9. LINEAR ALGEBRA

MAPLE can do symbolic and floating point matrix and linear algebra computations. There are two packages: the *linalg* package and the new *LinearAlgebra* package. The new *LinearAlgebra* package is more user-friendly for matrix algebra computations. It is also more efficient for numeric computations, especially with large matrices. The *linalg* package is recommended for more abstract computations. We will concentrate mainly on the *LinearAlgebra* package. Try

```
> ?LinearAlgebra
```

for an introduction to the *LinearAlgebra* package and a list of functions.

9.1 Vectors, Arrays, and Matrices

`Matrix`, `Array`, and `Vector` are the main data types used in the *LinearAlgebra* package. Note that the "M", "A" and "V" are capitalized. The lower-case `matrix`, `array`, and `vector` are used in the `linalg` package. `Matrix` and `Vector` are examples of what MAPLE calls an `rtable`. See `?rtable` for more information.

```
> Matrix(3);
```

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

```
> Matrix(3,4);
```

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

```
> Matrix(2,3,[[a,b,c],[d,e,f]]);
```

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

```
> Matrix(2,3,[[a,b],[d,e,f]]);
```

$$\begin{bmatrix} a & b & 0 \\ d & e & f \end{bmatrix}$$

```
> Matrix(2,3,[[a,b],[c,d,e,f]]);
Error, (in Matrix) initializer defines more columns
(4) than column dimension parameter specifies (3)
> Matrix(2,3,[a,b,c,d,e,f]);
```

```
Error, (in Matrix) initializer defines more columns
(6) than column dimension parameter specifies (3)
```

The call Matrix(m,n) returns an $m \times n$ matrix of zeros. Observe matrix entries are assigned by a list of rows.

```
> W:=Vector(4);
```

$$W := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> V:=Vector([x,y,z]);
```

$$V := \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The call Vector(m) returns an $m \times 1$ column vector of zeros. Observe that vector entries can be assigned using a list.

A fun way to create matrices is to use a function $f(x, y)$ of two variables. The function Matrix(m,n,f) produces the $m \times n$ matrix whose (i, j)th entry is $f(i, j)$.

```
> f := (i,j) -> x^(i*j);
```

$$F := (i, j) \mapsto x^{ij}$$

```
> A := Matrix(2,2,f);
```

$$A := \begin{bmatrix} x & x^2 \\ x^2 & x^4 \end{bmatrix}$$

Now try

```
> A := Matrix(4,4,f);
> factor(LinearAlgebra[Determinant](A));
```

The map function also works on matrices. Let's form a 5×5 matrix of the integers from 1 to 25.

```
> M:=Matrix(5,(i,j)->5*i+j-5);
```

$$M := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

Now let's use `map` and `ithprime` to form a table of the first 25 primes:

```
> map(ithprime,M);
```

$$\begin{bmatrix} 2 & 3 & 5 & 7 & 11 \\ 13 & 17 & 19 & 23 & 29 \\ 31 & 37 & 41 & 43 & 47 \\ 53 & 59 & 61 & 67 & 71 \\ 73 & 79 & 83 & 89 & 97 \end{bmatrix}$$

Of course, we could have done this without using `map`. Try

```
> Matrix(5,(i,j)->ithprime(5*i+j-5));
```

Try making a table of the first 100 primes:

```
> Matrix(10,(i,j)->ithprime(10*i+j-10));
```

9.1.1 Matrix and Vector entry assignment

It is easy to access entries in a matrix and reassign them.

```
> A:=Matrix(2,3,[[1,2,3],[5,10,16]]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 10 & 16 \end{bmatrix}$$

```
> A[2,3];
```

$$16$$

The entry in the second row and third column is 16. Let's change it to 15.

```
> A[2,3]:=15;
```

$$15$$

```
> A;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 10 & 15 \end{bmatrix}$$

In general, `A[i,j]` refers to the ijth entry of the matrix A (i.e., the entry in the ith row and jth column). It is also possible to access a block of entries.

```
> A := Matrix(4,(i,j)->(i+j));
```

$$A := \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

```
> A[2..3,2..4];
```

$$\begin{bmatrix} 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

```
> B := Matrix(2,3,[[0,1,2],[3,4,5]]);
```

$$B := \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

```
> A[2..3,2..4]:=B;
```

$$A_{2..3,2..4} := \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

```
> A;
```

$$\begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 0 & 1 & 2 \\ 4 & 3 & 4 & 5 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

In general, `A[a..b,c..d]` refers to the submatrix of A from rows a to b, and columns c to d. It is also possible to rearrange rows or columns.

```
> B:=Matrix(3,(i,j)->b[i,j]);
```

$$\begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

```
> B[[3,2,2,1],1..3];
```

$$\begin{bmatrix} b_{3,1} & b_{3,2} & b_{3,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{1,1} & b_{1,2} & b_{1,3} \end{bmatrix}$$

Observe how we created a generic matrix B. The call `B[[3,2,2,1],1..3]` created a new matrix whose rows are rows 3, 2, 2, and 1 of matrix B. Observe how the second row was repeated. In general, we use the syntax `B[L1,L2]`, where $L1$, $L2$ are either lists or of the form $a..b$. Try

```
> A := Matrix(3,4,[[1,2,3,4],[2,4,6,8],[3,6,9,12]]);
> A[[3,2],[4,3,2]];
```

```
> V := Vector([a,b,c,d]);
> W := V[[3,2]];
```

9.1.2 The Matrix and Vector palettes

The **Matrix** palette contains buttons for entering matrices up to a 4×4. To show the **Matrix** palette: in the menu bar click on View, select `Palettes`, slide to `Matrix Palette` and release. The **Matrix** palette should appear in a separate window. See Figure 9.1 below.

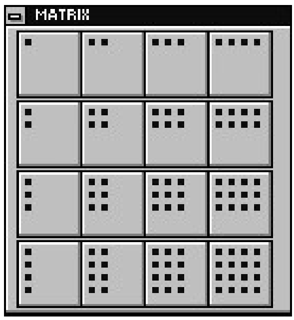
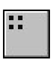


Figure 9.1 The **Matrix** palette.

Let's enter a 2×2 matrix. Click a place in the worksheet where you want to enter the matrix:

```
> |
```

Now click on . A matrix template should appear in the worksheet:

```
> Matrix([[%?, %?], [%?, %?]]);
```

Type 23:

```
> Matrix([[23, %?], [%?, %?]]);
```

To get to the next entry location, press `Tab`.

```
> Matrix([[23, %?], [%?, %?]]);
```

Type `int(1/x,x=1..2)` and press `Tab`:

```
> Matrix([[23, int(1/x,x=1..2), [%?, %?]]);
```

Type 25 and press `Tab`:

```
> Matrix([[23, int(1/x,x=1..2), [25, %?]]);
```

Finally, type 27 and press `Enter`:

$$\begin{bmatrix} 23 & \ln(2) \\ 25 & 27 \end{bmatrix}$$

The **Vector** palette works in a similar way. In the menu bar, click on View, select `Palettes`, slide to `Vector Palette`, and release. The **Vector** palette should appear in a separate window. See Figure 9.2 below.

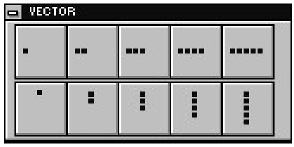
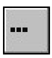


Figure 9.2 The Vector palette.

Let's enter a 3×1 row vector. Click a place in the worksheet where you want to enter the vector:

```
> |
```

Now click on . A vector template should appear in the worksheet:

```
> <%? | %? | %?>;
```

Type `11`:

```
> <11 | %? | %?>;
```

Press `Tab` and type `12`:

```
> <11 | 12 | %?>;
```

Press `Tab`, type `13` and press `Enter`:

```
> <11 | 12 | 13>;
```

$$[11, 12, 13]$$

9.1.3 Matrix operations

MAPLE can do the usual matrix operations of addition, multiplication, scalar multiplication, inverse, transpose, and trace.

Matrix Operation	Mathematical Notation	MAPLE Notation
Addition	$A + B$	`A + B`
Subtraction	$A - B$	`A - B`
Scalar multiplication	$c\,A$	`c*A`
Matrix multiplication	$A\,B$	`A . B` or `Multiply(A,B)`
Matrix power	A^n	`A^n`
Inverse	A^{-1}	`A^(-1)` or `1/A` or `MatrixInverse(A)`
Transpose	A^T	`Transpose(A)`
Trace	$\mathrm{tr}\,A$	`Trace(A)`

We illustrate matrix addition, subtraction and scalar multiplication.

```
> A := Matrix(2,[[1,2],[3,4]]);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> B := Matrix(2,[[-2,3],[-5,1]]);
```

$$\begin{bmatrix} -2 & 3 \\ -5 & 1 \end{bmatrix}$$

```
> A + B;
```

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

```
> A - B;
```

$$\begin{bmatrix} 3 & -1 \\ 8 & 3 \end{bmatrix}$$

```
> 5*A;
```

$$\begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

We continue with matrix multiplication, matrix power, and finding an inverse.

```
> A := Matrix(2,[[1,2],[3,4]]):
> B := Matrix(2,[[-2,3],[-5,1]]):
> A . B;
```

$$\begin{bmatrix} -12 & 5 \\ -26 & 13 \end{bmatrix}$$

```
> AI := 1/A;
```

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

```
> A . AI;
```

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

```
> A^3;
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

The functions `Multiply`, `MatrixInverse`, `Transpose`, and `Trace` are part of the *LinearAlgebra* package. Try

```
> with(LinearAlgebra);
```
to see a list of functions in the *LinearAlgebra* package.

```
> with(LinearAlgebra):
> A := Matrix(2,[[1,2],[3,4]]):
> B := Matrix(2,[[-2,3],[-5,1]]):
> Multiply(A , B);
```

$$\begin{bmatrix} -12 & 5 \\ -26 & 13 \end{bmatrix}$$

```
> Multiply(Multiply(A,A),A);
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

```
> AI := MatrixInverse(A);
```

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

```
> Transpose(A);
```

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

```
> Trace(A);
```

$$5$$

Now try the following:

```
> with(LinearAlgebra):
> A:=Matrix(2,3,[[1,2,3],[4,5,6]]);
> B:=Matrix(3,2,[[2,4],[-7,3],[5,1]]);
> C:=Matrix(2,2,[[1,-2],[-3,4]]);
> A . B;
> Multiply(A,B);
> A.B-2*C;
```

Now check your results with pencil and paper. You should have found that

$$AB - 2C = \begin{bmatrix} 1 & 17 \\ 9 & 29 \end{bmatrix}$$

9.1.4 Matrix and vector construction shortcuts

Angled brackets `< >` are used as a shortcut to construct matrices and vectors. We can construct a column vector:

```
> V := <1,2,3>;
```

$$V := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

The construction `<a, b, c, ... >` gives a column vector when a, b, c, ... are scalars. We can construct a row vector:

```
> R := <1|2|3>;
```

$$R := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

We can construct a matrix from column vectors:

```
> U := <a,b,c>;
```

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

```
> V := <i,j,k>;
```

$$\begin{bmatrix} i \\ j \\ k \end{bmatrix}$$

```
> W := <x,y,z>;
```

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

```
> M := <U | V | W>;
```

$$\begin{bmatrix} a & i & x \\ b & j & y \\ c & k & z \end{bmatrix}$$

Similarly, we can build a matrix from row vectors. Try the following:

```
> U := <a|b|c>;
> V := <i|j|k>;
> W := <x|y|z>;
> M := <U , V , W>;
```

Angled brackets can also be used to stack matrices.

```
> A:=Matrix(3,(i,j)->a^i*b^j):
> B:=Matrix(3,(i,j)->b^i*c^j):
> C:=Matrix(3,(i,j)->c^i*a^j):
> A,B,C;
```

$$\begin{bmatrix} ab & ab^2 & ab^3 \\ a^2b & a^2b^2 & a^2b^3 \\ a^3b & a^3b^2 & a^3b^3 \end{bmatrix}, \begin{bmatrix} bc & bc^2 & bc^3 \\ b^2c & b^2c^2 & b^2c^3 \\ b^3c & b^3c^2 & b^3c^3 \end{bmatrix}, \begin{bmatrix} ca & ca^2 & ca^3 \\ c^2a & c^2a^2 & c^2a^3 \\ c^3a & c^3a^2 & c^3a^3 \end{bmatrix}$$

Now we form a new matrix by stacking the matrices A, B, C, to the right of each other:

```
> <A|B|C>;
```

$$\begin{bmatrix} ab & ab^2 & ab^3 & bc & bc^2 & bc^3 & ca & ca^2 & ca^3 \\ a^2b & a^2b^2 & a^2b^3 & b^2c & b^2c^2 & b^2c^3 & c^2a & c^2a^2 & c^2a^3 \\ a^3b & a^3b^2 & a^3b^3 & b^3c & b^3c^2 & b^3c^3 & c^3a & c^3a^2 & c^3a^3 \end{bmatrix}$$

Similarly we can stack A above B:

```
> <A,B>;
```

$$\begin{bmatrix} ab & ab^2 & ab^3 \\ a^2b & a^2b^2 & a^2b^3 \\ a^3b & a^3b^2 & a^3b^3 \\ bc & bc^2 & bc^3 \\ b^2c & b^2c^2 & b^2c^3 \\ b^3c & b^3c^2 & b^3c^3 \end{bmatrix}$$

Now try stacking A, B, and C above each other:

```
> <A,B,C>;
```

9.1.5 Viewing large Matrices and Vectors

Only relatively small matrices and vectors will be displayed on the screen. For instance, a 50×20 matrix of the first 1000 primes is much too big to be displayed on the screen.

```
> M:=Matrix(50,20,(i,j)->ithprime(20*i+j-20));
```

$$M := \begin{bmatrix} 50 \times 20 \texttt{ Matrix} \\ \texttt{Data Type: anything} \\ \texttt{Storage: rectangular} \\ \texttt{Order: Fortran\_order} \end{bmatrix}$$

Observe that this 50×20 matrix was not displayed on the screen. In its place is a matrix giving the dimensions and some information on `Data Type`, `Storage`, and `Order`. To view entries in this matrix, we can use the context menu, which we will discuss in more detail in the next section. First click the right button of