Numerical Integration

Author	**Aaron Tresham**
Date	**2017-06-12T18:54:55**
Project	**9189c752-e334-4311-afa9-605b6159620a**
Location	**02 - Numerical Integration Assignment/Numerical Integration Notes.sagews**
Original file	**Numerical Integration Notes.sagews**

Numerical Integration

Some antiderivatives are difficult (or impossible) to compute. In such cases, numerical approximation of a definite integral may be a better (or the only) option.

In Calculus 1, we learned about the "numerical_integral" command in Sage. We are *not* going to use this command in this lab. Instead, we will explore various approximation techniques.

Riemann Sums

The simplest numerical approximation comes from the definition of the definite integral as a limit of Riemann sums. Each Riemann sum is an approximation of the definite integral using rectangles (with one side on the x-axis), and we can make the approximation better by increasing the number of rectangles.

Our goal is to approximate $\displaystyle\int_a^b f(x)\,dx$.

We will use n rectangles of equal width, which means the width of each rectangle is $\Delta x = \frac{b-a}{n}$.

The height of the rectangle will be given by the value of the function f at some point in the base of that rectangle.

If we choose the left endpoint of each rectangle, we will find the left Riemann sum. If we choose the right endpoint of each rectangle, we will find the right Riemann sum.

The endpoints of the rectangles are $a,\ a+\Delta x,\ a+2\Delta x,\ a+3\Delta x, \ldots,\ a+n\Delta x = b$.

So the left Riemann sum is $LS = f(a)\cdot\Delta x + f(a+\Delta x)\cdot\Delta x + \cdots + f(a+(n-1)\Delta x)\cdot\Delta x = \displaystyle\sum_{i=0}^{n-1}\left(f(a+i\Delta x)\cdot\Delta x\right)$.

Similarly, the right Riemann sum is $RS = f(a+\Delta x)\cdot\Delta x + f(a+2\Delta x)\cdot\Delta x + \cdots + f(b)\cdot\Delta x = \displaystyle\sum_{i=1}^{n}\left(f(a+i\Delta x)\cdot\Delta x\right)$.

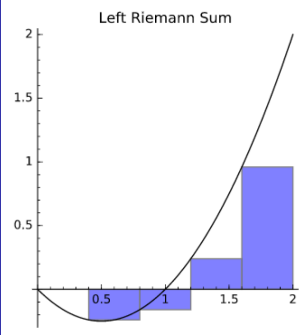
The only difference in these formulas is the index of summation.

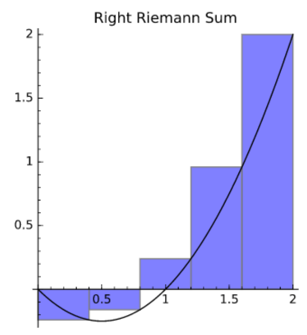
Example 1

Consider $\displaystyle\int_0^2 x^2 - x\,dx$.

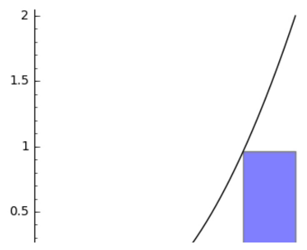
Here are pictures with $n=5$ rectangles.

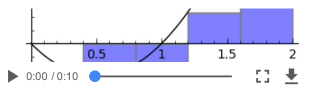
```
f(x)=x^2-x      #function to be integrated
a=0             #lower limit of integration
b=2             #upper limit of integration
n=5             #number of subintervals
dx=RR((b-a)/n) #delta x - the RR converts to a decimal, which simplifies things
p=plot(f,(a,b),color='black');q=p
for i in [0..n-1]: #adds rectangles for left sum
    p=p+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx)),(a+(i+1)*dx,f(a+i*dx)),(a+(i+1)*dx,0)],alpha=.5)+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx)),(a+(i+1)*dx,f(a+i*dx)),(a+(i+1)*dx,0)],fill=False,color='gray')
p.show(title='Left Riemann Sum')
for i in [0..n-1]: #adds rectangles for right sum
    q=q+polygon([(a+i*dx,0),(a+i*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,0)],alpha=.5)+polygon([(a+i*dx,0),(a+i*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,0)],fill=False,color='gray')
q.show(title='Right Riemann Sum')
```





The animation below shows graphs of left Riemann sums for increasing values of n. You can see that the rectangles begin to fill in the area under the curve.





The formulas below compute the left and right Riemann sums.

```
f(x)=x^2-x   #function to be integrated
a=0          #lower limit of integration
b=2          #upper limit of integration
n=5          #number of subintervals
dx=(b-a)/n   #delta x
%var i
LS=sum(f(a+i*dx)*dx,i,0,n-1) #calculates left sum
print 'The left Riemann sum is',N(LS)
RS=sum(f(a+i*dx)*dx,i,1,n)   #calculates right sum
print 'The right Riemann sum is',N(RS)
```

```
The left Riemann sum is 0.320000000000000
The right Riemann sum is 1.12000000000000
```

For the sake of comparison, the exact value is $\frac{2}{3} \approx 0.66667$.

Let's change the number of rectangles to 10 and see how our approximation improves.

```
f(x)=x^2-x
a=0
b=2
n=10        #Just changed this to 10
dx=(b-a)/n
%var i
LS=sum(f(a+i*dx)*dx,i,0,n-1) #calculates left sum
print 'The left Riemann sum is',N(LS)
RS=sum(f(a+i*dx)*dx,i,1,n)   #calculates right sum
print 'The right Riemann sum is',N(RS)
```

```
The left Riemann sum is 0.480000000000000
The right Riemann sum is 0.880000000000000
```

Remember that the exact answer is $\frac{2}{3}$. Both of these are closer to the correct answer than before, but they are still not very close.

Of course, we can increase our accuracy by using larger values of n, but it would be very nice if there were more accurate approximation methods. Fortunately, there are.

Midpoint Rule

The Midpoint Rule is another Riemann sum approximation, but instead of using the left or right endpoints, we will use the midpoint of each subinterval.

Notice in our example above, that each rectangle is either too big (an over-estimate) or too small (an under-estimate).

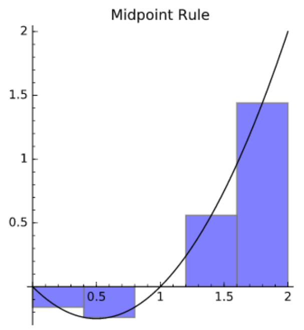
If we use the midpoint, then part of the rectangle will be above the curve and part will be below, so they will tend to cancel out and give us a better estimate.

Example 2

Consider $\int_0^2 x^2 - x\,dx$.

Here is a plot and calculation for the Midpoint Rule using $n = 5$.

```
f(x)=x^2-x    #function to be integrated
a=0           #lower limit of integration
b=2           #upper limit of integration
n=5           #number of subintervals
dx=(b-a)/n    #delta x
m=plot(f,(a,b),color='black')
for i in [0..n-1]: #sets up the plot
    m=m+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx+dx/2)),(a+(i+1)*dx,f(a+i*dx+dx/2)),(a+(i+1)*dx,0)],alpha=.5)+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx+dx/2)),(a+(i+1)*dx,f(a+i*dx+dx/2)),(a+(i+1)*dx,0)],fill=False,color='gray')
m.show(title='Midpoint Rule')
```



```
%var i
M=sum(f(a+i*dx+dx/2)*dx,i,0,n-1) #performs the calculation
print 'The Midpoint Rule gives',N(M)
```

```
The Midpoint Rule gives 0.640000000000000
```

The exact value is $\frac{2}{3}$, and the Midpoint Rule gives us the correct answer to one decimal place with only 5 rectangles.

Recall that for $n = 5$, the left sum was 0.32 and the right sum was 1.12. The Midpoint Rule has done a good job of balancing out the under- and over-estimates.

Trapezoidal Rule

So far we have been approximating our function using a constant (horizontal line = degree 0 polynomial) on each subinterval.

We may be able to improve our approximation if we use a degree 1 polynomial instead (i.e., a non-horizontal line). For each subinterval, we'll use the secant line based on the left and right endpoints. Of course, the resulting shape is not a rectangle but a trapezoid. Thus, this approach is called the Trapezoidal Rule.

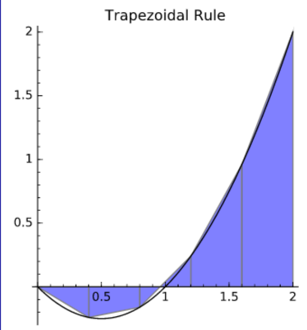
Recall that the area of a trapezoid is $\frac{b_1+b_2}{2}h$, where b_1 and b_2 are the lengths of the bases and h is the height. In this case, the trapezoid is sitting on its side, so the bases are actually vertical and the height is Δx.

Example 3

Consider $\displaystyle\int_0^2 x^2 - x\,dx$.

Here is the plot and calculation for the Trapezoidal Rule using $n = 5$.

```
f(x)=x^2-x   #function to be integrated
a=0          #lower limit of integration
b=2          #upper limit of integration
n=5          #number of subintervals
dx=(b-a)/n   #delta x
t=plot(f,(a,b),color='black')
for i in [0..n-1]: #sets up the plot
    t=t+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx)),(a+(i+1)*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,0)],alpha=.5)+polygon([(a+i*dx,0),(a+i*dx,f(a+i*dx)),(a+(i+1)*dx,f(a+(i+1)*dx)),(a+(i+1)*dx,0)],fill=False,color='gray')
t.show(title='Trapezoidal Rule')
```



```
%var i
T=sum((f(a+i*dx)+f(a+(i+1)*dx))*dx/2,i,0,n-1) #performs the calculation
print 'The Trapezoidal Rule gives',N(T)
```

```
The Trapezoidal Rule gives 0.720000000000000
```

Notice that this approximation is worse than what we got from the Midpoint Rule, but it is much better than either the left or right sum.

In fact, numerically the Trapezoidal Rule simply gives the average of the left and right sums!

In general (but not always), the Midpoint Rule will give a better approximation. If the function f is either concave up or concave down on the entire subinterval, then the linear approximation from the Trapezoidal Rule (the secant line) will be entirely above or below the curve. On the other hand, the horizontal line segment from the Midpoint Rule will be above the curve on part of the interval and below the curve on part of the interval, so the errors will cancel out.

Despite this numerical disappointment, the idea of increasing the degree of the approximating polynomial was sound. The next step is to use a degree 2 polynomial (quadratic, parabola) to approximate f on each subinterval. This is called Simpson's Rule.

Simpson's Rule

Instead of using a line to approximate our curve on each subinterval, we will use a parabola. Since a parabola is usually closer to the curve than a line, this should give us a better approximation (of course, this is not true if our original curve is a line, but in that case, why are we approximating the integral?).

A line is determined by 2 points, but it takes 3 points to determine a parabola.

The three points that are normally used are the left endpoints of three consecutive subintervals. This choice forces us to use an even number of subintervals (i.e., n must be even). The number of approximating parabolas is then $n/2$.

I won't take you through all the calculations (see the textbook, pages 454-456). The algebra is complicated, but here is the final result:

$$\begin{aligned}&\int_a^b f(x)\,dx\\ &\approx \frac{\Delta x}{3}(f(a)+4f(a+\Delta x)+2f(a+2\Delta x)+4f(a+3\Delta x)+2f(a+4\Delta x)+\cdots\\ &\qquad +2f(a+(n-2)\Delta x)+4f(a+(n-1)\Delta x)+f(b))\end{aligned}$$

Notice the pattern in the coefficients: $1, 4, 2, 4, 2, \ldots, 2, 4, 1$

Example 4

Consider $\displaystyle\int_0^2 x^2 - x\,dx$.

Here is the result for Simpson's Rule using $n = 10$ (5 parabolas).

```
f(x)=x^2-x     #function to be integrated
a=0            #lower limit of integration
b=2            #upper limit of integration
n=10            #number of subintervals (must be even)
dx=(b-a)/n     #delta x
n2=int(n/2)
coeffs = [4,2]*n2
coeffs = [1] +coeffs[:n-1]+[1]
S=(dx/3)*sum([coeffs[k]*f(a+dx*k) for k in [0..n]])
print 'Simpson\'s Rule gives',N(S)
```

```
Simpson's Rule gives 0.666666666666667
```

Of course, in this example $f(x) = x^2 - x$, which is already a parabola, so Simpson's Rule gives the exact answer.

It is interesting to note that numerically the answer from Simpson's Rule is a weighted average of the answers from the Trapezoidal Rule and the Midpoint Rule (with half the number of rectangles). In particular, if S_{2n} is the approximation from Simpson's Rule with $2n$ subintervals (n parabolas), T_n is the approximation from the Trapezoidal Rule with n trapezoids, and M_n is the approximation from the Midpoint Rule with n rectangles, then

$$S_{2n} = \frac{T_n + 2M_n}{3}$$

Here is the calculation for the examples above:

```
(T+2*M)/3
```

```
2/3
```

Example 5 (and a one-stop shop for copy-and-paste)

Here is one example that puts all the rules together. (I'll skip the graphs again.).

Let's approximate $\int_1^4 x^5 - 4x^2 + 6x - 9\,dx$ using $n = 10$, $n = 50$, and $n = 100$.

```
f(x)=x^5-4*x^2+6*x-9    #function to be integrated
a=1                     #lower limit of integration
b=4                     #upper limit of integration
n=10                    #number of subintervals
dx=RR(b-a)/n            #delta x (the RR converts to decimal, which speeds things up)
%var i

#Left Riemann Sum
LS=sum(f(a+i*dx)*dx,i,0,n-1) #calculates left sum
print 'The left Riemann sum is   ',N(LS)

#Right Riemann Sum
RS=sum(f(a+i*dx)*dx,i,1,n)   #calculates right sum
print 'The right Riemann sum is  ',N(RS)

#Midpoint Rule

M=sum(f(a+i*dx+dx/2)*dx,i,0,n-1)
print 'The Midpoint Rule gives   ',N(M)

#Trapezoidal Rule
T=sum((f(a+i*dx)+f(a+(i+1)*dx))*dx/2,i,0,n-1)
print 'The Trapezoidal Rule gives',N(T)

#Simpson's Rule
n2=int(n/2)
coeffs = [4,2]*n2
coeffs = [1] +coeffs[:n-1]+[1]
S=(dx/3)*sum([coeffs[k]*f(a+dx*k) for k in [0..n]])
print 'Simpson\'s Rule gives      ',N(S)
```

```
The left Riemann sum is    478.722375000000
The right Riemann sum is   773.022375000000
The Midpoint Rule gives    611.817609375000
The Trapezoidal Rule gives 625.872375000000
Simpson's Rule gives       616.540500000000
```

Compare these to the exact value of 616.5.

Now let's try $n = 50$:

```
f(x)=x^5-4*x^2+6*x-9    #function to be integrated
a=1                     #lower limit of integration
b=4                     #upper limit of integration
n=50                    #number of subintervals
dx=RR(b-a)/n
%var i

#Left Riemann Sum
LS=sum(f(a+i*dx)*dx,i,0,n-1) #calculates left sum
print 'The left Riemann sum is   ',N(LS)

#Right Riemann Sum
RS=sum(f(a+i*dx)*dx,i,1,n)   #calculates right sum
print 'The right Riemann sum is  ',N(RS)

#Midpoint Rule
M=sum(f(a+i*dx+dx/2)*dx,i,0,n-1)
print 'The Midpoint Rule gives   ',N(M)

#Trapezoidal Rule
T=sum((f(a+i*dx)+f(a+(i+1)*dx))*dx/2,i,0,n-1)
print 'The Trapezoidal Rule gives',N(T)

#Simpson's Rule
n2=int(n/2)
coeffs = [4,2]*n2
coeffs = [1] +coeffs[:n-1]+[1]
S=(dx/3)*sum([coeffs[k]*f(a+dx*k) for k in [0..n]])
print 'Simpson\'s Rule gives      ',N(S)
```

```
The left Riemann sum is    587.445283800000
The right Riemann sum is   646.305283800000
The Midpoint Rule gives    616.312364175000
The Trapezoidal Rule gives 616.875283800000
Simpson's Rule gives       616.500064800000
```

Compare these to the exact value of 616.5.

Now let's try $n = 100$:

```
f(x)=x^5-4*x^2+6*x-9    #function to be integrated
a=1                     #lower limit of integration
b=4                     #upper limit of integration
n=100                   #number of subintervals
dx=RR(b-a)/n
%var i
```

```
%var i

#Left Riemann Sum
LS=sum(f(a+i*dx)*dx,i,0,n-1) #calculates left sum
print 'The left Riemann sum is   ',N(LS)

#Right Riemann Sum
RS=sum(f(a+i*dx)*dx,i,1,n)   #calculates right sum
print 'The right Riemann sum is  ',N(RS)

#Midpoint Rule
M=sum(f(a+i*dx+dx/2)*dx,i,0,n-1)
print 'The Midpoint Rule gives   ',N(M)

#Trapezoidal Rule
T=sum((f(a+i*dx)+f(a+(i+1)*dx))*dx/2,i,0,n-1)
print 'The Trapezoidal Rule gives',N(T)

#Simpson's Rule
n2=int(n/2)
coeffs = [4,2]*n2
coeffs = [1] +coeffs[:n-1]+[1]
S=(dx/3)*sum([coeffs[k]*f(a+dx*k) for k in [0..n]])
print 'Simpson\'s Rule gives      ',N(S)
```

```
The left Riemann sum is    601.878823987500
The right Riemann sum is   631.308823987500
The Midpoint Rule gives    616.453088385937
The Trapezoidal Rule gives 616.593823987500
Simpson's Rule gives       616.500004050000
```

Compare these to the exact value of 616.5.

What do we see from this example?

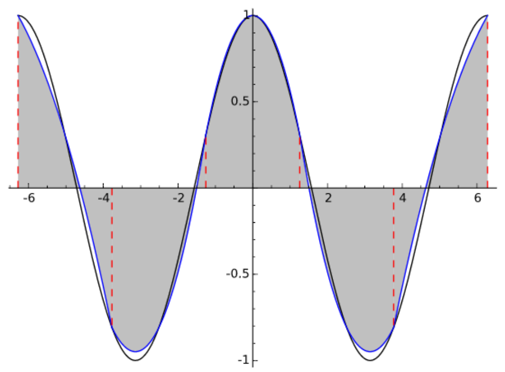
1. All the approximations improve as n increases.
2. The left and right Riemann sums are not very good approximations.
3. The Midpoint Rule and Trapezoidal Rule are better than the left and right sums.
4. The Midpoint Rule is better than the Trapezoidal Rule.
5. Simpson's Rule is the best.

Example 6 - Simpson's Rule Graph

The function to be integrated, $f(x) = \cos(x)$, is in black. The red vertical lines mark the ends of each parabola. The approximating parabolas are in blue.

I have set $n = 10$ (5 parabolas).

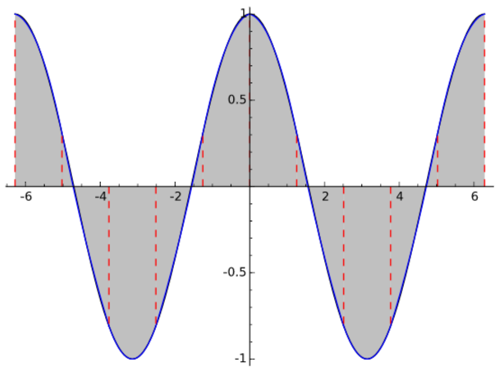
```
f(x)=cos(x)    #function to be integrated
a=-2*pi        #lower limit of integration
b=2*pi         #upper limit of integration
n=10           #number of subintervals
dx=(b-a)/n
v=[a]
w=[f(a)]
p=plot(f,(a,b),color='black')
for i in [0..n-1]:
    v=v+[a+(i+1)*dx]
    w=w+[f(a+(i+1)*dx)]
for i in [0,2..n-1]:
    %var A, B, C
    eqn1=A*v[i]^2+B*v[i]+C==w[i]
    eqn2=A*v[i+1]^2+B*v[i+1]+C==w[i+1]
    eqn3=A*v[i+2]^2+B*v[i+2]+C==w[i+2]
    coeff=solve([eqn1,eqn2,eqn3],A,B,C)
    A=coeff[0][0].rhs();B=coeff[0][1].rhs();C=coeff[0][2].rhs()
    p+=plot(A*x^2+B*x+C,(v[i],v[i+2]),fill='axis')+line([(v[i],0),(v[i],f(v[i]))],color='red',linestyle='--')
p+=line([(v[n],0),(v[n],f(v[n]))],color='red',linestyle='--');p
```



Here's the graph with $n = 20$ (10 parabolas).

```
f(x)=cos(x)    #function to be integrated
a=-2*pi        #lower limit of integration
b=2*pi         #upper limit of integration
n=20           #number of subintervals
```

```
dx=(b-a)/n
v=[a]
w=[f(a)]
p=plot(f,(a,b),color='black')
for i in [0..n-1]:
    v=v+[a+(i+1)*dx]
    w=w+[f(a+(i+1)*dx)]
for i in [0,2..n-1]:
    %var A, B, C
    eqn1=A*v[i]^2+B*v[i]+C==w[i]
    eqn2=A*v[i+1]^2+B*v[i+1]+C==w[i+1]
    eqn3=A*v[i+2]^2+B*v[i+2]+C==w[i+2]
    coeff=solve([eqn1,eqn2,eqn3],A,B,C)
    A=coeff[0][0].rhs();B=coeff[0][1].rhs();C=coeff[0][2].rhs()
    p+=plot(A*x^2+B*x+C,(v[i],v[i+2]),fill='axis')+line([(v[i],0),(v[i],f(v[i]))],color='red',linestyle='--')
p+=line([(v[n],0),(v[n],f(v[n]))],color='red',linestyle='--');p
```


1
0.5
-0.5
-1
-6
-4
-2
2
4
6