EXPLORATIONS

CHAPTER 6

Integral Calculus

In this chapter we will use various features of the TI-86 to investigate some of the fundamental concepts of the integral calculus. Topics discussed in this chapter include:

(a) The computation of Riemann sums using the **sum** and **seq** commands.

(b) The numerical approximation of Riemann integrals using the **fnInt** command.

(c) The trapezoid, midpoint, and Simpson approximation techniques.

(d) A probabilistic interpretation of the average value of a function.

(e) Graphical verification that $\frac{d}{dx}\left(\int_a^x f(t)dt\right) = f(x)$.

§1 – The seq and sum Commands

The **seq** and **sum** commands give an easy way to calculate Riemann sums on the calculator. The two commands are accessed via the MATH MISC menu (press [2nd] [MATH] ⟨**MISC**⟩). The **seq** command has the syntax

seq(*expression, variablename, begin, end, increment*).

It returns a real list in which the elements are the values of the expression evaluated at successive incremented values of the variable starting from *begin* and ending at the last incremented value of the variable that is no greater than *end*.

For example,

seq(2*x*+1, *x*, 3, 5, 0.5) and **seq**(2*x*+1, *x*, 3, 5.25, 0.5)

both return the list {7, 8, 9, 10, 11} as shown in (6.1.1).

```
seq(2 x+1,x,3,5,0.5)
          {7 8 9 10 11}
seq(2x+1,x,3,5.25,0.5
)
          {7 8 9 10 11}
```

(6.1.1)

Due to the fact that the machine only does approximate work, the *end* value of the **seq** command should be taken to be *half an increment greater than* the desired true ending value when the *begin*, *end*, and *increment* values are not integer values. Otherwise, your list may be prematurely truncated.

1. As an illustration, **seq**(3x, x, 1, 2+1/3, 2/3) returns {3, 5} instead of the desired {3, 5, 7}, whereas **seq**(3x, x, 1, 2+2/3, 2/3) returns {3, 5, 7} as expected. See (6.1.2).

```
seq(3x,x,1,2+1/3,2/3)
                    {3 5}
seq(3x,x,1,2+2/3,2/3)
                  {3 5 7}
```

(6.1.2)

2. The **sum** command sums the elements of a list. Thus

   sum seq(3x, x, 1, 2+2/3, 2/3)

   returns 15 as shown in (6.1.3) since we are summing the sequence {3, 5, 7}.

```
sum seq(3x,x,1,2+2/3,
2/3)
                     15
```

(6.1.3)

§2 – Riemann Sums

We now apply the **sum** and **seq** commands to calculate Riemann sums for a function f over the interval $[a, b]$. Using standard notation, we take $\Delta x = (b-a)/n$ and $x_i = a + i\Delta x$ for $i = 0, 1, 2, \ldots, n$. Then the left, right, and midpoint Riemann sums, denoted by LS(n), RS(n), MID(n) are given by

$$\Delta x\big(f(z_1) + f(z_2) + \ldots + f(z_n)\big)$$

where $z_i = x_{i-1}$ for LS(n), $z_i = x_i$ for RS(n), and $z_i = (x_{i-1} + x_i)/2$ for MID(n). Consequently,

(Δx)**sum seq**($f(x)$, x, a, b – Δx/2, Δx),

(Δx)**sum seq**($f(x)$, x, a + Δx, b + Δx/2, Δx),

(Δx)**sum seq**($f(x)$, x, a + Δx/2, b, Δx)

will be the syntax of the computations for LS(n), RS(n), and MID(n), respectively. *Note that in each of the three arguments, we have added half an increment to the theoretical end value.*

As a specific example, for $f(x) = \sqrt{x^3 + 1}$ over the interval [1, 4], we see in (6.2.1), (6.2.2), and (6.2.3) that

LS(113) = 12.7833125814,

RS(113) = 12.9598093297, and

MID(113) = 12.8713921341.

```
(3/113)sum seq(√(x^3+
1),x,1,4-3/226,3/113)
→LS
          12.7833125814
```

(6.2.1)

```
(3/113)sum seq(√(x^3+
1),x,1+3/113,4+3/226,
3/113)→RS
          12.9598093297
```

(6.2.2)

Note also in (6.2.1)–(6.2.3) that we have stored the three Riemann sums in the variables LS, RS, and MID.

Incidentally, if we use the theoretical end values 4 – 3/113, 4, and 4 – 3/226 for LS(113), RS(113), and MID(113) the TI-86 prematurely truncates the sums and gives 12.5713648609, 12.7457670886 and 12.6583980656, respectively.

```
(3/113)sum seq(√(x^3+
1),x,1+3/226,4,3/113)
→MID
          12.8713921341
```

(6.2.3)

When computing and storing the left, right, and midpoint Riemann sums as in (6.2.1)–(6.2.3), it is advantageous to use the ENTRY feature of the calculator to recall a previous command line. The recalled command line can then be edited. For example, with the display as in (6.2.1), press [CLEAR] to clear the display and then [2nd] [ENTRY] to retype the command line in (6.2.1). Then edit the command line with the aid of the arrow keys, and the delete/insert key to obtain the command line in (6.2.2). Finally, press [ENTER] to compute and store the right Riemann sum.

The size of a list that the TI-86 can sum is limited by the available free memory. With 96500 bytes of free memory you can sum a list of about 9600 terms. With 45500 bytes of free memory you can a sum a list of about 4500 terms.

§3 – Trapezoid and Simpson Approximations

The Trapezoidal and Simpson approximations with n subdivisions are given by

$$\text{TRAP}(n) = (\text{LS}(n) + \text{RS}(n))/2$$

$$\text{SIMP}(n) = (2\text{MID}(n) + \text{TRAP}(n))/3.$$

In (6.3.1) we see that TRAP(113) = 12.8715609556 and SIMP(113) = 12.8714484079 for $f(x) = \sqrt{x^3 + 1}$ over the interval [1, 4].

```
(LS+RS)/2→TRAP
          12.8715609556
(2MID+TRAP)/3
          12.8714484079
```

(6.3.1)

§4 – The NINT Program

In (6.4.1)–(6.4.4) we use the NINT program given in Appendix A of this book to verify the results in Sections 2 and 3.

```
 Plot1 Plot2 Plot3
\y1■√(x^3+1)


 Y(X)= WIND ZOOM TRACE GRAPH
  X    Y  INSF DELF SELCT▶
```

(6.4.1)

Readers are encouraged to use the methods found in Sections 2 and 3 rather than the program found in Appendix A. The rationale is that actually constructing the Riemann sums will greatly enhance the understanding of this important concept.

```
NINT
a=1
b=4
n=113
```

(6.4.2)

```
n=113
LEFT=
          12.7833125814
RIGHT=
          12.9598093297
TRAPEZOID=
          12.8715609556
```

(6.4.3)

```
TRAPEZOID=
          12.8715609556
MIDPOINT=
          12.8713921341
SIMPSON=
          12.8714484079
                  Done
```

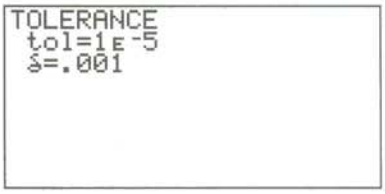
(6.4.4)

§5 – The fnInt Command

The **fnInt** command finds a numerical approximation for a definite integral. The command is accessed via the CALC menu (press [2nd] [CALC]). It has the syntax

fnInt(*expression, variable name, lower limit, upper limit*).

1. The accuracy of the approximation is controlled by the **tol** variable, which is found by pressing [2nd] [MEM] ⟨**TOL**⟩. See (6.5.1). The default **tol** setting is 1×10^{-5}.



(6.5.1)

For most integration purposes, the default setting is a good compromise. A smaller value for **tol** will give a better approximation but will require a longer time for the TI-86 to produce the result. A larger value will require less time but will give less accurate results. Any positive number greater than or equal to 1×10^{-12} can be used as the **tol** variable setting. Unless stated otherwise, we assume the **tol** setting is 1×10^{-5}.

2. In (6.5.2) we obtain approximations for

$$\int_1^2 \frac{1}{x}\,dx, \text{ and } \left(\frac{1}{\sqrt{2\pi}}\right)\int_0^1 e^{-x^2/2}\,dx.$$

In addition to computing the value of the definite integral, the **fnInt** command generates a value **fnIntErr** that indicates the possible integration error. After computing the value of an integral using **fnInt**, to see the value of **fnIntErr** press [2nd] [CATLG-VARS] ⟨**ALL**⟩ [2nd] [alpha] **[F]** [▼] [▼] [ENTER] [ENTER].

(6.5.2)

Note: *We find that accessing the* **fnIntErr** *variable is usually more trouble than it is worth. Its value will always be a positive or negative value whose magnitude is less than* **tol**.

§6 – A Shortcut

Keying in the sequence of steps necessary to use **fnInt** to compute an integral can become a tedious exercise if you have more than a few integrals to calculate. The following method can be used to expedite the process.

Step (1).	Enter the integrand as $y1$ in the ⟨**y(x)=**⟩ graph editor. Enter **fnInt**($y1$, x, A, B) as $y2$ in the ⟨**y(x)=**⟩ graph editor.
Step (2).	Return to the home screen (press [2nd] [QUIT]) and use the [STO▸] key to store the desired lower limit in the variable A and the desired upper limit in the variable B. Then press [2nd] [alpha] **y** [2] [ENTER] to compute the integral.

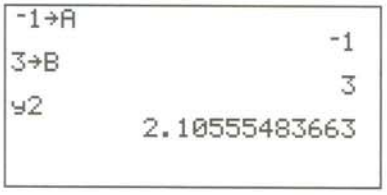
1a. As an illustration, we use the shortcut method to show that

$$\int_{-1}^{3} e^{-x^2/2}\,dx = 2.10555483663, \text{ and}$$

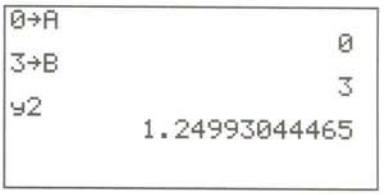
$$\int_{0}^{3} e^{-x^2/2}\,dx = 1.24993044465.$$

Step (1) for computing the two integrals is shown in (6.6.1). Step (2) is shown in (6.6.2) and (6.6.3).

(6.6.1)



(6.6.2)



(6.6.3)

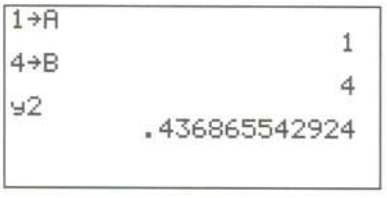
2a. For a second illustration, we show that

$$\int_{1}^{4} \sin\left(x^2\right)dx = .436865542924, \text{ and}$$

$$\int_{0}^{2} \sin\left(x^2\right)dx = .804776489344.$$

Step (1) for computing the two integrals is shown in (6.6.4). Step (2) is shown in (6.6.5) and (6.6.6).

(6.6.4)



(6.6.5)

(6.6.6)

§7 – Average Value of a Function

The simple program AVGF listed on the right can be used to give a probabilistic justification for the reason why

$$\left(\frac{1}{b-a}\right)\int_a^b f(x)dx$$

is called the average value of the function f over the interval $[a, b]$. The program uses the **rand** command to randomly select n points $x_1, x_2, \ldots, x_n$ in the interval $[a, b]$ and then it computes the average

$$AVG = (f(x_1) + f(x_2) + \ldots + f(x_n))/n.$$

A few comments regarding the **rand** command are in order. The command is found in the PROB submenu of the MATH menu (press [2nd] [MATH] ⟨**PROB**⟩). The command generates and returns a random number x in the interval (0, 1). To reseed this random number generator, store an integer value to **rand**. For example, **0 → rand**, **–2 → rand**, and **5 → rand** will each reseed the random number generator. (**0 → rand** gives the factory-set seed value.) We suggest you occasionally reseed when using the AVGF program.

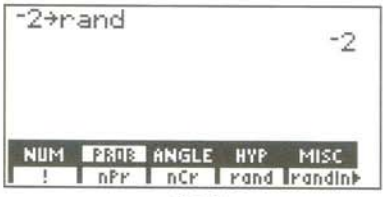
```
                AVGF
:Input "a=",A
:Input "b=",B
:Input "n=",N
:0→SM
:For(K,1,N,1)
:rand→R
:(B-A)R+A→x
:y1+SM→SM
:End
:SM/N→AVG
:Disp "AVG=",AVG
```

The program is run the same way the NINT program in Appendix A is run. In (6.7.1)–(6.7.9) below we run the program numerous times for the case $f(x) = 1/x$, $a = 1$, $b = 4$. Since

$$\left(\frac{1}{3}\right)\int_1^4 \frac{1}{x}dx = (\ln 4)/3 = 0.462098120373,$$

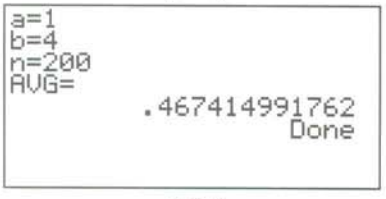
the data contained in (6.7.1)–(6.7.9) is consistent with what we expect to see.

(6.7.1)

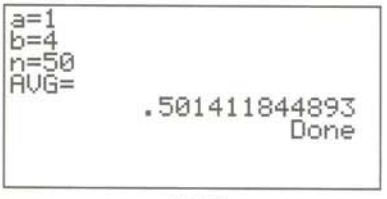


(6.7.2)

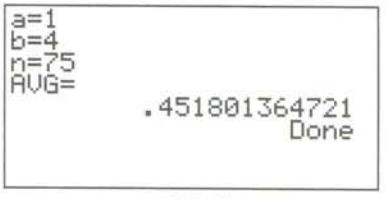
(6.7.3)



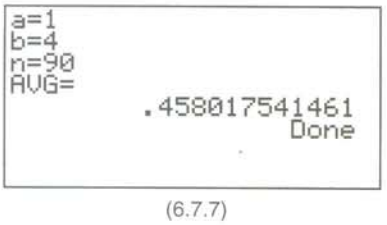
(6.7.4)



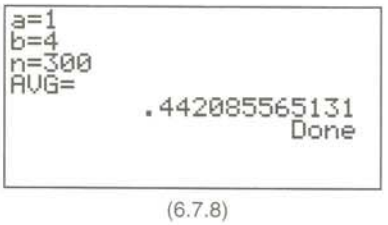
(6.7.5)



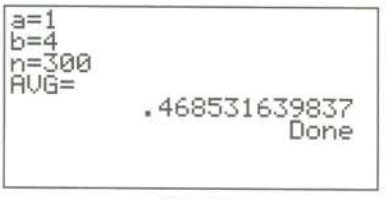
(6.7.6)



(6.7.7)



(6.7.8)



(6.7.9)

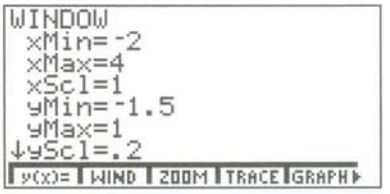
§8 – Graphing Integrals with Variable Upper Limit

One of the really nice features of the TI-86 is its ability to graph functions defined by integrals.

1. As an example, let's graph

$$F(x) = \int_1^x \sin(t^2)\,dt$$

in the viewing window $[-2, 4, 1] \times [-1.5, 1, 0.2]$ with **xRes** = 3. We first set the WINDOW values (see (6.8.1)) and then enter $F(x)$ as $y1$ in the ⟨**y(x)=**⟩ editor.



(6.8.1)

2. The obvious way to enter $F(x)$ is to enter

fnInt(sin(t^2), t, 1, x)

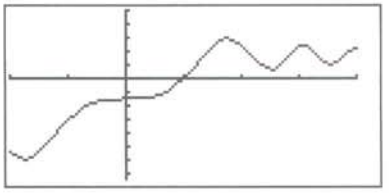
for $y1$. However, entering

fnInt(sin(x^2), x, 1, x)

for $y1$ as we did in (6.8.2) is also permissible, *i.e.*, the TI-86 can distinguish when x is used as a dummy integration variable and when x is used as an upper limit variable. This is convenient since it is much easier to enter x using the [x-VAR] key than it is to enter t by pressing [2nd] [alpha] **t**.

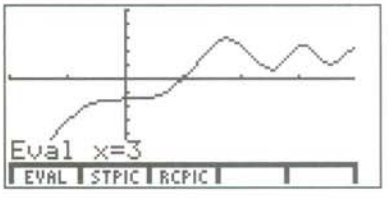
(6.8.2)

3. In (6.8.3) we see the graph of $F(x)$ in the indicated viewing window when **xRes** = 3 and **tol** = 1×10^{-5}. Be patient. It takes about 2.5 minutes for the graph to be completed. (With **xRes** = 1 and **tol** = 1×10^{-5} it takes about 7 minutes with negligible improvement in graph resolution.)



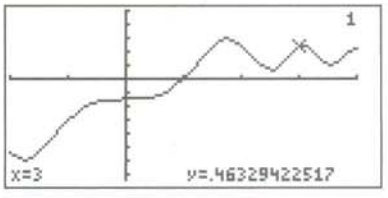
(6.8.3)

4. Now that we have the graph of $F(x)$, we can use ⟨**EVAL**⟩ to evaluate $F(x)$ for specific values of x in $[-2, 4]$. (To access ⟨**EVAL**⟩ press [MORE] twice when you are in the GRAPH menu.)



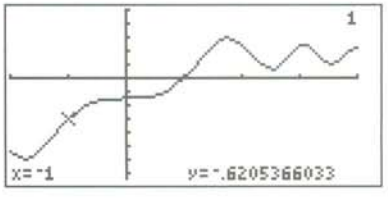
(6.8.4)

5. To find $F(3)$, select ⟨**EVAL**⟩, then press [3] [ENTER]. See (6.8.4) and (6.8.5).



(6.8.5)

6. With the display as in (6.8.5) type in –1, and press [ENTER] to evaluate $F(-1)$. See (6.8.6).

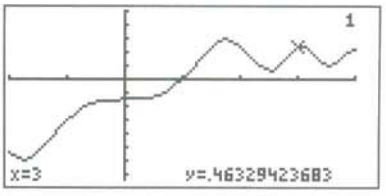


(6.8.6)

7. For graphing $F(x)$ in the viewing window $[-2, 4, 1] \times [-1.5, 1, 0.2]$ with **xRes** = 3, it would certainly have been sufficient to take **tol** = 0.01. If you set **tol** to this value and regraph, you will find that it takes the TI-86 about a minute to complete the graph.

(6.8.7)

8. In (6.8.7), and (6.8.8) we see that for **tol** = 0.01, the TI-86 computes $F(3) = 0.46329423683$, and that the graphs of $F(x)$ for **tol** = 1×10^{-5} and **tol** = 0.01 are visually identical.



(6.8.8)

§9 – The $\frac{d}{dx}\left(\int_a^x f(t)dt\right) = f(x)$ Result

In this section we obtain graphical verification that

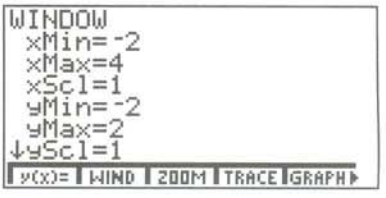
$$\frac{d}{dx}\left(\int_a^x f(t)dt\right) = f(x)$$

for the special case $f(x) = \sin(x^2)$, $a = 1$. We accomplish this by graphing both

$$y1 = \frac{d}{dx}\left(\int_1^x \sin(t^2)\,dt\right) \text{ and } y2 = \sin(x^2)$$

in the same $[-2, 4, 1] \times [-2, 2, 1]$ viewing window.

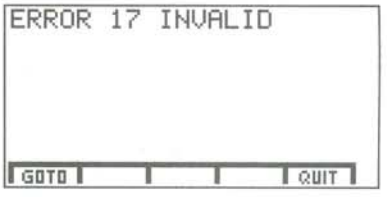
1. If we set **tol** = 0.01 as in (6.8.7), set the viewing window as in (6.9.1), enter $y1$ as



(6.9.1)

$$\textbf{der1(fnInt(}\sin(x^2), x, 1, x\textbf{)}, x\textbf{)}$$

and attempt to graph $y1$, we obtain the error message shown in (6.9.2). This message is to be expected. Recall that **der1** uses differentiation rules to compute the derivatives for special functions for which the TI-86 knows the functional form of the derivative. The TI-86 does not recognize



(6.9.2)

$$\textbf{fnInt(}\sin(x^2), x, 1, x\textbf{)}$$

as one of these functions. However, the numerical derivative command **nDer** does not have this limitation. Recall that the numerical derivative of f at x is given by

$$\frac{f(x+\delta) - f(x-\delta)}{2\delta}$$

where δ is accessed by pressing 2nd [MEM] ⟨**TOL**⟩.

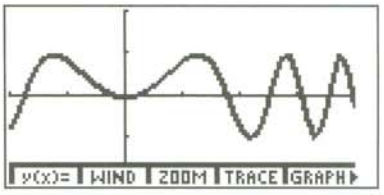
2. We will use the default value 0.001 for δ as shown in (6.8.7). We enter

$$\mathbf{nDer(fnInt(}\sin(x^2), x, 1, x\mathbf{)}, x\mathbf{)}$$

for $y1$ and $\sin(x^2)$ for $y2$ as shown in (6.9.3). Note that we have selected Line style for $y1$ and Thick style for $y2$.

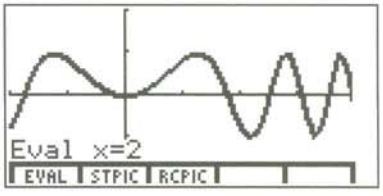
(6.9.3)

3. We then graph $y1$ and $y2$ with **xRes** = 3. Note on your calculator how the graph of $y1$ is first graphed in Line style and then $y2$ is graphed in Thick style covering up $y1$. After about 1.5 minutes we obtain the display shown in (6.9.4).



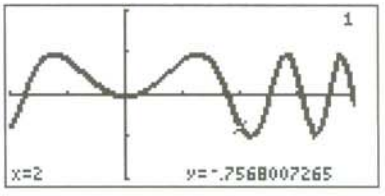
(6.9.4)

It is instructive to use ⟨**EVAL**⟩ to demonstrate that (6.9.4) contains both graphs and that the values of $y1$ and $y2$ are slightly different at $x = 2$ (due to the approximate nature of **fnInt** and **nDer**).



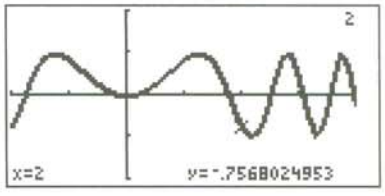
(6.9.5)

4. With the display as in (6.9.4), press [MORE] [MORE] ⟨**EVAL**⟩ [2] to obtain (6.9.5), and then press [ENTER] to obtain (6.9.6), which indicates that $y1 = -0.7568007265$ at $x = 2$.



(6.9.6)

5. Then press the up arrow key to obtain (6.9.7), which indicates that $y2 = -0.7568024953$ at $x = 2$.



(6.9.7)

§10 – Limitations of Numerical Integration: Part 1

In this section and the next we investigate the limitations of the **fnInt** command. The general principles employed by this command are common to graphing calculators and computer software (such as *Mathematica*) that have built-in numeric integration commands. We have found that they all have trouble dealing with the types of examples considered in Sections 10 and 11.

The general numeric integration algorithm approximates the integral of a function $f(x)$ by computing a weighted average of the function's values at values x (sample points) in the integration interval. The algorithm uses an iteration method, doubling the number of sample points in each successive iteration and computing the weighted average for each iteration. When the algorithm indicates that the last weighted average is within the stated tolerance (the **tol** setting on the TI-86), the algorithm terminates, and the current weighted average is reported as the value of the integral.

The procedure whereby the algorithm determines the initial number of sample points and the test to terminate the algorithm varies from one machine to the next.

All of the numeric integration algorithms appear to make similar assumptions about the smoothness and other properties of the integrand. For a sufficiently pathological integrand, these assumptions may be wrong and the algorithm may return an absurd answer or an answer that is reasonable but not within the stated tolerance.

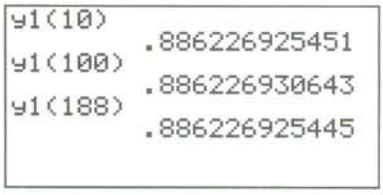
1. For our first pathological example, consider the function

$$G(x) = \int_0^x e^{-t^2}\,dt$$

which is an increasing function on $[0, \infty)$. Enter $G(x)$ as $y1$ in the ⟨**y(x)=**⟩ graph editor as shown in (6.10.1).

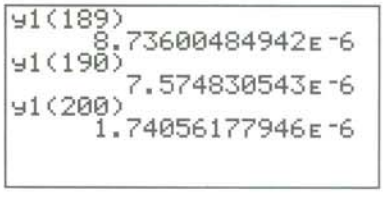
(6.10.1)

2. Then with **tol** = 1×10^{-5}, compute $G(10)$, $G(100)$, $G(188)$, $G(189)$, $G(190)$, and $G(200)$ as shown in (6.10.2) and (6.10.3).



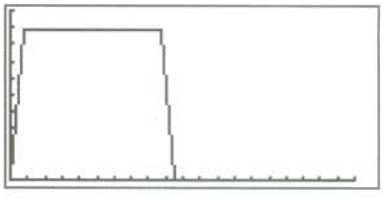
(6.10.2)

3. The reported values for $G(10)$, $G(100)$, and $G(188)$ are correct to the specified tolerance, but the reported values for $G(189)$, $G(190)$, and $G(200)$ are clearly wrong. What has happened? Well, the integrand $g(x) = e^{-x^2}$ approaches 0 very quickly as we allow x to increase from 0 along the positive x-axis; *i.e.*, $g(x)$ is close to zero for almost all x in the intervals [0, 189], [0, 190], and [0, 200]. In the cases $G(189)$, $G(190)$, and $G(200)$, the **fnInt** algorithm was fooled into not taking enough sample points near $x = 0$ to detect that $G(x)$ for x = 189, 190, 200 was significantly greater than 0.



(6.10.3)

4. In (6.10.4) we give the calculator's version of the graph of G in the viewing window $[0, 400, 20] \times [0, 1, 0.1]$ with **xRes** = 5 and **tol** = 1×10^{-5}. (The calculator needs about 1.5 minutes to do the graph.) This graph indicates that the TI-86 will fail to correctly compute $G(x)$ for $x \geq 189$.



(6.10.4)

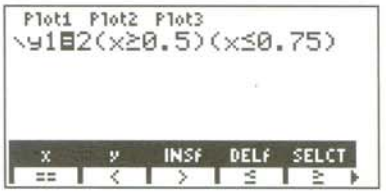
§11 – Limitations of Numerical Integration: Part 2

Our second pathological example is even more enlightening. Consider the function

$$H(x) = \int_0^x h(t)\,dt,$$

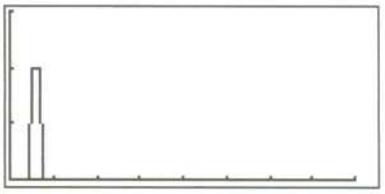
where the integrand is the step function

$$h(x) = \begin{cases} 2, & 0.5 \le x \le 0.75 \\ 0, & \textit{elsewhere} \end{cases}.$$



(6.11.1)

1. In (6.11.1) we enter $h(x)$ as $y1$ in the ⟨**y(x)=**⟩ graph editor using the TEST menu as explained in Sections 14–16 of Chapter 2. In (6.11.2) we show the graph of h in the viewing window $[0, 8, 1] \times [0, 3, 1]$ with **xRes** = 1.

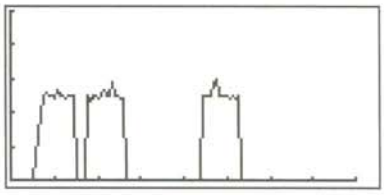


(6.11.2)

2. For all $x \ge 0.75$, it is clear that $H(x) = 1/2$. Let's see what the TI-86 thinks the graph of H should look like in the viewing window $[0, 8, 1] \times [0, 1, 0.2]$ with **xRes** = 1 and **tol** = 1×10^{-5}. We deselect $y1$ for graphing purposes and enter $y2$ as shown in (6.11.3). (With the chosen values for **xRes** and **tol**, the TI-86 will need about 5 minutes to draw the graph.)

(6.11.3)

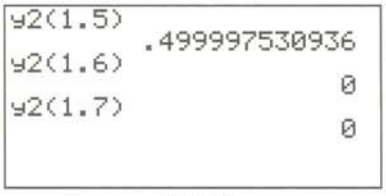
3. In (6.11.4) we see the TI-86 rendition of the graph of H in the specified viewing window. The graph in (6.11.4) is rather shocking! The TI-86 reports that the value of $H(x)$ is zero for a large fraction of x in $[0.75, 8]$. Even when it reports a value for $H(x)$ close to 1/2, this value is often not within the specified tolerance level (look at the jagged nature of the graph).



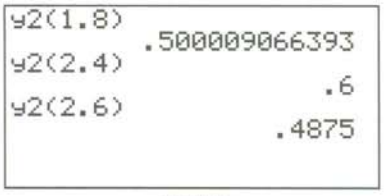
(6.11.4)

In (6.11.5)–(6.11.10) we give additional numerical evidence of the failure of the TI-86 to give the correct value of 1/2 for $H(x)$ when $x \ge 0.75$.

(6.11.5)

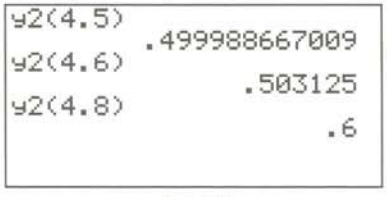


(6.11.6)

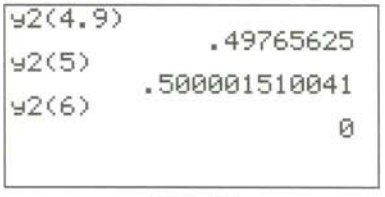


(6.11.7)

(6.11.8)



(6.11.9)



(6.11.10)

When the TI-86 reported a value of 0 for $H(x)$ when $x \geq 0.75$, it is evident that the **fnInt** algorithm did not sample any points x in the interval [0.5, 0.75] and, thus, thought that the integrand was zero throughout the interval of integration. When the TI-86 reported a value for $H(x)$ close to 1/2 but not within the stated tolerance (such as reporting $H(4.8) = 0.6$), evidently the bisection process (due to the step discontinuities at $x = 0.5$ and $x = 0.75$), did not produce a large enough difference for successive weighted averages in order for the algorithm to work as hoped.

The two pathological examples presented in the last two sections should not shake your faith in the **fnInt** command. For continuous integrands with no sharp-looking spikes, when you graph the integrand over the interval of integration, the **fnInt** command gives results that are correct to within the stated tolerance.

Exercises

1. Consider $\int_0^1 \sin(x^2)dx$. Compute LS(100), RS(100), TRAP(100), MID(100), and SIMP(100) for this integral.

2. Let I be given by $I = \int_1^3 2^{-x}\,dx$.

   (a) Find the exact value of I.

   (b) Approximate I using **fnInt** with **tol** = 1×10^{-5}.

   (c) Use **sum seq** to compute RS(50) for I.

   (d) The value of RS(50) is also given by $(1/25)\sum_{K=1}^{50} 2^{-(1+K/25)}$. Write down the appropriate **sum seq** command with variable K and increment value 1 to compute RS(50) using this form. Some people prefer this method of using **sum seq** to compute Riemann sums since integer increments avoid the need to adjust for premature truncation.

3. Use **fnInt** to compare the area under one hump of the graph of $f(x) = 2\sin(2x)$ with the area under one hump of the graph of $f(x) = \sin x$. Explain mathematically why the areas compare as they do.

4. It is usually shown in elementary calculus that the improper integral $\int_1^\infty \frac{1}{x^2}dx$ converges to 1.

   (a) Give numerical evidence in support of this fact.

   (b) Compute **fnInt**$(1/x^2,\ x,\ 1,\ 10^8)$ to see that one must be careful in interpreting the TI-86's response for problems of this type.

5. Consider $f(x) = x^{\sin x} - 0.8,\ 0 \leq x \leq 1$. Have the TI-86 sketch a graph of a function $F(x)$ with the property that $F'(x) = f(x)$ and $F(0) = 0.2$.

6. Is the graph of $f(x) = \sin\left(\int_0^x \sqrt{t^3+1}\,dt\right)$ increasing or decreasing at $x = 1.7$?

