Programs for Graphing Calculators

From time to time, various problem sets call upon students to write or download programs for their graphing calculators. The programs listed and described in this *Instructor's Resource Book* are relatively simple versions of what students might do. They are specific to the TI-83 and TI-84 graphing calculators, but can be adapted to other graphers.

It is instructive to have your students write their own programs for specific purposes. If there is time, the writing of such programs can be excellent learning activities, and can support students' understanding of the calculus concepts the programs address. You will find that some of your students—not necessarily the ones who make the best grades—can quickly come up with remarkably insightful and useful programs! Often, however, you may want to download a program you have written or adapted from the ones that follow so that students' attention will not be distracted from the topic at hand. In this case it is recommended that you have students work "by hand" a simple example, such as a trapezoidal rule problem. The first thing they do after receiving the downloaded program is to work the same example and show that the answers agree. From that time on, they will have confidence in the program and can use it for more complicated examples, such as exploring the limit a Riemann sum seems to approach as the number of increments increases.

You are invited to explore the publisher's Web page, www.keypress.com, for updated information on graphing calculator programs. You are encouraged to share with other instructors, via the publisher, any highly effective programs you or your students have written. Contact Key Curriculum Press at editorial@keypress.com or 1-800-338-7638.

Here, in the order that they are first used in the text, are listings of the programs.

TRAPRULE, Problem Set 1-4, Problem 5 (pages 22–23)

This program evaluates the definite integral of a given function between lower and upper limits of integration using the trapezoidal rule with any desired number of increments. Before you run the program, store the function for the integrand as Y1. When you run the program, the grapher will prompt you to enter the lower and upper limits of integration, *A* and *B*, and the number of increments, *N*. The grapher then computes and sums the successive *y*-values, using half the first and half the last, and saves the sum as *S*. The grapher displays the approximate value of the integral, which it has saved internally as the variable *I*. Along the way the grapher displays the successive *y*-values to give students a sense of how rapidly it is calculating. You may want to omit this instruction so that the program will run faster, particularly for larger numbers of increments.

TRAPRULE (TI-83, TI-83+, and TI-84+)

```
:Prompt A
:Prompt B
:Prompt N
:A→X
:0→S
:(B-A)/N→D
:0→C
:Lbl 1
:Y1+S→S
:X+D→X
:Y1+S→S
:IS>(C,N-1)
:Goto 1
:SD/2
:Disp "INTEGRAL"
:Disp Ans
```

TRAPRULD, Problem Set 1-4, Problem 6 (page 23)

The name comes from "trapezoidal rule from data." This program evaluates an integral of a function for which *y*-values are given, assuming that the *x*-values are evenly spaced. Before running the program, store the given *y*-values in list L1. Upon running the program, the grapher will prompt you to enter the number of increments (which must be one less than the number of data points). Then it will prompt you to enter the width of each increment, *DX*. The grapher then runs the program and displays the approximate integral, which it has stored in its memory as *I*. Note that the *x*-values themselves appear nowhere in the program.

TRAPRULD (TI-83, TI-83+, and TI-84+)

```
:Disp "INCREMENTS"
:Input N
:Disp "DX"
:Input D
:(D/2)*(2*sum(L1)-L1(1)-L1(N+1))→I
:Disp I
```

NEWTON, Problem Set 4-10, Problem C1 (pages 182–183)

This program finds zeros of a given function. Before running the program, store the equation for the function in Y1. Upon running the program, the grapher pauses to ask for the initial value of *x*, giving the prompt "FIRST X." The grapher then displays the next approximation for *x*. Press ENTER repeatedly for successive iterations. The current value is stored as *X* in the grapher's memory. The program as presently written has no elegant way of getting out of the loop. So simply press ON when you are ready to exit the program. Then clear the resulting "error" message.

NEWTON (TI-83, TI-83+, and TI-84+)

```
:Disp "FIRST X"
:Input X
:Lbl 1
:X-Y₁/nDeriv(Y₁,X,X)
:Ans→X
:Disp "NEXT X IS"
:Disp X
:Pause
:Goto 1
```

RIEMANN, Problem Set 5-4, Problem 12 (page 210)

This program finds a Riemann sum for a definite integral of a given function between $x = A$ and $x = B$. Before running the program, store the integrand function in Y1. The grapher will prompt you to enter the limits of integration, *A* and *B*; the number of increments, *N*; and the percent of the way through each interval at which the sample points are to be taken, PCT. By entering 0 for PCT, the grapher finds a left Riemann sum. By entering 50, it finds a midpoint sum. By entering 100, it finds a right Riemann sum. At the end of the run, the grapher will display *S*, the sum of the *y*-values before multiplying by Δx, and the approximate value of the integral, *I*. The integral is stored as *I* in the grapher's memory.

RIEMANN (TI-83, TI-83+, and TI-84+)

```
:Prompt A
:Prompt B
:Prompt N
:Disp "PCT?"
:Input P
:(B-A)/N→D
:A+P/100*D→X
:0→S
:1→K
:Lbl 1
:Y₁+S→S
:Disp X
:K+1→K
:X+D→X
:If K≤N
:Goto 1
:S*D→I
:Disp S
:Disp I
```

SIMPSONE, Problem Set 5-10, Problem 16 (page 259)

The name comes from "Simpson's rule from equation." The program evaluates an integral of a function from $x = A$ to $x = B$, for which an equation is given. Before running the program, store the equation for the function as Y1. Upon running the program, the grapher will prompt you to enter the limits of integration, A and B, and the number of increments, N. The grapher runs the program and displays the approximate integral, which it has stored in its memory as I. The weighted sum of the data, before multiplying by $\Delta x/3$, is stored as S. Note also that the program, as currently written, does not protect against students entering an odd number of increments.

SIMPSONE (TI-83, TI-83+, and TI-84+)

```
:Disp "A"
:Input A
:Disp "B"
:Input B
:Disp "INCREMENTS"
:Input N
:(B-A)/N→D
:0→S
:2→K
:A+D→X
:Lbl 1
:Y1(X-D)+4Y1(X)+Y1(X+D)
:Ans+S→S
:X+2D→X
:K+2→K
:If K≤N
:Goto 1
:S/3*D→I
:Disp I
```

SIMPSOND, Problem Set 5-10, Problem 16 (page 259)

The name comes from "Simpson's rule from data." The program evaluates an integral of a function for which *y*-values are given, assuming that the *x*-values are regularly spaced. Before running the program, store the given *y*-values in list L1. Upon running the program, the grapher will prompt you to enter the number of increments (which must be one less than the number of data points). Then it will prompt you to enter the width of each increment, *DX*. The grapher runs the program and displays the approximate integral, which it has stored in its memory as *I*. The weighted sum of the data, before multiplying by $\Delta x/3$, is stored as *S*. Note that the *x*-values themselves appear nowhere in the program. Note also that the program, as currently written, does not protect against students entering an odd number of increments.

SIMPSOND (TI-83, TI-83+, and TI-84+)

```
:Disp "INCREMENTS"
:Input N
:Disp "DX"
:Input D
:0→S
:2→K
:Lbl 1
:L1(K-1)+4L1(K)+L1(K+1)
:Ans+S→S
:K+2→K
:If K≤N
:Goto 1
:S/3*D→I
:Disp I
```

SLOPEFLD, Problem Set 7-4, Problem 14 (page 343)

The program plots a slope field for a given differential equation. Before running the program, store the differential equation in Y_1 in terms of both *x* and *y*. Upon running the program, the grapher prompts you to select the window by asking, "Same as last?" If you choose to change the window, be sure to remember that the number of *x*- and *y*-values must take into account a beginning and an end data point. For instance, a window of [0, 10] would require 11 values if the slope lines were to be spaced 1 unit apart. The grapher also asks if you want to "square" the window, using the same scales on both axes. The grapher then proceeds to draw the slope field. Note that as presently written, the program does not correctly compensate for slope fields where there are different scales on the two axes.

SLOPEFLD (TI-83, TI-83+, and TI-84+)

```
:((Ymax-Ymin)/Yscl)→L
:((Xmax-Xmin)/Xscl)→W
:(W*(.5^(W/10))→G
:(L*(.5^(L/10))→A
:(Ymax-Ymin)/L→V
:(Xmax-Xmin)/W→H
:ClrDraw
:FnOff
:0→R
:Ymin→Y
:Lbl 1
:R+1→R
:0→C
:Xmin→X
:If Y=0:.0000001→Y
:Lbl 2
:If X=0:.0000001→X
:If Y=0:.0000001→Y
:C+1→C
:Y1→M
: -M*H/A+Y→S
:M*H/A+Y→T
:X-H/G→P
:X+H/G→Q
:If abs((T-S))>V
:Goto 3
:Lbl 4
:Y→Z
:Line(P,S,Q,T)
:Z→Y
:X+H→X
:If C-1<W
:Goto 2
:Y+V→Y
:If R-1<L
:Goto 1
:Stop
:Lbl 3
:Y+V/A→T
:Y-V/A→S
:(T-Y)/M+X→Q
:(S-Y)/M+X→P
:Goto 4
```

SLOPEDRW, Problem Set 7-4, Problem 14 (page 343)

This program, written by the author's student Robert "Trae" Sawyer in 1994, plots a particular solution for a given differential equation on a slope field that has already been drawn by the program SLOPEFLD, described above. The differential equation in terms of *x* and *y* should already be stored as Y_1 in the grapher. Upon running the program, the grapher prompts you to enter the initial condition and the direction (right or left) to begin. Then it asks you to enter a factor, which is used to determine the step size in Euler's method of solving the differential equation. Entering 3 for the factor gives reasonable accuracy. The grapher will draw successive steps, including cyclical paths, until you stop it by pressing the ON key, or until the curve goes off-screen.

SLOPEDRW (TI-83, TI-83+, and TI-84+)

```
:ClrHome
:FnOff 1
:Disp "   SLOPEFIELD","DRAWER",""
:Input "INITAL X? ",X
:Input "INITAL Y? ",Y
:Disp "START GOING TO","THE (0=LEFT OR"
:Input "1=RIGHT?= ",L
:Input "FACTOR?=",F
:F*(Xmax-Xmin)/94→S
:F*(Ymax-Ymin)/62→T
:While ((X≥Xmin) and (X≤Xmax) and (Y≥Ymin)
and (Y≤Ymax))
:((abs(Y1)≤1)→A
:((abs(Y1)≠Y1)→B
:(abs((L-1))→C
:If B:Then
:If C:Then
: -1→U:1→V
:Else
:1→U: -1→V
:End
:Else
:If C:Then
: -1→U: -1→V
:Else
:1→U:1→V
:End:End
:If A:Then
:Line(X,Y,X+U*S,Y+V*T*(abs(Y1))
:Y1→P:(X+U*S)→X
:(Y+V*T*(abs (Y1))→Y:Y1→Q
:Else
:Line(X,Y,X+U*S/(abs (Y1)),Y+V*T)
:Y1→P:(X+U*S/(abs (Y1))→X
:(Y+V*T)→Y:Y1→Q
:End
:If (((abs (P)=P) and ((abs (P)>1) and ((abs (Q)≠Q)
and ((abs (Q))>1)):Then
:(abs ((L-1))→L
:End:End
```

EULER, Problem Set 7-5, Problem 6 (page 348)

The program calculates successive values of *y* for a given differential equation. Before running the program, store the differential equation in Y1 in terms of both *x* and *y*. Upon running the program, the grapher will ask you to enter the initial condition and increment size by asking for First *x*, First *y*, and Delta *x*. The grapher then displays the next *x*-value and the corresponding (approximate) *y*-value. To calculate subsequent values, press ENTER repeatedly. The program as presently written has no elegant way of getting out of the loop. So simply press ON when you are ready to exit the program. Then clear the resulting "error" message.

EULER (TI-83, TI-83+, and TI-84+)

```
:Disp "FIRST X"
:Input X
:Disp "FIRST Y"
:Input Y
:Disp "DELTA X"
:Input D
:Lbl 1
:Y+Y1D→Y
:X+D→X
:Disp "X"
:Disp X
:Disp "Y"
:Disp Y
:Pause
:Goto 1
```

ARCSUM, Problem Set 8-5, Problem 33 (page 407)

This program calculates the approximate arc length of a plane curve by summing the lengths of the chords for any chosen partition. Before running the program, store the equation for the curve in Y1. Upon running the program, the grapher will ask you to enter *A* (the lower limit of integration), *B* (the upper limit of integration), and *N* (the number of increments). The grapher then sums the chord lengths and displays the sum. The answer is also stored as *S* in the grapher's memory.

ARCSUM (TI-83, TI-83+, and TI-84+)

```
:Disp "A?"
:Input A
:Disp "B?"
:Input B
:Disp "N?"
:Input N
:(B-A)/N→D
:A→X
:0→S
:1→K
:Lbl 1
:Y1→V
:X+D→X
:S+√((Y1-V)²+D²)→S
:IS>(K,N)
:Goto 1
:Disp "SUM"
:Disp S
```

ACVELDIS, Section 10-2, Example 2 (pages 504–505)

The name comes from ACceleration, VELocity, DISplacement. The program, written by the author's student Patrick "P. J." Stevens in 1995, calculates numerically the approximate displacement of a moving object, using a table of time and acceleration data. Before running the program, store the times in list L1 and the corresponding accelerations in L2. Upon running the program, the grapher will pause for you to enter the number of data points and the initial velocity. The program then computes the displacements under the assumption that the acceleration varies linearly throughout each time interval. The displacements are obtained from the ordinary physics formula $s = v_0t + 0.5at^2$. (The same result could be computed by averaging the initial and final velocities in each time interval and multiplying by the time.) The results are stored in other lists. The grapher will direct you to the appropriate lists, as follows:

L3 has the average acceleration for the time interval.
L4 has velocity at the time in L1.
L5 has the total displacement at the time in L1.

ACVELDIS (TI-83, TI-83+, and TI-84+)

```
:Disp "NO. DATA PTS."
:Input N
:N→dim(L3)
:0→L3(1)
:N→dim(L4)
:0→L4(1)
:N→dim(L5)
:0→L5(1)
:Disp "INIT. V"
:Input V
:V→L4(1)
:For(A,2,N,1)
:L1(A)-L1(A-1)→T
:(L2(A-1)+L2(A))/2→L3(A)
:L3(A)*T+L4(A-1)→L4(A)
:.5L3(A)*(T²)+L4(A-1)*T+L5(A-1)→L5(A)
:End
:Disp "PRESS STAT."
:Disp "L1: TIME"
:Disp "L2: ACCELERATION"
:Disp "L3: AV. ACCEL."
:Disp "L4: VELOCITY"
:Disp "L5: DISPLACEMENT"
```

SERIES, Section 12-2 (page 590)

This program calculates a partial sum of a series where the formula for the *x*th term is given. Before running the program, store the formula for the *x*th term in Y1. Upon running the program, the grapher prompts you to enter the final value, *N*, of the term index and the initial value of the term index, which it stores as *A*. Then it calculates and displays each partial sum so that students can see what is happening as *x* increases. The display is particularly effective for a convergent series if you set the grapher to show a fixed number of decimal places. As more and more terms are displayed, more and more decimal places remain constant. Thus, students develop a practical sense of what it means for a series to converge.

SERIES (TI-83, TI-83+, and TI-84+)

```
:0→S
:Disp "HOW MANY TERMS?"
:Prompt N
:For(X,1,N)
:S+Y1(X)→S
:Disp S
:End
:Disp "NO. OF TERMS ="
:Disp N
:Disp "PARTIAL SUM ="
:Disp S
```