18.03: Differential Equations

A Book of Matlab Secrets Spring, 1999

In the old days computers were batch machines. This led to an impression that one could pass a problem off to a computer and let it solve it. Today we think of computers as interactive partners. This carries with it the implication that problems are solved by a synergetic interaction between human and machine. The machine adds muscle, but for much of the thinking the human is still much more agile. Besides, the end goal is an enrichment of human understanding—in this case, your understanding of differential equations.

MATLAB is a standard and powerful computational tool, used extensively in coursework at MIT and also in scientific and engineering applications. But any computer program is much less predictable than mathematics. This is in part because Mathematics is eternal, while programs are continually being replaced by new versions. John Polking's book *Ordinary Differential Equations using* MATLAB, for example, was written for the Student Edition of MATLAB 4. At home I have the Student Edition of MATLAB 5.0, and Athena and other MIT systems run MATLAB 5.2. There are many small differences, some of which are annoying.

Below are some more or less random notes, updating information in Polking's book in some cases, and clarifying MATLAB defaults in others. There is a danger in the use of MATLAB, or any other program, in a course such as this. MATLAB is optimized for certain applications. This results in frequent examples of specialized syntax and of obscure and hidden default parameter values. The result is that quite often when you ask it to compute something for you it will return an answer without complaint but it will be an answer to a different question than the one you intended. So be skeptical. Try to understand the answer MATLAB gives. Test it against your intuition. Don't let it make you lazy. It's supposed to *strengthen* your intuition, not weaken it.

These notes may expand as the term progresses. In fact, if you observe MATLAB behaving in hard-to-understand ways, let me (`hrm@math.mit.edu`) know about it and I will try to explain it. It may make it into these notes.

The entries in these notes are keyed to pages in Polking's book. *Ordinary Differential Equations using* MATLAB,

p 52 `simple` seems to have gotten dumber since 1995, and fails to return such a simple expression for `h`, despite trying a number of different strategies which it does list. (Thanks to Elizabeth Immen.)

p 56 `dsolve` defaults to `t` as independent variable, whether or not it occurs in the expression: `D` is interpreted as d/dt. If `x` occurs it will be interpreted as a constant. This default can be changed. To make x the independent variable, for example, add `'x'` as a final parameter in the call (after a comma). This default contradicts what is said on p. 104, at the bottom of the page. If you try out that example you will find `t` in the answer, not `x` as given.

There are other tiny changes in current versions of MATLAB. For example `dsolve` returns the two portions of the general solution on the bottom of p 57 in the reverse order, and the factors of each occur in the order opposite to what is shown by Polking.

p 63 `subs:` Current versions of Matlab have fields 2 and 3 exchanged in `subs`: so you must write `p = subs(p,'pzero',10)` in the example at the top of the page, and so on throughout.

p 87 `ode45` and presumably the other solvers require *column* vectors as input when you are using them to solve systems of differential equations. The construction x(1) = ... , x(2) = ..., as in Polking p 87, produces a *row* vector: so his routines don't work as written. You have to replace `xpr(1) = ...` by `xpr(1,:) = ...` to produce a column vector output; or, better, simply `xpr=[x(2)-x(1)^2;-x(1)-2*x(1)*x(2)];`. This is another point at which MATLAB defaults can be annoying: the meaning of `x(2)` depends upon the shape of the matrix `x`. If it is a row-vector `x(2)` gives the entry in the second column; if it is a column-vector, `x(2)` gives the entry in the second row; if it is neither, `x(2)` gives the first entry in the second row. As a constructor, `x(2)=...` constructs a row-vector of length 2 and fills the second entry.

p 97 The behavior of Polking's program `eul` in the "negative direction" seems to depend upon the implementation. `[t,x]=eul('  ',[0,-1],.1)` seems to work fine on my Student Edition 5.0 at home but gives bizarre results under Athena's 5.2.

p 119 Variable names like `x1` are apparently now allowed, contrary to the **Warning**.

p 126 `*`'s appear in a matrix if the entries get too big, not just too small.

p 129 Polking says Matlab always approximates, and he means it. If you type `format rational` you'll be surprised to find that `pi` is `355/113`, and even more surprised to find that `7^19` ends in `0`. MATLAB approximates everything to about 16 digits. MATLAB arithemetic is strange, as a result. It is a theorem of Ioanid Rosu that the smallest MATLAB number `x` greater than `1` which is such that `1-x*(1/x)` does not return zero is `1+(257736490*(2^(-52)))`.