2 Solving Differential Equations with Maple Fall 2003

Maple has very powerful capabilities for solving differential equations, both analytically and numerically. In learning to use this capability, we need to have a clear understanding of Maple's use of the terms *function*, *expression*, and *equation*. So let's start by reviewing these terms.

Consider straight-line motion of a particle with constant acceleration a. If the particle starts from rest, its distance x from the origin at time t is given by $x = at^2/2$. In Maple language we express this relationship by defining a *function* x with the statement

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

This notation shows explicitly that the function x is a *mapping*; we input a value of t and the function outputs a value of $at^2/2$.

We could have used a different variable name, such as q, to define the function x. The statement

```
x := q -> (a*q^2)/2;
```

defines the *same function* as the previous statement, even though the variable name is different. Thus a function has a meaning that is independent of the particular variable name used to define it. By contrast, $at^2/2$ is an *expression*; $aq^2/2$ is a different expression.

In Maple language, if we want to express a function s in terms of a specific variable name, such as t, we use the notation `s(t)` to show that we have substituted the particular variable name t. Thus in Maple nomenclature, if `s` is a function, then `s(t)` is not a function; it is an *expression*. If this looks like a trivial distinction, stay tuned; it becomes crucial when we need to describe initial conditions for a solution of a differential equation.

This distinction is also important in plotting functions and expressions. In the `plot` command, you must identify the independent variable when plotting an *expression*, but you must NOT identify it when plotting a *function*. If `fcn := t -> 3*t^2`, `expr := 3*t^2`, and we want to plot the range from 0 to 2, the following forms are OK:

```
plot(fcn, 0..2);        or        plot(expr, t = 0..2);
```

but the following ones won't work:

```
plot(fcn, t = 0..2)     or        plot(expr, 0..2);
```

Try them; you'll get an error message that may contain the enigmatic term "empty plot."

To belabor the point a little more, `fcn := z -> cos(z);` defines a *function* that we could equally well write simply as `fcn := cos;` But `cos(z)` is an *expression*, not a function.

In a differential equation Maple always considers the dependent variable to be a *function*; but the terms in the differential equation are *expressions*. Remembering this simple fact will save you a lot of heartache and misery. Suppose we have a differential equation containing x and dx/dt. (The independent variable is t and the dependent variable x.) Then to Maple x is a *function*, but we have to write x and its derivative as *expressions*.

There are two ways to do this. (1) We can first make x into an expression `x(t)` and then use the Maple command `diff` for the derivative of an expression: `diff(x(t), t)`. Or (2) we can use the Maple command `D` for the derivative of a function, `D(x)` and then substitute in the variable name t: `D(x)(t)` (Remember, once again, that `x` and `D(x)` are *functions*, but `D(x)(t)` is an *expression*.)

Thus the derivative dx/dt may be written in Maple notation either as `diff(x(t), t)` or as `D(x)(t)`. Similarly, d^3x/dt^3 may be written as `diff(x(t), t, t, t)`, as `diff(x(t), t$3)`, or as `(D@@3)(x)(t)`. (Note the use of the symbol @@ and the parentheses for repeated differentiation of a function.)

Now, finally, to differential equations! Follow these simple steps:

1. Write the differential equation in Maple form and give it a name, such as "diffeq." For example,

   $$m\frac{d^2x}{dt^2} + kx = Ft^2 \qquad \text{(where } m, k, \text{ and } F \text{ are constants)} \qquad \text{becomes}$$

   ```
   diffeq := m*diff(x(t), t$2) + k*x(t) = F*t^2;
   ```

   Because x is a *function*, the dependence of x on t must be written explicitly (i.e., `x(t)`, not just `x`), so that the terms in the equation become expressions. Otherwise, Maple wouldn't know how to take the derivatives with respect to t.

2. The command for solving differential equations is `dsolve`. First state the name of the equation and then what you are solving for, i.e., the expression $x(t)$. Always give the solution a name, such as "sol." For example,

   ```
   sol := dsolve(diffeq, x(t));
   ```

   If no initial conditions are given, the general solution always contains arbitrary constants, which Maple denotes as *_C1*, *_C2*, ... The solution is given as an *equation*, in the form $x(t)$ = (an *expression*). In our example, the Maple output is

   $$sol := x(t) = -\frac{F(2m - t^2k)}{k^2} + \_C1\cos\left(\frac{\sqrt{km}\,t}{m}\right) + \_C2\sin\left(\frac{\sqrt{km}\,t}{m}\right)$$

If you want to work with the expression on the right side of the equation, you can peel it off by giving it a name, say `xx`, and using

```
xx := rhs(sol);
```

3. To solve a differential equation with initial conditions (and thus evaluate the arbitrary constants), write the initial conditions as separate equations and use assignment statements to give them names, such as "init1," "init2," and so on. If an initial condition involves the value of a *derivative* of x at a particular time, it *must* be written with the `D` notation, not the `diff` notation. Example: If at time $t = 0$, x has the value x_o and dx/dt has the value v_o, we write

```
init1 := x(0) = xo;      and      init2 := D(x)(0) = vo;
```

In this case we must tell Maple to solve simultaneously a *list* or *set* of equations consisting of the differential equation and the initial conditions, as follows:

```
sol := dsolve([diffeq, init1, init2], x(t));      or
sol := dsolve({diffeq, init1, init2}, x(t));
```

(Note the use of brackets to denote a list or set.) For the example on page 2-2, the Maple output is

$$x(t) = -\frac{F\,(2\,m - t^2\,k)}{k^2} + \frac{(2\,F\,m + xo\,k^2)\cos\left(\frac{\sqrt{k\,m}\;t}{m}\right)}{k^2} + \frac{vo\,m\,\sin\left(\frac{\sqrt{k\,m}\;t}{m}\right)}{\sqrt{k\,m}}$$

4. If you want to graph the solution $x(t)$, you must first insert numerical values for all parameters (such as m, k, and F), and for all initial values (such as x_o and v_o). This can be done in two ways; one way is to use a string of assignments, such as

```
m := 4;   k := 100;   F := 4;   xo := 1;   vo := 3;
```

Alternatively, use the command `subs`:

```
xx := subs(m = 4, k = 100, F = 4, xo = 1, vo = 3, rhs(sol));

plot(rhs(sol), t = 0..6);      or      plot(xx, t = 0..6);
```

5. It's always a good idea to check your solution to make sure it satisfies the differential equation and the initial conditions, by substituting it back into the differential equation and also checking that the initial conditions are satisfied.

6. Often the solutions of a differential equation cannot be expressed in terms of familiar functions. In such cases Maple can do an approximate *numerical* solution to get an approximate numerical value of the expression $x(t)$ for any value of t, using one of several methods such as Euler, Runge-Kutta, and others. If you haven't heard of these, don't worry. We'll study these methods briefly in Section 3, to get a rough idea of what Maple does to obtain a numerical solution.

To get a numerical solution of a differential equation, you *must* specify initial conditions. Before the dsolve command, you must give numerical values for all parameters (such as m, k, and F), and for all initial values (such as x_o and v_o). The appropriate command for the example on page 2-2 is

```
sol := dsolve([diffeq, init1, init2], x(t), numeric);
```

The solution emerges as a *procedure*; that is, sol behaves in Maple like a *function*. Then the value of x for some particular value of t, say 5, is given by sol(5). Actually, the output from sol(5) is a *list*, containing three items: (1) the value of t, (2) the value of x at that t, and (3) the value of dx/dt at that t. You can give this list a name, such as value. That is, value := sol(5); If you just want the value of x, which you might call xvalue, you can use value[2] to select the second item in the list. That is,

```
xvalue := value[2];        or        xvalue := sol(5)[2];
```

7. Numerical solutions can't be graphed with the ordinary plot command. Instead, use a special command odeplot that computes numerical values of the function at many points within your specified range and then plots them. This is part of the plots package, so you first have to load this package using with(plots, odeplot); Then to get a graph from $t = 0$ to 6 (for example), use odeplot(sol, 0..6); Note once again that sol is a *function*, not an expression, so you must specify the range as 0..6, and *not* as t = 0..6. If you use t = 0..6 , you'll get a graph, but not necessarily the one you expect. We invite you to carry out a numerical solution of the example on page 2-2 and compare the resulting graph with the one from the analytic solution.

8. An alternative form is odeplot(sol, [t, x(t)], 0..6); where [t, x(t)] is a *list* of the horizontal and vertical coordinates. With this command, t (the first item inside the square brackets) is plotted on the horizontal axis and x (second in the brackets) on the vertical axis. This is analogous to the parametric form of the ordinary plot command. You can also use this form to make *a phase plot*, which is a plot with dx/dt on the vertical axis and x on the horizontal. To do this, use

```
odeplot(sol, [x(t), diff(x(t), t)], 0..6);
```

The numbers 0..6 are the limits for the independent variable t. We'll use phase plots a lot in this course to study behavior of dynamical systems.

9. All the usual options for plots, such as color, line weight, axis labels, titles, and so on, can be used with odeplot just as with plot. For a listing of all the available plot options, type ?plot[options]. Also check out ?plot[setoptions] and ?plot[style]. The command odeplot can also be used to generate a *plot structure*, which then enables you to superimpose this graph on other graphs. More about plot structures later.