Numerical Integration

Trapezoid Rule and Simpson's Rule

TI Calculator Program

The program steps are listed below with commentary on their function. Before you run the program, you must have the function to be integrated entered as y1= as you would a function to be graphed. (Program written for the TI-86, but modified for the TI-83/84 series.)

You must enter a function to be integrated in the graphing section as the first function, y1, just as if you were going to graph it. (Use the [Y=] key.) The programs asks you to input the left bound and right bound of the area estimate and to select the number of steps to use in the estimation process. Note the number of steps must be even for Simpson's Rule to work correctly.

ClrHome

A command in the catalog to clear the screen.

Input "Left Bound: ",L
Input "Right Bound: ",R
Input "Steps: ",N

When the program runs, words in quotes are displayed on the screen and the calculator waits for the user to enter a value which is stored in the variable named in the command. The words inside the quotes are optional, *but* if you don't put something in, the calculator just shows a blinking question mark, and *you* have to remember what it wants to see. These commands request the left and right limits of integration, then the number of intervals (steps). The author did not add a check for an even number here which is required for Simpson's Rule.

L→x

x is the variable key, [X,T,θ,n], and → is the [STO] key. Left bound is stored in variable x as a starting value.

0→T
0→S

0 is the number 0. These steps put zero into variables T and S as starting values. T and S are the Trapezoid function value sum and a partial Simpson function sum (explained later) which will be done as part of a loop.

(R-L)/N→D

Right bound minus left bound divided by the number of steps is stored in D. This value is Δx.

For(I,0,N-1,1)

Start the loop to calculate the summation in the Trapezoid Rule. The program will repeat the steps below down to the command, End. The number of repetitions is counted by the variable *I* (first item in the list inside the parentheses), starting at 0 (second entry in the list), and adding the value of the fourth item in the list (here it is 1) each time the program bounces back to the For command to start the loop over. Repeating will continue until the variable, *i*, becomes larger than N-1 (third entry in the list).

L+I*D→ x

Takes the left bound, L, and adds the loop counter, I, times Δx (D in the program) and stores it as the value of x, x, used to calculate the value of the function, y1, in the next step. We are stepping through the x values, Δx by Δx, to calculate the function values we need.

y1+T→T

Enter y1 by using the [VARS] key and selecting [Y-VARS], [Function], [y1].The Adds the function value $f(x)$ (function f defined by y1) to the Trapezoid total (T) and replaces it in the Trapezoid Total (T). The variable, T, is a running total of the function values.

x+D→x

Increments the value of x by Δx, and places it in x. This is clever as explained in the next step.

y1+T→T

Adds the function value $f(x)$ (the function defined at y1), but now computed with $(x+\Delta x)$ to the Trapezoid total (T) and replaces it in the Trapezoid Total (T). This is the second time in the loop this is done. Since the counter, *i*, is not changed, the next time through the loop, this calculation will be repeated in the first addition. As a result, the loop calculates the running sum two values at a time:

$$\left(f(x_0)+f(x_1)\right)+\left(f(x_1)+f(x_2)\right)+\left(f(x_2)+f(x_3)\right)+\cdots+\left(f(x_{n-1})+f(x_n)\right)$$

or: $\left[f(x_0)+2\cdot f(x_1)+2\cdot f(x_2)+\cdots+2\cdot f(x_{n-1})+f(x_n)\right]$.

This is a clever way of getting exactly the correct Trapezoid Rule coefficients on each of the function values without having to do a special accounting for the first and last terms.

End

Once the counter, *i*, hits the value N-1, the program falls out of the loop and performs the next steps. Notice it still hasn't done the multiplication by $\frac{R-L}{2N}$ to finish the Trapezoid Rule calculation. The author waits, with good reason, until the end.

Now the program starts doing additional calculations for Simpson's Rule.

L→x

Variable *x* is once again set to the left bound.

For(I,1,N,2)

Start of the loop for Simpson's Rule. The counter, i, is set to 1, the increment for i in this loop is 2. The program will go through the loop counting by 2 until *i* becomes larger than N.

L+I*D→x

Value for *x* is set by adding the step number times Δx to the left bound. Since the loop is stepping by two, this will produce the values x_1, x_3, x_5, etc.

2y1+S→S

Program multiplies two times the value of the function at step i and adds it to S. That sum is stored in S. Again, since it is stepping by two, this is producing the sum: $2 \cdot f(x_1) + 2 \cdot f(x_3) + 2 \cdot f(x_5) + \cdots + 2 \cdot f(x_{n-1})$. This is not the Simpson Rule sum by itself—the coefficients are in the pattern 0, 2, 0, 2, 0, etc. This sum, however, when added to the Trapezoid sum (coefficients 1, 2, 2, 2, ... 2, 1) computed above, will produce coefficients in the 1, 4, 2, 4, 2, 4, ... 4, 1 pattern for the Simpson sum. Again this is some programming cleverness to do the calculations necessary without having to keep track of the proper coefficient for each step.

End

End of Simpson's Rule calculation loop.

(D/2)T→U

Now $\frac{R-L}{2N}$ is multiplied times the sum of the function values for the Trapezoid rule. The result is stored in the variable U (our calculated trapezoid area) because the value T is still needed.

T+S→S

T, the sum of the Trapezoid Rule function values, is added to S, the sum of the extra function values needed to produce the sum of the function values in Simpson's Rule (as explained above). The result is stored in S.

(D/3)S→R

The factor for Simpson's Rule, $\frac{R-L}{2N}$, is multiplied times the sum of the function values and stored in R, Simpson's Rule Area.

Disp "Trapezoid: ", U
Disp "Simpson: ",R

Prints the words in quotes on the display screen followed by the calculation results.