Sums of Rectangle Areas to Approximate Integrals

by Dave Slomer

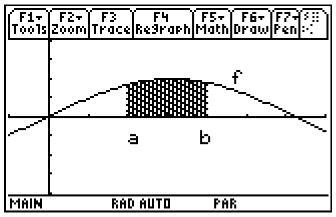


Fig. 1

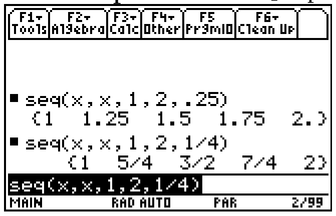
Areas of regions such as the one in figure 1 cannot be computed using formulas from geometry. But by *inscribing* or *circumscribing* enough rectangles (or trapezoids) whose areas approximate the area of the shaded region, we can get as close as necessary to the exact area. (You might want to scan figures 3a through 4b now to get a preview of the process.) In general, the area of a region bounded by $y = f(x)$, $x = a$, $x = b$, and the x-axis is symbolized as $\int_a^b f(x)dx$ (read "the *integral* of *f* of *x with respect to x*, from *a* to *b*").

The first step in approximating such an area is to *partition* (*subdivide*) the interval $[a,b]$ into n equal pieces (called *subintervals*) by locating $n + 1$ *subdivision points*. (Refer to the tick marks on the x-axis in figure 2b.) The length of each subinterval is called Δx, which, therefore, is always defined by $\Delta x = \frac{b-a}{n}$. If $n = 2$ or 4 or something not much bigger, it is advisable to perform the process by hand to get a feel for this important process. But if $n = 10$ or more, you need help from a TI-89.

The TI-89 "sequence function" **seq** can help create the partition. It has the general form

seq(*sequence formula***,***variable in the formula***,***variable start value***,***end value***,***step***)**.

It returns a *list* of values enclosed in "curly brackets"—**{ }**—according to the *sequence formula*, starting at the *variable start value*, going up by *step*, and stopping at the *end value*. For example, the command **seq(x,x,1,2,1/4)** gives the sequence $a_x = x$ for $x = 1$ to 2 by steps of 1/4—five x-coordinates, as shown in figure 2a. {To get decimals instead of fractions, make the step a decimal [or press ◆ENTER (≈)].}



```
■ seq(x,x,1,2,.25)
   {1   1.25   1.5   1.75   2.}
■ seq(x,x,1,2,1/4)
      {1   5/4   3/2   7/4   2}
seq(x,x,1,2,1/4)
```

Fig. 2a

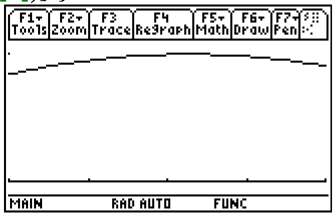


Fig. 2b

These values of **x** are the endpoints of the bases of the rectangles whose areas will be used to approximate $\int_a^b f(x)dx$. Note that the left-hand edge of the region, $x = 1$, and the right-hand edge, $x = 2$, are the *start value* and *end values* in the **seq** command in figure 2a. Also note that $\Delta x = \frac{2-1}{4} = \frac{1}{4}$, the *step* used in the **seq** command. Finally, note that while $n = 4$, there are $n + 1$ (that is, 5) subdivision points, as shown on the x-axis in figure 2b, where the boundaries of the region of figure 1 have been displayed in the window [1,2] by [-.1,1.1] with **xscl** = .25, which is **Δx**. (It is not necessary to make **xscl** = **Δx**.)

To begin constructing the approximating rectangles, construct vertical line segments, extending from each subdivision point to the curve (see figure 3a).

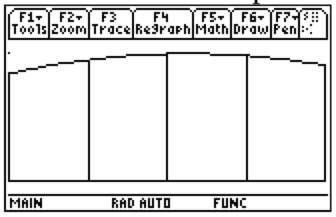


Fig. 3a

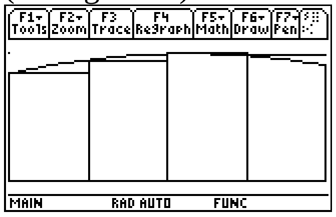


Fig. 3b

This gives 3 sides (bottom and vertical sides) of each of 4 rectangles. Do you see that?

The final geometric step is to draw the 4th sides of each rectangle. How to draw it depends on whether the approximation is to be done using *left*, *right*, or *midpoint* sums. In figure 3b, you see 4 rectangles that would compute the area sum of 4 left rectangles, so-called because the rectangles' heights are function values at the left endpoint of each subinterval. Do you see that the first two rectangles' areas are a little too small, while the last two are a little too big?

In figures 4a and 4b are 4 right and 4 midpoint rectangles. Note that the heights of the midpoint rectangles are computed at midpoints of subintervals. Do you see that?

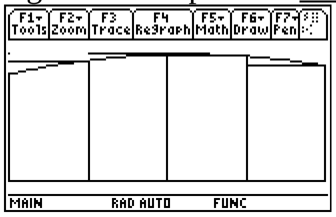


Fig. 4a

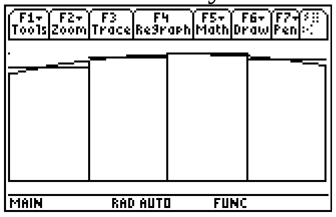
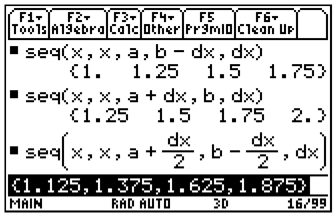


Fig. 4b

Do you see (fig. 4a) that the first two right rectangles' areas are a little too big, while the last is too small, and the third is too close to call? Do you see that each midpoint rectangle's area is a lot closer to the actual area because of being partly above (too big) and partly below (too small) the curve (although the middle two are too close to call)?

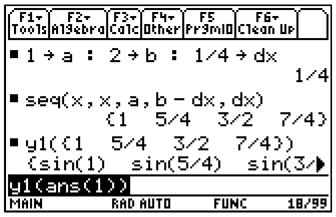
Computing the area sums by hand is easy enough with only 4 rectangles, although it is a little tedious. But no matter how many approximating rectangles are used, the '89 can help with its **seq** command.

The command **seq(x,x,1,2,1/4)** that was used to generate the partition earlier involves 5 *x*-coordinates, but there are only 4 rectangles. Study figures 3a through 4b, noting that, for left sums, the rightmost endpoint does not enter into the computation; for right sums, the leftmost endpoint isn't used; and for midpoint sums, only midpoints of subintervals are used. For a change, symbols might make it clearer than numbers. Consider the screen in figure 5a. Prior commands had been given to store 1 into **a**, 2 into **b**, and 1/4 into **dx** [**dx** muse be used instead of **Δx**]. Decimals were obtained by pressing $\boxed{◆}\boxed{\text{ENTER}}$.



```
■ seq(x, x, a, b - dx, dx)
        {1.   1.25   1.5   1.75}
■ seq(x, x, a + dx, b, dx)
        {1.25   1.5   1.75   2.}
■ seq(x, x, a + dx/2, b - dx/2, dx)
{1.125,1.375,1.625,1.875}
```

Fig. 5a



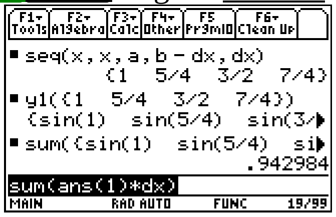
```
■ 1 → a : 2 → b : 1/4 → dx
                              1/4
■ seq(x, x, a, b - dx, dx)
        {1   5/4   3/2   7/4}
■ y1({1   5/4   3/2   7/4})
  {sin(1)   sin(5/4)   sin(3/▸
y1(ans(1))
```

Fig. 5b

Note that, in figure 5a, as expected from the graphs in figures 3b through 4b:

- The first **seq** command includes 1 but not 2 (because the ending value is **b-dx**, the next-to-last subdivision point) and so gives *x*-coordinates for a left sum.
- The second does not include 1 (because the starting value is **a+dx**, the second subdivision point) but does include 2 and so gives *x*-coordinates for a right sum.
- The third includes neither 1 nor 2 because it starts at **a+dx/2** (the midpoint of the first subinterval) and ends at **b-dx/2** (the midpoint of the last subinterval) and so gives *x*-coordinates for a midpoint sum, hitting each subinterval's midpoint en route, since the increment is **dx**. Think about this one for a moment. See figure 4b for clarification. Drawing dotted line segments from the midpoint of each subinterval to the curve and noting that this is the height of the rectangles may help.

Having constructed the rectangles and having decided which type of approximating sum to compute (left, right, or midpoint), you can make the '89 find all *n* heights at once by one command: **y1(ans(1))**. See figure 5b for details, where a left sum has been used with **y_1(x) = sin(x)**. Once you have all of the heights available as a list, all that remains is to multiply each by the base (**dx**) and **sum** the results by typing **sum(ans(1)*dx)** and to press ◆ ENTER to get a decimal approximation. (See figure 6a)



```
■ seq(x, x, a, b - dx, dx)
        {1   5/4   3/2   7/4}
■ y1({1   5/4   3/2   7/4})
  {sin(1)   sin(5/4)   sin(3/▸
■ sum({sin(1)   sin(5/4)   si▸
                          .942984
sum(ans(1)*dx)
```

Fig. 6a

So, using 4 left rectangles, $\int_1^2 \sin(x)dx \approx .94$.

Exercise 1: Approximate $\int_1^2 \sin(x)dx$ with 4 right rectangles.

Define **y_1(x) = sin(x)** and follow the process shown in figure 6a, but modify the **seq** command to give right instead of left endpoints of subintervals. (Since the left sum was about 1, you would expect the result here to be around 1, too, wouldn't you?)

Exercise 2: Approximate $\int_1^2 \sin(x)dx$ with 4 midpoint rectangles.

Exercise 3: On any given subinterval, trapezoids could be constructed instead of rectangles by drawing line segments from the point on the curve at the left endpoint of the subinterval to the point on the curve at the right endpoint (see fig. 7). It can be shown that the area of any such trapezoid is the average of the areas of the left and right rectangles on that interval. (Make a sketch and convince yourself of that.) Hence, the average of the areas of *n* left and *n* right rectangles will always equal the area of *n* trapezoids. The trapezoid area sum would seem to not miss by much at all, since you can hardly distinguish curve from upper trapezoid side in figure 7. Compute the sum of 4 trapezoids by averaging the left and right sums.

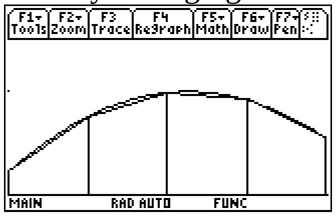


Fig. 7

Exercise 4: Compute the <u>exact</u> value of $\int_1^2 \sin(x)dx$ by using the TI-89's built in $\boxed{\int}$ key ($\boxed{\text{2nd}}\boxed{7}$) by typing $\boxed{\int}$ **y1(x),x,1,2)**. Do it again, but press $\boxed{\blacklozenge}\boxed{\text{ENTER}}$ to see a very accurate decimal approximation of the exact value so you can rank the left, right, midpoint and trapezoid approximations from worst to best. Do you think it will <u>always</u> be so ordered?

Exercise 5: Double the number of rectangles from 4 to 8, changing **dx** from 1/4 to 1/8. Repeat the approximations—left, right, midpoint, and trapezoid—and rank again from worst to best. The order changed. Why do you think this happened?

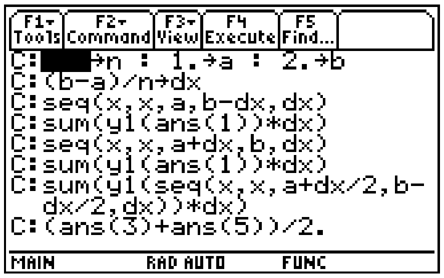
Exercise 6: Clearly, the larger *n* is, the smaller **Δx** will be, and the better the approximation. Suppose you did not know the exact value of $\int_1^2 \sin(x)dx$. How would you know how close you were for the given *n* and **Δx**? Get the most powerful automated help you can find (whether a TI-89 program, such as **RectAprx** below, or the text file shown below, or a more powerful PC program, such as Derive™, that you might have access to). Use it to compute the left, right, midpoint, and trapezoid sums for 4,8,16, 32, 64, … subintervals. Fill in the table below, stopping when you are sure that you have <u>4 decimal place accuracy</u> (accuracy to <u>ten-thousandths</u>). Why did you stop when you did? Which method produced the desired accuracy first? Compare the approximation to the exact value from Exercise 4. Did you stop too soon?Conclusions?

n	Left sum	Right sum	Midpoint sum	Trapezoid sum
4				
8				
16				
32				
64				
128				
256				

While not very accurate in comparison to trapezoid and midpoint sums, sums of left and right rectangle areas are still important to approximate integrals for two reasons.

(1) Their average gives a trapezoid sum that is usually much more accurate than either left or right sums (but don't forget the method accuracy switch in Exercises 4 and 5).

(2) The much more accurate, harder to apply midpoint sums are made considerably easier by studying the easier to apply left and right sums first.



```
C:[illegible]→n : 1.→a : 2.→b
C:(b-a)/n→dx
C:seq(x,x,a,b-dx,dx)
C:sum(y1(ans(1))*dx)
C:seq(x,x,a+dx,b,dx)
C:sum(y1(ans(1))*dx)
C:sum(y1(seq(x,x,a+dx/2,b-
  dx/2,dx))*dx)
C:(ans(3)+ans(5))/2.
```

Fig. 8

```
(a,b,n)
Prgm RectAprx
setMode("Exact/Approx","Approximate")
Ø→l:Ø→m:Ø→r
(b-a)/n→dx
For x,a,b-dx/2,dx
  l+dx*y1(x)→l
  r+dx*y1(x+dx)→r
  m+dx*y1(x+dx/2)→m
EndFor
Disp "n,l,r,m, and t:"
Disp n,l,r,m,(l+r)/2.
EndPrgm
```

Calculus Generic Scope and Sequence Topics: Definite and Indefinite Integrals
NCTM Standards: Number and operations, Algebra, Geometry, Measurement, Problem solving, Connections, Communication, Representation