

Taking Calculus with *Mathematica*

A student and teaching assistant presents his experiences with *Mathematica* in the classroom and offers some lessons for teachers.

Sandy D. Balkin

The Student's View

Mathematica is a powerful program that is used by many professors as an aid in teaching calculus. As one of the *Mathematica* guinea pigs in the laboratory section of the calculus course described by L. Thomas Hill and Clifford A. Reiter of Lafayette College in *The Mathematica Journal* 2(1), and now as a lab assistant for these classes, I have seen both the benefits *Mathematica* affords some students and the hardships it causes others. This article offers insight into the student's frame of mind and suggests ways to design the labs to work more efficiently towards teaching calculus and other subjects using *Mathematica*.

Many students view *Mathematica* as an excellent tool containing outstanding mathematical capabilities, but one that virtually requires a Ph.D. to be of any use. Anyone who is familiar with *Mathematica* appreciates the fact that it has a high and long learning curve. Some students, whether because of their own inexperience or poor teaching, never quite get the hang of it. Instead of aiding in the learning process, the use of *Mathematica* contributes to these students becoming easily frustrated and disenchanted with mathematics as a whole. Most students do not understand what computers can do for them in math at the undergraduate level. However, with a proper understanding of the student's perspective, professors may design labs properly to achieve maximum comprehension and enjoyment.

Computer Experience

Computers are relatively new compared to the theories they are being used to help teach. Students have a dozen years of background doing math by hand and by calculator, but probably very little by computer. It is important to remember the saying "A computer is only as perfect as the person using it." *Mathematica* is definitely no exception. The concept of strictness may be new to many users. For students who have never used computers before, *Mathematica* can be an unending source of frustration. Avoiding all unnecessary setbacks should be a top concern in organizing computer labs.

It is crucial to ensure that the lab be uninterrupted and self explanatory. First, the computer system must be stable. Nothing is worse than informing students that their entire

lab work was lost due to a memory parity error or server shutdown. Second, it is important to reinforce the strictness of *Mathematica* syntax as often as necessary. As with many programming languages, *Mathematica*'s parser is very specific about the case and order of the input commands. All students need to be told exactly what symbols such as the underscore character (`_`) and the rule signs (`/.`) do and when to use [], { }, or (). These simple misunderstandings account for much of the frustration and wasted time associated with the program. The computer lab should be set up so that the instructor, and possibly an assistant, can easily see where students are in working through the material and what may be hanging them up.

It is also very important that students know what to expect from a given statement. For example, if a `Plot` command is issued and error messages and no plot result, the student should realize that a mistake was made and that the error should be found before continuing. It may sound ridiculous, but students will usually try to simply plug and chug and barrel through a lab without trying to comprehend what they are doing. This was most evident to me when, even after using the `FindRoot` command in a previous lab, a dozen students literally typed into *Mathematica* `FindRoot[lhs==rhs,{x,x0}]` without defining the variables and *did not* realize that they might have done something wrong! To avoid this problem, the professor should develop the students' *Mathematica* vocabulary by introducing new functions and reiterating their uses in subsequent labs. This will familiarize the students with the commands and help them to recall from memory the right command for a particular task.

The Lab Exercise

The actual creation of the lab is where the instructor must exercise the greatest amount of care. First and most importantly, the professor must be careful of whether the course is Calculus with *Mathematica* or *Mathematica* with Calculus. In other words, is the professor using the program to enhance the learning of the subject, or using the subject to aid in teaching the program? Both approaches have their pros and cons. Calculus is definitely a base for upper level mathematics, but *Mathematica* can be used by students in other classes such as linear algebra and differential equations. However, I believe the calculus curriculum should not be compromised to add a lab.

Sandy D. Balkin is a senior at Lafayette College in Easton, Pennsylvania. In the Fall, he will enter Pennsylvania State University for a Ph.D. in Statistics.

Secondly, theoretical topics not covered in the classroom should not be introduced in the lab. Obviously, subjects concerning basic graphic visualization, recursion, or a large amount of arithmetic may be suitable, but topics requiring theoretical knowledge, such as curvature or methods of integration, should be taught in the classroom and then applied in the lab. Introducing a new subject in the lab causes frustration during lab time and results in only partial comprehension of the topic. Using *Mathematica* exercises to introduce concepts, such as limits, sequences and series, and integration in the form of area under a curve, is an excellent way to acquaint students with such concepts prior to teaching the theory. Attempting to teach the underlying concepts of these subjects in the lab itself may not be as beneficial. If implemented correctly, the labs have the potential to explain better than any textbook topics such as the relationship between the derivatives and the graph of a function and the differences between various methods for approximating integrals.

Thirdly, one should try not to ask students to use *Mathematica* expressions for which they do not know the meaning. For example, when demonstrating Newton's Method, it is much clearer to use a recursive function:

```
f[x_] := function
x[n_] := x[n] = x[n-1] - f[x[n-1]] / f'[x[n-1]]
```

where $x[0]$ is the initial guess, rather than:

```
f[x_] := function
g[x_] := x - f[x]/f'[x]
newton[x0_, n_] := NestList[g, f[x0], n]
```

Once students are comfortable with the technique, `FindRoot` can be introduced. If `NestList` or `FindRoot` are introduced before the concept is absorbed, the student will not have a complete understanding of what is happening. Even though `NestList` shows how quickly Newton's Method converges to a root of a function, it is important for students to understand the process so they can *play around* with parameters such as the initial guess and the number of iterations. This experimentation lets students do their own exploration of such relationships. Preparing packages that let students do things such as filling the area under a curve with a variable number of rectangles is a good way to demonstrate Riemann Sums without bogging students down with complicated syntax.

Conclusion

Many articles have appeared extolling the uses of *Mathematica* in teaching calculus and other subjects. The program can increase comprehension of the material taught very effectively, but its high and long learning curve may cause students frustration. It is up to each professor to determine whether the lab section of Calculus will be a boon to the students or will just be a way for students to avoid another hour of lecture time. Laboratory projects with *Mathematica* should be captivating and interesting, while their materials are thoroughly taught. They should come to a definite point, elucidating the concepts that were learned in the classroom or foreshadowing future lectures. 