will give you a summation expression for the Riemann sum obtained by dividing the interval from `a` to `b` into `n` equal subintervals and erecting on each subinterval a rectangle whose height is the value of *expression* at the righthand end of the subinterval. If you use `rightbox` instead of `rightsum`, you get the diagram for the same Riemann sum.

There are corresponding commands `leftsum` and `leftbox` which use values at the lefthand ends of the subintervals and `middlesum` and `middlebox` which use values in the middle of the subintervals.

The command

`simpson(`*expression*`,x=a..b,n);`

gives the summation expression which would arise if you used Simpson's Rule to get an approximate value for the integral of *expression* from `a` to `b`.

The expressions given by the comands `rightsum`, `leftsum`, `middlesum` and `simpson` are not numerical values — they are in the form of a sum such as

$$\frac{1}{10}\left(\sum_{i=1}^{10}\left(\frac{1}{100}i^2\right)\right).$$

To see the actual value of the summation, apply `value`. This will, in general, give an algebraic expression and you may need to use `evalf` in order to get a meaningful answer. If the value of `n` is large then `value` will produce an enormous amount of messy output, so it is best to combine all these procedures into one command by entering

```
evalf( value(rightsum(f(x),x=a..b,n)) );
```

(or alternatively, use a colon at the end of the `value` command line).

2.10 Vectors and Matrices

2.10.1 Vectors.

Maple uses a capital V for vectors when you use the LinearAlgebra package. (Be careful not to use vector with a small 'v'. You will not get an error as this is used by an older package called `linalg`, but things may not work as you expect.)

You create Vectors by using the `Vector` procedure, or more easily by angle brackets (the $<$ and $>$ signs):

`<`*sequence*`>;`

where *sequence* is a sequence of expressions (which *can* be numbers). For example, the command

```
v := < 1,-5,4 >;
```

assigns to the variable `v` a value which is a 3-dimensional Vector with components 1, -5 and 4 (in that order). Alternatively, one could use

```
v := Vector( [ 1,-5,4 ] );
```

You can also use fuctions to define Vectors. For example, to create a Vector in $\mathbb{R}^3$ whose entries are 1,4 and 9 respectively, use

```
Vector(3, i -> i^2);
```

2.10.2 Matrices

Maple uses a capital M for a matrix in the `LinearAlgebra` package. (Be careful not to use matrix with a small 'm'. You will not get an error as this is used by an older package called `linalg`, but things may not work as you expect.)

You create a Matrix using pairs of angle brackets (or the `Matrix` procedure). You enclose the columns of the Matrix as a collection of `Vectors`, separated by vertical lines `|` and linked together with an outer set of angle brackets. For example, you can assign the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 5 \\ 3 & -2 & 4 \end{pmatrix}$$

to the variable `A` with the command

```
A := < <1,3,3> | <2,-1,-2> | < 3,5,4> >;
```

Note the outer set of angle brackets `<  >` which make the object into a `Matrix`. These outer brackets must be there even if the matrix only has one column, as in

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

Note that Maple does NOT regard a Vector as the same thing as a Matrix with one column, but the differences will probably not be relevant to you in first year.

An alternative way to enter matrices is by rows rather than columns: all you need to do is "swap the commas and bars around". So the prevous matrix can be defined by

```
A := < <1|2|3> , < 3|-1|5> , <3|-2|-4> >;
```

However, it is better to think of matrices in columns rather than rows so the 'collection of columns' method is preferable. It is also easier to type commas rather than the vertical bars, of course, and there are fewer bars in the collection of columns method.

You can also create a Matrix using a function that tells Maple how to calculate each entry. For a Matrix you need a function of two variables (and to give two dimensions), for example,

```
Matrix(3,2, (i,j) -> i^2+j^3);
```

It is conventional in mathematics to use a single capital letter to denote a matrix, but `Matrix` names in Maple do not have to follow this convention — they could be a lower case letter or any valid Maple name. But remember that `A` and `a` are not the same name. Also be warned that some letters (such as `D`) are reserved.

Note that the default behaviour of Maple, for both Vectors and Matrices, is to display a placeholder when any dimension of a Vector or Matrix is larger than 10. For example, try

```
Vector(11, i -> i^3);
```

To see how to alter this behaviour look up the Maple help page for `interface`.

2.10.3 Selecting Components of Vectors and Matrices

If the value of `A` is a Matrix, you can use the notation `A[i,j]` to refer to the entry in the `i`th row and `j`th column of `A`. For example, if the value of `A` is

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 5 \\ 3 & -2 & 4 \end{pmatrix}$$

then `A[3,2]` has the value `-2`.

Similarly, if the value of a variable `v` is a Vector, you can use `v[n]` to refer to its `n`th entry. For example, if (the list) `v` has been assigned a value by

```
v := [1,2,-4,9];
```

then `v[3]` will have the value -4 because the third entry in the list is -4. Note that the same notation also works for lists.

You can use these notations in commands to *change* the value of entries. For example, the commands

```
v[2] := 12:     A[2,1] := 13:
```

will assign the value 12 to the second entry of the list `v` and the value 13 to the entry in the second row and first column of the Matrix `A`.

You can do the same thing with a set, but this is dangerous because the order of entries in a set is not fixed, so you cannot be sure which entry Maple will regard as being the `n`th entry.

2.10.4 Manipulating Vectors and Matrices

The last section shows how to create and display Vectors and Matrices. However, we cannot do any useful linear algebra without first loading the `LinearAlgebra` package, which provides many procedures for dealing with Vectors, Matrices and linear equations. In these Notes we will only give an outline of some of the basic procedures. To find out more about these and other procedures available in `LinearAlgebra`, use Maple's Help.

To use the procedures in this package, it is easiest to first load it with the command

```
with(LinearAlgebra):
```

Note the use of the *colon* here, which supresses the (very long) list of new funcitons in `LinearAlgebra`. You can use a command from this package directly by giving its **full name**: you do this by adding `LinearAlgebra:-` to the start of the command. This is what Maple will do if you operate with the context sensitive menus (see section 1.3.4).

The package `LinearAlgebra` also allows you to enter a *diagonal* matrix (i.e. a matrix A with $a_{ij} = 0$ for $i \neq j$) by using `DiagonalMatrix`. You can use either a list or a Vector as the argument of `DiagonalMatrix`. For example,

```
A := DiagonalMatrix([1,2]);
```

$$A := \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

There is the useful command `IdentityMatrix` for creating (of course) identity matrices. For example, the 6×6 identity matrix is

```
id6 := IdentityMatrix(6);
```

Also, new Matrices can be constructed by combining Vectors and/or Matrices. If we have Vectors, say `v1`, `v2` and `v3`, then we can form a Matrix which has these vectors as *columns* by using the angle brackets and vertical bars

```
< v1 | v2 | v3 >;
```

Similarly, if we have matrices `A`, `B` and `C` of appropriate sizes we can combine them *side by side* (called **augmenting**) into a large matrix by using

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

or *one on top of the other* (called **stacking**) by using

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

Conversely, you can extract vectors from matrices by using `Column(A,i)` to create a *Vector* whose components are the entries in the `i`th column of the Matrix `A`. For example,

```
v := Column(A,2);
```

We also have `Row` for rows — but this creates a **row Vector**, which is not something we'll be dealing with.

If you use a range (for example `2..3`) you can extract more than one column (as a sequence).

Note that the things created by `Column` (and `Row`) are *Vectors* (or sequences of Vectors) and not Matrices

Another procedure which selects parts of matrices is `SubMatrix`. The command

```
SubMatrix( A, a..b, c..d );
```

(where `a..b` and `c..d` are *ranges* of positive integers) produces a smaller Matrix whose entries are the values of `A[i,j]` for $\texttt{a} \le \texttt{i} \le \texttt{b}$ and $\texttt{c} \le \texttt{j} \le \texttt{d}$, arranged in the same relative positions as they had in `A`.

You can also use `SubMatrix` in the form

```
SubMatrix( A, [1,4..6], [2..4] );
```

to get the submatrix whose entries are the values of `A[i,j]` for `i` in the list `[1,4,5,6]` and `j` in the list `[2,3,4]`.

Maple also let's you define Vectors and Matrices using an *indexing function*. If f is a function that takes one argument, `Vector(5, f)` gives a vector with the 5 entries, $f(1)$, $f(2)$, $f(3)$, $f(4)$, $f(5)$. Similarly, if g is a function that takes two arguments, `Matrix(2, 3, g)` produces a Matrix with 2 rows and 3 columns in which the element in the i^{th} row and j^{th} column is given by $g(i,j)$. See section 2.21 for examples.

2.11 Gaussian Elimination.

The `LinearAlgebra` package provides the procedure `RowOperation` which allows you to solve systems of simultaneous linear equations by going step by step through the steps of Gaussian elimination. The one procedure will do different things, depending on what arguments you give it. So:

`< A | b>`	augment Matrix `A` by Vector `b` to give `(A|b)`
`RowOperation(A,[i,j])`	swap rows `i` and `j` in `A`
`RowOperation(A,i,m)`	multiply row `i` of `A` by the value of `m`
`RowOperation(A,[i,j],m)`	replace row `i` with row `i` + (row `j`)`*m` in `A` i.e. add `m` times row `j` to row `i` (NOTE THE ORDER CAREFULLY)

Note that the Matrices produced by these procedures are NOT assigned as new values for the original variable `A` (unless you specifically instruct Maple to do that by adding the option `inplace=true`, see the help page). Usually you will assign the resulting matrix to a new variable (as in the following example) OR just allow Maple to display the result and use `%` to refer to it in your next command.

Example

If you want to apply Gaussian elimination to the augmented matrix

$$(\mathrm{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} 0 & 1 & 2 & 1 & 1 \\ 2 & -1 & -2 & 1 & -1 \\ 1 & 2 & 4 & 0 & 5 \end{array}\right)$$

then you could proceed as follows. (Note that some of the command lines end in a colon. This is to save space by suppressing display of the result.)

```
> with(LinearAlgebra):
> A := < <0,2,1> | <1,-1,2> |< 2,-2,4> | <1,1,0> >:
> b := < 1,-1,5 >:
> m1 := < A | b >;
```

$$m1 := \begin{bmatrix} 0 & 1 & 2 & 1 & 1 \\ 2 & -1 & -2 & 1 & -1 \\ 1 & 2 & 4 & 0 & 5 \end{bmatrix}$$

```
> m2 := RowOperation( m1,[1,2] );
```

$$m2 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 0 & 5 \end{bmatrix}$$

```
> m3 := RowOperation( m2,[3,1],-1/2 );
```

$$m3 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 5/2 & 5 & -1/2 & 11/2 \end{bmatrix}$$

```
> m4 := RowOperation( m3,[3,2],-5/2 );
```

$$m4 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

This matrix is now in echelon form. We could use the procedure `BackwardSubstitute` at this point but instead we will carry out the backsubstitution on the matrix itself.

```
> m5 := RowOperation( m4,3,-1/3 );
```

$$m5 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> m6 := RowOperation( m5,[2,3],-1 ):
> m7 := RowOperation( m6,[1,3],-1 );
```

$$m7 := \begin{bmatrix} 2 & -1 & -2 & 0 & 0 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> m8 := RowOperation( m7,[1,2],1 );
```

$$m8 := \begin{bmatrix} 2 & 0 & 0 & 0 & 2 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> reduced := RowOperation( m8,1,1/2 );
```

$$\text{reduced} := \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

We now have the fully reduced form and we can read off the general solution as

$$[1, 2 - 2\lambda, \lambda, -1]$$

where λ is an arbitrary parameter representing an arbitrary choice of the third variable.

When Maple is solving systems of linear equations, it uses names like $\_t_1$, $\_t_2$ for arbitrary parameters which may occur in the general solution. So the solution of the above set of equations would be displayed as

$$[1, 2 - 2\_t_1, \_t_1, -1]$$

if you were using Maple, or as `[1, 2-2*_t[1], _t[1], -1]` in text files or when you are using plain Maple without a Maple window.

In the above example, intermediate matrices were assigned to named variables so that we could go back to an earlier stage if we found that we had made a mistake. Alternatively you could simply use `%` to apply each command to the previous result. The above example would then start with something like

```
<A | b>;
RowOperation( %,1,2 );
RowOperation( %,[3,1],-1/2 );
RowOperation( %,[3,2],-5/2 );
```

The `LinearAlgebra` package also provides procedures which do all or parts of the Gaussian elimination process in one step. In particular, you can use:

`GaussianElimination(A)`	gives the row-echelon form of `A`
`ReducedRowEchelonForm(A)`	gives the fully-reduced form of `A`
`BackwardSubstitute(W)`	does backsubstitution on a matrix `W` in echelon form, assuming that `W` has been been derived from an augmented matrix of the form `(A|b)`
`LinearSolve(A,b)`	solves `A.x=b`
`Basis( {v1,v2,...,vn} )`	finds a basis for the vector space spanned by the set $\{v1, v2, \ldots, vn\}$ of Vectors.

NOTES:

1. Maple does all its reductions using rational numbers, unless the original matrix had some floating point numbers in it. If you want to see the results in floating point, apply `evalf` in the way which was described in section 2.5.7. If the matrix involved is large and has floating point entries, the procedures `GaussianElimination`, `ReducedRowEchelonForm` and `LinearSolve` *will not always give exactly correct answers* because of problems with rounding errors when doing floating point arithmetic.

2. Maple can solve linear sytems with unknown parameters in them. In particular, it can solve systems in which each of the entries in the righthand side is an *unassigned* variable, for example

```
b := < b1 , b2 , b3 >;
```

3. If you are doing MATH1231 or MATH1241, you should look up the Maple help files for the procedures `Rank`, `RowSpace`, `ColumnSpace`, `Transpose`, `Determinant`, `Eigenvalues`, `Eigenvectors`, `CharacteristicPolynomial`. To anyone who has studied the linear algebra section of MATH1231/1241, the effects of these procedures should be obvious from their names.

2.12 Vector and Matrix Arithmetic

The last section shows how to create and display Vectors and Matrices. However, we cannot do any useful linear algebra without first loading the `LinearAlgebra` package, which provides many procedures for dealing with Vectors, Matrices and linear equations. (Technically, `LinearAlgebra` is called a **module**, but you can ignore this distinction.) In these Notes we will only give an outline of some of the basic procedures. To find out more about these and other procedures available in `LinearAlgebra`, use Maple's Help.

To use the procedures in this package, it is easiest to first load it with the command

```
with(LinearAlgebra):
```

Note the use of the *colon* here, which supresses the (very long) list of new funcitons in `LinearAlgebra`.

However, you can use a command directly by giving its **full name**: you do this by adding `LinearAlgebra:-` to the start of the command. This is what Maple will do if you operate with the context sensitive menus (see section 1.3.4).

Vectors and Matrices can be added, subtracted and multipled by scalars using the usual operators `+`, `-` and `*`. Two Matrices can be multipled using the `.` operator. The `.` operator is also used to give the dot product of two Vectors. A square Matrix can be raised to an integer power using `^`.

For example, to find the linear combination $2\mathbf{v} - 3\mathbf{w}$ of the vectors $\mathbf{v}$ and $\mathbf{w}$ we use

```
2*v - 3*w;
```

and we can enter the formula for a general point on the line through the points with position vectors $\mathbf{v}$ and $\mathbf{w}$ (i.e. the formula $\mathbf{x} = (1-\lambda)\mathbf{v} + \lambda\mathbf{w}$, where λ is an arbitrary parameter). We do this by

```
x := (1-lambda)*v + lambda*w;
```

Note that `A^(-1)` gives the inverse matrix.

When using these operations, you must of course ensure that the Matrices and Vectors are of the appropriate dimensions and, in the case of the negative power of a matrix, that the matrix is invertible.

If you enter an expression involving these operations, Maple will automatically carry out the operations. For example, to enter the mathematical expression $A\mathbf{x} + \mathbf{b}$, we use

```
A.x + b ;
```

If values have already been assigned to `A`, `x` and `b` then `A.x+b` will be evaluated, but if some of these variables are unassigned then you will get an answer involving the unassigned variables.

A useful convention in Maple is that, in expressions involving matrices, a scalar constant can be treated as that scalar multiple of an appropriate identity matrix I. For example, if `A` is a square matrix then the command

```
1 + A + 2*A^2 ;
```

evaluates the polynomial expression $I + A + 2A^2$, where I is the appropriate identity matrix. This convention enables you to substitute matrices into polynomials. For example,

```
subs( x=A, 1+x+2*x^2 );
```

gives the expression $I + A + 2A^2$ (because the 1 is interpreted as an identity matrix).

If you use the dot with two Vectors of the same size, Maple will calculate their dot (inner) product. So

```
v := <2,1,2> : w := <-4,2,1>:
v.w;
```

will return -4, the dot product of the two vectors.

The transpose of a Matrix or Vector is given by the `Transpose` function from the `LinearAlgebra` package. For example, if `A` is a 2 by 3 Matrix, then `Transpose(A)` is the 3 by 2 Matrix which is the transpose of `A`. If `v` is a Column vector, then `Transpose(v)` is the corresponding row Vector.

2.13 Vector Geometry

Maple provides facilities for doing geometry in $\mathbb{R}^n$ and, in particular, in $\mathbb{R}^3$.

2.13.1 Dot and Cross Products, Length

The `LinearAlgebra` package provides the procedures

`DotProduct(v,w)` (or `v.w`)	dot product of **v** and **w**
`CrossProduct(v,w)`	cross product of **v** and **w**

To calculate the length of a vector, you need to use the procedure `Norm`, in the `LinearAlgebra` package, but it must be used with care. If `v` is a Vector, then the command `Norm(v,2)` will give the length of the Vector. (The reason for the `2` is that `Norm` can be used for rather more general purposes which you will not need to know about in first year mathematics.)

There is also a `distance` procedure in the `student` package, which gives the distance between two points (represented by lists or Vectors). Use Maple's Help for more information.

2.13.2 Three-dimensional Geometry.

The package `geom3d` contains many useful procedures for solving problems in $\mathbb{R}^3$. In fact, many of the geometric problems involving planes from the MATH1131 Algebra Notes can by solved with procedures in `geom3d`. The following is an outline of some of the things which you can do with `geom3d`.

The method of assigning a name to a point, line or plane is not quite what you might expect. To say that A is the point $(1, 2, 3)$, you do NOT enter `A := [1,2,3];` — this is assigning a point to a name. You enter `point(A,[1,2,3]);` — you are assigning a name to a point. The command `line` is used to assign a name to a line. The line may be specified by giving two points on it or a point on it and a direction parallel to it. The command `plane` is used to assign a name to a plane. The plane may be specified

by giving a cartesian equation for it or three (non-collinear) points on it or a point on it and a normal direction or a point on it and two lines parallel to it. The command `sphere` is used to assign a name to a sphere. A sphere may be specified by giving its cartesian equation or four points on it or the end-points of a diameter or its center and its radius. To display the specifications of one of these things you need to use `detail` (see the example below). For a plane, the detail includes a cartesian equation for the plane. If you want to find a normal to a plane `p` use

```
NormalVector(p);
```

Be warned that if you specify a plane or sphere by means of an equation then Maple will want you to specify the names of the variables which are associated with the three axes. You can do this by listing them as a third argument to the `plane` or `sphere` command, as in

```
plane(P,x+y+z=1,[x,y,z]);
```

If you leave out the `[x,y,z]` then Maple will, rather strangely, prompt you with the request `enter the name of the x-axis`, to which you reply `x;`, and similarly for the other two axes. The semi-colons are *compulsory* here. Depending on the way Maple is set up, this might be done using a pop-up dialogue box.

When you have set up objects of these types you can, for example, use the command `distance` to find the distance between two of them or the command `intersection` to find the intersection of two of them or the command `FindAngle` to find the angle between two of them (where appropriate).

You can also use `Equation` to find a cartesian equation for a line, plane or sphere and `center` for the center of a sphere and `coordinates` to find the coordinates of a point.

Use the Maple `Help` to find out more about any of these commands and to find out about the many other commands available in `geom3d`.

In the following example, we first label the points $A(0,1,2)$ and $B(2,3,1)$ and the line AB through A and B. Applying `detail` to AB shows that the direction of the line is $(2,2,-1)$ and the line can be expressed in parametric vector form as

$$\mathbf{x} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}, \quad t \in \mathbb{R}.$$

Then we assign the label P to the plane through $C(4,5,6)$ with normal $(1,1,1)$ and use `detail` to find that P can be described by the cartesian equation

$$x + y + z = 15.$$

Then we assign the label X to the point of intersection of the line AB and the plane P and find that the coordinates of X are $(8,9,-2)$. Next we find that the plane ABC through the three points A, B, C can be described by the cartesian equation

$$12x - 12y + 12 = 0.$$

Finally, we find the sphere S passing through A, C, X and $Y(0,7,4)$ this time giving the coordinates axes as x, y and z and calling its centre Q, and see that it has volume 288π, its equation is

$$5 + x^2 + y^2 + z^2 - 8x - 10y = 0,$$