VECTORS and MATRICES

Before starting matrix calculations, load the linear algebra package:

\[
> \text{with(linalg)}:\n\]

Vektors and matrices are created with the commands vector and matrix respectively. A vector (which is not a special case of a matrix) is regarded as a column vector by multiplication. Some examples:

\[
> A := \text{matrix}([[4,34],[-a,sqrt(2)]]):
> a12 := A[1,2]:
> v := vector([x,-12]):
> w := v[2]:
\]

The command array can also be used to create a matrix.

Whenever doing matrix calculations the function evalm has to be applied. Matrix multiplication is written \( \text{&*} \) which has to be surrounded by spaces. Multiplication with a scalar is called \( * \). Thus to perform \( A = 2B^TC + D \), where the operands are matrices, you type

\[
> A := \text{evalm}(2* \text{transpose}(B) \text{&*} C + D):
\]

The characteristic polynomial of \( A \) with \( s \) as variable is obtained by \( \text{charpoly}(A,s) \); and \( \text{eigenvals}(A) \) gives the eigenvalues of \( A \) as an expression sequence. Example:

\[
> B := \text{matrix}([[1, sqrt(2)], [-ln(3), 3]]):
> \text{evalf}([\text{eigenvals}(B)],5):
\]

\[
[2. + .74404 I, 2. - .74404 I]
\]

TYPES and TYPE CHECKING

Maple is not strongly typed. Variables may be assigned an object of any type. The type function is used to test if a value is of a given type, e.g.

\[
> \text{type}(3/2,\text{integer});
\]

\[
\text{false}
\]

\[
> \text{type}(3/2,\text{rational});
\]

\[
\text{true}
\]

\[
> \text{type}([1,3/2],\text{list(\text{integer})});
\]

\[
\text{false}
\]

\[
> \text{type}([1,3/2],\text{list(\text{rational})});
\]

\[
\text{true}
\]
> int(sin(x)*x,x);
\[ \sin(x) - x \cos(x) \]

> evalf(int(sin(x)/x,x=0..1),15);
\[
0.946083070367183
\]

> sum(binomial(n,k)*x^k,k=0..n);
\[
n (1 + x)
\]

EQUATIONS

The call solve(eq, var) solves the equation eq w.r.t. var. If there is no = in eq the right hand side is supposed to be zero, and if var is left out all variables occurring are considered unknown. If there are several solutions to eq then solve returns an expression sequence, i.e., several expressions separated by commas. It is up to the user to collect the solutions in a set or a list etc. An equation system is given as a set of equations; in this case the second argument is a set of variables.

> eq:=x^2-x+1=0;
> solve(eq);
\[
I 3
\]

> S:=[solve(2*z^3-z^2+11*z-21)];
\[
I 3
\]

> solve({x+y-y,x^2-y^2});
\[
\{ x = 0, y = 0 \}, \{ x = 1, y = 1 \}, \{ x = 1, y = -1 \}
\]

Sometimes Maple answers using RootOf, which is short for “the solution of the equation”. This function uses the global parameter \_Z. To force a (numerical) solution, use fsolve or allvalues.

To solve the differential equation deq with y as unknown, type \texttt{dsolve(deq,y(x))};

An example of \texttt{deq}:

> deq:=diff(y(x),x,x)-y(x)=0;

Initial conditions are treated like this:

> deq:={diff(y(x),x,x)-y=0,y(0)=1, D(y)(0)=0};

You can also solve a DE numerically with a 4-5th order Runge-Kutta. Type

> ?dsolve[numeric] for more information about this.
SIMPLIFICATION

Some pretty obvious commands are: simplify, expand, factor

\[ \text{factor}(x^4 + 4) = 2 \, (x - 2 \, x + 2) \, (x + 2 \, x + 2) \]

\[ \text{expand}(\sin(x+y)) = \sin(x) \cos(y) + \cos(x) \sin(y) \]

Sometimes simplify(expand(expr)) differs from simplify(expr).
The opposite of expand is combine, in some sense.
For simplification of rational expressions, use normal:

\[ \text{normal}((x+1)/y+x/(x+y)) = \frac{2 \, x + 2 \, x \, y + x + y}{y \, (x + y)} \]

\[ \text{collect}(\text{pol}, \text{var}); \text{ returns the multivariate polynomial } \text{pol written as a uni-} \]

\[ \text{vari ate one, in the variable } \text{var. More advanced uses are possible.} \]

\[ \text{collect}(y^2x^2+7xy+y+a)y^2-y+ux, y) = 2 \, 2 \]

\[ (a + x) \, y + (7 \, x - 1) \, y + u \, x \]

SOME MATHEMATICAL FUNCTIONS

\[ \text{diff}(f,x); \text{ means “differentiate } f \text{ with respect to } x \]

\[ \text{diff}(f,x^n); \text{ yields the } n \text{th derivative.} \]

\[ \text{grad}(f,[x_1,x_2]); \text{ computes the gradient of } f \text{ w.r.t. } [x_1,x_2]. \]

\[ \text{jacobian}(f,[x_1,x_2]); \text{ jacobian matrix of the vector } f. \]

\[ \text{laplace}(f,t,s); \text{ yields the laplace transform of } f(t) \]

\[ \text{int}(f,x); \text{ int}(f,x=a..b); \text{ indefinite and definite integration} \]

for numerical integration apply evalf. for complex integration, apply evalc.

\[ \text{sum}(f,i); \text{ sum}(f,i=a..b); \text{ indefinite and definite summation} \]

for products use product which has the same syntax as sum.

\[ \text{limit}(f,x=a); \text{ limit of } f \text{ as } x \text{ goes to } a \]

\[ \text{limit}(f,x=a,\text{right}); \text{ a directional limit} \]

\[ \text{taylor}(f,x=a,n); \text{ a Taylor expansion of } f \text{ about } x=a \text{ to } O(x^n) \]

See also series for more general series expansions.
g := (x, y) -> x * y - 1

> g(f(1), 2);
- 2 a + 1

If the function is more complicated and local variables are needed the function proc should be used. In the example below the function myfcn takes one argument (a number) and returns another number using the local variables t and u during the computation.

myfcn := proc(x)
local t, u;
if x > 0 then t := evalf(sqrt(x), 3) else t := x^2 fi;
if t > 2 then u := t + x else u := t - x fi;
max(t, u)
end:

The value returned from the procedure is the one of the last expression - the one before end. If the number of in-arguments is not predetermined, use args and nargs.

The alias facility is quite useful. An example:

alias(tp = transpose, SE = simplify@expand, choose = binomial):

LOOPS

General for-loop: for i from 1 by 1 to 10 do x[i] := i^2; y[i] := i^3 od:
Simplified: for i to 10 do x[i] := i^2; y[i] := i^3 od:
Writing for t in L do t^2 od; is equivalent to
for i to nops(L) do t := op(i, L); t^2 od;
While-loop: while i < 3 do i := i^2 + 1 od;

CONDITIONAL CLAUSES

General syntax: if x > 0 then a := x else a := -x fi:

if x > 0 then s := 1 elif x = 0 then s := 0 else s := -1 fi;

PLOTTING

If you are interested in plotting, run Maple under X-windows or on a 386, etc. Useful plot commands are: plot, plot3d, tubeplot, spacecurve

The syntax of plot and plot3d is hopefully obvious from the examples below. They can both take a set of functions as their first argument.

The advanced plot-functions have to be loaded with the command > with(plots);
Some simple examples:
> b := 4:
> [a, b, c];
>
> [4, 4, 3]

> b := 'b'; [a, b, c];
>
> [b, b, 3]

This principle does not work for vectors and matrices. E.g. no evaluation of the entries of the matrix \( A \) is made at the call \( \text{op}(A) \); The best way to circumvent this is to use \text{subs} (see Elementary Manipulation below).

For complex evaluation of \( \text{expr} \), use \( \text{evalc}(\text{expr}) \);
To evaluate the expression \( \text{expr} \) numerically with \( n \) digits, type \( \text{evalf}(\text{expr}, n) \); If \( n \) is left out the floating point precision is determined by the variable \text{Digits}.

The function \text{evalhf} uses the hardware floating point of the system to enhance speed.

**ELEMENTARY MANIPULATION**

> \text{op}(n, \text{expr}); extracts operand number \( n \) in \( \text{expr} \). Example:

> a := x + 2*y*z^2;
>
> 2
> a := x + 2 \cdot y \cdot z

> \text{op}(2, a);
>
> 2
> 2 \cdot y \cdot z

> \text{op}(2, "");
>
> y

> \text{nops}(\text{expr}); number of operands of \( \text{expr} \), e.g. the length, if \( \text{expr} \) is a list.

> \text{subs}(a = b, \text{expr}); Substitute \( a \) by \( b \) in \( \text{expr} \). If the second argument is a vector, matrix etc. you have to use \( \text{op} \): \( > \text{subs}(a = b, \text{op}(M)) \);

\text{rhs}, \text{lhs} right and left hand sides of equations
\text{Re}, \text{Im} Real and imaginary part

> \text{coeff}(\text{pol}, \text{var}); gives the coefficient of \( \text{var} \) in the polynomial \( \text{pol} \).

**USER-DEFINED FUNCTIONS**

There are several possibilities for writing your own functions in \texttt{Maple}. The simplest is probably the arrow construct. An example:

> f := x -> x^2 - a;
>
> 2
> f := x \rightarrow x - a

> g := (x, y) -> x*y - 1;
> 1, (2,3,4), 5;
   1, 2, 3, 4, 5

> seq( i!, i=1..5 );
   1, 2, 6, 24, 120

A list is of the form [op1, ..., opn] and a set {op1,...,opn}. The i:th element of the list L is addressed L[i]. (also works for sets). Examples:

> L:=[17,f,ln(y),[1,9]];  L[2];
   f

> S:={9,L[1],L[4][2],9};  S := {9, 17}

There is no special append function; instead, to append x to the list L you write

> L:=[op(L),x];
   L := [17, f, ln(y), [1, 9], x]

Some operations you can perform on sets are: union, intersect, minus:

> {1,2,3} union {a,2,b} minus {c,3};
   {1, 2, a, b}

> map(fcn,L);  means "apply the function fcn on every element in the list L".
This also works for sets, vectors, matrices etc. If fcn takes several arguments you type

> map(fcn,L,arg2,arg3);

ASSIGNMENT and EVALUATION

Assignments are made with := whereas the equality sign = is used for equations.
To check the current value of a variable type its name: > a;
The exceptions are vectors, matrices, arrays, tables and procedures for which the print
or op command has to be used: > print(A);  or > op(A);
To check if the variable a has been assigned a value you can type > assigned(a);
To unassign the variable a write > a:='a';
The main evaluation rule in Maple is that every expression is evaluated as far as possible, i.e. all variables in the expression are substituted by their value if they have been assigned one. Thus the order in which assignments are made is of importance, as is illustrated by the following example.

> a:=b;
   a := b

> b:=3;
> c:=b;
   c := 3
> evalf();
   2.197224577

> [0$4];
   [0, 0, 0, 0]

> {$1..5};
   {1, 2, 3, 4, 5}

> diff(x(t),t$6): #Sixth derivative of x(t) w.r.t. t
> x.(1..4);
   x1, x2, x3, x4

> (sin@arcsin)(x);
   x

The function convert(expr,type) converts between different data types.
If you run Maple under UNIX clear the screen by typing > !clear;
(Thus ! means “escape to host.”)

For translation of an expression into LATEX type > latex(expr,foo);
The second argument, which is optional, is a file name.
There are similar functions for C and FORTRAN.

HELP FACILITIES

> ?fname; displays a help message about fname, and a general help message can be
   obtained by typing a sole question mark: > ?
The call > anames(); returns a list of all names that have been assigned a value.
Assign the global variable printlevel to an integer and Maple will trace the execution
of all its code.
The trace function can be used to trace the execution of a specific procedure.

READING AND SAVING FILES

To save all data on the file foo type > save(foo);
If you just want to save some assignments type > save(name1,name2,foo);
The command used for reading from a file is called read: > read(foo);

SEQUENCES, LISTS AND SETS

Sequences are expressions separated by commas. Sequences can be generated using
the seq command.
A Short Maple Primer

Krister Forsman, Dept. of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden
email: krister@isy.liu.se
1992-02-28

Preface

Maple is a nice and powerful computer algebra system. There is also a very readable manual for Maple. However, many people do not do symbolic calculations every day and may need to use the program only occasionally. This is a manual for those people. It is totally incomplete, it has examples instead of general syntax description of the commands, it only accounts for a small fraction of the existing functions.

The basic idea is: look here for the function you need, and if there is not enough information about it here (there typically won't be) use the on-line documentation or read the real manual. The Maple version described here is version V. Caveat: since the functions are mostly explained by examples, most of them have more facilities and are more powerful than it appears here.

STARTING MAPLE

Start Maple by typing maple. If you wish the file foo to be read when Maple is started, type maple < foo. If there is a file called .mapleinit in your home directory it is automatically read. There is also an X windows version of Maple, which is started with the command xmaple.

BASIC FACTS

The prompt is >
Each command is terminated by a semicolon (;) or a colon (:).
; => the result is showed, : => the result is not showed.

Maple is case sensitive. Comments start with # and run to the end of the line.

> k! ; denotes k factorial. Exponentiation is written ^ or **. The logarithm function is called ln. π is exp(πi). ∞ is called infinity. For numerical evaluation apply evalf.

The repetition operator, $, is very useful. The dot-operator (.) denotes concatenation. The composition operator is written @.

"The latest expression" is called by ". The latest expression but one is "", etc. For a history encompassing more than the three latest expressions returned, use the command history. Here are some examples

> ln(3^2);

\begin{align*}
\ln(9)
\end{align*}