Graphing Functions and their Derivatives

by Kathy McCabe and Mike Meehan

Kathy and Mike are seniors at Woburn High School, 88 Montvale Ave., Woburn, Massachusetts 01801. They can be reached in care of the Mathematics Department.

As you probably know, it is easy to find computer software that will graph functions, but we decided to invent our own. Why? Because most store-bought programs are limited in what they can do. With our program, we are able to define the functions that we want to have graphed and compare their graphs against the graphs of their derivatives. The following is a brief summation of what we have accomplished over the past fifteen weeks.

As you probably know, it is easy to find computer software that will graph functions, but we decided to invent our own. Why? Because most store-bought programs are limited in what they can do.

The need for an X-Y axis "setup" program was the first step in making a useful graphing procedure. First, we just used a trial and error procedure to determine the number of turtle movements that were needed to cross the computer screen both vertically and horizontally. It turned out that the horizontal distance, which would represent our X-axis, was five-hundred turtle movements in length. (We are using Object Logo on a Macintosh computer.) Because the vertical distance, which is also our Y-axis, is shorter than the horizontal it was not necessary to determine the actual length of it; when we set up our axis the extra length simply ended up off the screen. Upon discovering this, we instituted a procedure that could take a number *n*, the desired number of tick marks, and graph *n* tick marks with a distance of 500/*n* between them.

The actual graphing procedure was by far the most complicated in this project. First, we needed to define some simple functions that could be input to the graphing procedure. We enabled the computer to calculate the outputs for these functions over a specified interval and then move the turtle to the coordinate and place a dot. The program uses an increment of only .2 (although other increments can be inputted as options.) Even the .2 increment doesn't give the impression of a smooth curve, so we modified the program to "connect the dots."

```
TO GRAPH :FUNC :L :H [ :D .2]
IF :L > :H [PD STOP]
PD HT
SETPOS SE (250 / :SCALE) * :L (250 / :SCALE) *
(APPLY :FUNC :L)
(GRAPH :FUNC :L + :D :H :D)
END
```

This procedure seemed to work fine; however, we did find one major flaw in it. It seemed that the computer's turtle was leaving a line on its way to the first coordinate. This was a problem which would have no actual bearing on the graph itself, but did seem to make things more complicated when two or more graphs were drawn on the same axes. Thus came the need for the program to take this graph procedure and incorporate it into one which could erase such stray lines. This we called **M&M.** This program took the same inputs as the graphing one before it, but instead of leaving the line behind we instituted the small clause in it which would keep the turtle from writing any marks until it reached the first coordinate and evaluated it. We decided it would be helpful to design a program that would only make a small dot on the coordinates in the interval and leave a space between them. We called this **DOTGRAPH.**

```
TO DOTGRAPH :FUNC :H [ :D .1]
IF :L > :H [PD STOP]
HT
SETPOS SE (250/ :SCALE) * :L (250 / :SCALE) * (APPLY
:FUNC :L)
IF (ABS 250/ :SCALE) * (APPLY :FUNC :L) ) < 4095 [DOTT]
```

Note: *We needed the 4095 because of a bug in Logo.*

continued on next page