Riemann, Trapezoids, and Simpson

Introduction

OBJECTIVE: To visualize the process of using Riemann sums, the trapezoid rule, and Simpson's rule for approximating definite integrals, and to understand the error associated with each method.

To evaluate a definite integral, it is often necessary to use a numerical estimate of the integral in place of an exact value. This occurs in three instances: 1) when there is no simple formula for an antiderivative of the integrand; 2) when the antiderivative of the integrand is difficult to determine and/or evaluate; and, 3) when the integrand function is represented by a table of numeric values rather than by a formula. In this module, we investigate several numerical methods for estimating definite integrals.

Technology Guidelines

NOTE: If you have just finished a module, restart *Mathematica* or close the *Kernel* before executing a new module.
TO OPEN CELLS, put your cursor on the right cell bracket and double click.
INITIALIZATION CELLS
When asked if you want to ". . . automatically evaluate all the initialization cells in the notebook . . . ," respond by pressing the "Yes" button.
TO STOP AN EXECUTION
Select the *Kernel* pull-down menu and click on *Abort Evaluation.*
ORDER OF EXECUTION
Execute cells in the order given. Do not skip any Input cells within a given notebook.
SAVING NOTEBOOKS
You can save anytime to any directory you choose, and it is wise to save often.
However, before you do your final save, delete all your output by selecting the
Delete All Output selection under the *Kernel* pull-down menu.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, then shut down *Mathematica* and start it up again.

Part I: Riemann Sums and Errors

The most direct method for estimating a definite integral is to replace it with a Riemann sum. That is, we estimate the integral as follows: $\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(c_i)h$, where the interval $[a, b]$ is divided into n subintervals, each of length $h=\frac{b-a}{n}$, and where c_i is any value of x taken from the i^{th} subinterval. To assess the error in the approximation, we will perform a numerical experiment wherein we use a Riemann sum to estimate an integral for which we know the exact value. (You should keep in mind, however, that numerical integration is primarily used for estimating integrals in situations where we can't determine a decimal or fraction representation for the exact value of the integral.) Our first goal is to determine what factors affect the error when we use left- or right-hand Riemann sums to estimate the exact value of an integral.

The integral we choose for our experiment is $\int_0^{\frac{\pi}{2}} \cos x dx=1$. First, we look at a graphical depiction of the situation. For this, we include a specially designed command that shows the areas of the rectangles that are accumulated in the Riemann sum and the errors in the estimate. The command is **riemannsum[f_, x_, a_, b_, n_, rightleft_]**. The arguments are the function, **f**, the independent variable, **x**, the lower bound on the integral, **a**, the upper bound, **b**, the number of rectangles, **n**, and a "right" or "left" sum indicator. Red areas are positive, and blue areas are negative. The error is taken as the Riemann estimate minus the exact value of the integral.

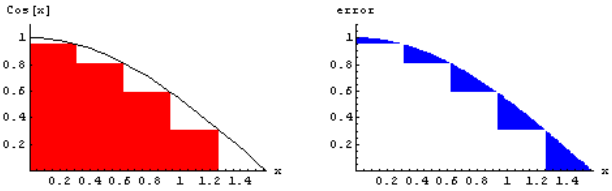
The special command **riemannsum[ ]** is only available in this module and is not a built-in *Mathematica* command. Here's how it works.

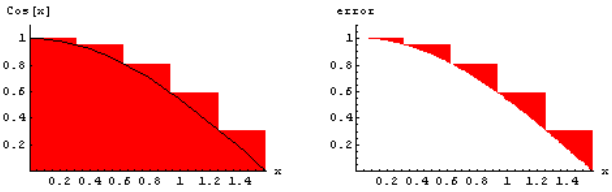
In[183]:=

```
n = 5;

riemannsum[Cos[x], x, 0, Pi / 2, n,
 "right"];

riemannsum[Cos[x], x, 0, Pi / 2, n, "left"];
```





The area of each small triangular-shaped region on the error graph is the local error associated with each rectangle in the Riemann sum. The sum of the areas of these triangular-shaped regions is the global or total error in the estimate of the integral.

In general, the areas can be positive, negative, or 0. In the graphs generated by **riemannsum[ ]**, positive areas are red, and negative areas are blue. Because the areas are signed, we will refer to them hereafter as "signed areas."

You Try It: Visualizing the Errors

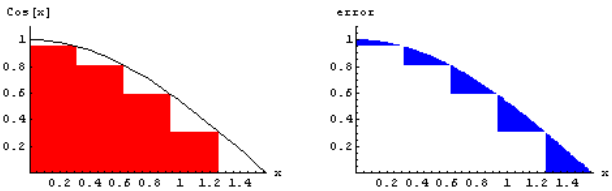
Use the **riemansumm[ ]** command in the preceding cell (copied below) to respond to the items that follow.

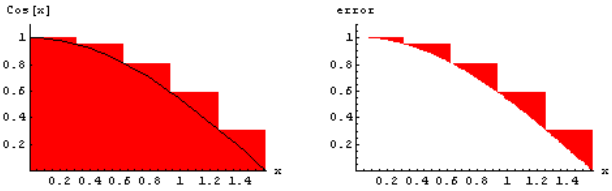
In[186]:=

```
n = 5;

riemannsum[Cos[x], x, 0, Pi/2, n,
 "right"];

riemannsum[Cos[x], x, 0, Pi/2, n, "left"];
```





a) Increase the number of rectangles, highlighted in red, in the preceding input cell, and describe qualitatively what happens to the error in the estimate of the integral each time you double the number of rectangles.

b) What effect does the slope of the function have on the error?

c) The function cos x decreases on the interval from 0 to $\frac{\pi}{2}$, and its first derivative is negative. In this case, the right-hand Riemann sum underestimates the integral (i.e., the error is negative), and the left-hand sum overestimates the integral (i.e., the error is positive). Describe what happens to the error when we use left- and right-hand Riemann sums to estimate $\int_0^{\frac{\pi}{2}} \sin x \, dx$.

d) While the error for each rectangle (i.e., the local error) on the left-hand sum appears to be nearly equal in magnitude to the local error for each corresponding rectangle on the right-hand sum, they aren't exactly the same because the graph of the function curves. Where does the difference in the local errors appear to be largest, and how is the difference in the local errors related to the second derivative of the function? (To see this effect more dramatically, try using two and/or three rectangles in the **riemannsum[ ]** command.)

e) Based on your observations in part (d), specify two different ways in which you could use Riemann sums to improve the estimate of the integral by reducing the errors. The trick is to look for ways to estimate the integrals so that the local errors nearly cancel. For your improved methods, which feature of the integrand function would you expect to have the most significant effect on the error?

f) Try the **riemannsum[ ]** command on a function that we can't integrate exactly. Two examples are $\int_1^2 \frac{1}{x} \, dx$ and $\int_0^4 e^{-x^2} \, dx$.

Part II: Analyzing the Errors

Now that we have made some qualitative observations about Riemann estimates of an integral and the associated errors, we are ready to be more quantitative in our analysis. For this we write the following **riemSumLeft[f_,x_,a_,b_,n_]** and **riemSumRight[f_,x_,a_,b_,n_]** commands to calculate left-hand and right-hand Riemann sums. The arguments are **f**, the integrand function, **x**, the independent variable, **a**, the lower bound, **b**, the upper bound, and **n**, the number of subintervals.

In[189]:=

```
riemSumLeft[f_, x_, a_, b_, n_] :=
 Sum[(f /. x → (a + (i - 1) * (b - a) / n)) *
    (b - a) / n, {i, 1, n}];


riemSumRight[f_, x_, a_, b_, n_] :=
 Sum[(f /. x → (a + i * (b - a) / n)) * (b - a) / n,
  {i, 1, n}];
```

Note that h=(b-a)/n and, for left-hand sums, c_i =a+(i-1)h=a+(i-1)(b-a)/n, whereas for right-hand sums, c_i =a+i h=a+i(b-a)/n. Inside the **Sum[ ]** command for the left-hand sum, the code **f/.x→(a+(i-1)*(b-a)/n)** means that the value of **f** should be used with **x** replaced with **(a+(i-1)*(b-a)/n)** which is c_i. Similarly, **x** is replaced with **(a+i*(b-a)/n))*(b-a)/n** for the right-hand sum.

Now we use these commands, together with the fact that the exact value of the integral $\int_0^{\frac{\pi}{2}} \cos x \, dx$ is 1, to build a table of values that includes the step size **h**, the numerical estimates of the integral, **RleftI** and **RrightI**, and the errors in the estimates, ***Eleft*=RleftI-**1 and ***Eright*=RrightI-**1. We start with two rectangles and double the number of rectangles ten times.

In[191]:=

```
f[x_] = Cos[x];
 a = 0;
 b = π/2;
 exactvalue = ∫_a^b f[x] dx;
 Print[
  "The exact value of the integral is ",
  exactvalue];

 t1 =
  Table[{2^i, π/2/2^i,
    RleftI =
     riemSumLeft[f[x], x, a, b, 2^i] // N,
    RrightI =
     riemSumRight[f[x], x, a, b, 2^i] // N,
    RleftI - exactvalue,
    RrightI - exactvalue}, {i, 1, 11}];

TableForm[t1,
  TableHeadings →
   {None, {"n", "h", "left estimate",
     "right estimate", "left error",
     "right error"}}]
```

```
The exact value of the integral is 1
```

Out[197]//TableForm=

n	h	left estimate	right estimate	left error	right error
2	$\frac{\pi}{4}$	1.340758530667244`	0.5553603672697958`	0.34075853066724404`	-0.44663963273020424`
4	$\frac{\pi}{8}$	1.1834653418221375`	0.7907662601234133`	0.18346534182213747`	-0.20923373987658667`
8	$\frac{\pi}{16}$	1.0949599423108507`	0.8986104014614886`	0.0949599423108507`	-0.10138959853851137`
16	$\frac{\pi}{32}$	1.0482840656974128`	0.9501092952727318`	0.04828406569741284`	-0.0498907047272682`
32	$\frac{\pi}{64}$	1.024342886926189`	0.9752555017138487`	0.024342886926189022`	-0.02474449828615133`
64	$\frac{\pi}{128}$	1.0122216463951867`	0.9876779537890162`	0.012221646395186747`	-0.012322046210983761`
128	$\frac{\pi}{256}$	1.006123373269069`	0.9938515269659839`	0.006123373269069088`	-0.006148473034016111`
256	$\frac{\pi}{512}$	1.0030648241110593`	0.9969289009595166`	0.00306482411105935`	-0.003071099040483416`
512	$\frac{\pi}{1024}$	1.0015331964220768`	0.9984652348463051`	0.0015331964220768324`	-0.0015347651536948836`
1024	$\frac{\pi}{2048}$	1.0007667943025131`	0.9992328135146272`	0.0007667943025131407`	-0.0007671864853727728`
2048	$\frac{\pi}{4096}$	1.000383446174115`	0.9996164557801718`	0.0003834461741150097`	-0.0003835442198282246`

The data in the table should confirm your qualitative observations from "You Try It: Visualizing Errors" above, but now we can be more specific. We make the following quantitative observations.

1) As the number of rectangles increases, the error decreases, and, in fact, each time we double the number of rectangles or cut *h* in half, we cut the error roughly in half. A numerical estimate that exhibits this characteristic is said to have an error of order *h*, and we designate this as O(h).

2) The left-hand sum overestimates the integral, whereas the right-hand some underestimates it, and, for the same number of rectangles, the errors are roughly equal in magnitude.

3) When the number of rectangles is small, the difference between the errors for the left- and right-hand estimates is larger, showing the effect of the second derivative and the concavity of the graph on the difference in errors.

The one effect that is not evident from the numeric data is that the local error is larger when the magnitude of the first derivative, that is, the magnitude of the slope of the graph, is larger. We explore this effect further in Part III below.

You Try It: On Another Function

Perform a numerical experiment like the one in Part II on the integral $\int_0^{\frac{\pi}{2}} \sin^2 x\,dx$, and then answer the questions that follow. To help, we include the commands to generate the table of values like the one in Part II.

In[198]:=

```
f[x_] = Cos[x];
 a = 0;
 b = π/2;
 exactvalue = ∫_a^b f[x] dx;
 Print[
  "The exact value of the integral is ",
  exactvalue];

 t1 =
  Table[{2^i, π/2/2^i,
    RleftI =
     riemSumLeft[f[x], x, a, b, 2^i] // N,
    RrightI =
     riemSumRight[f[x], x, a, b, 2^i] // N,
    RleftI - exactvalue,
    RrightI - exactvalue}, {i, 1, 11}];

TableForm[t1,
  TableHeadings →
   {None, {"n", "h", "left estimate",
     "right estimate", "left error",
     "right error"}}]
```

```
The exact value of the integral is 1
```

Out[204]//TableForm=

n	h	left estimate	right estimate	left error	right error
2	$\frac{\pi}{4}$	1.340758530667244`	0.5553603672697958`	0.34075853066724404`	-0.44463963273020424`
4	$\frac{\pi}{8}$	1.1834653418221375`	0.7907662601234133`	0.18346534182213747`	-0.20923373987658667`
8	$\frac{\pi}{16}$	1.0949599423108507`	0.8986104014614886`	0.0949599423108507`	-0.10138959853851137`
16	$\frac{\pi}{32}$	1.0482840656974128`	0.9501092952727318`	0.04828406569741284`	-0.0498907047272682`
32	$\frac{\pi}{64}$	1.024342886926189`	0.9752555017138487`	0.024342886926189022`	-0.02474449828615133`
64	$\frac{\pi}{128}$	1.0122216463951867`	0.9876779537890162`	0.012221646395186747`	-0.012322046210983761`
128	$\frac{\pi}{256}$	1.006123373269069`	0.9938515269659839`	0.006123373269069088`	-0.006148473034016111`
256	$\frac{\pi}{512}$	1.0030648241110593`	0.9969289009595166`	0.0030648241105935`	-0.003071099040483416`
512	$\frac{\pi}{1024}$	1.0015331964220768`	0.9984652348463051`	0.0015331964220768324`	-0.0015347651536948836`
1024	$\frac{\pi}{2048}$	1.0007667943025131`	0.9992328135146272`	0.0007667943025131407`	-0.0007671864853727728`
2048	$\frac{\pi}{4096}$	1.000383446174115`	0.9996164557801718`	0.0003834461741150097`	-0.0003835442198282246`

1. What happens to the errors each time the number of rectangles is doubled? What is the order of the error for each Riemann sum estimate of the

integral?

2. Does the left-hand Riemann sum overestimate or underestimate the integral? What about the right-hand Riemann sum?

3. What effect does the second derivative of the integrand have on the difference of the errors for the left and right Riemann sums?

Part III: The Effect of f' on the Error

To quantify the effect of the slope on the error, we investigate straight-line functions, f(x)=m x, with varying slopes. First, let's look at a graphical depiction.

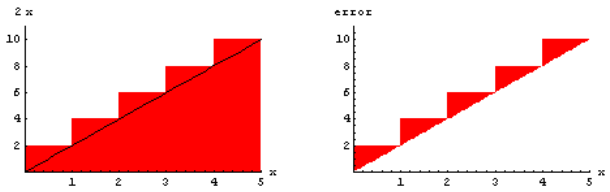
In[205]:=

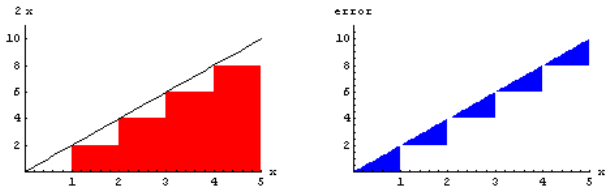
```
n = 5;

m = 2;

riemannsum[m*x, x, 0, 5, n, "right"];

riemannsum[m*x, x, 0, 5, n, "left"];
```





Now we set up a table of numeric values, but in this case we keep the number of rectangles fixed at 100 and change *m*, the slope of the straight-line graph, starting with m=0.1 and doubling it ten times. The exact values of the integrals are $\frac{25\,m}{2}$.

In[209]:=

```
Clear[m];

m[i_] = 0.1*2^i;

f[x_] = m[i] x;

a = 0;
```

```
b = 5;

exactvalue = ∫_a^b f[x] dx;

Print[
 "The exact value of the integral is ",
 exactvalue];

t2 =
 Table[{m[i], exactvalue,
   RleftI =
    riemSumLeft[f[x], x, a, b, 100] // N,
   RrightI =
    riemSumRight[f[x], x, a, b, 100] // N,
   RleftI - exactvalue,
   RrightI - exactvalue}, {i, 0, 10}];

TableForm[t2,
 TableHeadings →
  {None, {"slope", "exact", "left estimate",
    "right estimate", "left error",
    "right error"}}]

The exact value of the integral is 1.25 2^i
```

Out[217]//TableForm=

slope	exact	left estimate	right estimate	left error	right error
0.1`	1.25`	1.2375000000000005`	1.2625000000000004`	-0.012499999999999512`	0.0125000000000004`
0.2`	2.5`	2.475000000000001`	2.525000000000001`	-0.024999999999999023`	0.0250000000000008`
0.4`	5.`	4.950000000000002`	5.050000000000002`	-0.049999999999998046`	0.0500000000000016`
0.8`	10.`	9.900000000000004`	10.100000000000003`	-0.09999999999999609`	0.1000000000000032`
1.6`	20.`	19.800000000000008`	20.200000000000006`	-0.19999999999999218`	0.2000000000000064`
3.2`	40.`	39.600000000000016`	40.40000000000001`	-0.39999999999998437`	0.4000000000000128`
6.4`	80.`	79.20000000000003`	80.80000000000003`	-0.7999999999999687`	0.8000000000000256`
12.8`	160.`	158.40000000000006`	161.60000000000005`	-1.5999999999999375`	1.6000000000000512`
25.6`	320.`	316.8000000000001`	323.2000000000001`	-3.199999999999875`	3.2000000000001023`
51.2`	640.`	633.6000000000003`	646.4000000000002`	-6.39999999999975`	6.400000000000205`
102.4`	1280.`	1267.2000000000005`	1292.8000000000004`	-12.7999999999995`	12.80000000000041`

What happens to the error each time you double the slope of the line?

Note that for nonlinear functions, the effect of the slope on the error is local in that it varies from point to point on the graph. But if we double all of the slopes, we would also double the error in using a left- or right-hand Riemann sum to estimate the integral.

Note also that the differences in the magnitudes of the errors are all 0. This supports the observation that the difference in the errors is dependent upon the second derivative or the concavity of the graph. One method for improving our numerical estimate of the integral is to add the left- and right-hand estimates and then divide the result by two. The idea that motivates this is that the errors for the two estimates will nearly cancel each other out when we add the left- and right-hand sums together. For linear functions, the second derivative is zero, and the graph has no concavity; therefore, this improvement will give the exact value of the integral.

You Try It: The Effect of f'' on the Error Difference

Design a numerical experiment like the one in Part III to demonstrate the effect of the value of the second derivative on the difference between the errors in the estimate of the integral that are obtained using right- and left-hand Riemann sums. To do this, select a simple function that has a constant second derivative, is easy to integrate, and allows you to change the value of the second derivative. Generate a table of numerical values that shows the values of the second derivative and the difference in the errors for the left and right Riemann sums. Take 1 as the starting value of the second derivative, and double its value ten times. Try functions that are concave down as well as ones that are concave up, and summarize the results of your

investigation. (We have put the solution for this problem in the next input cell with the code hidden. Try to do it on your own, and then look at our solution. To reveal the hidden code, select the small cell bracket to the right, pull down the Cell menu, select Cell Properties, and click on Cell Open.)

The exact value of the integral is $\frac{125\, 2^{-1+i}}{3}$

Out[226]//TableForm=

f''(x)	exact	left estimate	right estimate	left error	right error	error difference
1	20.833333333333332`	20.521875`	21.146875`	-0.31145833333333073`	0.31354166666666927`	-0.625`
2	41.666666666666664`	41.04375`	42.29375`	-0.6229166666666615`	0.6270833333333385`	-1.25`
4	83.33333333333333`	82.0875`	84.5875`	-1.245833333333323`	1.254166666666677`	-2.5`
8	166.66666666666666`	164.175`	169.175`	-2.491666666666646`	2.508333333333354`	-5.`
16	333.3333333333333`	328.35`	338.35`	-4.983333333333292`	5.016666666666708`	-10.`
32	666.6666666666666`	656.7`	676.7`	-9.966666666666583`	10.033333333333417`	-20.`
64	1333.3333333333333`	1313.4`	1353.4`	-19.933333333333167`	20.066666666666833`	-40.`
128	2666.6666666666665`	2626.8`	2706.8`	-39.86666666666633`	40.13333333333367`	-80.`
256	5333.333333333333`	5253.6`	5413.6`	-79.73333333333267`	80.26666666666733`	-160.`
512	10666.666666666666`	10507.2`	10827.2`	-159.46666666666533`	160.53333333333467`	-320.`
1024	21333.333333333332`	21014.4`	21654.4`	-318.93333333333067`	321.06666666666933`	-640.`

Part IV: Trapezoids and Midpoint Riemann Sums

In this part, we investigate two ways to improve the efficiency of numerical estimates of definite integrals.

▪ Trapezoids

We achieve the first improvement by adding together the left and right Riemann sum estimates of the integral and dividing by two. The motivation for this approach is that the errors in the left and right sums should nearly cancel each other out. In the next cell, we form the new function **trapezoid[ ]** that does this.

In[227]:=

```
trapezoid[f_, x_, a_, b_, n_] :=
 (riemSumLeft[f, x, a, b, n] +
    riemSumRight[f, x, a, b, n]) / 2;
```

(The trapezoid method gets its name from the signed area of a trapezoid in each subinterval that results from adding the signed areas of the left and right rectangles in each subinterval and dividing by 2. That is, $(f(c_{left})h+f(c_{right})h)/2=\frac{f(c_{left})+f(c_{right})}{2}h$, the area of a trapezoid, where $f(c_{left})$ is the length of one parallel side, $f(c_{right})$ is the length of the other parallel side, and *h* is the distance between the two sides.)

Now we use **trapezoid[ ]** to estimate $\int_0^{\frac{\pi}{2}} \cos x\,dx$ with ten subintervals and compare the result with the left- and right-hand Riemann sums. First, we calculate the exact value of the integral.

In[228]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

exactvalue = ∫_a^b f[x] dx;

Print[
 "The exact value of the integral is ",
 exactvalue];
```

The exact value of the integral is 1

Now we calculate the estimates of the integral.

In[233]:=

```
n = 10;

trapezoid[f[x], x, a, b, n] // N

riemSumLeft[f[x], x, a, b, n] // N

riemSumRight[f[x], x, a, b, n] // N
```

Out[234]=

```
0.997943
```

Out[235]=

```
1.07648
```

Out[236]=

```
0.919403
```

And we compute the errors.

In[237]:=

```
(trapezoid[f[x], x, a, b, n] // N) -
 exactvalue

(riemSumLeft[f[x], x, a, b, n] // N) -
 exactvalue

(riemSumRight[f[x], x, a, b, n] // N) -
 exactvalue
```

Out[237]=

```
-0.00205701
```

Out[238]=

```
0.0764828
```

Out[239]=

```
-0.0805968
```

For the same number of subintervals, we get a much more precise estimate of the integral from the trapezoid method.

▪ Midpoint Riemann Sums

The second improvement we consider is to use the midpoint of each subinterval in a Riemann sum. The effect of doing this is to nearly cancel the error on either side of each midpoint in each subinterval. If the function we integrate is increasing over a subinterval, then the midpoint rectangle will overestimate the actual area on the left side of the midpoint and will underestimate, by nearly the same amount, the actual area on the right side of the midpoint. Again we expect that these errors will nearly cancel each other out. Here is the **midpoint[ ]** command.

In[240]:=

```
riemSumMid[f_, x_, a_, b_, n_] :=
 Sum[(f /. x → (a + (i - 1 / 2) * (b - a) / n)) *
   (b - a) / n, {i, 1, n}];
```

Let's try it on our old friend and compare the result with left and right Riemann sums and with the trapezoid estimate. First, we calculate the exact value of the integral.

In[241]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

exactvalue = ∫_a^b f[x] dx;

Print[
 "The exact value of the integral is ",
 exactvalue];
```

```
The exact value of the integral is 1
```

Now we calculate the estimates of the integral.

In[246]:=

```
n = 10;

riemSumMid[f[x], x, a, b, n] // N

trapezoid[f[x], x, a, b, n] // N

riemSumLeft[f[x], x, a, b, n] // N

riemSumRight[f[x], x, a, b, n] // N
```

Out[247]=

```
1.00103
```

Out[248]=

```
0.997943
```

Out[249]=

```
1.07648
```

Out[250]=

```
0.919403
```

Again, we compute the errors.

In[251]:=

```
(riemSumMid[f[x], x, a, b, n] // N) -
 exactvalue

(trapezoid[f[x], x, a, b, n] // N) -
 exactvalue

(riemSumLeft[f[x], x, a, b, n] // N) -
 exactvalue

(riemSumRight[f[x], x, a, b, n] // N) -
 exactvalue
```

Out[251]=

```
0.00102882
```

Out[252]=

```
-0.00205701
```

Out[253]=

```
0.0764828
```

Out[254]=

```
-0.0805968
```

For the same number of intervals, we get a much more precise estimate of the integral using the midpoint rule; it is even more precise than the estimate given by the trapezoid rule. What relationship does the error for the midpoint estimate seem to have with that for the trapezoid estimate?

You Try It: On Some Other Functions

Compare the Riemann left, Riemann right, trapezoid, and midpoint estimates of some integrals that you pick. How does the error for the midpoint rule compare with the error for the trapezoid rule with the same number of subdivisions? Also, try increasing the number of subintervals, highlighted in blue. What happens to the errors when you double the number of subintervals?

Here are some integrals that you might want to try: $\int_0^{\frac{\pi}{2}} \sin x\,dx$, $\int_{-1}^{1} e^x\,dx$, $\int_0^1 \sqrt{1-x^2}\,dx$. To help you out, we provide some commands in the two cells that follow.

In[255]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

exactvalue = ∫_a^b f[x] dx;

n = 10;


Print[
 "The exact value of the integral is ",
 exactvalue];

Print["The midpoint estimate is ",
 riemSumMid[f[x], x, a, b, n] // N];

Print["The trapezoid estimate is ",
 trapezoid[f[x], x, a, b, n] // N];

Print["The left Riemann estimate is ",
 riemSumLeft[f[x], x, a, b, n] // N];

Print["The right Riemann estimate is ",
 riemSumRight[f[x], x, a, b, n] // N];
```

```
The exact value of the integral is 1

The midpoint estimate is 1.00103

The trapezoid estimate is 0.997943

The left Riemann estimate is 1.07648
```

```
The right Riemann estimate is 0.919403
```

In[265]:=

```
Print["The midpoint error is ",
 (riemSumMid[f[x], x, a, b, n] -
    exactvalue) // N];

Print["The trapezoid error is ",
 (trapezoid[f[x], x, a, b, n] - exactvalue) //
  N];

Print["The left Riemann error is ",
 (riemSumLeft[f[x], x, a, b, n] -
    exactvalue) // N];

Print["The right Riemann error is ",
 (riemSumRight[f[x], x, a, b, n] -
    exactvalue) // N];
```

```
The midpoint error is 0.00102882

The trapezoid error is -0.00205701

The left Riemann error is 0.0764828

The right Riemann error is -0.0805968
```

Part V: Experiment with Trapezoids and Midpoints

In this part, we do some numerical experiments to determine what factors affect the errors when we use trapezoids and midpoint Riemann sums to estimate a definite integral.

▪ The Effect of the Step Size on the Error

First, we would like to determine the effect of the step size *h* on the error. We build a table similar to the one in Part II.

In[269]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

exactvalue = ∫_a^b f[x] 𝕕x;

Print[
 "The exact value of the integral is ",
 exactvalue];
```

```
t1 =
 Table[{2^i, π / 2 / 2^i,
   trap = trapezoid[f[x], x, a, b, 2^i] // N,
   mid = riemSumMid[f[x], x, a, b, 2^i] // N,
   trap - exactvalue, mid - exactvalue},
  {i, 1, 11}];


TableForm[t1,
 TableHeadings →
  {None, {"n", "h", "trap estimate",
    "mid estimate", "trap error",
    "mid error"}}]

The exact value of the integral is 1
```

Out[275]//TableForm=

n	h	trap estimate	mid estimate	trap error	mid error
2	$\frac{\pi}{4}$	0.9480594489685199`	1.026172152977031`	-0.0519405510314801`	0.026172152977030905`
4	$\frac{\pi}{8}$	0.9871158009727754`	1.006454542799564`	-0.012884199027224597`	0.006454542799563923`
8	$\frac{\pi}{16}$	0.9967851718861697`	1.001608189083975`	-0.003214828113830337`	0.0016081890839749757`
16	$\frac{\pi}{32}$	0.9991966804850724`	1.0004017081549654`	-0.0008033195149276251`	0.00040170815496543`
32	$\frac{\pi}{64}$	0.999799194320019`	1.0001004058641836`	-0.00020080567998104204`	0.00010040586418358366`
64	$\frac{\pi}{128}$	0.9999498000921014`	1.0000251001429512`	-0.00005019990789856266`	0.00002510014295120655`
128	$\frac{\pi}{256}$	0.9999874501175264`	1.0000062749530498`	-0.000012549882473567031`	6.274953049834053`*^-6
256	$\frac{\pi}{512}$	0.999996862535288`	1.0000015687330954`	-3.1374647120330224`*^-6	1.5687330954250456`*^-6
512	$\frac{\pi}{1024}$	0.9999992156341907`	1.0000003921829501`	-7.843658093031891`*^-7	3.921829501152274`*^-7
1024	$\frac{\pi}{2048}$	0.9999998039085699`	1.0000000980457187`	-1.960914300935812`*^-7	9.804571865501543`*^-8
2048	$\frac{\pi}{4096}$	0.9999999509771432`	1.0000000245114289`	-4.902285677399476`*^-8	2.451142888659774`*^-8

▪ The Effect of f'' on the Error

Based on the observations made in Parts II, III, and IV, we conjecture that the error for these methods depends on the magnitude of the second derivative of the integrand function. We test our conjecture by comparing the error in estimating the area under the quadratic functions of the form $f(x) = \frac{ax^2}{2}$ between x=0 and x=5 for various values of *a*. The exact values of the integrals are $\int_0^5 \frac{ax^2}{2}\,dx = \frac{125a}{6}$, and the second derivative of f(x) is equal to *a*. We set up a table in which *a* starts at 0.1 and doubles ten times. The number of subintervals is kept constant at 100.

In[276]:=

```
Clear[ac];

ac[i_] = 0.1 * 2^i;

f[x_] = ac[i] x^2 / 2;

a = 0;

b = 5;

exactvalue = ∫_a^b f[x] dx;
```

```
Print[
 "The exact value of the integral is ",
 exactvalue];


t2 =
 Table[{ac[i], exactvalue,
   trap = trapezoid[f[x], x, a, b, 100] // N,
   mid = riemSumMid[f[x], x, a, b, 100] // N,
   trap - exactvalue, mid - exactvalue},
  {i, 0, 10}];


TableForm[t2,
 TableHeadings →
  {None, {"f″(x)", "exact", "trap estimate",
    "mid estimate", "trap error",
    "mid error"}}]


The exact value of the integral is 2.08333 2^i
```

Out[284]//TableForm=

f″(x)	exact	trap estimate	mid estimate	trap error	mid error
0.1`	2.0833333333333335`	2.0834375`	2.08328125`	0.00010416666666657193`	-0.00005208333333285964`
0.2`	4.166666666666667`	4.166875`	4.1665625`	0.00020833333333314386`	-0.00010416666666657193`
0.4`	8.333333333333334`	8.33375`	8.333125`	0.00041666666666662877`	-0.00020833333333314386`
0.8`	16.666666666666668`	16.6675`	16.66625`	0.00083333333333325754`	-0.00041666666666662877`
1.6`	33.333333333333336`	33.335`	33.3325`	0.0016666666666651508`	-0.00083333333333325754`
3.2`	66.66666666666667`	66.67`	66.665`	0.0033333333333303017`	-0.0016666666666651508`
6.4`	133.33333333333334`	133.34`	133.33`	0.006666666666660603`	-0.0033333333333303017`
12.8`	266.6666666666667`	266.68`	266.66`	0.013333333333321207`	-0.006666666666660603`
25.6`	533.3333333333334`	533.36`	533.32`	0.026666666666642413`	-0.013333333333321207`
51.2`	1066.6666666666667`	1066.72`	1066.64`	0.05333333333328483`	-0.026666666666642413`
102.4`	2133.3333333333335`	2133.44`	2133.28`	0.10666666666656965`	-0.05333333333328483`

In "You Try It" below, we ask you to summarize the results of these experiments.

You Try It: Summarize the Experiment's Results

Summarize the results of the numerical experiments in Part V. In your summary, address the following items.

a) How is the error for each method related to the number of subintervals n and the interval width h? What is the order of the error for each method? (Note: If cutting the interval width in half reduces the error by a factor of $\left\{\frac{1}{2}\right\}^n$, then the error is of order h^n, and we designate this by $O(h^n)$.)

b) How is the error for the trapezoid method related to the error for the midpoint Riemann sum?

c) How is the error related to the second derivative of the integrand function? (To answer this completely, repeat the second experiment in Part V for negative values of a.

d) Are your conclusions consistent with the trapezoid error formula that is found in the text?

e) From the pattern of errors for the trapezoid method and the midpoint Riemann sum method, specify how you might combine these two estimates of the integral so that the errors will nearly cancel each other. Compare your idea for combining the estimates with those of other students.

Part VI: Simpson's Method

Noting that the magnitude of the error for the trapezoid method is approximately two times the magnitude of the error for a midpoint Riemann sum with the same number of subintervals, we can make yet another improvement in the efficiency of our numerical estimations of the integrals. If we add two midpoint estimates and one trapezoid estimate and divide the result by three, the errors for the two methods should nearly cancel. The result of this combination is called Simpson's method.

To illustrate this, consider the following estimates of the integral $\int_0^{\frac{\pi}{2}} \cos x dx$. Using 10 subintervals, we compare the midpoint and trapezoid estimates and their errors, and then we combine them as indicated above and calculate the error for the new estimate.

In[285]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

n = 10;

exactvalue = ∫_a^b f[x] dx;

trap = trapezoid[f[x], x, a, b, n] // N;

mid = riemSumMid[f[x], x, a, b, n] // N;


simps = (2 * mid + trap) / 3;


Print[
 "The exact value of the integral is ",
 exactvalue];

Print["The trapezoid estimate is ",
 trap];

Print["The trapezoid error is ",
 (trap - exactvalue) // N];

Print["The midpoint estimate is ",
 mid];

Print["The midpoint error is ",
 (mid - exactvalue) // N];

Print[
 "The Simpson estimate of the integral
   is ", simps];

Print[
 "The error for Simpson's estimate is ",
 simps - exactvalue];
```

```
The exact value of the integral is 1

The trapezoid estimate is 0.997943

The trapezoid error is -0.00205701

The midpoint estimate is 1.00103

The midpoint error is 0.00102882
```

```
The Simpson estimate of the integral is 1.

The error for Simpson's estimate is 2.11547 × 10^-7
```

Since Simpson's method requires three function evaluations in each subinterval, one at each end point for the trapezoid plus one at the mid point, we count the number of subintervals differently. If we take the dividing points for the subintervals to be all of the places where the integrand function is evaluated, then there are actually twice as many subintervals for Simpson's method than there would be for the trapezoid method or the midpoint Riemann sum. One consequence of this is that we need an even number of subintervals for Simpson's method to work.

In the next cell we write a command that gives the Simpson estimate of a definite integral.

In[300]:=

```
simpson[f_, x_, a_, b_, n_] :=
 (2 * riemSumMid[f, x, a, b, n / 2] +
    trapezoid[f, x, a, b, n / 2]) / 3;
```

Let's use it to estimate $\int_0^{\frac{\pi}{2}} \cos x dx=1$ and then calculate the errors, comparing the results with those obtained using trapezoids and midpoint Riemann sums.

In[301]:=

```
f[x_] = Cos[x];

a = 0;

b = π/2;

exactvalue = ∫_a^b f[x] dx;

Print[
 "The exact value of the integral is ",
 exactvalue];

Print[
 "The Simpson estimate of the integral
   is ", simpson[Cos[x], x, a, b, 6] // N];

Print[
 "The trapezoid estimate of the
   integral is ",
 trapezoid[Cos[x], x, a, b, 6] // N];

Print[
 "The midpoint estimate of the integral
   is ", riemSumMid[Cos[x], x, a, b, 6] //
  N];
```

```
The exact value of the integral is 1

The Simpson estimate of the integral is 1.00003

The trapezoid estimate of the integral is 0.994282

The midpoint estimate of the integral is 1.00286
```

Now we compare the errors.

In[309]:=

```
Print[
 "The error in the Simpson estimate is ",
 (simpson[Cos[x], x, a, b, 6] // N) -
  exactvalue];

Print[
 "The error in the trapezoid estimate
   is ",
 (trapezoid[Cos[x], x, a, b, 6] // N) -
  exactvalue];

Print[
 "The error in the midpoint estimate is ",
 (riemSumMid[Cos[x], x, a, b, 6] // N) -
  exactvalue];
```

```
The error in the Simpson estimate is 0.0000263122

The error in the trapezoid estimate is -0.00571811

The error in the midpoint estimate is 0.00286151
```

Using only six subintervals, Simpson's method gives very good results.

While left and right Riemann sums integrate constant functions exactly, and the trapezoids and midpoint Riemann sums integrate linear functions exactly, we might conjecture that Simpson's rule integrates quadratic functions exactly. This conjecture is in fact correct, but it is even better than that. Simpson's method integrates cubic functions exactly. We show this in the following cell by applying Simpson's method to a general cubic function with only two subintervals and then comparing the result with the exact value.

In[312]:=

```
Clear[fcubic, a, b];

fcubic[x_] = c x^3 + d x^2 + e x + f;

simpsonresult =
 simpson[fcubic[x], x, a, b, 2] // Expand
```

Out[314]=

$$-\frac{a^4 c}{4}+\frac{b^4 c}{4}-\frac{a^3 d}{3}+\frac{b^3 d}{3}-\frac{a^2 e}{2}+\frac{b^2 e}{2}-a f+b f$$

In[315]:=

```
integralresult =
 Integrate[fcubic[x], {x, a, b}] // Expand
```

Out[315]=

$$-\frac{a^4 c}{4}+\frac{b^4 c}{4}-\frac{a^3 d}{3}+\frac{b^3 d}{3}-\frac{a^2 e}{2}+\frac{b^2 e}{2}-a f+b f$$

In[316]:=

```
simpsonresult == integralresult
```

Out[316]=

```
True
```

You Try It: Experiment with Simpson

Perform some numerical experiments like those in the preceding parts to show that the order of the error for Simpson's method is fourth. Also show that the local error is proportional to the fourth derivative of the integrand function. (We have put the solutions for this problem in the next two input cells with the code hidden. Try to do it on your own, and then look at our solutions. To reveal the hidden code, select the small cell bracket to the right of each cell, pull down the Cell menu, select Cell Properties, and click on Cell Open.)

```
The exact value of the integral is 1
```

Out[323]//TableForm=

n	h	Simpson estimate	Simpson error
2	$\frac{\pi}{4}$	1.00227987749221047770786711361673566658`20.	0.00227987749221047770786711361673566658`17.356922499855397
4	$\frac{\pi}{8}$	1.00013458497419390447591703613772821029`20.	0.00013458497419390447591703613772821029`16.128938129993145
8	$\frac{\pi}{16}$	1.00000829552396775878348786152484436433`20.	8.29552396775878348786152484436433`14.918840219756975*^-6
16	$\frac{\pi}{32}$	1.00000051668470650034553532299179356267`20.	5.16684706500345535322991793562 67`13.713225382535013*^-7
32	$\frac{\pi}{64}$	1.00000003226500096155111774425145743756`20.	3.22650009615511177442514574375 6`12.508731668373338*^-8
64	$\frac{\pi}{128}$	1.00000000201612870346150380267124483122`20.	2.01612870346150380267124483122`11.304518251808116*^-9
128	$\frac{\pi}{256}$	1.00000000012600126656435463902499224898`20.	1.26001266564354639024992248 98`10.100374910611617*^-10
256	$\frac{\pi}{512}$	1.00000000000787497326964557194418308038`20.	7.87497326964557194418308038`8.896249088316617*^-12
512	$\frac{\pi}{1024}$	1.00000000000049218417483633550431445695`20.	4.92184174836335504314456 95`7.692127645750676*^-13
1024	$\frac{\pi}{2048}$	1.00000000000003076148507554626758567327`20.	3.07614850755462675856732 7`6.488007298117233*^-14
2048	$\frac{\pi}{4096}$	1.00000000000000192259241328881752122619`20.	1.92259241328881752122619`5.283887224216928*^-15

```
The exact value of the integral is 2.60417 2^i
```

Out[332]//TableForm=

$f^{(4)}(x)$	exact	Simpson estimate	Simpson error
0.1`	2.60416666666666651863693005`12.	2.60416668402777773394518590 07`11.999999999999998	1.7361111215308256`*^-8
0.2`	5.20833333333333330372738601`12.	5.20833336805555546789037180 13`11.999999999999998	3.472222243061651`*^-8
0.4`	10.41666666666666607454772019 99`12.	10.41666673611111093578074360 27`11.999999999999998	6.944444486123302`*^-8
0.8`	20.83333333333333214909544039 98`12.	20.83333472222221871561487205 3`11.999999999999998	1.3888888972246605`*^-7
1.6`	41.66666666666666429819088079 97`12.	41.66666694444444374312297441 07`11.999999999999998	2.777777794449321`*^-7
3.2`	83.33333333333332859638176159 93`12.	83.33333388888888748624594882 13`11.999999999999998	5.555555588898642`*^-7
6.4`	166.66666666666665719276352319 87`12.	166.66666777777777497249189764 26`11.999999999999998	1.1111111177797284`*^-6
12.8`	333.33333333333333143855`12.	333.333335555555549945`11.999999999999998	2.2222222355594567`*^-6
25.6`	666.66666666666666287711`12.	666.66667111111109989`11.999999999999998	4.4444444711189135`*^-6
51.2`	1333.33333333333332575421`12.	1333.33334222222221997799`11.999999999999998	8.888888942237827`*^-6
102.4`	2666.66666666666665150842`12.	2666.66668444444443995599`11.999999999999998	0.000017777777884475654`