

MATLAB 5 for Windows (Student Edition)

You are prompted for each line of input in *MATLAB 5 for Windows (Student Edition)*¹ with the symbol “EDU>>”. Note that *MATLAB 5* distinguishes between uppercase and lowercase letters. For example, the variable `M9` is not the same in *MATLAB 5* as the variable `m9`.

Input of Vectors and Matrices; Fundamental Operations

To enter a vector, simply type the list of entries (separated by commas) within a pair of brackets (such as “[]”). Matrix input is done similarly, with a semicolon at the end of each row. 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. Notice that usually no brackets/braces are displayed when vectors and matrices are printed out.

Figures D.19 and D.20 illustrate the input of vectors/matrices as well as the fundamental operations of dot product, scalar multiplication, matrix multiplication, transpose, and inverse. The `dot` function calculates the dot product of two vectors. Addition and subtraction are performed using “+” and “-”, respectively. Scalar multiplication can never be implied simply by putting a scalar next to a vector; instead, you must use “*”. Matrix multiplication is also indicated by “*”. The “^” symbol is used to find integer powers of a square matrix. In particular, “^(-1)” can be used to calculate the inverse of a square matrix, although the answer may be given as a decimal approximation. The `inv` function can also be used to find the inverse of a square matrix. The “'” symbol denotes the transpose of a matrix.

The `sym` function is used to create a symbolic matrix or vector. This is used for vectors or matrices that contain variables or fractions, as illustrated with the vector `v2` in Figure D.19. When typing in such a vector or matrix, it must be put inside single quotes, as shown for the vector `v2` in Figure D.19. Using `sym` on an existing vector or matrix will turn it into a symbolic vector or matrix. Symbolic vectors and matrices are displayed with brackets around each row, while nonsymbolic vectors and matrices have no such brackets displayed. Note that the combination of `sym` and `inv` used on matrix `M4` in Figure D.20 displays the exact inverse of that matrix.

Input	Output
EDU>> <code>v1=[5,7,-4]</code>	<code>v1 = 5 7 -4</code>
EDU>> <code>v2=sym(' [x,y,z]')</code>	<code>v2 = [x, y, z]</code>
EDU>> <code>v3=2*v1+3*v2</code>	<code>v3 = [10 + 3 × x, 14 + 3 × y, -8 + 3 × z]</code>
EDU>> <code>dot(v1,v2)</code>	<code>ans = 5 × x + 7 × y - 3 × z</code>

Figure D.19: *MATLAB 5* session: vectors; fundamental vector operations

¹For brevity, we will henceforth refer to the *Student Edition* of *MATLAB 5 for Windows* as “*MATLAB 5*.” There is a *Professional Edition* of *MATLAB 5*, with greater capabilities than the *Student Edition*. Our comments regarding *MATLAB 5* pertain to the *Student Edition*, and not necessarily to the *Professional Edition*.

Input	Output
EDU>> M1=[4,-1,6,-2; -3,2,-3,2;-6,8,1,3]	M1 = $\begin{bmatrix} 4 & -1 & 6 & -2 \\ -3 & 2 & -3 & 2 \\ -6 & 8 & 1 & 3 \end{bmatrix}$
EDU>> M2=[2,-3,0; 6,8,-1; 3,1,-2; 2,-4,-2]	M2 = $\begin{bmatrix} 2 & -3 & 0 \\ 6 & 8 & -1 \\ 3 & 1 & -2 \\ 2 & -4 & -2 \end{bmatrix}$
EDU>> M3=M1-2*M2'	M3 = $\begin{bmatrix} 0 & -13 & 0 & -6 \\ 3 & -14 & -5 & 10 \\ -6 & 10 & 5 & 7 \end{bmatrix}$
EDU>> M4=M1*M2	M4 = $\begin{bmatrix} 16 & -6 & -7 \\ 1 & 14 & 0 \\ 45 & 71 & -16 \end{bmatrix}$
EDU>> M5=sym(inv(M4))	M5 = $\begin{bmatrix} -224/233, & -593/233, & 98/233 \\ 16/233, & 59/233, & -7/233 \\ -559/233, & -1406/233 & 230/233 \end{bmatrix}$
EDU>> M5=sym(M4^(-1))	M5 = $\begin{bmatrix} -224/233, & -593/233, & 98/233 \\ 16/233, & 59/233, & -7/233 \\ -559/233, & -1406/233 & 230/233 \end{bmatrix}$
EDU>> M5=M4^(-1)	M5 = $\begin{bmatrix} -0.9614 & -2.5451 & 0.4206 \\ 0.0687 & 0.2532 & -0.0300 \\ -2.3991 & -6.0343 & 0.9871 \end{bmatrix}$

Figure D.20: *MATLAB 5* session: matrices; fundamental matrix operations

Note the difference in output in Figure D.20 between the symbolic version of **M5** and the nonsymbolic (final) version. Although we do not illustrate it here, the **numeric** function gives approximate values for a symbolic object containing only numbers.

Solving a Linear System; Gauss-Jordan Row Reduction Method

You can solve a linear system with the **solve** function. Type a pair of brackets containing the variables of the system in lexicographic order, and then an equal sign followed by **solve**, and finally a pair of parentheses containing each equation in the system in turn (separated by commas). Note that each equation in the system should be enclosed within a pair of apostrophes (‘ ’). When printing out the solution set of such a system, *MATLAB 5* expresses each dependent variable in terms of the independent variables. If there is no solution to the system, the output is “x = [empty sym]”, where “x” represents the first variable in the system.

You can also solve a system using the `rref` function. This function calculates the reduced row echelon form of a (possibly augmented) matrix. The matrix `M6` in Figure D.21 is the augmented matrix for a linear system with an infinite solution set.

These functions are illustrated in Figure D.21, in which the same linear system is solved using both `solve` and `rref`. Using either method, you can easily see that the general solution set is $\{(-3c + 4, 2c + 5, c, -2)\}$.

Input	Output
<pre>EDU>> [w,x,y,z] = solve ('3*x+y+7*z+2*w=13', '2*x-4*y+14*z-w=-10', '5*x+11*y-7*z+8*w=59', '2*x+5*y-4*z-3*w=39')</pre>	<pre>w = -2 x = -3*z + 4 y = 2*z + 5 z = z</pre>
<pre>EDU>> M6=[3,1,7,2,13; 2,-4,14,-1,-10; 5,11,-7,8,59; 2,5,-4,-3,39]</pre>	<pre>M6 = 3 1 7 2 13 2 -4 14 -1 -10 5 11 -7 8 59 2 5 -4 -3 39</pre>
<pre>EDU>> M7=rref(M6)</pre>	<pre>M7 = 1 0 3 0 4 0 1 -2 0 5 0 0 0 1 -2 0 0 0 0 0</pre>

Figure D.21: *MATLAB 5* session: solution of a linear system; row reduction

Determinants; Eigenvalues/Eigenvectors; Characteristic Polynomial:

The `det` function calculates the determinant of a square matrix. The coefficients (alone) of the characteristic polynomial (from highest to lowest degree) are computed using the `poly` function. Using `poly` on a symbolic matrix causes the characteristic polynomial to be displayed in the customary polynomial form.

Eigenvalues for a given square matrix can be found by calculating the roots of its characteristic polynomial. One way to do this is to use the `roots` function on the coefficients found by `poly`. To find the roots of a symbolic polynomial, use the `solve` command instead.

A more direct way to calculate eigenvalues (and eigenvectors as well) is to create an appropriate equation involving the `eigensys` function. Place $[V, E]$ on the left side of the equation, and `eigensys(M)` on the right side, where `M` is the given matrix. This will create a matrix `E` containing all the eigenvalues of `M` on its main diagonal, and a matrix `V` whose *columns* are a set of linearly independent eigenvectors for these eigenvalues. Regardless of the method you use to find eigenvalues and eigenvectors, the complex number “ i ” (see Appendix C of the textbook) may appear in the result.

These functions are illustrated in Figure D.22, where the matrix $\mathbf{M8}$ has two eigenvalues, $\lambda_1 = -5$, with algebraic and geometric multiplicity 1, and $\lambda_2 = 3$, with algebraic multiplicity 3 and geometric multiplicity 2. The *columns* of \mathbf{V} give a linearly independent set of eigenvectors for $\mathbf{M8}$: $[1, 0, 4, -2]$, $[0, 1, 2, -2]$, and $[-1, 1, -2, 8]$.

Input	Output
EDU>> M8=[5,2,0,1;-2,1,0,-1; 4,4,3,2;16,0,-8,-5]	$\mathbf{M8} = \begin{bmatrix} 5 & 2 & 0 & 1 \\ -2 & 1 & 0 & -1 \\ 4 & 4 & 3 & 2 \\ 16 & 0 & -8 & -5 \end{bmatrix}$
EDU>> det(M8)	ans = -135
EDU>> poly(M8)	ans = 1.0000 - 4.0000 - 18.0000 108.0000 - 135.0000
EDU>> roots(poly(M8))	ans = -5.0000 3.0000 3.0000 + 0.0000i 3.0000 - 0.0000i
EDU>> poly(sym(M8))	ans = $x^4 - 4 \times x^3 - 18 \times x^2 + 108 \times x - 135$
EDU>> solve(poly(sym(M8)))	ans = $\begin{bmatrix} -5 \\ 3 \\ 3 \\ 3 \end{bmatrix}$
EDU>> [V,E] = eigensys(M8)	$\mathbf{V} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 4 & 2 & -2 \\ -2 & -2 & 8 \end{bmatrix}$ $\mathbf{E} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$

Figure D.22: *MATLAB 5* session: characteristic polynomial; eigenvalues; eigenvectors

Of course, a basis of eigenvectors for any eigenvalue λ of a (square) matrix \mathbf{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. The $k \times k$ identity matrix is represented by `eye(k)`.

In Figure D.23, linearly independent eigenvectors for the earlier matrix $\mathbf{M8}$, for eigenvalue $\lambda_2 = 3$, are found from the reduced row echelon form matrix $\mathbf{M10}$ of $\mathbf{M9} = 3\mathbf{I}_4 - \mathbf{M8}$. Letting the third column variable of $\mathbf{M10}$ equal 1 and its fourth column variable equal 0, we obtain $[\frac{1}{2}, -\frac{1}{2}, 1, 0]$, and letting its third column variable equal 0 and fourth column variable equal 1, we obtain $[\frac{1}{2}, -1, 0, 1]$. (You can easily verify 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 $\{[1, 0, 4, -2], [0, 1, 2, -2]\}$ of eigenvectors obtained earlier from the `eigensys` function.)

Input	Output
EDU>> M9 = 3*eye(4)-M8	M9 = $\begin{bmatrix} -2 & -2 & 0 & -1 \\ 2 & 2 & 0 & 1 \\ -4 & -4 & 0 & -2 \\ -16 & 0 & 8 & 8 \end{bmatrix}$
EDU>> M10 = rref(M9)	M10 = $\begin{bmatrix} 1.0000 & 0 & -0.5000 & -0.5000 \\ 0 & 1.0000 & 0.5000 & 1.0000 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

Figure D.23: *MATLAB 5* session: direct calculation of eigenspace

Gram-Schmidt Process

The necessary calculations for the Gram-Schmidt Process can be performed in *MATLAB 5* in the manner illustrated in Figure D.24. We begin with a given set of 3 linearly independent vectors $\{[2, 1, 0, -1], [1, 0, 2, -1], [0, -2, 1, 0]\}$ in \mathbb{R}^4 (vectors **v5**, **v6**, and **v7**), and construct an orthogonal basis (**v5**, **v8**, **v9**) for the span of those vectors. We then produce an orthonormal basis (**v10**, **v11**, **v12**) for the span by dividing each orthogonal basis vector by its length. The **norm** function calculates the length of a given vector. The results of all of these computations can be converted to symbolic vectors using the **sym** function.

Analogous calculations can be made if you begin with symbolic vectors. However, the **norm** function is not defined for symbolic vectors. Instead, the norm of a symbolic vector can be calculated by taking the square root of the dot product of the vector with itself. (Another approach is to first apply the **numeric** function to the symbolic vector, take the **norm**, and then convert that result to a symbolic value.)

Input	Output
EDU>> v5=[2,1,0,-1]	v5 = 2 1 0 -1
EDU>> v6=[1,0,2,-1]	v6 = 1 0 2 -1
EDU>> v7=[0,-2,1,0]	v7 = 0 -2 1 0
EDU>> v8=v6-(dot(v6,v5)/dot(v5,v5))*v5	v8 = 0 -0.5000 2.0000 -0.5000
EDU>> sym(v8)	ans = [0, -1/2, 2, -1/2]
EDU>> v9=v7-(dot(v7,v5)/dot(v5,v5))*v5 -(dot(v7,v8)/dot(v8,v8))*v8	v9 = 0.6667 -1.3333 -0.3333 0
EDU>> sym(v9)	ans = [2/3, -4/3, -1/3, 0]
EDU>> v10=v5 / norm(v5)	v10 = 0.8165 0.4082 0 -0.4082
EDU>> sym(v10)	ans = [sqrt(2/3), sqrt(1/6), 0, -sqrt(1/6)]
EDU>> v11=v8 / norm(v8)	v11 = 0 -0.2357 0.9428 -0.2357
EDU>> sym(v11)	ans = [0, -sqrt(1/18), sqrt(8/9), -sqrt(1/18)]
EDU>> v12=v9 / norm(v9)	v12 = 0.4364 -0.8729 -0.2182 0
EDU>> sym(v12)	ans = [sqrt(4/21), -sqrt(16/21), -sqrt(1/21), 0]

Figure D.24: *MATLAB 5* session: Gram-Schmidt Process; orthogonal and orthonormal bases

You can easily verify that $\{[2, 1, 0, -1], [0, -\frac{1}{2}, 2, -\frac{1}{2}], [\frac{2}{3}, -\frac{4}{3}, -\frac{1}{3}, 0]\}$ (vectors v_5 , v_8 , and v_9) is an orthogonal set of vectors spanning the same subspace of \mathbb{R}^4 as $\{[2, 1, 0, -1], [1, 0, 2, -1], [0, -2, 1, 0]\}$. An orthonormal basis for the same subspace is given by the set $\left\{ \left[\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{6}}, 0, -\sqrt{\frac{1}{6}} \right], \left[0, -\sqrt{\frac{1}{18}}, \sqrt{\frac{8}{9}}, -\sqrt{\frac{1}{18}} \right], \left[\sqrt{\frac{4}{21}}, -\sqrt{\frac{16}{21}}, -\sqrt{\frac{1}{21}}, 0 \right] \right\}$ (vectors $\text{sym}(v_{10})$, $\text{sym}(v_{11})$, and $\text{sym}(v_{12})$).