Getting from x to y without crashing: Computer syntax in mathematics education

By David J. Jeffrey

Department of Applied Mathematics, University of Western Ontario, London, Ontario, Canada N6A 5B7
djeffrey@uwo.ca

Received: 1 September 2009 Revised: 1 February 2010

When we use technology to teach mathematics, we hope to focus on the mathematics, restricting the computer software systems to providing support for our pedagogy. It is a matter of common experience, however, that students can become distracted or frustrated by the quirks of the particular software system being used. Here, experience using the systems Maple and Matlab in the classroom is described with a view to highlighting the places where students are most likely to experience frustration. Strategies to help them avoid the common pitfalls are given.

1 INTRODUCTION

One sometimes hears people talking about cars say "For me, they are just a means of getting from A to B." This is usually in response to someone who is in the middle of analyzing the latest cars on the market, or explaining how an engine works. Certainly, modern drivers can get from A to B with little technical knowledge, but they cannot escape entirely from learning something about the inner workings of their car, if they do not want to crash. The same is true of mathematical software. Mathematics students may not be sitting at our keyboards to enjoy programming, but they cannot use the computer to get from x to y without some technical knowledge.

Cleve Moler (2004a) has described how Matlab was originally written so that students who could not write programs (in FORTRAN) could still access the newly collected computational routines known as LINPACK and EISPACK for matrix computations. The world of computer use has advanced greatly since the 1980s, when Matlab was first released. Now the Matlab interface looks a lot like a programming interface when viewed by students who have grown up with word-processors, web browsers, and email. In addition, Matlab is no longer a program just for students; it has become a popular tool for engineering and other advanced applications. As a result, it contains many features that speed the life of a professional user. These conveniences, however, can trip up the faltering steps of a student.

The software system Maple is another popular system that is used when teaching mathematics. It also has an interface which started in the 1980s and which now presents its student users with challenges which get in the way of teaching, even though in the past the interface was an enabling technology.

It seems to be the case that an ability to study mathematics and to do well at assignments does not always partner an ability to use computers, particularly in a programming or semi-programming mode (it would be interesting to see an objective study of this observation). Numerical methods are still not a prominent part of a mathematical education. Instructors who use computers in their teaching are typically people who feel at home with computers, and find it difficult to sympathize with students who "can do the maths, but not the computer stuff". Even though computer use is not formally a part of a mathematics course on calculus or linear algebra, it is important to take some time specifically to teach the software tools to the students and to inoculate them against the most common difficulties. It is worthwhile to reinforce the precautionary lessons by setting explicit assignment questions or projects on the most common difficulties.

2 TEACHING WITH MATLAB

Some of the observations here are more relevant to courses that make more intensive use of Matlab, such as a numerical analysis course or a course on differential equations, in the latter case, when it is taught with a significant component of numerical approaches. None the less, even lighter use, such as supplementing a linear algebra course, can bring students in touch with the difficulties described.

Problem: entering commands. A book on canoeing (McNair 1972) recommends that the first paddling stroke to teach students is the *back*stroke. Most people think that the purpose of canoeing is to go forward, and so are puzzled by this strategy. The author, however, explains:

> "*... because we note a tendency to revert to the most familiar stroke when flustered*."

The author thinks that if a beginning canoeist has got into a troublesome situation, then it is most likely that the paddler needs to reverse away from the trouble. The beginner, however, will also be flustered and not thinking clearly. Hence they will revert to the first thing they learned, and that, if they followed the author's advice, is paddling backwards.

This advice applies to Matlab. Most books open with a description of the interactive nature of Matlab. The tutorials that come with Matlab do that, as do textbooks such as Moler (2004b) and Higham and Higham (2000). The problem with this is that when the material being taught becomes more advanced, it becomes more and more difficult to succeed by working only interactively. For example, entering a complicated set of differential equations becomes very error prone. It is better at this stage to open an editor window and work there. If, however, the instructor waits until the editing window is unavoidable, I have found that for weaker students it is too late. When the time comes to solve a problem under pressure (for example, in an exam setting or when rushing to hand in that last assignment), the students who started out in the interactive window return to it, and cannot tear themselves away to the editor.

A solution: It is better therefore to start the very first laboratory session with the instructions:

1. Chose the directory (folder) where you will save your work.
2. Make that directory (folder) the current directory. This can be done easily using the current directory bar in the Matlab window.
3. Open a new file.
4. Type a comment line with the current date.
5. Type a comment line describing what you are going to do.
6. Save the file (giving it a name of course).
7. Start working on the first problem.

This not only prepares the student for later work, but solves the problem of how they hand in their work. There are other options, such as the diary command, but that command preserves all the confusion of the interactive window.

Problem: entering matrices. What is the best way to type in matrices? The Matlab tutorials and the textbooks like to separate elements by spaces. Thus they recommend

```
>> A= [1 2 3; 4 5 6]
```

This is probably because matrices are printed in books with spaces. In Matlab, it is equally legal to type

```
>> A= [1, 2, 3; 4, 5, 6]
```

Is one better that the other? Yes! Because typing accurately and close proofreading are not skills that every student possesses. A student who is not a good typist, or not a good proof-reader, could be held up for long periods trying to discover what is wrong with the command such as

```
>> A= [0 1 2 3 4; 5 6 78 9]
```

(The space between 7 and 8 has been skipped by the person typing the input). More recent versions of Matlab allow multi-line input. Thus we can type

```
>> A= [0 1 2 3 4
 5 6 78 9]
```

where the mistype is still present, or, preferably,

```
>> A=[0 1 2 3 4
        5 6 7 8 9]
```

This multiline format (especially when the last form is used) encourages a more tabular style of thinking, which is another way of reducing the danger of mistyping.

A final comment is that the point of this discussion is how to input data into Matlab, not the way textbooks print matrices. Below, matrices are printed without commas, and I am not suggesting changing this traditional mathematical practice.

A solution: A useful exercise would be to present the correct and incorrect entries given above to a class and encourage them to discover the difference between the typed lines. Comma-separated lists are easier to correct, and I recommend them in spite of the current textbooks and Matlab's own help pages. I think that ultimately an instructor must not aim to impose a standard set of behaviours on a class, but present them with enough information to make their own choices. My personal style is to use both commas and spaces to make the syntax as clear as pedantically possible, but I would not go as far as to demand such compulsive behaviour from all of my students.

Problem: non-mathematical extensions. We move on to the first mathematical point. There is no textbook of linear algebra that defines the addition of a scalar and a matrix, but in spite of this, Matlab does define such addition:

$$a + \begin{pmatrix} b & c \\ d & e \end{pmatrix} = \begin{pmatrix} a+b & a+c \\ a+d & a+e \end{pmatrix}$$

Other software systems are much stricter in their interpretation of mathematics. The feature above is useful for experts, but not helpful for instructors introducing matrices.

A solution: One often reads the statement "In Matlab, everything is a matrix". This is not accurate. It is better to say that everything is an array. This brings the question "When is an array a matrix and when is an array an array?" The answer is that it depends on the operation being carried out. This brings us to "dot" notation.

Students find it difficult to see that the following two lines are different:

```
[1, 2, 3] * [4, 5, 6]
[1, 2, 3] .* [4, 5, 6]
```

As with the earlier typing problem, that dot is easily overlooked by students. The first line says that the arrays are to be treated as matrices, meaning that in this case the multiplication will be an error, and the second line says that the arrays are to be treated as arrays. This distinction should be given special attention.

Problem: not seeing dots before the eyes. The way Matlab switches interpretations between arrays and matrices combines with the next feature to cause students endless trouble. Matlab defines division of matrices. Consider

```
[1, 2, 3] / [4, 5, 6]
```

Again, we note that no textbook of linear algebra defines division of matrices. The only operation that exists in linear algebra is multiplication (on left or right) by an inverse. If we write A=[1, 2, 3] and B=[4, 5, 6], then Matlab defines division A/B as the solution of the matrix equation

$$XB = A$$

If this equation does not have a solution in the ordinary sense, then Matlab proceeds by computing the solution defined in the sense of least-squares. We can join this up to the usual way in which the theory of least squares is taught by taking the transpose of each side: $B^T X^T = A^T$. Many students now can find the normal equation: $BB^T X^T = BA^T$. This could then be solved for X (Matlab would use a different method, but that does not matter.) The reader who knows linear algebra can check that this gives [1,2,3]/[4,5,6]=0.4156.

These three features now combine as follows. Let us consider a specific problem to help with our understanding. The problem is to use the Matlab quad command (or similar command) to evaluate the integral

$$\int_0^1 \left(\frac{x}{(1+x)} + \frac{1}{(2+x)}\right) \mathrm{d}x$$

We must write a function to evaluate the integrand; this function will then be handed to quad. The Matlab function for the integrand is

```
function out = Ex1(x)
out = x ./ (1+x) + 1 ./ (2+x);
```

Now we suppose that the student makes a mistake and leaves out the first dot.

```
function out = Ex1error(x)
out = x/(1+x) + 1./(2+x);
```

What happens?

1. The function Ex1error is called with an array of values. It is important to know that Matlab functions such as quad can call a function with an array of values.
2. 1+x and 2+x evaluate to arrays.
3. x/(1+x) evaluates to a 1x1 matrix. 1./(2+x) evaluates to an array.
4. The 1x1 matrix is re-interpreted as a scalar.
5. The scalar and the array are combined together by using the above rule for adding a scalar and an array.
6. A plausible, but wrong, array is returned by the function.
7. The student cannot see the error or understand it and is frustrated because the answer to the problem is wrong.

A solution: Asking questions along the lines "How will Matlab evaluate [2 , 4] / [3 , 4] and [2 , 4] ./ [3 , 4]" can encourage students to be more sensitive to the differences between the two constructions. My preference is to include such questions on assignments and then repeat them on exams; other teachers may prefer other approaches.

Problem: sometimes a solution, sometimes not. Matlab tries to be helpful to professionals by automatically returning approximate solutions to inconsistent equations. This can confuse students who are struggling to digest the differences between systems with one solution, families of solutions or no solutions.
Let us take some specific examples. Consider a set of equations having no solution, i.e., that are inconsistent:

$$x+y = 1 ,$$
$$x+y = 2 .$$

We can convert this into the matrix equation

$$AX = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Then we can try to solve this using the Matlab command

```
[1,1 ; 1,1] \ [1 ; 2]
```

This will produce an error message from Matlab that the matrix is singular. However, now we try a slightly more difficult problem:

$$x+y = 1 ,$$
$$x+y = 2 ,$$
$$x+2y = 3 .$$

This becomes the matrix equation

$$AX = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

It is probably still the case that in a first course on linear algebra, the instructor would like the students to recognize that this system is inconsistent. However, if students try using Matlab to verify their ideas, they might type

```
[1,1 ; 1,1 ; 1,2 ] \ [1 ;2 ; 3]
```

and obtain the "solution" [0 ; 1.5], which happens to be an approximation to a solution based on the theory of least squares.

A solution: There are two ways to handle this problem. The first way is to include the theory of least squares in the curriculum of the course. Many instructors object to this, however, because it is an admission that software, rather than the instructor's view of mathematics, is dictating the contents of the course. The second way is to warn students explicitly that they must not use the Matlab operator \ to decide consistency and inconsistency. In fact, it might be best to require every solution to be obtained using the RREF (Reduced Row Echelon Form) command explicitly. This means that Matlab is emulating the methods used by a human without Matlab.

Problem: function names. A first course on Linear Algebra that is using Matlab will probably not require the use of functions. Since the object of using Matlab is to illustrate mathematics, rather than teach programming, this is understandable. At second year, however, when we teach differential equations, functions become important.
The Matlab syntax for a function is the keyword function followed by the output variables of the function, an "equals" sign, a name and then a list of input variables. Thus

```
function output = Name (list of input variables)
  % Matlab commands that compute the output
  % go here
```

This function is written in the editor window and then saved to disk as an ".m" file. In this case "Name.m" would be pre-selected by the editor. A difficulty for a student arises if they now use this function as a model for others. It now is possible for a student to write a function

```
function output = NextTry (list of input variables)
  % Matlab commands
```

And make the mistake of saving it to disk under the name "Name.m" or some other name different from "NextTry". Which name does Matlab use? The answer is the name of the disk file. The name inside the file is a dummy name that Matlab does not use.

A solution: As with other problems, there are several steps to take to help students protect themselves. One way is to set the class tasks or examination questions such as "Consider the following text saved in the file 'Name1.m'. How do you call this function from Matlab?" Another way is to give students explicit instructions, for example, "Write a function called 'Quest1' and save it in a file called 'Quest1.m'. The function will do the following things ..." It is worth recalling again that this discussion assumes the purpose is to teach mathematics, and teaching programming is taking place in a different course (which the mathematics students may not be taking). However, one cannot generate the solution of a differential equation without some knowledge of functions.

Problem: variable assignments written on the screen. When a function is executed, any statement that does not end in a semicolon will cause output to be written in the command window. Thus the function

```
function y = TwoTimes(x)
    y=2*x
```

will cause the command screen to show

```
>> TwoTimes(3)
   y = 6
  ans = 6
```

(The output has been compressed relative to typical Matlab printing.) Students trying to access "y", which they see was assigned "6", are puzzled to receive an error message.

A solution: Many non-computer science students find the idea of scoping a variable difficult. Either one can include it in the course material (thus losing time to teach mathematics) or simply dictatorially demand that every line end in semicolon (this, however, prevents good students from using the printing of intermediate results for debugging purposes). A balance needs to be struck, and who better than the teacher to gauge the class, and decide what to do?

Problem: imaginary unit re-assigned. The symbol "i" will be used for $\sqrt{-1}$ unless it is re-assigned. Unfortunately, ever since the days of FORTRAN, the variable "i" is invariably the first thing that pops into people's brains when writing a loop.

A solution: If complex numbers are likely to arise, then training oneself and one's students to use K, L, M as loop variables is useful. Notice the use of capital letters: that is to avoid the other great confusion of lower-case L and 1.

3. TEACHING WITH MAPLE

As with Matlab, Maple offers facilities for advanced users that can cause confusion for beginning students. Some of the differences between Maple and Matlab reflect different philosophies, while some reflect the fact that Maple is an algebraic system, whereas Matlab is a numerical system.

Problem: matrices and arrays are different.

In Maple, one must say explicitly whether an array of numbers is to be treated as an array or as a matrix. This is done at the declaration stage primarily. Thus the inputs

```
> A:=Matrix(2,2,[[a,b],[c,d]]);B:=Matrix(2,2,[[e,f],[g,h]]);
> E:=Array(1..2,1..2,[[a,b],[c,d]]);
> F:=Array(1..2,1..2,[[e,f],[g,h]]);
```

create different objects. This can be seen when we combine them using the "." operator, which stands for non-commutative multiplication. We see

```
> A.B;
```

$$\begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

```
> E.F;
```

$$\begin{pmatrix} a.e & b.f \\ c.g & d.h \end{pmatrix}$$

One can switch between interpretations just as one can in Matlab, but we do it by changing the objects rather than the operations. This is an important distinction. Both systems change interpretations, but Matlab chooses to do it through the operation syntax (the dot notation) while Maple chooses to do it by changing the object before the operation. Thus in Maple we can operate on A, B as arrays even though we first declared them as matrices by using Maple's convert facility:

```
> convert(A,Array).convert(B,Array);
> convert(E,Matrix).convert(F,Matrix);
```

where the second line converts arrays to matrices so that the product will be a matrix product.

A solution: One of the reasons that the array-matrix dichotomy is so important for Matlab is the need to use the one data structure for non-matrix activities such as plotting. In Maple, such activities are better performed using other commands. Therefore it is possible to avoid referring to Maple arrays at all in a first course on matrices.

Problem: copying matrices is surprising. In Matlab, the sequence

```
>> A=[1,2]
>> B=A
>>A(2)=5
```

results in two different matrices being stored in memory. B remains the original [1,2] and A becomes [1,5]. In Maple, a sequence apparently the same

```
>> A:=<1 | 2 >;
>> B:= A;
>> A[2]:=5;
```

results in a single matrix being stored in memory. Both A and B are the matrix (1 5).

A solution: To obtain the desired effect, one must explicitly make a copy of the first matrix.

```
>> A:=<1 | 2 >;
>> B:= copy(A);
>> A[2]:=5;
```

Now there are two matrices in memory: A is (1 5) and B is still (1 2).

As with other topics in this discussion, we are faced with a tempting side track that leads away from mathematics and into computer science. How much we let these topics distract us is up to the teacher.

Problem: defining functions. The mathematical way to write down the definition of a function is to write $f(x,y) = x^2 + y^2$. Students naturally translate this into Maple as f(x,y):=x^2+y^2. This, however, does not achieve their intention. Each Maple function can have a "remember table" associated with it. The idea behind remember tables when they were introduced into Maple was to increase the efficiency of function evaluation. When a function has been evaluated for a particular case of its arguments, the result is stored (remembered) in case it is needed again. If a function takes a long time to compute its value, then looking up old results is a lot faster.

An entry in a remember table is only for specific values of the argument, and so f(u,v) would not evaluate to $u^2 + v^2$.

A solution: The correct syntax for specifying a rule, as opposed to a special value is f:= (x,y) ->x^2+y^2.

It may seem that remember tables are just nuisances, so here is a simple example of how they can be useful. Suppose we have defined the function g:=x->sin(x)/x. If we now ask Maple to evaluate g(0), we shall receive an error because of division by zero. However, we all know that the limit at x=0 is g(0)=1. We can quickly teach this to Maple by entering the command g(0):=1. Now the remember table contains this special case, and whenever g(0) is entered, there will no longer be an error message.

Problem: function or expression. There are two ways to represent a mathematical function in Maple, the two methods having been introduced at different times in Maple's

development. The first way is as an expression. It is the earlier method (meaning it was used in the first versions of Maple) and it is simpler to start with. Thus we write

```
Ex:=x^2+y^2.
```

To use this expression, we now use the commands diff, int, subs. An important point is that if we write

```
Excopy:=Ex;
Ex:=x^3+y^3;
```

Then we have two expressions, with Excopy still holding x^2+y^2.

The second method defines a function by a rule as explained above. `Exfun:=(x,y)->x^2+y^2`. With a function, one can obtain special values easily, without needing the subs command. However, now differentiation requires either reverting to the expression form, namely diff(Exfun(x,y),x), or one uses the D operator, as in D[1](Exfun). In more detail, Exfun is a function, but Exfun(x,y) and Exfun(p,q) are both expressions obtained by evaluating the function, first for the arguments (x,y) and secondly for (p,q). The differentiation operator diff expects an expression, while D expects a function. There is no equivalent for integration, and one must return to an expression first. To combine two functions, one uses the composition operator, namely @.

As with matrices, the copying behaviour of functions can puzzle students. If we write

```
Exfuncopy:=Exfun;
Exfun:=x->x^2;
```

We obtain only one function, with both Exfuncopy and Exfun pointing to the same definition.

A solution: As with matrices, to obtain the effect that most students want, we must use the copy command:

```
Exfuncopy:=copy(Exfun);
Exfun:=x->x^2;
```

It might seem that this behaviour is undesirable. To obtain some background, we must understand "last name evaluation". Usually, when working with a computer algebra system, one ends up defining chains of expressions. For example, Ex1:=x^2 is followed by Ex2:=Ex1^3. A user who asks for the value of Ex2 now expects Maple to calculate that Ex2 = x^6. This is full recursive evaluation. The designers of Maple decided that this is not appropriate for functions and procedures, because they can become very large. To get the effect of evaluation with functions, one explicitly invokes evaluation, using the eval command.

Problem: plotting functions. Although plotting functions in Maple is mostly straightforward, there are some situations (admittedly, not ones that a first-year student would meet) in which it can be difficult to figure out what has gone wrong. An example is given by plotting functions that are defined with if statements. For example, students often find it difficult to become comfortable with piecewise-defined functions. Even the simple absolute value function, or a step function, can be challenging for them. For this reason, one might want to get students to plot a function with an if statement. Consider

```
F:= proc(x) if x<2 then x else 2 end if; end;
```

If we simply ask Maple to plot this, using

```
plot(F(x),x=0..4)
```

we shall receive an error message. The reason is that Maple tries to evaluate or simplify the function before handing it to plot, and finds that it cannot decide whether x<2 is a true statement.

A solution. We can cure this problem two ways. First we can rewrite the function so that it behaves correctly.

```
f:= proc(x)
  if not type(x,numeric) then return('f'(x)) end if;
  if x<2 then x else 2 end if;
end proc;
```

This code makes sure that if x is a symbol, as it is when plot gets hold of it, then the test x<2 is not allowed to happen.

A simpler solution is to use Maple's delay mechanism and write `plot('f'(x),x=0..4)`.

Either way, the key thing is to understand the use of the single forward quote to prevent Maple from trying to evaluate a symbol.

4. CONCLUDING REMARKS

When using computers and software systems to teach mathematics, we must always balance the time spent mastering the tool (Maple, Matlab, etc.) and the time spent mastering mathematics. This is not unique to mathematics, because all students are expected these days to deliver their homework in any subject as a computer file. What makes mathematics special is the complication of the tools we use. Somehow, we seem to be still years away from user interfaces that are as universally accessible as are word processors. (Not that all students are happy with the more obscure features of their word processors.)

On the other hand, Maple and Matlab are very useful tools for many computational jobs and students really should get to know how to use them. As mathematics teachers, we are not happy to see time being taken away from mathematics, but there is some consolation, perhaps, in the knowledge that students are developing useful and important skills in addition to their mathematics.

Some years ago, the University of Western Ontario required students to purchase and use a specific calculator for linear algebra. When this happened, considerable class time was taken up teaching calculator usage. At the time, it often seemed to the instructors to be time wasted, but many students thanked their instructors in later years for teaching them calculator skills as well as linear algebra. We must be grateful when our students thank us, even if it is for teaching something that we did not regard as primary to our course.

All of the items discussed here are ones that I have taken up with some class that I have taught. Not all the classes are first-year classes, so some of the questions raised here have been of an advanced nature. However, even topics only of interest to advanced students benefit from studies in mathematical education, and this is certainly true when the software we use increases in complexity as it keeps up with the advanced level of the mathematics. On the whole, I spend about 3 hours of laboratory time in a course explicitly presenting the material I have given here. After that I concentrate on the mathematics, but project work and homework inevitably force me to remind the students of our earlier software discussions. For myself, this is a reasonable allocation of time resources, and I hope that other courses will benefit from our laboratory discussions, and that the

material eventually becomes an additional benefit taken from my course.

ACKNOWLEDGEMENTS

I thank Professor James Davenport, who listened to my opinions and even encouraged me to write my prejudices down on paper!

REFERENCES

Moler, Cleve B. (2004a) The origins of Matlab, *Matlab News and Notes, December 2004. www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html*.

Moler, Cleve B. (2004b) Numerical Computing with Matlab. SIAM, Philadelphia.

McNair, Robert E. (1972) Basic River Canoeing. American Camping Association, Martinsville, Indiana, USA.

Higham, Desmond J., Higham, Nicholas J. (2000) Matlab Guide. SIAM, Philadelphia.

BIOGRAPHICAL NOTES

Dr. David Jeffrey is a professor of Applied Mathematics at The University of Western Ontario. He uses Matlab and Maple for teaching a variety of courses on linear algebra, numerical analysis and symbolic computing.