

Sage Calculus Commands

Although our initial Introduction to *Sage* included discussion of a number of *Sage* functions related to calculus, you may not have noticed that fact if you didn't look carefully at what *Sage* can do. This handout is, in the first instance, just a summary of *Sage's* calculus-related functionality. As always, I'm only scratching the surface. The *Sage* help command (say **derivative?** to get help on **derivative**, for instance) offers a lot more information on any of the commands used in this Worksheet.

Basic Commands

Limits

Sage's command for limits is, logically enough, **limit**. Notice how *Sage* handles the limits below, distinguishing among infinite limits, undefined limits, and indeterminate limits for functions that are still locally bounded.

```
var('x')
limit(sin(x)/x, x=0)
```

1

```
limit(1/x, x=0)
```

und

```
limit(1/x^2, x=0)
```

+∞

```
limit(sin(1/x), x=0)
```

ind

Our investigation of derivatives could have been speeded up substantially had we been willing to use *Sage* uncritically as a black box for taking limits. Here, for instance, are calculations of the derivatives of the sine, exponential, and cube root functions. It would be useful review to try to remember how we obtained these results in class. Are you sure *Sage* doesn't have bugs that affect these calculations?

These calculation show a pretty serious annoyance in *Sage*: In order to compute two of the limits, it needs to know whether various quantities are positive and whether they are integers, even though the result of the calculation doesn't depend on the answers to these questions. We therefore end up having to use *Sage's* **assume()** function to give it various choices of sign and type for numbers.

```
var('h')
limit((sin(x+h) - sin(x))/h, h=0)
```

```
Traceback (click to the left for traceback)
...
Is sin(x) positive, negative, or zero?
```

```
assume(sin(x) > 0)
limit((sin(x+h) - sin(x))/h, h=0)
```

```
Traceback (click to the left for traceback)
...
```

Is $\cos(x)$ positive, negative, or zero?

```
assume(cos(x) > 0)
limit((sin(x+h) - sin(x))/h, h=0)
```

$\cos(x)$

```
forget()
assume(sin(x)<0, cos(x)>0)
limit((sin(x+h) - sin(x))/h, h=0)
```

$\cos(x)$

```
forget()
assume(sin(x)<0, cos(x)<0)
limit((sin(x+h) - sin(x))/h, h=0)
```

$\cos(x)$

```
forget()
```

```
limit((exp(x+h) - exp(x))/h, h=0)
```

Traceback (click to the left for traceback)

...

Is x an integer?

```
assume(x, 'integer')
limit((exp(x+h) - exp(x))/h, h=0)
```

e^x

```
forget()
assume(x, 'noninteger')
limit((exp(x+h) - exp(x))/h, h=0)
```

e^x

```
forget()
```

```
var('a')
limit((x^(1/3) - a^(1/3))/(x-a), x=a)
```

$\frac{1}{3} \frac{1}{a^{2/3}}$

Here's one last limit, just for fun. You might think about how to do it by hand.

```
limit(x^x, x=0)
```

Traceback (click to the left for traceback)

...

Is x positive or negative?

```
limit(x^x, x=0, dir='plus')
```

1

```
limit(x^x, x=0, dir='minus')
```

Derivatives

Sage can also compute derivatives, either of expressions or of functions. **derivative()**, **differentiate()**, and **diff()** are three different names for the same function (or method) for doing this.

```
nestrاد = sqrt(x+sqrt(x+sqrt(x)))
nestrاد
```

$$\sqrt{x + \sqrt{x + \sqrt{x}}}$$

```
diff(nestrاد,x)
```

$$\frac{1}{8} \frac{(\sqrt{x}+2)}{\sqrt{x+\sqrt{x}}}$$

```
_.simplify_rational()
```

$$\frac{1}{8} \frac{(4\sqrt{x+\sqrt{x}}\sqrt{x}+2\sqrt{x}+1)}{\sqrt{x+\sqrt{x}}\sqrt{x+\sqrt{x+\sqrt{x}}}}$$

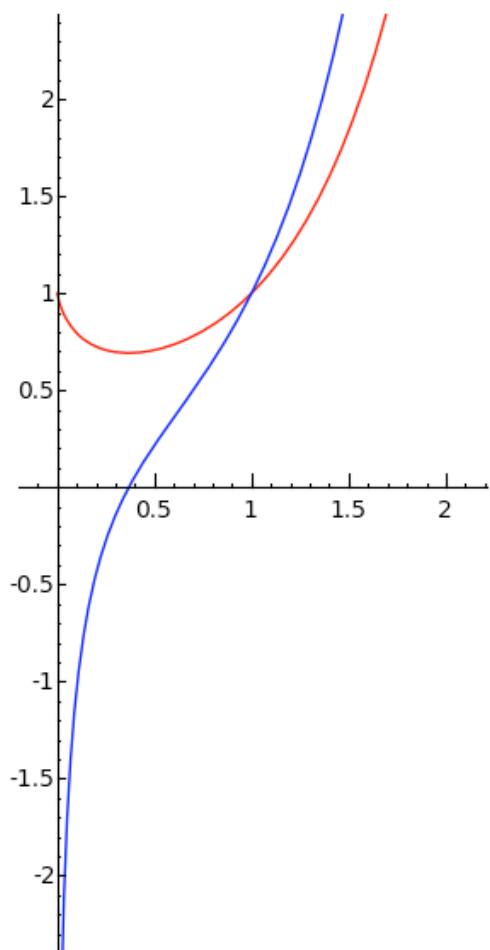
```
f(x)=x^x
f
```

$$x \mapsto x^x$$

```
fprime = diff(f)
fprime
```

$$x \mapsto (\ln(x) + 1)x^x$$

```
pf = plot(f, (x,0,2), color='red')
pfprime = plot(fprime, (x,0,2), color='blue')
show(pf+pfprime,aspect_ratio=1, ymin=-2, ymax=2)
```



$\text{diff}(\sin(x),x)$

$\cos(x)$

$\text{diff}(\sin(x),x,4)$

$\sin(x)$

$\sin(x).\text{diff}(x,21)$

$\cos(x)$

$f(x)=\sin(x)$

$\text{diff}(f)$

$x \mapsto \cos(x)$

$\text{diff}(f,x,4)$

$x \mapsto \sin(x)$

$\text{diff}(f,x,4)(\pi/4)$

$\frac{1}{2}\sqrt{2}$

Sage can even handle differentiation rules for arbitrary functions, like this:

```
function('f g h')
var('x')
```

x

```
diff(f(x)*g(x),x)
```

$$D[0](f)(x)g(x) + f(x)D[0](g)(x)$$

```
diff(f(x)/g(x), x)
```

$$\frac{D[0](f)(x)}{g(x)} - \frac{f(x)D[0](g)(x)}{g(x)^2}$$

```
diff(f(x)*g(x)*h(x),x)
```

$$D[0](f)(x)g(x)h(x) + f(x)D[0](g)(x)h(x) + f(x)g(x)D[0](h)(x)$$

```
diff(f(g(x)),x)
```

$$D[0](f)(g(x))D[0](g)(x)$$

```
diff(sqrt(f(x)),x)
```

$$\frac{1}{2} \frac{D[0](f)(x)}{\sqrt{f(x)}}$$

Integrals

Sage can also compute integrals, naturally using the function (or method) **integral()**.

```
integral(sin(x), x,0,pi)
```

2

```
(1/x).integrate(x,1,e)
```

```
Traceback (click to the left for traceback)
...
Is x+%e positive or negative?
```

Aargh! Another squabble over signs. This could get old.

```
assume(x>0)
```

```
(1/x).integrate(x,1,e)
```

1

```
forget()
```

```
integral(x/(1+x^2),x)
```

$$\frac{1}{2} \ln(x^2 + 1)$$

```
integral(x/(1+x^4),x)
```

$$\frac{1}{2} \arctan(x^2)$$

Big deal so far, but here's one I bet you can't do by hand, $\int \frac{x}{1+x^3} dx$.

```
integral(x/(1+x^3),x)
```

$$\frac{1}{3} \sqrt{3} \arctan\left(\frac{1}{3}(2x-1)\sqrt{3}\right) - \frac{1}{3} \ln(x+1) + \frac{1}{6} \ln(x^2-x+1)$$

```
integral(x^3*e^x,x)
```

$$(x^3 - 3x^2 + 6x - 6)e^x$$

```
var('n')
```

```
integral(x^n,x)
```

```
Traceback (click to the left for traceback)
```

```
...
```

```
Is n+1 zero or nonzero?
```

Ah, this is good! *Sage* should ask whether or not $n = -1$. After all, it makes a difference:

```
assume(n != -1)
```

```
integral(x^n,x)
```

$$\frac{x^{(n+1)}}{(n+1)}$$

```
forget()
```

```
integral(x^(-1),x)
```

$$\ln(x)$$

Sums

Sums are done with slightly unusual notation. **[1..10]** means the list **[1,2,3,4,5,6,7,8,9,10]**. For convenience, **range(n)** is the list of integers x such that $0 \leq x < n$. (Watch the endpoints here!)

```
range(5)
```

$$[0, 1, 2, 3, 4]$$

```
sum(k for k in [1..10])
```

$$55$$

```
var('n')
```

$$n$$

Unfortunately, *Sage* doesn't directly support sums with variable or infinite endpoints. *Maxima*, another computer algebra system that is one of the tools called by *Sage* behind the scenes, does support sums like this. Here's the notation we would use to get *Maxima* to compute the sums

$$\sum_{k=1}^n k, \quad \sum_{k=1}^n k^2, \quad \sum_{k=1}^n k^3, \quad \sum_{k=1}^{\infty} \frac{1}{k^2}, \quad \sum_{k=1}^{\infty} \frac{1}{k^4}, \quad \text{and} \quad \sum_{k=1}^{\infty} \frac{1}{k^3}.$$

Notice that *Maxima's* notation for sums is somewhat different from *Sage's*.

```
maxima('sum(k, k, 1, n), simpsum')
```

$$\frac{n^2+n}{2}$$

For technical reasons (for instance, if we want to take limits of the sums we produce or to evaluate them numerically), it is better to use a slightly modified version of *Maxima* and to turn the output from a string to a *Sage* symbolic ring object using the command **SR**. A slightly better set of magic words to use is therefore (just trust me on this)

```
SR(sage.calculus.calculus.maxima('sum(k, k, 1, n), simpsum'))
```

$$\frac{1}{2}n^2 + \frac{1}{2}n$$

```
factor(_)
```

$$\frac{1}{2}(n+1)n$$

Of course, if we're using that string again and again, we can define a shorter name for it.

```
def mymax(s):
```

```
    return SR(sage.calculus.calculus.maxima(s))
```

```
mymax('sum(k^2, k, 1, n), simpsum')
```

$$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

```
factor(_)
```

$$\frac{1}{6}(n+1)(2n+1)n$$

```
factor(mymax('sum(k^3, k, 1, n), simpsum'))
```

$$\frac{1}{4}(n+1)^2n^2$$

```
mymax('sum(1/k^2, k, 1, inf),simpsum')
```

$$\frac{1}{6}\pi^2$$

```
_.n()
```

$$1.64493406684823$$

```
mymax('sum(1/k^4, k, 1, inf),simpsum')
```

$$\frac{1}{90}\pi^4$$

```
mymax('sum(1/k^3, k, 1, inf),simpsum')
```

$$\zeta(3)$$

```
zeta(3).n()
```

$$1.20205690315959$$

Sage can compute integrals using Riemann sums, though again, we have to use *Maxima* and we fall afoul of some problems with the *Maxima* interface.

The right hand Riemann sum for computing the integral

$$\int_0^x t^3 dt$$

is

$$\sum_{k=1}^n \left(\frac{kx}{n}\right)^3 \cdot \frac{x}{n},$$

which *Maxima* can write and evaluate.

```
mymax('sum((k*x/n)^3 * x/n, k, 1, n),simpsum')
```

$$\frac{1}{4} \frac{(n^4+2n^3+n^2)x^4}{n^4}$$

We now want to take the limit of this as $n \rightarrow \infty$ in order to get the integral, right?

```
limit(_, n=infinity)
```

$$\frac{1}{4} x^4$$

Bingo! So

$$\int_0^x t^3 dt = \frac{x^4}{4},$$

just as we would have suspected.

More Complicated Tools

As one learns to work with *Sage*, one finds more and more ways to visualize and to work with the ideas of calculus. Here, for instance, is a little interactive tool for illustrating the idea of the derivative of a function as the slope of the tangent line. The blue curve is the function $f(x)$, the dotted black curve is its derivative $f'(x)$. Moving the slider moves a point along the curve and the tangent line at that point. As the tangent line moves, the green dot on the derivative curve traces out its slope.

```
var('x')
f(x) = sin(x)*e^(-x)
fp = f.diff()
p = plot(f,-1,5, thickness=2)
pp = plot(fp, -1, 5, color='black', linestyle=':')
pt_list = (-0.4+0.1*j for j in range(45))
@interact
def_(pt=pt_list):
    dot = point((pt,f(pt)),pointsize=80,rgbcolor='red')
    dotp = point((pt, fp(pt)), pointsize=80, rgbcolor='green')
    slope = fp(pt)
    sp = plot(f(pt)+slope*(x-pt),(x,-1, 5), color='green', thickness=2)
    html('<font color=red>Tangent to y = exp(-x)sin(x) at x = %s</font>%pt)
    show(dot + dotp + p + pp + sp, ymin = -.5, ymax = 1)
```

And here are two functions for working with Riemann sums. **right_riemann_plot** plots a curve and its right-hand Riemann sum, and **right_riemann_value** gives the value of this sum.

```
def right_riemann_plot(f, a, b, n):
    p = plot(f, a, b, thickness=2)
    deltax = (b-a)/n
```

```

rectops = Piecewise([[(a+j*deltax, a+(j+1)*deltax),f(a+(j+1)*deltax)] for j in range(n)])
prectops = plot(rectops, color='green', fill='axis', fillcolor='green')
outline = []
for j in range(n):
    outline = outline + [(a+j*deltax,0), (a+j*deltax,f(a+(j+1)*deltax)), (a+(j+1)*deltax,f(a+(j+1)*deltax)), (a+(j+1)*deltax, 0)]
poutline = plot(line(outline, rgbcolor='black'))
show(p + prectops + poutline)

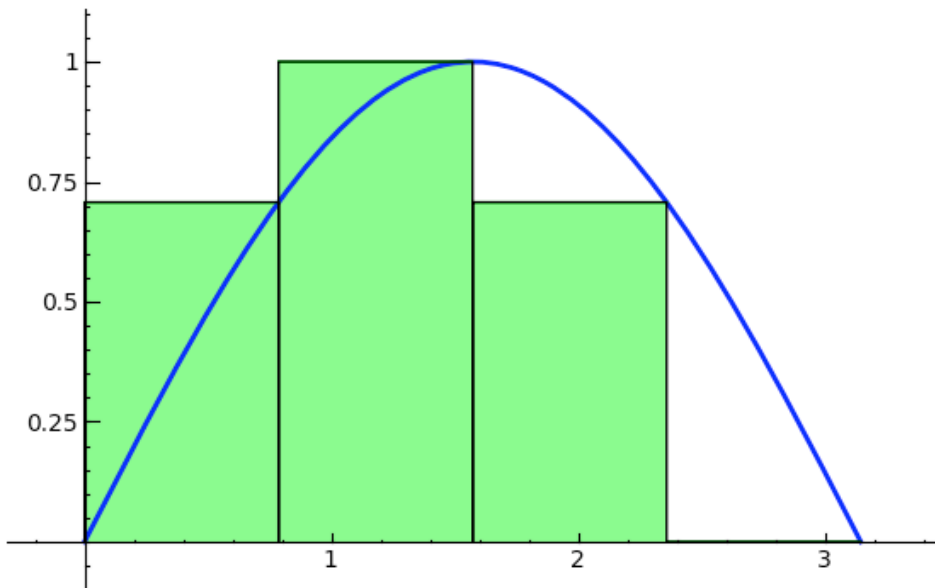
```

```

def right_riemann_value(f, a, b, n):
    deltax = (b-a)/n
    return(sum(f(a+(j+1)*deltax)*deltax for j in range(n)))

```

```
right_riemann_plot(sin,0,pi,4)
```



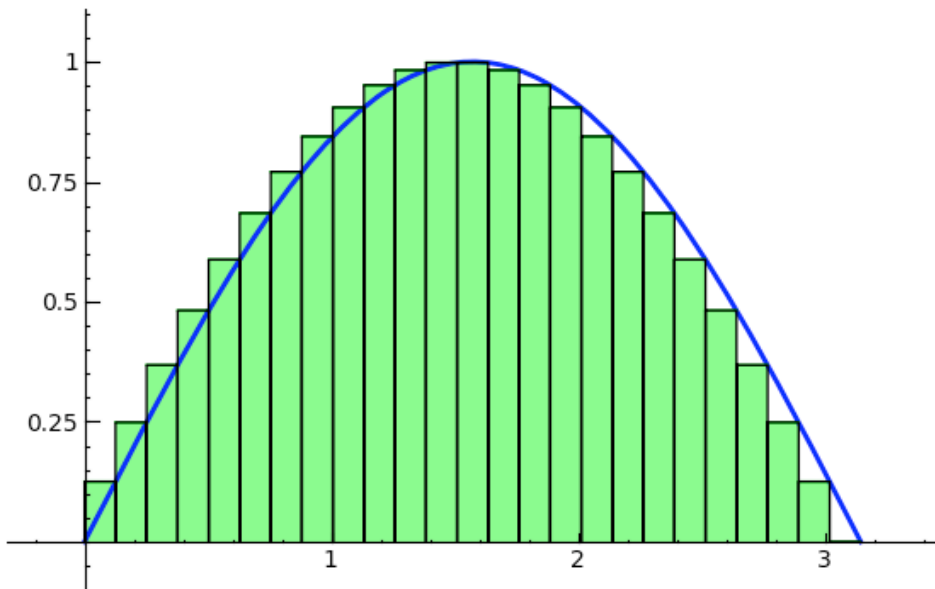
```
right_riemann_value(sin,0,pi,4)
```

$$\frac{1}{4}\pi + \frac{1}{4}\pi\sqrt{2}$$

```
right_riemann_value(sin,0,pi,4).n()
```

1.89611889793704

```
right_riemann_plot(sin,0,pi,25)
```



```
right_riemann_value(sin,0,pi,25).n()
```

```
1.99736741254563
```

```
integral(sin(x),x,0,pi)
```

```
2
```

Finally, here's a function that wraps f , a , b , and n up in a string and sends it to *Maxima*, which writes the formal Riemann sum. There are some twists and turns here because of the difficulties of moving between *Sage* and *Maxima*, but at least for simple functions, we end up with expressions where *Sage* can take the limit to evaluate the integral.

```
def right_riemann_sum(f,a,b,n):
```

```
    deltax = (b-a)/n
```

```
    return SR(sage.calculus.calculus.maxima('sum(' + str(f(a + sage.calculus.calculus.var('k') *deltax))*deltax) + ', k, 1,' + str(n) + ') ,simsup'))
```

```
right_riemann_sum(sin,0,1,3)
```

```
 $\frac{1}{3} \sin\left(\frac{1}{3}\right) + \frac{1}{3} \sin\left(\frac{2}{3}\right) + \frac{1}{3} \sin(1)$ 
```

```
right_riemann_sum(sin,0,1,3).n()
```

```
0.595678494891262
```

```
right_riemann_sum(sin,0,pi,4)
```

```
 $\frac{1}{4} \pi + \frac{1}{4} \pi \sqrt{2}$ 
```

```
f(x)=x^2
```

```
right_riemann_sum(f,0,1,5)
```

```
 $\frac{11}{25}$ 
```

```
var('n')
```

```
rs = right_riemann_sum(f,0,1,n)
```

```
rs
```

$$\frac{1}{6} \frac{(2n^3 + 3n^2 + n)}{n^3}$$

limit(rs, n=infinity)

$$\frac{1}{3}$$

integral(f(x), x, 0, 1)

$$\frac{1}{3}$$

var('z')

rs = right_riemann_sum(f, 1, z, n)

rs

$$\frac{1}{6} \frac{(z-1)(6n-6\frac{(n^2+n)}{n}+6\frac{(n^2z+nz)}{n}+\frac{(2n^3+3n^2+n)}{n^2}-2\frac{(2n^3z+3n^2z+nz)}{n^2}+\frac{(2n^3z^2+3n^2z^2+nz^2)}{n^2})}{n}$$

rs = rs.simplify_rational()

rs

$$\frac{1}{6} \frac{((2n^2+3n+1)z^3-3(n+1)z^2-3(n-1)z-2n^2+3n-1)}{n^2}$$

limit(rs, n=infinity)

$$\frac{1}{3} z^3 - \frac{1}{3}$$

integral(f(x), x, 1, z)

$$\frac{1}{3} z^3 - \frac{1}{3}$$

rs = right_riemann_sum(exp, 0, z, n)

rs

$$\frac{(e^{\frac{(n+1)z}{n}} - e^{\frac{z}{n}})z}{(e^{\frac{z}{n}} - 1)n}$$

rs = rs.simplify_rational()

rs

$$\frac{(ze^z - z)e^{\frac{z}{n}}}{(ne^{\frac{z}{n}} - n)}$$

limit(rs, n=infinity)

Traceback (click to the left for traceback)

...

Is z positive, negative, or zero?

assume(z>0)

limit(rs, n=infinity)

$$e^z - 1$$

integral(e^x, x, 0, z)

$$e^z - 1$$

Some Calculus Problems

As far as I can tell at this point, *Sage* doesn't have inert integrals like *Maple*, so some of the exercises in the *Maple* version of this worksheet which involve doing explicit changes of variable in inert integrals aren't so easy in *Sage*. Others of our exercises from the *Maple* version are easy. For instance, we can explicitly explore Taylor polynomials for the sine function by equating derivatives of $\sin x$ with derivatives of a generic polynomial at $x = 0$, and we can then interactively plot the first few Taylor polynomials.

```
var('a0 a1 a2 a3 a4 a5')
f(x) = a2*x^2 + a1*x + a0
s(x)=sin(x)
```

```
solve([f(0)==s(0),diff(f,x)(0)==diff(s,x)(0),diff(f,x,2)(0)==diff(s,x,2)(0)], a0,a1,a2)
```

```
[[a0 == 0, a1 == 1, a2 == 0]]
```

```
f(x) = a3*x^3 + a2*x^2 + a1*x + a0
```

```
solve([f(0)==s(0),diff(f,x)(0)==diff(s,x)(0),diff(f,x,2)(0)==diff(s,x,2)(0),diff(f,x,3)(0)==diff(s,x,3)(0)], a0,a1,a2,a3)
```

```
[[a0 == 0, a1 == 1, a2 == 0, a3 == (-1/6)]]
```

```
f(x) = a5*x^5 + a4*x^4 + a3*x^3 + a2*x^2 + a1*x + a0
```

```
solve(list(diff(f,x,k)(0)==diff(s,x,k)(0) for k in range(6)), a0,a1,a2,a3,a4,a5)
```

```
[[a0 == 0, a1 == 1, a2 == 0, a3 == (-1/6), a4 == 0, a5 == (1/120)]]
```

```
@interact
```

```
def _(terms=(j for j in range(17))):
```

```
    x = sage.calculus.calculus.var('x')
```

```
    sinplot = plot(sin(x), x,-18, 18, rgbcolor='blue')
```

```
    taypoly = sum((-1)^k*x^(2*k+1)/(2*k+1).factorial() for k in range(terms+1))
```

```
    tayplot = plot(taypoly, x, -15,15, rgbcolor='red')
```

```
    html('<font color=red>y = sin(x) and its Maclaurin series of degree %s</font>'%(2*terms+1))
```

```
    show(sinplot + tayplot, xmin=-18, xmax=18, ymin = -2, ymax = 2)
```