matrix `M`. The matrix `M6` in Figure D.15 is the augmented matrix for a linear system with an infinite solution set.

In Figure D.15, the same linear system is solved using both `Solve` and `RowReduce`. Using either method, you can easily see that the general solution set is $\{(-3c + 4, 2c + 5, c, -2)\}$.

Input	Output
<code>Solve[{3x+y+7z+2w==13, 2x-4y+14z-w==-10, 5x+11y-7z+8w==59, 2x+5y-4z-3w==39},{x,y,z,w}]</code>	$\left\{ \begin{aligned} x &\longrightarrow 4 - 3z, \\ y &\longrightarrow 5 + 2z, \\ w &\longrightarrow -2 \end{aligned} \right\}$
<code>MatrixForm[M6={{3,1,7,2,13}, {2,-4,14,-1,-10},{5,11,-7,8,59}, {2,5,-4,-3,39}}]</code>	$\begin{pmatrix} 3 & 1 & 7 & 2 & 13 \\ 2 & -4 & 14 & -1 & -10 \\ 5 & 11 & -7 & 8 & 59 \\ 2 & 5 & -4 & -3 & 39 \end{pmatrix}$
<code>MatrixForm[M7 = RowReduce[M6]]</code>	$\begin{pmatrix} 1 & 0 & 3 & 0 & 4 \\ 0 & 1 & -2 & 0 & 5 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Figure D.15: *Mathematica 3.0* session: solution of a linear system;
row reduction

Determinants; Eigenvalues/Eigenvectors; Characteristic Polynomial

The `Det[M]` function calculates the determinant of the square matrix `M`. The function `Eigenvalues[M]` calculates the eigenvalues of a square matrix `M`. The `Eigenvectors[M]` function returns a set of n eigenvectors for the $n \times n$ matrix `M`. However, if `M` is not diagonalizable, then `Eigenvectors` returns as many linearly independent eigenvectors for `M` as possible, but includes the zero vector as often as necessary to ensure that a total of n vectors is output.

The matrix **M8** in Figure D.16 has two eigenvalues, $\lambda_1 = -5$, with algebraic and geometric multiplicity 1, and $\lambda_2 = 3$, with algebraic multiplicity 3 and geometric multiplicity 2.

Notice in the list of eigenvectors for **M8** that $[-1, 1, -2, 8]$ is an eigenvector for $\lambda_1 = -5$, while the other two nonzero vectors are eigenvectors for $\lambda_2 = 3$. Since the algebraic multiplicity of $\lambda_2 = 3$ is exactly 1 greater than its geometric multiplicity, a single zero vector is included in the list.

The characteristic polynomial of a square matrix can be computed directly in *Mathematica 3.0* using determinants. The $k \times k$ identity matrix is created using the function `IdentityMatrix[k]`. If the most recent output line consists of a polynomial, you can factor it using the command `Factor[%]`.

Input	Output
<code>MatrixForm[M8={{5,2,0,1},{-2,1,0,-1}, {4,4,3,2},{16,0,-8,-5}}]</code>	$\begin{pmatrix} 5 & 2 & 0 & 1 \\ -2 & 1 & 0 & -1 \\ 4 & 4 & 3 & 2 \\ 16 & 0 & -8 & -5 \end{pmatrix}$
<code>Eigenvalues[M8]</code>	$\{-5, 3, 3, 3\}$
<code>MatrixForm[Eigenvectors[M8]]</code>	$\begin{pmatrix} -1 & 1 & -2 & 8 \\ 1 & -2 & 0 & 2 \\ 1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Figure D.16: *Mathematica 3.0* session: eigenvalues; eigenvectors

Of course, a basis of eigenvectors for any eigenvalue λ of a square matrix **M** can also be calculated by row reducing the matrix $\lambda \mathbf{I}_n - \mathbf{M}$, setting each independent variable in turn equal to 1 with all others equal to 0, and then solving for the dependent variables. These operations are illustrated in Figure D.17.

Linearly independent eigenvectors for the earlier matrix **M8**, for the eigenvalue $\lambda_2 = 3$, are found from the reduced row echelon form matrix **M10** for $\mathbf{M9} = 3\mathbf{I}_4 - \mathbf{M8}$ in Figure D.17. Letting the third column variable in **M10** equal 1 and fourth column variable equal 0, we obtain $[\frac{1}{2}, -\frac{1}{2}, 1, 0]$, and then by letting the third column variable equal 0 and fourth column variable equal 1, we obtain $[\frac{1}{2}, -1, 0, 1]$. Since these are scalar multiples of $[1, -1, 2, 0]$ and $[1, -2, 0, 2]$, respectively, it is obvious that $\{[\frac{1}{2}, -\frac{1}{2}, 1, 0], [\frac{1}{2}, -1, 0, 1]\}$ spans the same two-dimensional subspace of \mathbb{R}^4 as the set $\{[1, -2, 0, 2], [1, -1, 2, 0]\}$ of eigenvectors for $\lambda_2 = 3$, obtained earlier from the `Eigenvectors` function.

Input	Output
<code>Det[x*IdentityMatrix[4]-M8]</code>	$-135 + 108x - 18x^2 - 4x^3 + x^4$
<code>Factor[%]</code>	$(-3 + x)^3(5 + x)$
<code>MatrixForm[M9 = 3*IdentityMatrix[4]-M8]</code>	$\begin{pmatrix} -2 & -2 & 0 & -1 \\ 2 & 2 & 0 & 1 \\ -4 & -4 & 0 & -2 \\ -16 & 0 & 8 & 8 \end{pmatrix}$
<code>MatrixForm[M10 = RowReduce[M9]]</code>	$\begin{pmatrix} 1 & 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & \frac{1}{2} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Figure D.17: *Mathematica 3.0* session: characteristic polynomial via determinant; direct calculation of eigenspace

Gram-Schmidt Process

There is a function to perform the Gram-Schmidt Process in *Mathematica 3.0*, but to use it, the utility `LinearAlgebra`Orthogonalization`` must be loaded. (This is done by typing “<<” before the name of the utility, as in Figure D.18. Note that a single left quotation mark is used before and after the word `Orthogonalization`.)

The `GramSchmidt[vecs]` (no hyphen) function replaces a given set `vecs` of linearly independent vectors with an orthonormal basis for the span of those vectors. To produce an orthogonal basis instead, include `Normalized -> False` after the set `vecs`. The function `Normalize[v]` normalizes a given vector `v`. (The `LinearAlgebra`Orthogonalization`` utility is also needed for the `Normalize` function.)

These operations are illustrated in Figure D.18. We first find an orthogonal basis for the subspace of \mathbb{R}^4 spanned by the vectors `v5`, `v6`, and `v7` using the `GramSchmidt` function. Each vector in the orthogonal basis can be normalized individually to produce an orthonormal basis, and we illustrate this for the vector `v5`. Finally, we display the entire orthonormal basis using the `GramSchmidt` function.

Input	Output
<<LinearAlgebra'Orthogonalization'	(loads necessary utility)
v5 = {2, 1, 0, -1}	{2, 1, 0, -1}
v6 = {1, 0, 2, -1}	{1, 0, 2, -1}
v7 = {0, -2, 1, 0}	{0, -2, 1, 0}
GramSchmidt[{v5, v6, v7}, Normalized->False]	$\left\{ \{2, 1, 0, -1\}, \left\{0, -\frac{1}{2}, 2, -\frac{1}{2}\right\}, \left\{\frac{2}{3}, -\frac{4}{3}, -\frac{1}{3}, 0\right\} \right\}$
Normalize[v5]	$\left\{ \sqrt{\frac{2}{3}}, \frac{1}{\sqrt{6}}, 0, -\frac{1}{\sqrt{6}} \right\}$
GramSchmidt[{v5, v6, v7}]	$\left\{ \left\{ \sqrt{\frac{2}{3}}, \frac{1}{\sqrt{6}}, 0, -\frac{1}{\sqrt{6}} \right\}, \left\{ 0, -\frac{1}{3\sqrt{2}}, \frac{2\sqrt{2}}{3}, -\frac{1}{3\sqrt{2}} \right\}, \left\{ \frac{2}{\sqrt{21}}, -\frac{4}{\sqrt{21}}, -\frac{1}{\sqrt{21}}, 0 \right\} \right\}$

Figure D.18: *Mathematica 3.0* session: Gram-Schmidt Process;
orthogonal and orthonormal bases

You should verify that $\{[2, 1, 0, -1], [0, -\frac{1}{2}, 2, -\frac{1}{2}], [\frac{2}{3}, -\frac{4}{3}, -\frac{1}{3}, 0]\}$ is an orthogonal set of vectors spanning the same subspace of \mathbb{R}^4 as $\{[2, 1, 0, -1], [1, 0, 2, -1], [0, -2, 1, 0]\}$, and that $\left\{ \left[\sqrt{\frac{2}{3}}, \frac{1}{\sqrt{6}}, 0, -\frac{1}{\sqrt{6}} \right], \left[0, -\frac{1}{3\sqrt{2}}, \frac{2\sqrt{2}}{3}, -\frac{1}{3\sqrt{2}} \right], \left[\frac{2}{\sqrt{21}}, -\frac{4}{\sqrt{21}}, -\frac{1}{\sqrt{21}}, 0 \right] \right\}$ is an orthonormal set of vectors spanning this same subspace.