

# Hints and common pitfalls for Maple

Kees Lemmens,  
Department of Applied Mathematical Analysis

February 28, 2007

Experience shows that most people starting with Maple struggle with the same problems and make the same sometimes obvious errors. This document tries to list and discuss the most important ones below.

1. **Use restart** Many problems arise while going randomly around the sheet, reusing or redefining older variables. This often results in strange answers or errors or infinite loops caused by recursion. If you get these : give a restart to clear the memory and re-enter your statements in the correct order (simply by going there and pressing return).
2. **Screen vs. memory** If a statement is visible on the screen it won't necessarily be in the memory as well ! Only by pressing return on a statement it will be executed and the result stored in memory.
3. **Use sequential order** Put your statements in such a way that entering them sequential (e.g. by using "execute worksheet") always works. If you first need to enter the statement on line 1, then on line 4 and then go back to line 2 your asking for trouble.
4. **Assignments versus equations** Consider the following 2 examples :

```
a := b;  
a = b;
```

**Assignment** :  $a:=b$ ; implies 'store the value of **variable** b in **macro** a'  
**Equation** :  $a=b$ ; defines an equation stating '**variable** a is equal to **variable** b'

Confusion between these 2 is one of the most commonly made mistakes in either programming language, so be careful !

It is even possible to store an **equation** inside a **macro** like in

```
c := a = b;
```

Note that assigned macros may contain ANY statement, list of statements, equation(s), array(s) and even complete plots !

5. **Colon and semi-colon** Every statement in Maple should end with a (semi-)colon or it won't be executed. The difference is that ":" suppresses the output and only stores results in memory, so nothing seems to happen, while ";" will show the result on the screen (in blue).

6. **Lists and sequences** Consider the following 2 examples :

```
b := sin(t),cos(t);  
c := {sin(t),cos(t)};
```

The first one creates a macro with a sequence of 2 functions. You can select either of them using `b[1]` or `b[2]` or the whole sequence by using `b`.

The second one creates a macro with a **single** list of 2 functions. Selecting is also possible here but it is not guaranteed that `c[1]` gives you indeed the first element, as lists are automatically sorted.

If Maple expects an equation or list of functions as a **single argument** like in a `plot` you can supply the second one but NOT the first one.

Example :

```
c := {sin(t),cos(t)};  
plot(c,t=0..2*Pi);
```

7. **Equations versus functions** Equations and functions are not the same thing and can't be mixed : if you only need the functionpart on the rightside of an equation you'll need to separate it using the **rhs** command !

```
eq := x(t) = sin(t);  
fun := rhs(eq);  
plot(fun,t=0..10);  
sys := {x(t) = cos(t), y(t) = sin(t)};  
fun1 := rhs(sys[1]);  
plot(fun1,t=0..10);
```

8. **Nested lists cannot replace single lists** In Maple it is important that an array or list has the expected shape : for many commands the argument should be a one-dimensional list. The same applies for arrays : a 2x2 array is not the same as a 1x4 array or a 4x1 array. Consider the following example :

```
des := {D(x)(t) = y(t), D(y)(t) = -x(t)};  
ics := {x(0) = 1, y(0) = 0};  
dsolve({des,ics},{x(t),y(t)});
```

This command fails because the first argument for `dsolve` is not a one-dimensional list, but a nested list with 2 child lists inside the parent one.

The following however will work fine :

```
des := D(x)(t) = y(t), D(y)(t) = -x(t);
ics := x(0) = 1, y(0) = 0;
dsolve({des,ics},{x(t),y(t)});
```

Or, equivalent :

```
des := {D(x)(t) = y(t), D(y)(t) = -x(t)};
ics := {x(0) = 1, y(0) = 0};
dsolve({des[1],des[2],ics[1],ics[2]},{x(t),y(t)});
```

## 9. Functions versus simple macros

Although some macros may look very much like a real function (especially the results that Maple itself often gives as an answer) , they are definitely NOT the same. Consider the following example :

```
x := sin(t) * t^2;
x(2);
x(t) := sin(t) * t^2;
x(2);
```

In both cases you won't get  $\sin(2) * 2^2$  but a completely different statement. This is because both  $x$  and  $x(t)$  are here simply macros and not functions that can pass arguments.

If you want to create a real function of one or more variables there are 2 ways to do so : either using "unapply" :

```
x := unapply(sin(t) * t^2, t);
x(2);
```

Or with the  $\rightarrow$  operator:

```
x := t-> sin(t) * t^2;
x(2);
```

## 10. Function plots versus parametric plots

If you want to plot several functions with a dependency on the same parameter in one graph you can use the following statement :

```
plot({sin(t),cos(t)},t=0..2*Pi);
```

But if you want to make a *parametric* plot of the same functions (with one on the X-axis and the other on the Y-axis) you use this syntax :

```
plot([sin(t),cos(t),t=0..2*Pi]);
```

So, this command doesn't show any sine and cosine but a circle instead and the dependent variable can **not** explicitly be found on any axis of the graph.

A phaseportrait is just a special form of a parametric plot, with on the X-axis a function  $f(t)$  and on the Y-axis it's derivative  $f'(t)$  :

```
x:=sin(2*t)*exp(t/2);  
y:=diff(x,t);  
plot([x,y,t=0..2*Pi]);
```

Hope this makes your life with Maple a little easier !

Best regards,  
Kees Lemmens, February 2006.