Newton's Method

Author	**Aaron Tresham**
Date	**2017-04-11T22:58:00**
Project	**a8975d68-235e-4f21-8635-2051d699f504**
Location	09 - Newton's Method Assignment/Newton's Method Notes.sagews
Original file	Newton's Method Notes.sagews

Newton's Method

Newton's Method is a tool that uses calculus to estimate the roots (zeros or x-intercepts) of a function.

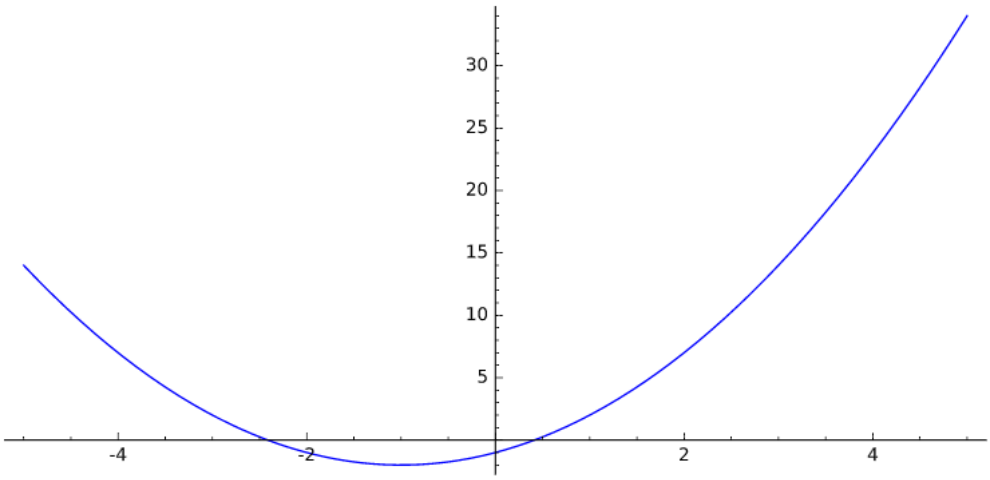
In a previous lab, we considered the find_root and solve commands in Sage, which can be used to find roots. The purpose of this lab is to give us some idea of the math behind these commands. There is no practical reason to use Newton's Method (unless you really need to find a root without a computer/calculator). However, it is instructive to see how calculus (wh not directly related to roots) can be used to estimate roots.

Newton's basic idea is that a differentiable function is "close" to its tangent lines (at least near the point of tangency). So if we know the root of a tangent line (easy algebra), we suppos the root of the function is somewhere close.

Example 1

Estimate the roots of $f(x) = x^2 + 2x - 1$. [Of course, we can find the roots of a quadratic function with the Quadratic Formula, but it's just an example.] Let's look at a graph:

```
1 f(x)=x^2+2*x-1
2 plot(f,xmin=-5,xmax=5)
```

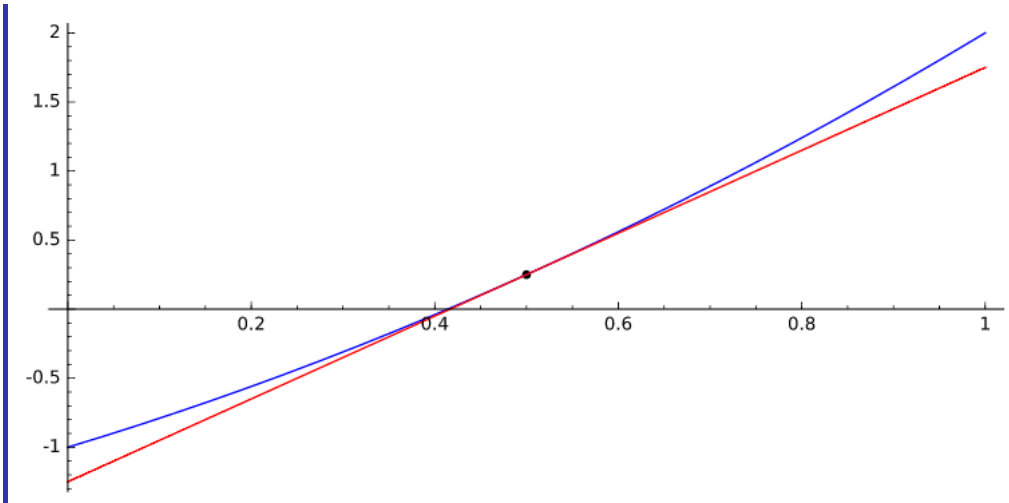


From this graph, it appears that there is a root near $x = \frac{1}{2}$.

Let's find the tangent line at this point. Remember, an equation for the tangent line at $x = x_0$ is $y = f(x_0) + f'(x_0)(x - x_0)$.

```
3 f(x)=x^2+2*x-1
4 df(x)=derivative(f,x)        #find the derivative
5 TL(x)=f(1/2)+df(1/2)*(x-1/2) #find the tangent line
6 print 'The tangent line is y =',TL(x)
7 plot(f,xmin=0,xmax=1)+plot(TL,xmin=0,xmax=1,color='red')+point((1/2,f(1/2)),color='black',size=25)
```

```
The tangent line is y = 3*x - 5/4
```



8

Notice that the tangent line and the graph of f are close together near $x = \frac{1}{2}$, and we can see that the root of the tangent line is close to the root of f.

Sage gives the equation of the tangent line as $y = 3x - \frac{5}{4}$. To find the root of this line, just plug in 0 for y and solve for x:

$0 = 3x - \frac{5}{4}$

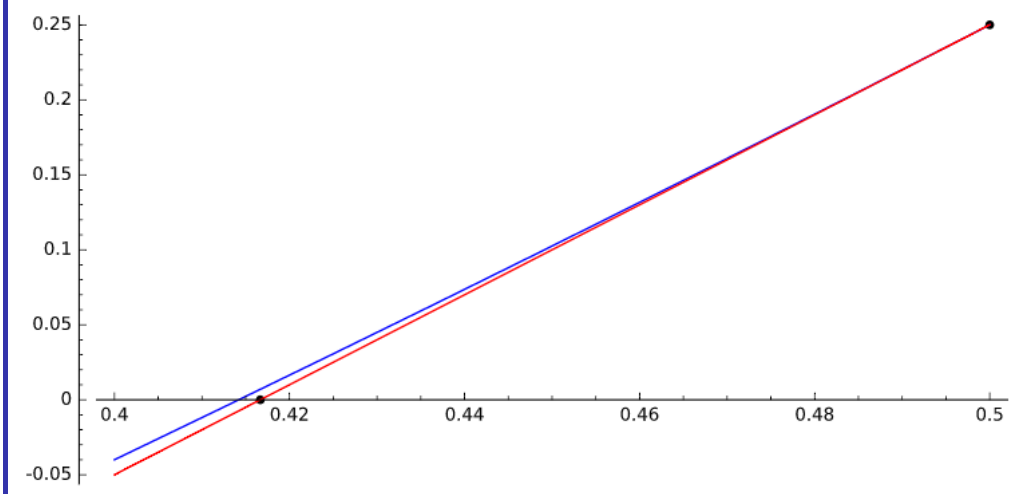
$\frac{5}{4} = 3x$

$\frac{\frac{5}{4}}{3} = x$

$x = \frac{5}{12} \approx 0.41666$

Let's zoom in on the graph and plot this root.

```
f(x)=x^2+2*x-1
df(x)=derivative(f,x)
TL(x)=f(1/2)+df(1/2)*(x-1/2)
plot(f,xmin=0.4,xmax=.5)+plot(TL,xmin=0.4,xmax=.5,color='red')+point((5/12,0),size=25,color='black')+point((1/2,f(1/2)),color='black',size=25)
```



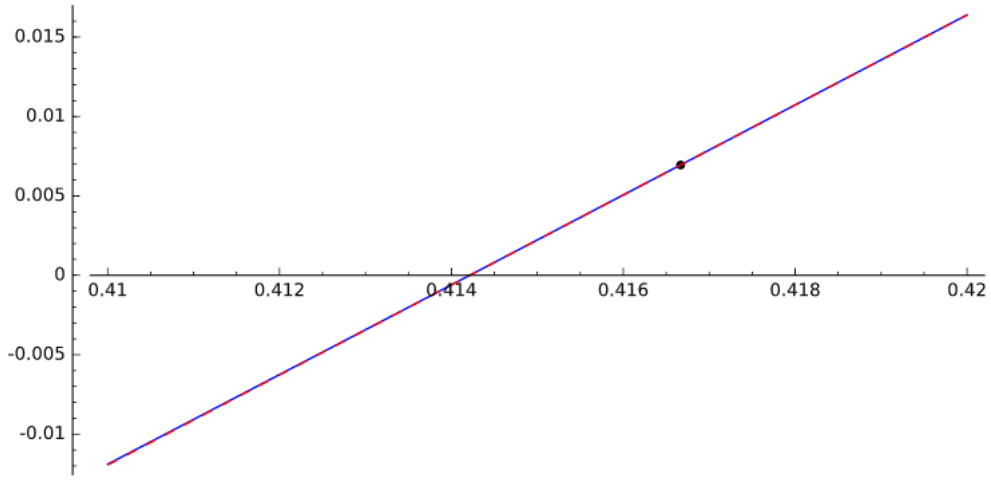
On this window, we can see that the root of f is not exactly $\frac{5}{12}$, but this is a good approximation.

If we want a better approximation, we can repeat the process, starting with a point of tangency that's closer to the root. Remember we started this example using $x = \frac{1}{2}$ as our point of tangency. This time, we'll use $x = \frac{5}{12}$, the approximation we got last time. We can see from the graph that this is closer to the root of f than $\frac{1}{2}$, so the tangent line we get from $x = \frac{5}{12}$ sh be even closer to the actual root of f.

So let's find a new tangent line:

```
f(x)=x^2+2*x-1
df(x)=derivative(f,x)
TL(x)=f(5/12)+df(5/12)*(x-5/12)
print 'The tangent line is y =',TL(x)
plot(f,xmin=.41,xmax=.42)+plot(TL,xmin=.41,xmax=.42,color='red',linestyle='--')+point((5/12,f(5/12)),size=25,color='black')
```

```
The tangent line is y = 17/6*x - 169/144
```



We can see that the tangent line and f are very close together on this window, so the root of the tangent line is very close to the root of f.

Let's find the root of the tangent line:

$0 = \frac{17}{6}x - \frac{169}{144}$

$\frac{169}{144} = \frac{17}{6}x$

$\frac{\frac{169}{144}}{\frac{17}{6}} = x$

$x = \frac{169}{408} \approx 0.414216$

Just for comparison, let's find the exact root of f.

```
f(x)=x^2+2*x-1
solve(f(x)==0,x)
```

```
[x == -sqrt(2) - 1, x == sqrt(2) - 1]
```

There are two solutions, $x = -\sqrt{2} - 1$ and $x = \sqrt{2} - 1$. We have been trying to estimate the postive root, which is $x = \sqrt{2} - 1 \approx 0.414214$.

This is very close to our estimate, which is off by only $\frac{169}{408} - (\sqrt{2} - 1) \approx 0.0000021239$ (about 0.0005% error).

If we wanted to improve our estimate even more, then we could repeat the process again, using $x = \frac{169}{408}$ as our new point of tangency.

I hope you see from this short example that this process involves repetition of the same steps over and over with different numbers. It lends itself very easily to be made into an algorith that can be automated.

Before we discuss this algorithm, I have two comments:

- We have to start the process with a point of tangency (we used $\frac{1}{2}$ in our example above). The closer this point is to the actual root, the better. In fact, if our point of tangency is chosen poorly, Newton's Method might never get close to the actual root, or it might take more repetitions than we want to use.
- Newton's Method only estimates one root at a time. If we had chosen a different point of tangency, we may have ended up at the other root of f (in our example, f has two roots)

Now let's develop the algorithm.

Generalizing Newton's Method

Given a function f with at least one root, the first step is to get a ballpark estimate for the root to use as the x-coordinate of our first point of tangency.

We'll use a graph to get this initial estimate, which we'll call x_0.

Now we find the tangent line to f when $x = x_0$:

$$TL(x) = f(x_0) + f'(x_0)(x - x_0)$$

Then we find the root of the tangent line by setting it equal to 0 and solving for x:

$$0 = f(x_0) + f'(x_0)(x - x_0)$$

First, distribute:

$$0 = f(x_0) + f'(x_0) \cdot x - f'(x_0) \cdot x_0$$

Then gather like terms:

$$f'(x_0) \cdot x_0 - f(x_0) = f'(x_0) \cdot x$$

Now divide both sides by $f'(x_0)$:

$$\frac{f'(x_0) \cdot x_0 - f(x_0)}{f'(x_0)} = x$$

And simplify:

$$x_0 - \frac{f(x_0)}{f'(x_0)} = x$$

This is the root of the tangent line, so this becomes our new estimate for the root of f. Let's call it x_1. Thus,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

To improve our approximation, use $x = x_1$ as our new point of tangency and repeat the process. The algebra is exactly the same. Simply replace all the x_0 in the above equations with x

What we get is a new approximation, x_2, given by

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

In general, if we have repeated this process n times to get an estimate x_n, then repeating the process again will give a new estimate, x_{n+1} given by:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Example 2

Now we're ready to lay out an algorithm for Newton's Method.

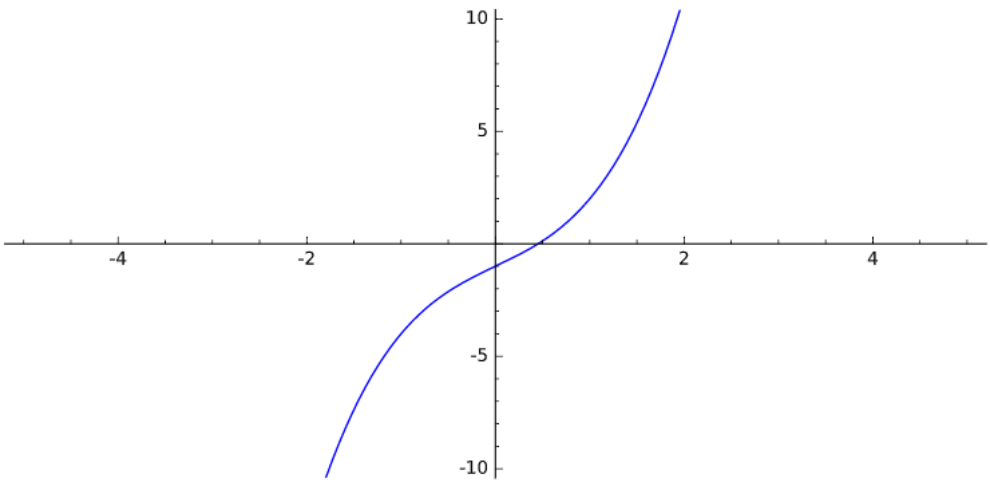
As I go through the steps, I'll use the example of $f(x) = x^3 + 2x - 1$.

Step 1 Define the function

```
20 f(x)=x^3+2*x-1
```

Step 2 Choose an initial value by graphing the function

```
21 plot(f,xmin=-5,xmax=5,ymin=-10,ymax=10)
```



From this graph, we can see that f has one (real) root, somewhere in the neighborhood of $x = \frac{1}{2}$.

We'll use this as our initial guess. I'm going to abandon subscripts to make it easier to do this in Sage. Instead, I'll call our estimate "r," and we'll replace r with better approximations as go:

```
22 r=1/2  #initial guess
```

Step 3 Define the number of repetitions

Now we have to decide how many times we want to repeat the process. I'll call this number "n."

```
n=10    #number of repetitions
```

Step 4 Define the derivative

```
df(x)=derivative(f,x)    #find the derivative
```

Step 5 Iterate Newton's Method

```
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)  #I will use N() to convert to decimal - it makes it run faster
    r
```

```
0.454545454545454545454545454545
0.453398336679093776885574992831
0.453397651516647791761892775232
0.453397651516403767644746569954
0.453397651516403767644746539000
0.453397651516403767644746539000
0.453397651516403767644746539000
0.453397651516403767644746539000
0.453397651516403767644746539000
0.453397651516403767644746539000
```

Our estimates get better as we go down the list, so our best estimate for the root of f is $x \approx 0.453397651516404$ (rounded to 15 decimal places).

For comparison, the exact solution is

$$\left(\frac{1}{18}\sqrt{59}\sqrt{3}+\frac{1}{2}\right)^{1/3}-\frac{\frac{2}{3}}{\left(\frac{1}{18}\sqrt{59}\sqrt{3}+\frac{1}{2}\right)^{1/3}} \approx 0.453397651516404$$

To 15 decimal places, our estimate matches the actual root exactly - and that's after only 5 repetitions.

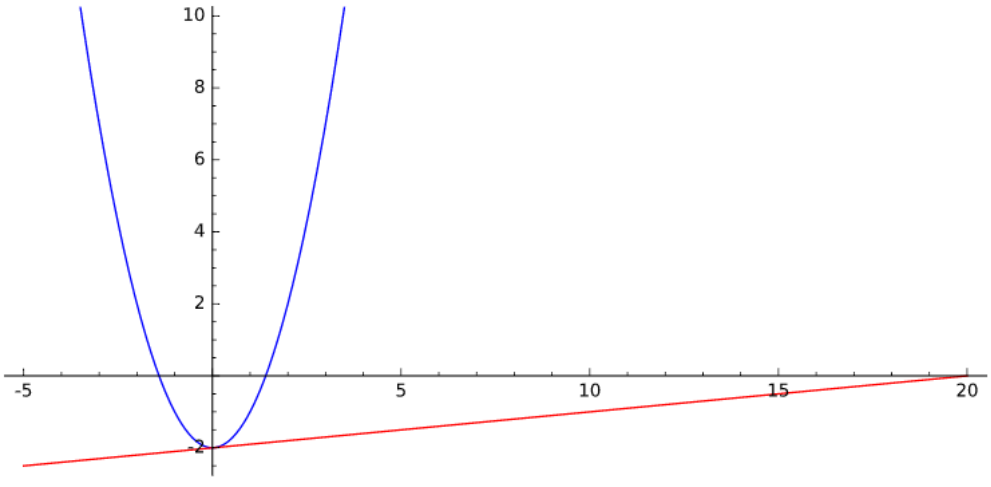
Note 1 It is possible to use Newton's Method to find complex roots as well (just start with a complex guess), but we will not do this in this class.

Note 2 As mentioned above, Newton's Method does not always work. An obvious example is when our initial guess is a local minimum or maximum. Then we get a horizontal tangent l which has no roots. Another problem arises if the function has a limited domain; in this case, the root of the tangent line may end up outside the domain of the function.

Example 3

Here's an example of a poor choice of initial guess. I have chosen an initial guess of $x = 0.05$ which is close to the minimum at $x = 0$. Although the root of the function is around 1.5, the of the tangent line is about 20.

```
f(x)=x^2-2
df(x)=derivative(f,x)
TL(x)=f(.05)+df(.05)*(x-.05)
plot(f,xmin=-5,xmax=10,ymax=10)+plot(TL,xmin=-5,xmax=20,color='red')
```



In this case, Newton's Method will eventually get back to the root, but it requires a few more repetitions.

```
f(x)=x^2-2
df(x)=derivative(f,x)
r=.05
n=10
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```

```
20.025000000000002131628207280 3
10.062437578027466728413436977 2
5.1305982874946600272337515606 9
2.7602081863120936282305430749 1
1.7423956061509768013391697841 1
1.4451202788151058365483661935 6
1.4145440625865478956176294748 8
1.4142136009828471171272810459 4
1.4142135623730955758482938981 2
1.4142135623730950488016887242 1
```

```
N(sqrt(2),digits=30) #This is the exact answer for comparison.
```

```
1.4142135623730950488016887242 1
```

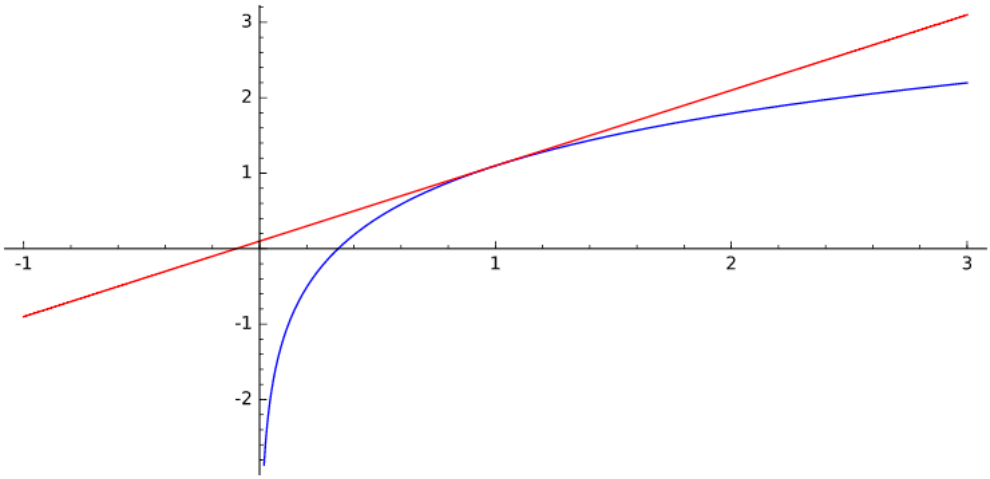
Example 4

Here's another poor choice of initial guess. Consider the function $f(x) = \ln(3x)$. I'm going to choose an initial guess of $x = 1$.

Below is a graph of f with the tangent line at $x = 1$.

Notice that the root of the tangent line is a negative number, which is not in the domain of f. That means we can't find a new tangent line using that root for the point of tangency.

```
f(x)=ln(3*x)
df(x)=derivative(f,x)
TL(x)=f(1)+df(1)*(x-1)
plot(f,xmin=0,xmax=3)+plot(TL,xmin=-1,xmax=3,color='red')
```



If we try Newton's Method, we end up with complex numbers!

```
f(x)=ln(3*x)
df(x)=derivative(f,x)
r=1
n=10
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```

```
-0.098612288668109691395245236922 5
-0.218716840163818111319368620162 2 + 0.309799641633409422971177200538*I
0.486573380029164645618123813404 + 0.747851351286347649636703893786*I
0.750863295538137729101406222465 - 0.472094422407517673683036387351*I
0.281017333228159361240218105002 + 0.411360592167483474474911835095*I
0.567723052665054031979560084438 - 0.026937088073690029971638885283 6*I
```

```
0.26605460242254438640723413331 6 + 0.014353849895062520460130807618 5*I
0.32642134552753086058981898136 1 + 0.003228995266481792215425401334 14*I
0.33327713995827883546095934632 0 + 0.000067607618066291580399193245288 7*I
0.33333333545386504380662686782 3 + 1.139779781613618460198803001 47e-8*I
```

Newton's Method in One Cell

You can copy and paste the input below to implement Newton's Method.

Change the function f, then hit run.

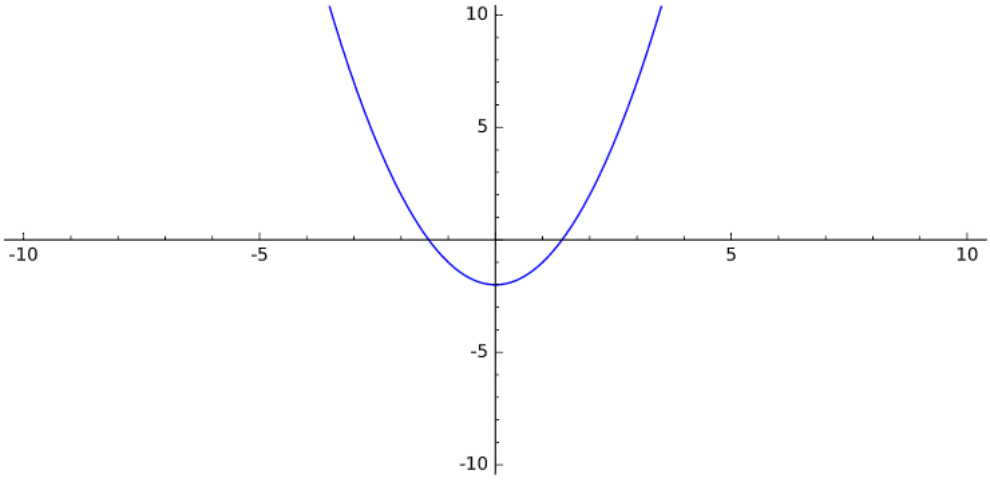
If no root is visible, adjust the plot window and run until you find a root.

Once you see a root, choose an initial guess in the ballpark of the root, change the number r, and hit run.

Example 5

Find the roots of $f(x) = x^2 - 2$.

```
f(x)=x^2-2                                  #change this (original function)
plot(f,xmin=-10,xmax=10,ymin=-10,ymax=10)   #you may need to adjust the plot window
r=1.5                                         #change this based on the graph (initial guess)
n=10                                        #number of iterations
df(x)=derivative(f,x)                       #only change the rest if you change the function's name
print ''
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```



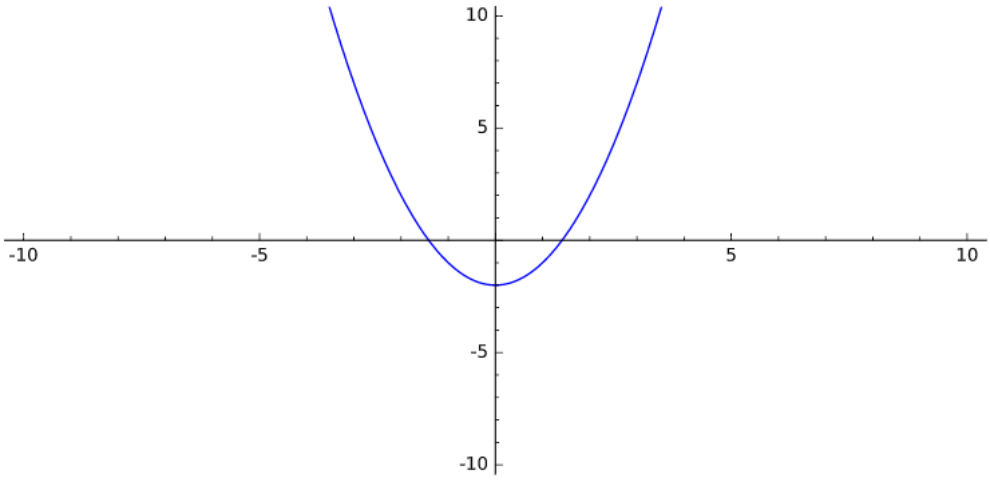

```
1.4166666666666667406815349750 1
1.4142156862745098040496220328 3
1.4142135623746899106262957712 0
1.4142135623730950488016896235 0
1.4142135623730950488016887242 1
1.4142135623730950488016887242 1
1.4142135623730950488016887242 1
1.4142135623730950488016887242 1
1.4142135623730950488016887242 1
1.4142135623730950488016887242 1
```

We found a root near 1.4142.

Notice how the answers stabilize after a few repetitions. This suggests we have the right answer.

For this example, the graph shows two roots. We have found the positive root; to find the negative root, change the initial guess and run it again.

```
f(x)=x^2-2                                  #change this (original function)
plot(f,xmin=-10,xmax=10,ymin=-10,ymax=10)   #you may need to adjust the plot window
r=-1.5                                        #change this based on the graph (initial guess)
n=10                                        #number of iterations
df(x)=derivative(f,x)                       #only change the rest if you change the function's name
print ''
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```




```
-1.4166666666666674068153497501
-1.4142156862745098404962203283
-1.4142135623746899106262957712 0
-1.4142135623730950488016896235 0
-1.4142135623730950488016887242 1
-1.4142135623730950488016887242 1
-1.4142135623730950488016887242 1
-1.4142135623730950488016887242 1
-1.4142135623730950488016887242 1
-1.4142135623730950488016887242 1
```

Now we have found the second root, around -1.4142.

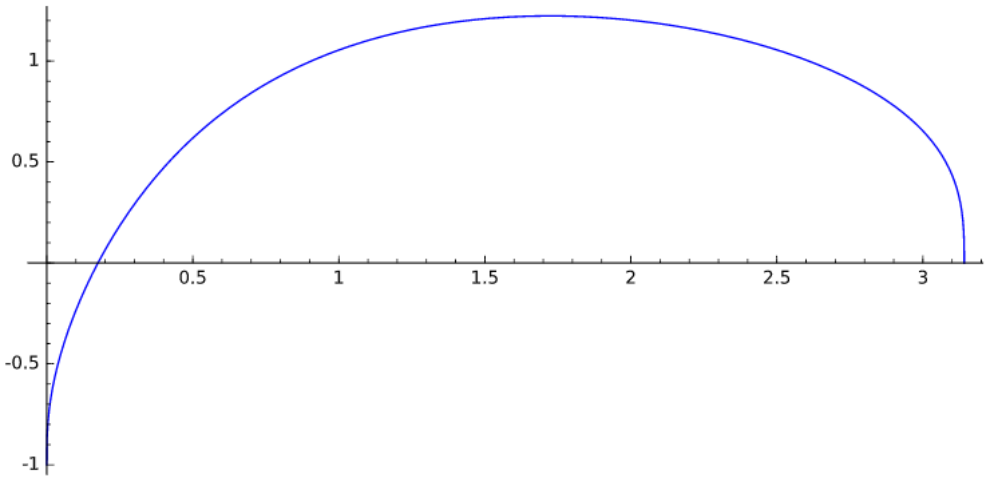
Using Newton's Method to find a point of intersection

If you have two functions, f_1 and f_2, and you want to know where they cross, then define a new function $f = f_1 - f_2$ and find the roots of f. This gives you the x-coordinate of a point of intersection. Plug this into either f_1 or f_2 to get the corresponding y-coordinate.

Example 6

Find the points of intersection of $f_1(x) = \sqrt[3]{2\sin(x)}$ and $f_2(x) = e^{-2x}$ for $0 \leq x \leq \pi$.

```
f1(x)=(2*sin(x))^(1/3)
f2(x)=e^(-2*x)
f(x)=f1(x)-f2(x)       #find the root of f1-f2
plot(f,xmin=0,xmax=pi) #I have adjusted the window
r=.2                   #Based on the graph, one root is close to 0.2
n=10
df(x)=derivative(f,x)
print ''
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```




```
0.17456461404644615248775388992 9
0.17542226188344320158022013411 4
0.17542334711394842425591723999 0
0.17542334711567936037319812910 3
0.17542334711567936037320253259 3
0.17542334711567936037320253259 3
0.17542334711567936037320253259 3
0.17542334711567936037320253259 3
0.17542334711567936037320253259 3
0.17542334711567936037320253259 3
```

This tells us that the x-coordinate of one point of intersection is approximately 0.175423347115679.

To find the y-coordinate, plug this into f_1 or f_2. I'll plug it into both as a check (I'd better get the same answer).

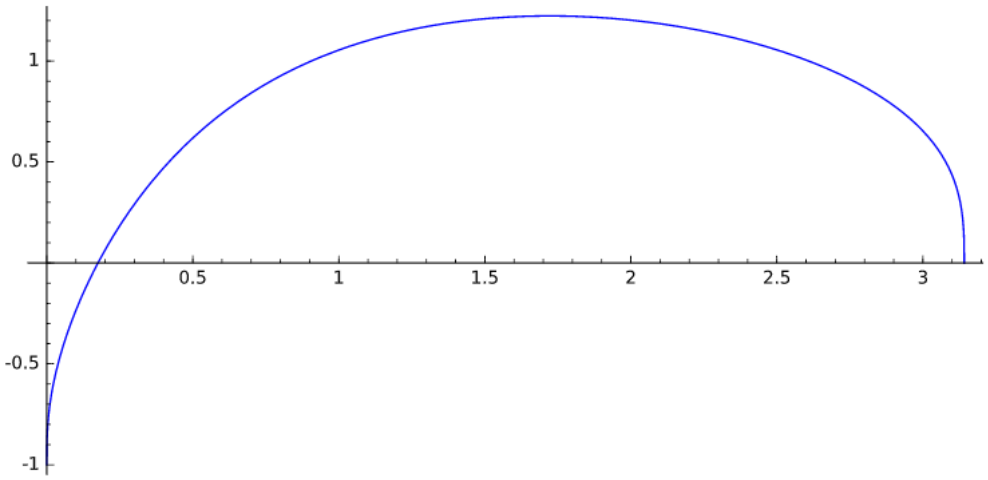
```
f1(x)=(2*sin(x))^(1/3)
f2(x)=e^(-2*x)
N(f1(0.17542334711567936037320253259),digits=30)
N(f2(0.17542334711567936037320253259),digits=30)
```

```
0.704091686899284448083383801316
0.704091686899284448083383801316
```

So our point of intersection is approximately $(0.175423347115679, 0.704091686899284)$.

The second root is hard to approximate with Newton's Method (at least in Sage), since Sage does not allow fractional powers of negative numbers (this happens when x goes past π). W need a very good initial guess to avoid ending up with complex numbers.

```
f1(x)=(2*sin(x))^(1/3)
f2(x)=e^(-2*x)
f(x)=f1(x)-f2(x)
plot(f,xmin=0,xmax=pi)
r=3.14159265
n=10
df(x)=derivative(f,x)
print ''
for i in range(n):
    r=N(r-f(r)/df(r),digits=30)
    r
```




```
3.14159265034448884600068852662
3.14159265033359929124197408890
3.14159265033358710682062112553
3.14159265033358710680542340382
3.14159265033358710680542340382
3.14159265033358710680542340382
3.14159265033358710680542340382
3.14159265033358710680542340382
3.14159265033358710680542340382
3.14159265033358710680542340382
```

This is the x-coordinate; now find the y-coordinate:

```
96 N(f1(3.14159265033358710680542340382),digits=30)
97 N(f2(3.14159265033358710680542340382),digits=30)
```

```
0.00186744274386954580104325212193
0.00186744274386954580104327322816
```

So our second point of intersection is approximately $(3.141592650333587, 0.001867442743870)$.

Here's a graph of the two functions:

```
98 f1(x)=(2*sin(x))^(1/3)
99 f2(x)=e^(-2*x)
100 plot(f1,xmin=0,xmax=pi)+plot(f2,xmin=0,xmax=pi,color='red')+point((0.175423347115679,0.704091686899284),size=25,color='black')+point((3.1415926503
```