

Maple Tutorial #4: Matrices and Vectors*

Christos A. Xenophontos and Scott R. Fulton
Department of Mathematics and Computer Science
Clarkson University

Maple V Release 5
Fall 1999

Introduction

In this tutorial we will learn how to manipulate matrices and vectors using Maple. The symbolic capabilities of Maple allow us to define matrices (and vectors) whose entries are not necessarily numbers! Generic constants, expressions and even functions can be used as entries, giving Maple a clear advantage over other mathematical software packages such as Matlab. We will comment on some of the differences between these two at the end of this tutorial.

The field of *linear algebra* contains many topics, some maybe too advanced to address in this tutorial. The focus here will be on the simple matrix (and vector) manipulations and the solution of linear systems, with some additional topics towards the end of this tutorial. The goal is to learn how to:

- Define and manipulate matrices and vectors
- Solve linear systems
- Compute eigenvalues and eigenvectors

As with the previous tutorials, you'll get the most out of this if you read the text carefully and *try* the commands as shown. After finishing this tutorial, you should have enough background to be able to explore the linear algebra capabilities of Maple in whatever direction your interests and needs might dictate.

*Technical Report 99-06, Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY 13699-5815.

Matrices

Matrices arise naturally in the solution of linear systems of equations, such as

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n.$$

The coefficients a_{ij} , $i, j = 1, \dots, n$ and the right hand sides b_i , $i = 1, \dots, n$ are given and the goal is to find x_i , $i = 1, \dots, n$ that satisfy the above equations simultaneously. This system can be written in matrix form as

$$A\mathbf{x} = \mathbf{b},$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

is an $n \times n$ matrix and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

are $n \times 1$ matrices. When the number of columns in a matrix is one, the matrix is often called a vector, hence \mathbf{x} and \mathbf{b} are essentially n -dimensional vectors. (Even though there are some mathematical differences between n -dimensional vectors and $n \times 1$ matrices, we will not distinguish them in this tutorial.)

In Maple we can define matrices in a very natural way and perform all of the standard operations on them, just like we would “by hand” in a linear algebra course. The first step is to ask Maple to load the *linear algebra* package, in order for all the commands associated with matrices to be made available to us. (Maple has a number of special-purpose packages that must be loaded before they can be used, such as the *student* package that you encountered in Tutorial 3.) To read the *linear algebra* package into Maple you use

```
> with(linalg):  
Warning, new definition for norm  
Warning, new definition for trace
```

(The colon suppresses the listing of the many commands being read in; if you would like to see this list, use a semi-colon instead). The warning we get from Maple simply means that the definitions for the `norm` and `trace` commands have been updated. This will not affect us and this tutorial. There are several ways to define matrices in Maple. One is by using the `matrix` command:

```
> A:=matrix([[2,1],[-1,5]]);
```

$$A := \begin{bmatrix} 2 & 1 \\ -1 & 5 \end{bmatrix}$$

Note that each bracketed “list” of elements (separated by commas) corresponds to a row of A and all rows are enclosed in brackets to compose the matrix. The same command could also be used in the following way:

```
> A:=matrix(2,2,[2,1,-1,5]);
```

$$A := \begin{bmatrix} 2 & 1 \\ -1 & 5 \end{bmatrix}$$

Note that when defining the matrix this way, we first let Maple know that it is a 2×2 matrix and then list all of its elements in order (within one set of brackets), without separating each row.

Since a matrix is a special case of a two-dimensional *array*, we could define A using the `array` command.

```
> A:=array(1..2,1..2,[[2,1],[-1,5]]);
```

$$A := \begin{bmatrix} 2 & 1 \\ -1 & 5 \end{bmatrix}$$

Independently of the way we define a matrix, when we ask Maple to display it we must use the `evalm` command, which stands for *evaluate matrix*, otherwise Maple will not “know” that the variable A represents a matrix.

```
> A;
```

A

```
> evalm(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ -1 & 5 \end{bmatrix}$$

Another way to “see” the matrix displayed is the use of the `print` command; `print(A)` will produce the same result as `evalm(A)` but without evaluating any possible expressions within the matrix.

Using “brackets”, we can refer to a specific element of a matrix, for example

```
> A[2,1];
```

$$-1$$

gives the element in the second row and first column of A .

```
> A[3,1];
```

Error, 1st index, 3, larger than upper array bound 2

produces an error, as expected, since there is no third row in the matrix.

Let us now define a few other matrices that we will use in this part of the tutorial.

```
> B:=matrix([[a,-b],[c,1/d]]);
```

$$B := \begin{bmatrix} a & -b \\ c & 1/d \end{bmatrix}$$

```
> C:=matrix([[7,-2,1],[9,3,8]]);
```

$$C := \begin{bmatrix} 7 & -2 & 1 \\ 9 & 3 & 8 \end{bmatrix}$$

As mentioned earlier, the elements of a matrix do not have to be numbers, so the definition of B above is allowed. We could add or subtract two *compatible* matrices (i.e. two matrices of the same *size*), as long as we remember to use the `evalm` command.

```
> evalm(A+B);
```

$$\begin{bmatrix} 2+a & 1-b \\ -1+c & 5+1/d \end{bmatrix}$$

If we forget to use the `evalm` command, Maple will not evaluate the variables as matrices.

```
> A+B;
```

$$A + B$$

The same holds for scalar multiplication and just about any other operation we perform with matrices in Maple.

```
> evalm(2*A);
```

$$\begin{bmatrix} 4 & 2 \\ -2 & 10 \end{bmatrix}$$

Note that if we attempt to perform an operation that is not allowed (due to incompatibility) then Maple lets us know with an error message.

```
> evalm(C+B);  
Error, (in linalg[matadd]) matrix dimensions incompatible
```

Multiplication of matrices, when it is defined, can be achieved using the `multiply` command. Be careful - *matrix multiplication is not commutative*, so the order of the matrices in this command is important!

```
> multiply(A,C);
```

$$\begin{bmatrix} 23 & -1 & 10 \\ 38 & 17 & 39 \end{bmatrix}$$

```
> multiply(C,A);  
Error, (in multiply) non matching dimensions for vector/matrix product
```

Another way to multiply two matrices is using the `&*` symbol, in conjunction with the `evalm` command, as usual. (Note that there is no space between the `&` and the `*`).

```
> evalm(A &* C);
```

$$\begin{bmatrix} 23 & -1 & 10 \\ 38 & 17 & 39 \end{bmatrix}$$

A number of other operations/commands on matrices are possible, such as the `transpose` command, which interchanges rows and columns within a matrix.

```
> transpose(B);
```

$$\begin{bmatrix} a & c \\ -b & 1/d \end{bmatrix}$$

Also, `det` finds the determinant of a square matrix.

```
> det(A);
```

```
> det(B);
```

$$\frac{a + bcd}{d}$$

The `inverse` command finds the inverse of a non-singular matrix (one whose determinant is not zero).

```
> inverse(A);
```

$$\begin{bmatrix} 5/11 & -1/11 \\ 1/11 & 2/11 \end{bmatrix}$$

Note that the products AA^{-1} and $A^{-1}A$ give the *identity* matrix.

```
> evalm(inverse(A) &* A);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
> evalm(A &* inverse(A));
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The inverse of a non-singular matrix can be obtained in a variety of other ways. For example,

```
> evalm(1/A);
```

$$\begin{bmatrix} 5/11 & -1/11 \\ 1/11 & 2/11 \end{bmatrix}$$

as well as

```
> evalm(A^(-1));
```

$$\begin{bmatrix} 5/11 & -1/11 \\ 1/11 & 2/11 \end{bmatrix}$$

We must be a little careful when verifying that $AA^{-1} = I$.

```
> evalm(A &* 1/A);
```

Error, (in evalm/amperstar) `&*` is reserved for matrix multiplication

The error message is due to the fact that since we used $1/A$ to compute the inverse, we should use

```
> multiply(A,1/A);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

See what happens when we type

```
> evalm(A &* (1/A));
```

```
&*()
```

or

```
> evalm(A &* A^(-1));
```

```
&*()
```

Maple gives an “ambiguous” response. This time, however, we did not get an error message but simply Maple’s representation of the identity matrix. (The sequence of symbols `&*()` represent the identity matrix in Maple.) To avoid this, we should use the `multiply` command, as was done above.

Vectors

We already mentioned that n -dimensional vectors can be “thought of” as $n \times 1$ arrays, hence we could use the `matrix` command that we learned in the previous section to define a vector. There is, however, another way to define vectors in Maple, through the command `vector` that only requires one dimension. For example, in order to define the vector with entries $1, \cos(t)$ and $\sin(t)$ we type

```
> v:=vector(3,[1,cos(t),sin(t)]);
      v := [1, cos(t), sin(t)]
```

(One of the differences between the `matrix` and `vector` commands in Maple is the way the results are being displayed; `matrix` produces a column vector while `vector` produces a row vector.)

To display a vector, regardless of how we define it, we must use the `evalm` command, as before.

```
> evalm(v);
```

```
[1, cos(t), sin(t)]
```

Addition, subtraction and scalar multiplication are performed the same way as with matrices and always using the `evalm` command.

Given two vectors $u = [u_1, u_2, \dots, u_n]$ and $v = [v_1, v_2, \dots, v_n]$, we define the *inner product* as

$$\langle u, v \rangle = \sum_{i=1}^n u_i v_i = u_1 v_1 + \dots + u_n v_n$$

which is a scalar. In Maple, we use the `innerprod` command. Define u as

```
> u:=vector(3, [-1, cos(t), sin(t)]);
      u := [-1, cos(t), sin(t)]
```

and ask Maple to calculate the inner product of u and v .

```
> innerprod(u, v);
      -1 + (cos(t))^2 + (sin(t))^2
```

Realizing that there is a simplification to be made, we ask Maple to

```
> simplify(%);
      0
```

getting that the inner product between u and v is 0. This means that the two vectors are *orthogonal* (i.e. the angle between them is 90°). Maple can tell us that as well.

```
> angle(u, v);
      arccos( $\frac{-1 + (\cos(t))^2 + (\sin(t))^2}{1 + (\cos(t))^2 + (\sin(t))^2}$ )
> simplify(%);
      1/2  $\pi$ 
```

One other possible quantity of interest is the *norm* of a vector (which gives its *length*), defined as

$$\|u\| = \sqrt{\sum_{i=1}^n u_i^2}.$$

In Maple this is obtained through the `norm` command.

```
> norm(u, 2);
       $\sqrt{1 + (|\cos(t)|)^2 + (|\sin(t)|)^2}$ 
```

Note that we use a second argument (the number 2) in this command to specify which norm we want (there are other norms that one could define for vectors), this being the 2-norm. To simplify the answer above we use

```
> simplify(% , assume=positive);
```

$$\sqrt{2}$$

where, even though we shouldn't have to, we "let Maple know" that the quantities within the expression are all positive.

As a final command in this section, we have `crossprod`, which calculates the cross product of two 3-dimensional vectors, defined as

$$u \times v = [u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1].$$

```
> crossprod(u,v);
```

$$[0, 2 \sin(t), -2 \cos(t)]$$

Solution of linear systems

In the introduction we mentioned the connection between linear systems and matrices. Let us now see how to actually solve a system of linear equations using matrix commands in Maple. Consider the system

$$\begin{aligned} 2x_1 + 2x_2 - x_3 &= 1 \\ 3x_1 + 4x_2 + 2x_3 &= -2 \\ x_1 + 4x_2 + 3x_3 &= 3 \end{aligned}$$

which can be expressed as $A\mathbf{x} = \mathbf{b}$, with

$$A = \begin{bmatrix} 2 & 2 & -1 \\ 3 & 4 & 2 \\ 1 & 4 & 3 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}.$$

In Maple, we define the matrix A and the vector \mathbf{b} as follows:

```
> A:=matrix([[2,2,-1],[3,4,2],[1,4,3]]);b:=vector(3,[1,-2,3]);
```

$$A := \begin{bmatrix} 2 & 2 & -1 \\ 3 & 4 & 2 \\ 1 & 4 & 3 \end{bmatrix}$$

$$b := [1, -2, 3]$$

There are several ways to solve such a linear system. Perhaps the most straight forward one is `linsolve`, which “automatically” gives the solution vector \mathbf{x} .

```
> x:=linsolve(A,b);
```

$$x := \left[-\frac{24}{7}, 3, -\frac{13}{7}\right]$$

Let us verify the above result by multiplying $A\mathbf{x}$ and seeing that we indeed get \mathbf{b} .

```
> multiply(A,x);
```

$$[1, -2, 3]$$

Keep in mind that \mathbf{x} is (still) an array, so to refer to its elements we must use brackets and to refer to \mathbf{x} as a whole we must use the `evalm` command.

```
> evalm(x);
```

$$\left[-\frac{24}{7}, 3, -\frac{13}{7}\right]$$

```
> x[3];
```

$$-\frac{13}{7}$$

Another way to solve the linear system is to use the fact that $\mathbf{x} = A^{-1}\mathbf{b}$.

```
> x:=multiply(inverse(A),b);
```

$$x := \left[-\frac{24}{7}, 3, -\frac{13}{7}\right]$$

This is not generally recommended in practice due to the possible presence of round-off error and the (computational) cost involved.

Yet another way to solve the linear system is *Gaussian elimination* and *back substitution*. In the elimination part of this procedure we form the augmented matrix $[A \mid \mathbf{b}]$ and “eliminate” (set to zero) all the entries below the main diagonal of A through elementary row operations. Once this is done, the resulting equivalent system is solved through *back substitution*.

In Maple this is done using the commands `augment`, `gausselim` and `backsub`.

```
> A_b:=augment(A,b);
```

$$A_b := \begin{bmatrix} 2 & 2 & -1 & 1 \\ 3 & 4 & 2 & -2 \\ 1 & 4 & 3 & 3 \end{bmatrix}$$

```
> A_b_elim:=gausselim(A_b);
```

$$A_b := \begin{bmatrix} 2 & 2 & -1 & 1 \\ 0 & 1 & 7/2 & -7/2 \\ 0 & 0 & -7 & 13 \end{bmatrix}$$

```
> x:=backsub(A_b_elim);
```

$$x := \left[-\frac{24}{7}, 3, -\frac{13}{7}\right]$$

The above procedure could also be illustrated (for educational purposes) step-by-step using commands such as `mulrow`, `addrow` and `swaprow`, but this beyond the scope of this tutorial. If you would like to find out more about this consult the help facilities of Maple.

One final way to solve a linear system in Maple is to continue the elimination of the augmented matrix to obtain the *row reduced echelon form* and simply *read* the solution. The command is

```
> rref(A_b);
```

$$\begin{bmatrix} 1 & 0 & 0 & -\frac{24}{7} \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -\frac{13}{7} \end{bmatrix}$$

and the last column above contains our solution.

Eigenvalues and eigenvectors

Another topic one encounters in a linear algebra course is the calculation of the eigenvalues and eigenvectors of a square matrix. These satisfy

$$A\mathbf{x} = \lambda\mathbf{x}$$

where A is an $n \times n$ matrix, λ is a constant (an eigenvalue) and \mathbf{x} is an $n \times 1$ vector (a corresponding eigenvector). The goal is to find the constants λ and the vectors \mathbf{x} associated with them, such that the above equation holds. Let us illustrate how to do this in Maple for the matrix A below, using the command `eigenvals`.

```
> A:=matrix([[2,-4,5],[0,2,-7/5],[1,1,-1]]);
```

$$A := \begin{bmatrix} 2 & -4 & 5 \\ 0 & 2 & -7/5 \\ 1 & 1 & -1 \end{bmatrix}$$

```
> lambda:=eigenvals(A);
      λ := 1, 1 + 1 + 1/5 √165, 1 - 1/5 √165
```

Two things to note here - first Maple recognized the Greek letter λ (lambda) as reflected in the output and second, the eigenvalues are calculated *exactly* as irrational numbers, rather than numerically. The eigenvalues can now be referred to individually using brackets, for e.g.

```
> lambda[2];
      1 + 1 + 1/5 √165
```

To obtain the eigenvectors associated with these eigenvalues we use the `eigenvecs` command.

```
> x:=eigenvecs(A);
      x : = [1, 1, { [1, 7/3, 5/3] }], [1 + 1/5 √165, { [-30/7 - 1/7 √165, 1, 5/7 - 1/7 √165] }],
           [1 - 1/5 √165, { [-30/7 + 1/7 √165, 1, 5/7 + 1/7 √165] }]
```

Before we explain Maple's response note that since we computed *eigenvectors*, Maple gave us the answer in vector (row) form. The first entry in the "vector" above corresponds to the first eigenvalue $\lambda = 1$, with (algebraic) multiplicity 1 and associated eigenvector $[1, 7/3, 5/3]$. We can see this by asking Maple to display the "elements" of this array.

The first eigenvalue

```
> x[1][1];
      1
```

its multiplicity

```
> x[1][2];
      1
```

and the associated eigenvector

```
> x[1][3];
      {[1, 7/3, 5/3]}
```

The second entry in the array x above corresponds to the second eigenvalue $\lambda = 1 + \frac{1}{5}\sqrt{165}$, with multiplicity 1 and associated eigenvector $[-\frac{30}{7} - \frac{1}{7}\sqrt{165}, 1, \frac{5}{7} - \frac{1}{7}\sqrt{165}]$. In Maple this information can be retrieved as follows.

The second eigenvalue

```
> x[2][1];
```

$$1 + 1/5 \sqrt{165}$$

its multiplicity

```
> x[2][2];
```

$$1$$

and the associated eigenvector

```
> x[2][3];
```

$$\left\{ \left[-\frac{30}{7} - 1/7 \sqrt{165}, 1, 5/7 - 1/7 \sqrt{165} \right] \right\}$$

The third eigenvalue and eigenvector can be retrieved in a similar way. Let us now verify that indeed $A\mathbf{x}_1 = \lambda_1\mathbf{x}_1$, with $\lambda_1 = 1$ and

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 7/3 \\ 5/3 \end{bmatrix}$$

as Maple found. Define

```
> x1:=vector(3,[1,7/3,5/3]);
```

$$x1 := [1, 7/3, 5/3]$$

and recall that

```
> lambda[1];
```

$$1$$

Then

```
> evalm(A &* x1 = lambda[1]*x1);
```

$$[1, 7/3, 5/3] = [1, 7/3, 5/3]$$

as expected.

Comments

In the beginning of this tutorial we mentioned another mathematical package called Matlab, which stands for MATrix LABoratory. There are some important differences between Maple and Matlab, which we mention here. In Matlab matrices can be defined in an “easier” way than in Maple and operations on them can be performed without special commands such as `evalm`. On the other hand, Matlab is a *numerical* software package, whereas Maple is a *symbolic* one. This not only means that in Maple one can define matrices (as well as other mathematical objects) with non-numeric entries, but the calculations are done exactly! In Matlab, matrices (and in general all objects) must have an assigned numerical value to them and most importantly, the calculations are susceptible to round-off error. Each program, however, has its place in the world of computational mathematics.

Review

You have now seen many of the basic linear algebra features and commands of Maple. You should now be able to enter and work with matrices, vectors and solve linear systems of equations. Additional commands and information on the linear algebra package found in Maple can be obtained by typing `?linalg` at the prompt and choosing which command you wish to explore. The most important commands introduced in this tutorial are summarized in the table on the next page.

Maple Linear Algebra Commands

Command	Description
<code>with(linalg):</code>	load linear algebra package
<code>A:=matrix(m,n,[lv]);</code>	defines an m by n matrix A where lv is a list of its elements
<code>evalm(A);</code>	displays the array (matrix or vector) A
<code>evalm(A+B)</code>	adds matrices A and B
<code>evalm(A*B)</code>	multiplies matrices A and B
<code>multiply(A,B)</code>	multiplies matrices A and B
<code>inverse(A)</code>	calculate the inverse of a matrix A
<code>evalm(A^(-1))</code>	calculate the inverse of a matrix A
<code>vector(n,[lv])</code>	defines an n dimensional vector where lv is a list of its elements
<code>innerprod(u,v)</code>	calculates the inner product of vectors u and v
<code>angle(u,v)</code>	calculates the angle between the vectors u and v
<code>norm(u,n)</code>	calculates the n -norm of the vector u
<code>linsolve(A,b)</code>	solves the linear system $A\mathbf{x} = \mathbf{b}$
<code>augment(A,b)</code>	forms the augmented matrix $[A \mid \mathbf{b}]$
<code>gausselim(Ab)</code>	performs Gaussian elimination on the augmented matrix Ab
<code>rref(Ab)</code>	calculates the row reduced echelon form of the augmented matrix Ab
<code>eigenvals(A)</code>	calculates the eigenvalues of the matrix A
<code>eigenvecs(A)</code>	calculates the eigenvectors of the matrix A

Remember:

- Always use `evalm` when working with matrices or vectors.
- Always end each command with a semicolon;
- Use capital letters only where needed—Maple is Case Sensitive.