Using Matlab in a Multivariable Calculus Course

Mark D. Schlatter
Department of Mathematics
Centenary College of Louisiana
2911 Centenary Blvd.
Shreveport, LA 71134
Phone: (318) 869-5206
Email: mschlat@centenary.edu

Abstract: The benefits of high-level mathematics packages such as Matlab include both a computer algebra system and the ability to provide students concrete visual examples. This paper will discuss how both capabilities of Matlab were used in a multivariable calculus class. Graphical user interfaces developed by the author which display three-dimensional surfaces, contour plots, gradient fields, and parametric curves and vector fields in both two and three dimensions will be presented with comments on their effectiveness. In addition, the benefits and difficulties of using computer algebra in a multivariable class will be discussed. Finally, we will examine student responses to the uses of Matlab.

Introduction

Prior to fall 1998, the highest level of technology used in our department's multivariable calculus course was graphing calculators. In the summer of 1998, I decided to try incorporating a higher-level mathematics package into the course. The resources and restrictions I had were the following:

- Our department had a license for Matlab with the Symbolic Toolbox (a package that allows Matlab to do the same symbolic calculations as Maple).
- Our students included math majors (who would use Matlab later in their curriculum), physics majors (who would be using Mathematica later), and chemistry majors (who would be using MathCad later).
- Our textbook would be the Harvard Multivariable Calculus text.

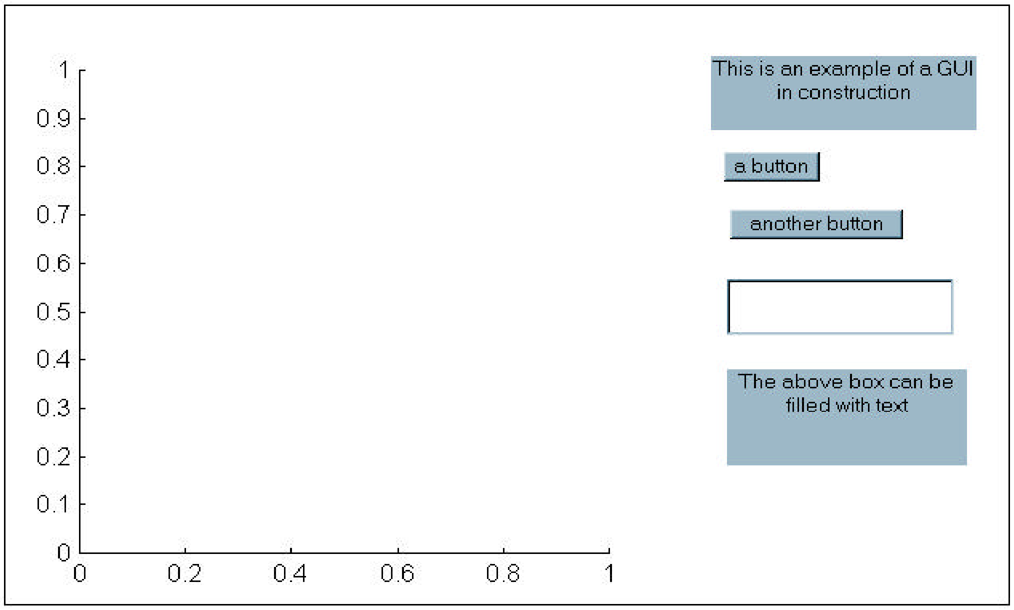
My goals for incorporating a higher-level mathematics package were twofold:

1. I wanted to help the students visualize the objects and surfaces we would encounter in the course.
2. I wanted the students to have a CAS (computer algebra system) available to assist with difficult calculations.

Based on these goals and facts, I decided to develop software packages for Matlab that would display most of the objects found in a multivariable calculus course.

Making Matlab GUI's

Matlab Version 5 allows the user to build GUI's (or graphical user interfaces). A GUI is simply an interface that can call upon different Matlab operations and perform them without the user needing to know the Matlab language. In Matlab, typing the command for the GUI calls up a window created by the GUI programmer where possible window elements include graphs, text boxes that can be filled in, and clickable buttons that carry out a series of Matlab commands. An example of a GUI in the process of construction is seen in Figure 1.



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
This is an example of a GUI in construction
a button
another button
The above box can be filled with text

Figure 1 (A GUI in construction)

The advantages of using GUI's include

- a very small learning curve for the students in terms of learning how to use the GUI and the software, and
- interfaces familiar to students (in building my GUI's, I attempted to mimic the inputs needed on a TI graphing calculator).

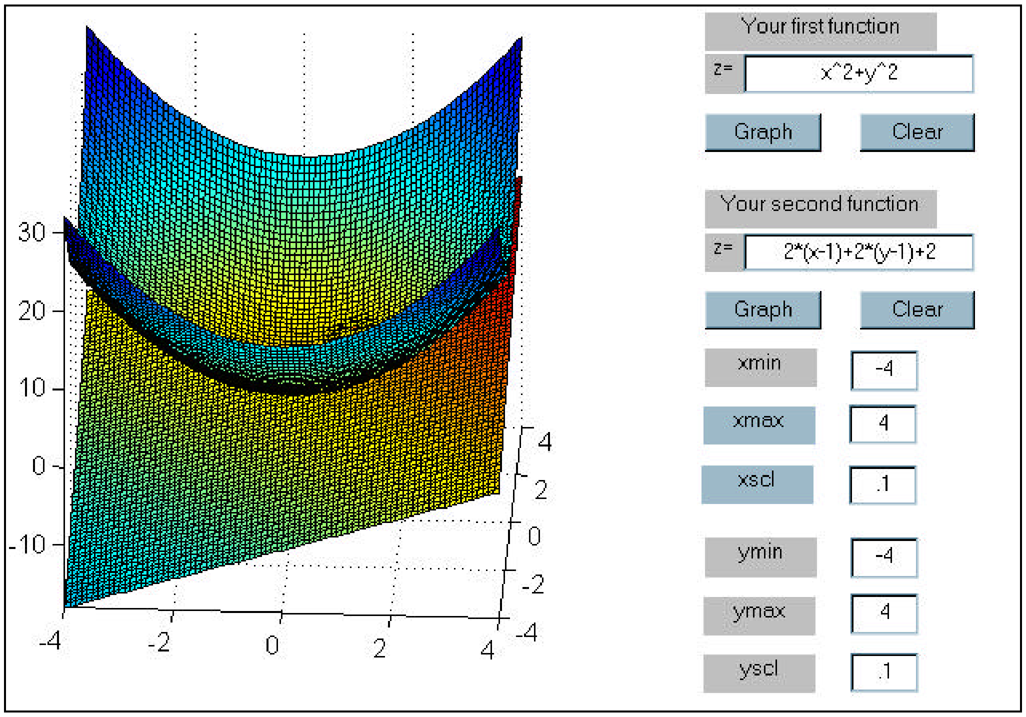
The disadvantages of using GUI's include

- the instructor must know the intricacies of the software and be able to program in it, and
- students learn how to use visualization tools on only one piece of software.

The Developed Software

For my multivariable calculus course, I developed four programs to aid in visualization. The first, `graph3d`, allows students to graph one or two functions of two variables and then rotate the surfaces in real time. (See Figure 2 below for an example of a function graphed along with its tangent plane at a point.) In lecture, I used `graph3d` to

- visualize functions of two variables
- explore noncontinuous functions
- see how differentiable functions are "locally planar"
- estimate the signs of 1st and 2nd partial derivatives
- help visualize the limits for definite integrals



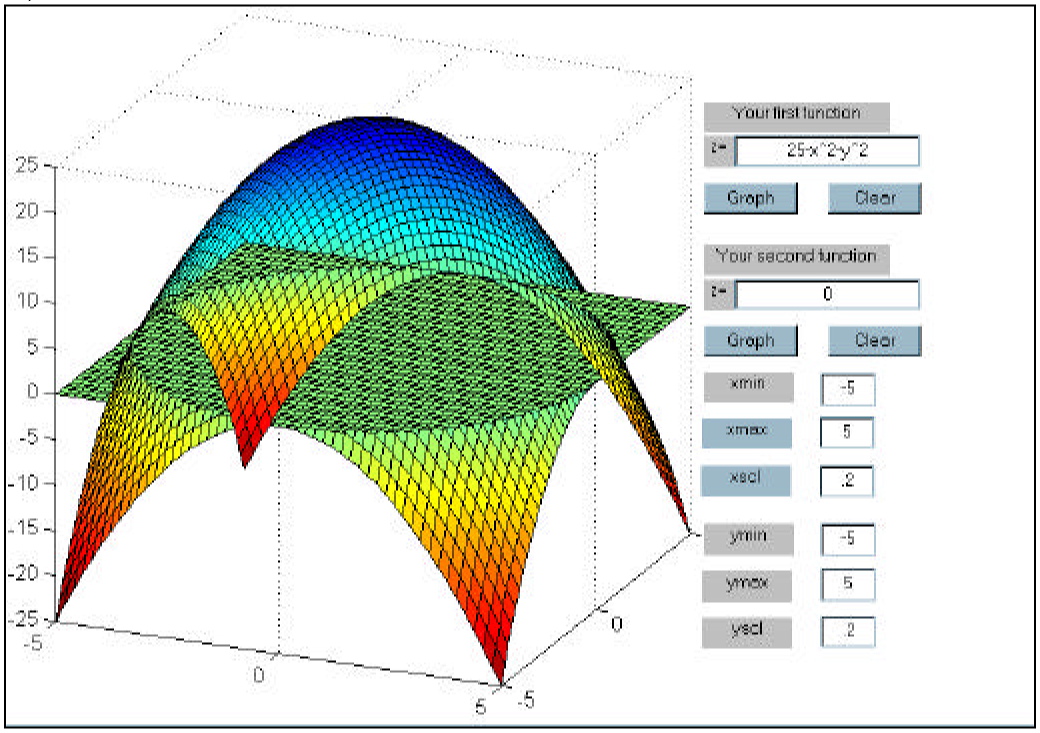
Your first function
z= x^2+y^2

Your second function
z= 2*(x-1)+2*(y-1)+2

xmin -4
xmax 4
xscl .1
ymin -4
ymax 4
yscl .1

Figure 2 (Using graph3d to illustrate tangent planes)

As an example for the last item, to find the volume under the surface defined by `z=25-x^2-y^2` but above the xy plane, I would graph both `z=25-x^2-y^2` and `z=0` and then rotate the object to see how the surfaces intersected in a circle of radius 5. This would help the students see how to set up the integral in polar coordinates. (See Figure 3 below.)



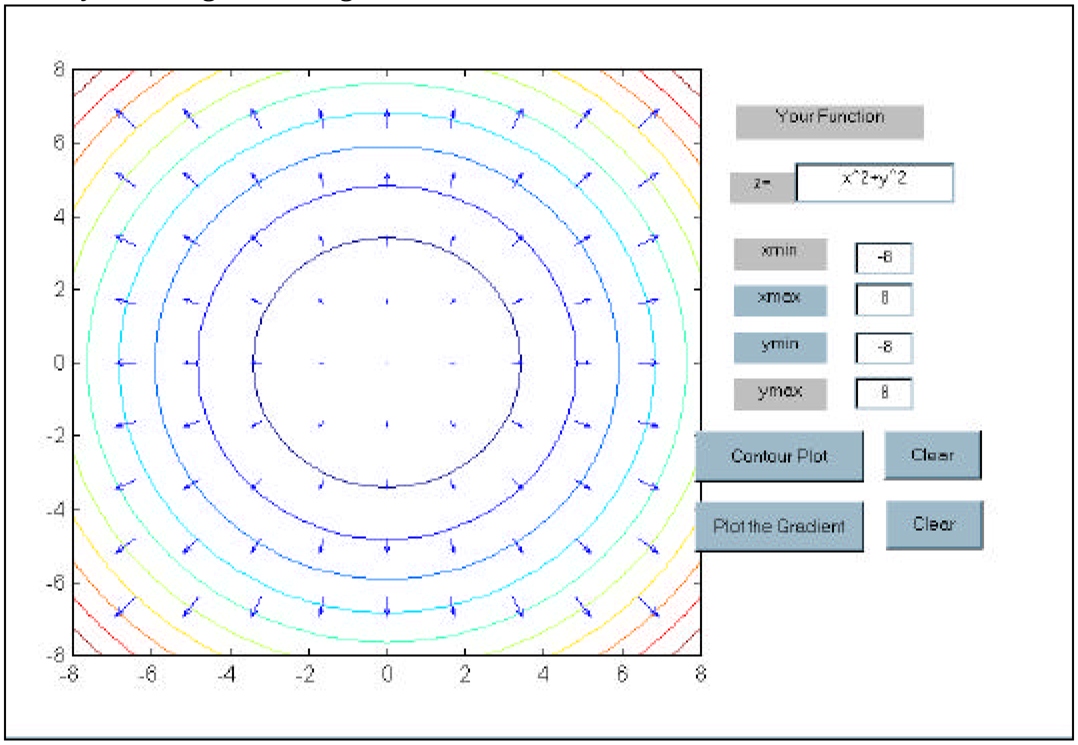
Your first function
z= 25-x^2-y^2

Your second function
z= 0

xmin -5
xmax 5
xscl .2
ymin -5
ymax 5
yscl .2

Figure 3 (using graph3d to illustrate volumes)

The second program, `congrad`, takes a function of two variables and over a specified domain plots contours of the function along with vectors indicating the gradient. Besides using this software to get the students comfortable with the idea of contour plots and the gradient, I also used it to investigate 1st and 2nd partial derivatives and motivate the idea behind Lagrange multipliers. Figure 4 below shows the contour plot of `z=x^2+y^2` along with the gradient field.



z= x^2+y^2

xmin -8

xmax 8

ymin -8

ymax 8

Figure 4 (Using congrad to illustrate contours and the gradient)

Of all the above-mentioned uses, I found `congrad` most helpful in constrained optimization problems. For example, if we wanted to minimize `z=x^2+x*y+y^2` subject to the constraint `x+y=1`, a contour plot of the region along with the gradient vectors (and the line added with the Matlab command `line([0 1],[1 0])`) helped the students see how the gradient of the function to be maximized matched up with the gradient of the constraint function. (See Figure 5.)

The last two programs, `pcurve2d` and `pcurve3d`, have the same purpose. Both can plot parametric curves and vector fields --- the first in two dimensions, the second in three dimensions with the resulting graphs being rotatable in real time. In addition, the vector and acceleration vectors can be plotted for the parametric curves. Besides using these to illustrate the basic concepts of parametric curves and vector fields, they were used later in the course to illustrate line integrals and estimate their signs. Figure 6 shows a parametrization of the circle of radius 1 along with velocity vectors and the vector field `F(x,y)=-y`**`i`**`+x`**`j`**. Figure 7 shows a helix in 3-space with associated velocity vectors and the vector field `F(`**`r`**`)=`**`r`**.

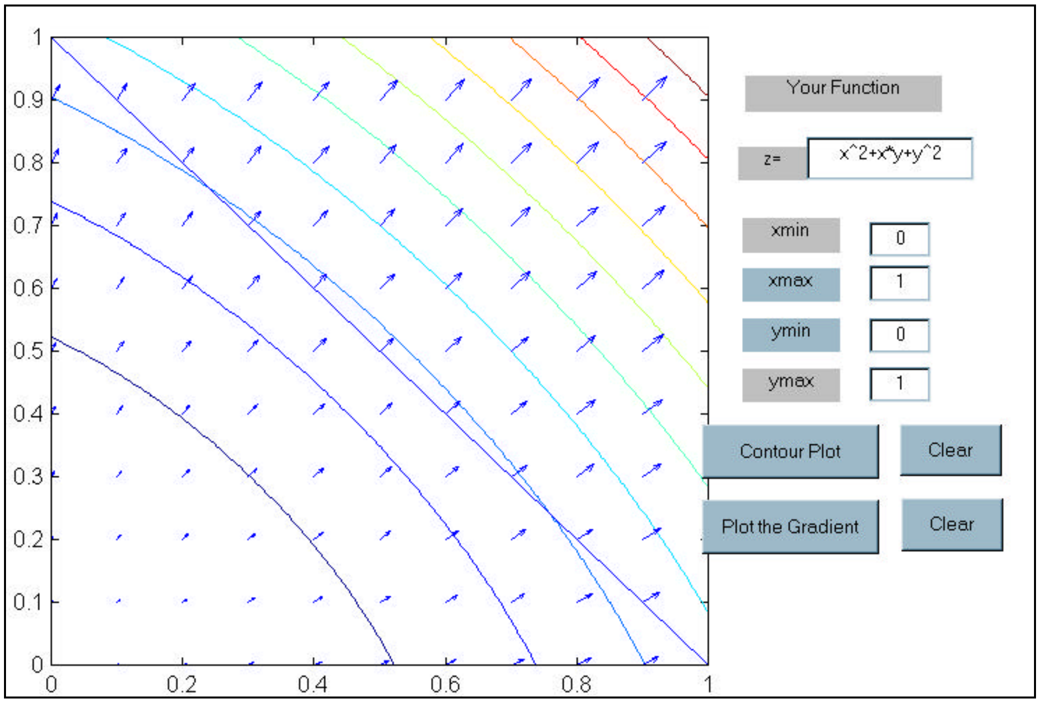


Figure 5 (Using congrad to help solve a Lagrange multipliers problem)

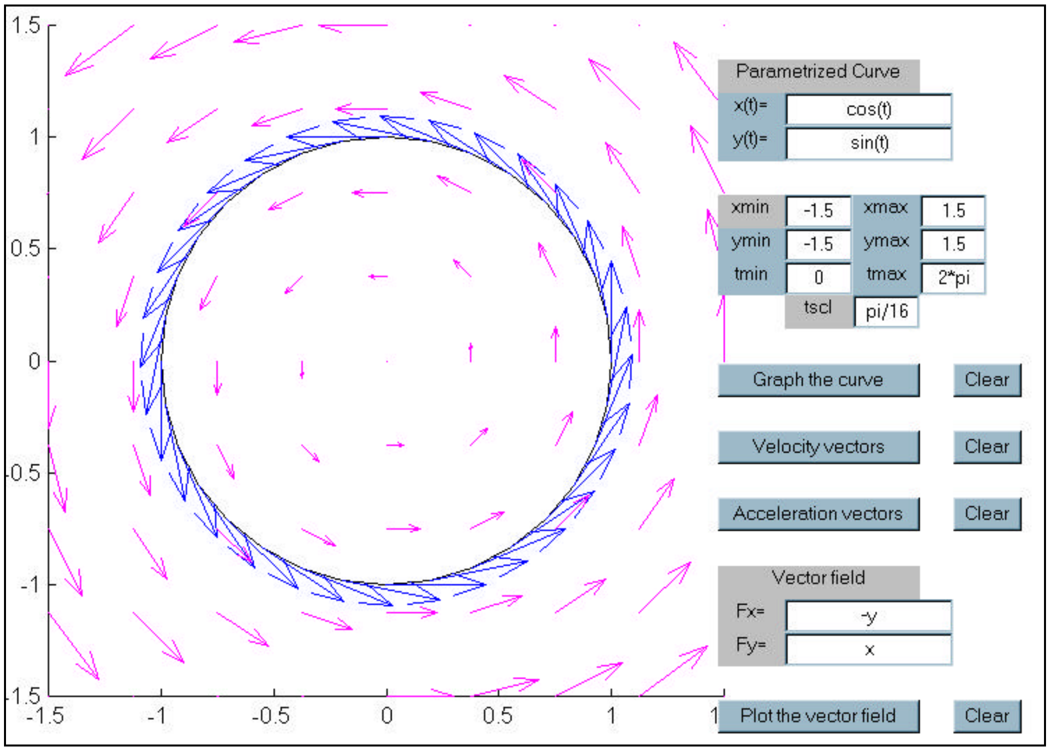
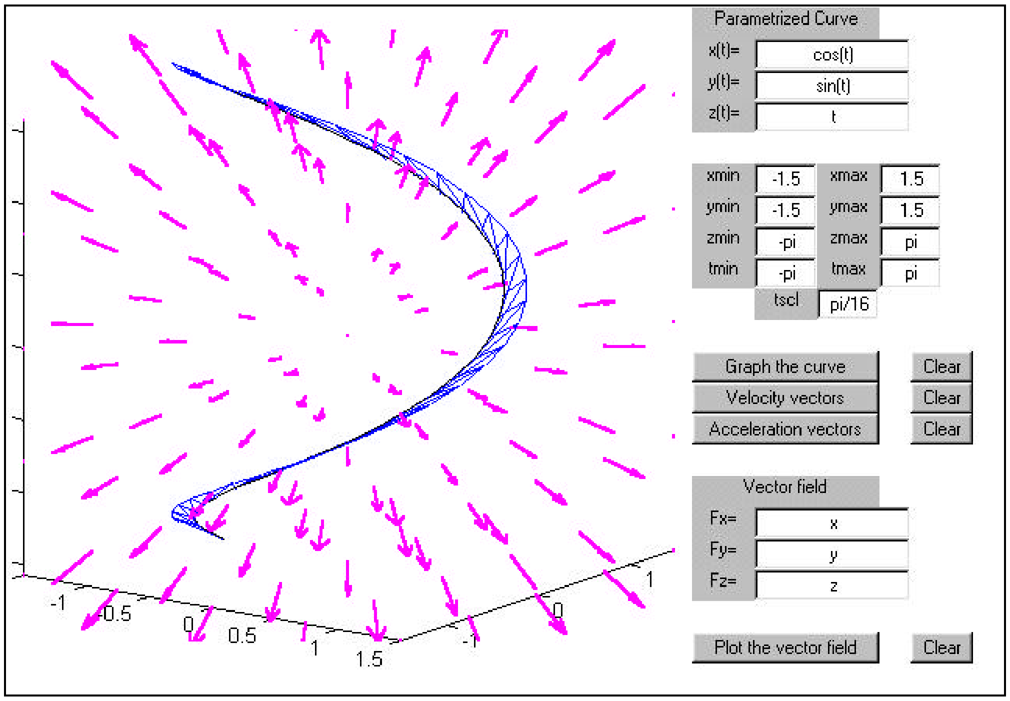


Figure 6 (Using pcurve2d to illustrate vector fields and parametric curves)



Parametrized Curve
x(t)= cos(t)
y(t)= sin(t)
z(t)= t

xmin -1.5 xmax 1.5
ymin -1.5 ymax 1.5
zmin -pi zmax pi
tmin -pi tmax pi
tscl pi/16

Vector field
Fx= x
Fy= y
Fz= z

Figure 7 (Using pcurve3d to illustrate vector fields and parametric curves)

Responses to the Developed Software

During fall 1998, the software was used in a class of 12 students which met in a computer lab to use the software about 40% of the time. During those times, I primarily taught in a lecture format using the software as a means of illustrating the concepts. Students were encouraged to use the software on homework and exams.

Response to the `graph3d` and `congrad` packages was positive. While students did not often use the packages on the homework (according to self-reporting and my observations), they used the packages on the exams quite a bit. I helped in this by labeling some exam questions as 'Matlab recommended'. Specifically, during exams I saw students

- use `graph3d` and `congrad` to plot a function near a point and estimate the signs of 1st and 2nd partial derivatives
- use `graph3d` to help visualize cross sections of three-dimensional surfaces, and
- use `congrad` to help see the solutions to Lagrange multiplier problems.

Due to the many difficulties students had with these latter problems, I was most pleased with the students' use of Matlab on them. After showing them a picture like Figure 4, I often saw students tracing the constraint curve with their finger over the contour plot on the screen to estimate solutions. Also, using Matlab, students found an error in my Lagrange multiplier exam problem (I asked for a maximum instead of a minimum) that I do not believe students without the technology would have noticed.

Responses to the `pcurve2d` and `pcurve3d` packages were positive, but not as enthusiastic. By my observations, uses of both packages on homework were mostly limited to printing out pictures of vector fields and sketching parametric curves. There was some use on exams (on the same types of tasks), but not to the level of the previous packages. This is primarily due to the fact that after we did line integrals, I did not utilize the software for looking at flux integrals --- it was simply easier in most lectures to draw a picture rather than develop Matlab pictures of a surface and a vector field.

Using Matlab's CAS capabilities

Our version of Matlab includes the Symbolic Toolbox, an add-on that allows the user to do most of the CAS operations found in Maple. During the fall 1998 class, we primarily used the CAS capabilities of Matlab for one purpose: symbolic integration. After students had learned the basics of iterated integrals, I showed them how to do integration in Matlab. The purpose of this was to focus attention on the setting up of integrals (a difficult task for students learning cylindrical and spherical coordinates) rather than on the evaluation.

For example, if a student wished find the average distance to the origin over the top half of a ball of radius 1 centered at the origin, in Matlab the user would type:

```
>> syms rho theta phi

>>int(int(int(rho*rho^2*sin(phi),rho,0,1),phi,0,pi/2),theta
,0,2*pi)

ans =
1/2*pi

>> ans/(2*pi/3)

ans =
3/4
```

Responses to Matlab's CAS capabilities

I did not use symbolic integration a great deal in class. Most of the time, I preferred to set up an integral and let students pick their methods of evaluation. Of those students who did use the symbolic integration (a few out of the class of twelve did not), it was heavily used on the homework in the latter half of the course. From student reports, the use of symbolic integration was the only tool from Matlab used after the course (except for use in later math courses).

As expected, the main advantage of using CAS was in allowing myself and the students more time to focus on setting up the integrals. I felt more able to cover a wider variety of integrals in class, and the students who used CAS could focus more time on finding the limits of integration.

The main disadvantage of using symbolic integration came near the end of the course. Students had grown so used to inputting integrals that when shortcuts like the divergence theorem were presented, students did not necessarily feel a need to use them. As an example, on my final exam I placed the following problem:

> Suppose $\mathbf{H} = x\mathbf{i} + y\mathbf{j}$.
>
> a) (7 points) Set up (but do not evaluate) an integral for the flux of $\mathbf{H}$ through the top half of a sphere of radius 2 centered at the origin oriented up.
>
> b) (7 points) Find the flux of $\mathbf{H}$ out through an entire sphere of radius 2 centered at the origin.

Four out of the twelve students, instead of using the divergence theorem on part b), simply altered the limits on their integral from part a) and entered the integral into Matlab. While those who got part a) correct got part b) correct, I lost an opportunity to assess these students on the divergence theorem.

Moving Forward

As mentioned above, during fall 1998, I taught the multivariable course with about 40% of the classes taking place in a computer lab. This semester (fall 1999) all classes are taking place in a lab setting, allowing for an increased use of the software. In addition, I have introduced the students to Matlab's ability to symbolically differentiate and compute cross and dot products with some use seen on homework. I also expect to spend more time during this semester carefully discussing how to wisely use symbolic integration (and more carefully designing exam questions). In the future, I plan to add one more package showing parametric surfaces together with vector fields in three dimensions to help students visualize flux integrals.

Overall, I have been very pleased with the use of these tools in a multivariable class. As an instructor, I have felt more able to show the students the basic concepts using visualization tools. I also believe the students, especially with the help of the symbolic integration capabilities, have been more able to handle difficult questions by focussing on setting up the problems correctly.

Notes on the Software

My Matlab packages are at http://personal.centenary.edu/~mschlat/teaching.html. Please note that you have to download both the *.mat and *.m files for the package to work. Some details and known bugs:

- The packages were written for Matlab 5.2.1.1420 and have not been tested (to my knowledge) on other versions.
- The figure sizes were chosen to work well on a 640x480-resolution screen.
- All the packages that plot vectors scale those vectors to fit nicely on the screen --- please note that vectors are usually scaled down from their true size. In some cases, vectors will not appear (for example, try plotting the gradient vectors for `z=x^(2/3)*y^(1/3)`).

- One minor bug: I have had problems printing figures, especially using pcurve2d or pcurve3d.
- One serious bug: all of the packages use the variables x, y, z. If students use x,y,z as symbolic variables, and then run one of these packages, they will have to clear x, y, z and then redefine them as symbolic variables to use them again symbolically.