Lecture 29: Newton's method

Objectives:

(29.1) Use Newton's method to approximate the zeros of a function.

(29.2) Use tangent lines to illustrate Newton's method.

Newton's method

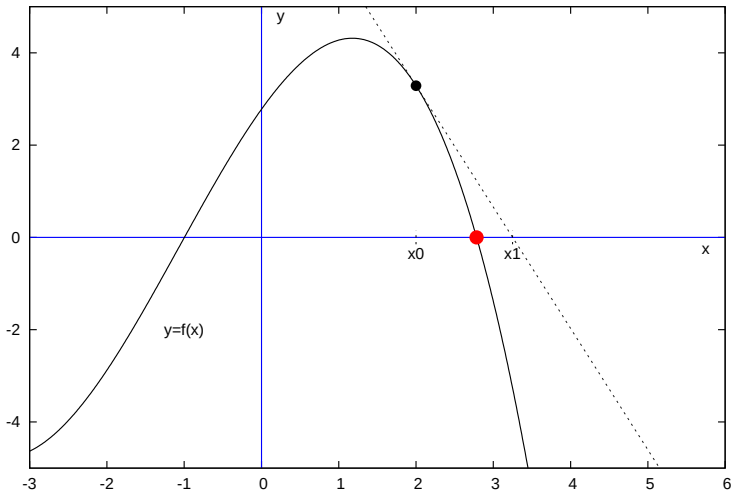
In the last example of lecture 20, we used the linearization of a function to approximate a zero of that function. Repeated use of this idea gives rise to a very effective method for approximating solutions of $f(x) = 0$. The method is called Newton's method or the Newton-Raphson method, named for Isaac Newton and Joseph Raphson.

Suppose we would like to solve $f(x) = 0$. Referring to the graph below, let's say we are looking for the indicated point (in red). Unfortunately, we have only an initial guess, x_0. We will approximate our solution of $f(x) = 0$ by solving $L(x) = 0$, where L is the linearization of f at x_0.

$$L(x) = f(x_0) + f'(x_0)(x - x_0) = 0 \Longrightarrow x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

We now give this x-value the name x_1 so that

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



We have gone from an initial guess at the solution of $f(x) = 0$ to a new approximation based on the linearization. From the graph, it looks like the new approximation, x_1, is slightly better than our original guess, x_0. At this point, we can simply repeat the process using x_1 in place of x_0. This would lead to the new approximation

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

This process by which we repeatedly refine our approximations is Newton's method.

Newton's method
Suppose we wish to approximate a solution of $f(x) = 0$. If x_0 is an initial approximation, then Newton's method generates the following sequence of "improved" approximations:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}; \quad n = 0, 1, 2, 3, \ldots$$

As we will see, Newton's method is not guaranteed to converge to a solution, but when it does, it often does so very quickly.

Example 1 Use Newton's method, starting with $x_0 = 1$, to approximate the solution of $x = \cos x$. Continue the process until two consecutive approximations agree to six decimal places.

$f(x) = x - \cos x$ and $f'(x) = 1 + \sin x$. According to Newton's method,

$$x_{n+1} = x_n - \frac{x_n - \cos(x_n)}{1 + \sin(x_n)}; \quad n = 0, 1, 2, 3, \ldots.$$

This generates the following sequence of approximations:

$$x_0 = 1$$

$$x_1 = 0.7503638678402439$$

$$x_2 = 0.7391128909113617$$

$$x_3 = 0.739085133385284$$

$$x_4 = 0.7390851332151606$$

The solution is $x \approx 0.7390851$.

Example 2 Use Newton's method to approximate $\sqrt{2}$.

In order to solve this problem, we need an equation with solution $\sqrt{2}$. The equation $x^2 - 2 = 0$ will do just fine. With this equation, we have

$$f(x) = x^2 - 2 \quad \text{and} \quad f'(x) = 2x.$$

According to Newton's method,

$$x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n}; \quad n = 0, 1, 2, 3, \ldots.$$

Using $x_0 = 1$, this generates the following sequence of approximations:

$$x_0 = 1.0$$

$$x_1 = 1.5$$

$$x_2 = 1.416666666666667$$

$$x_3 = 1.41421568627451$$

$$x_4 = 1.41421356237469$$

$$x_5 = 1.414213562373095$$

$$x_6 = 1.414213562373095$$

It looks like $\sqrt{2} \approx 1.414213562373095$.

In the examples above Newton's method worked very well. The final results are exactly correct up to the number of digits shown. In addition, the results were obtained very quickly. This is fairly typical of Newton's method:

Suppose α is a zero of f of multiplicity 1 and that f' is continuous on an interval containing α. Once the Newton's method approximations are sufficiently close to α, the number of correct digits roughly doubles with each iteration.

This behavior can be seen in the previous examples. However, Newton's method is not guaranteed to produce fast or accurate results.

Example 3 (Slow convergence) Use Newton's method with $x_0 = 1.2$ to approximate a solution of $x^3 - 3x + 2 = 0$.

The details are omitted, but we should find slow convergence to the exact solution $x = 1$.

Example 4 (No convergence) Use Newton's method with $x_0 = 0$ to approximate the single real solution of $x^3 - x + 3 = 0$.

The details are omitted, but we should find that Newton's method does not converge to the solution, $x \approx -1.6717$, regardless of the number of iterations.

Example 5 (Worse than no convergence) Use Newton's method to approximate the solution of $x^{1/3} = 0$. Use any nonzero value for x_0.

Notice that $x = 0$ is the exact solution. We let $f(x) = x^{1/3}$ and then $f'(x) = (1/3)x^{-2/3}$. According to Newton's method,

$$x_{n+1} = x_n - \frac{x_n^{1/3}}{(1/3)x_n^{-2/3}} = -2x_n.$$

If we start with any nonzero x_0, Newton's method will produce results that get farther and farther (twice the distance to be precise) from the exact solution with each iteration.

Example 6 (Convergence to a nonsolution) Use Newton's method with $x_0 = 1/2$ to approximate a solution of $\pi - 2x\sin(\frac{\pi}{x}) = 0$.

The details are omitted, but we should find slow convergence to $x = 0$, even though the equation has no real solutions.

Examples 3–6 should be analyzed in more detail. In each case, the behavior of Newton's method is easy to explain after thinking about the method graphically.

Newton's method in Python

Python code for using Newton's method is given below. The user input is between the hashtags, and the required modifications should be (somewhat) obvious. You can run the code by cutting and pasting into a SageMath Cell.

```
# Newton's method to approximate a solution of f(x)=0 given an initial guess, x0.
# The user input is between the hashtags. You can run the code by cutting and
# pasting into a SageMath Cell.
#
def f(x):
   return x - math.cos(x)
def fp(x):
   return 1.0 + math.sin(x)
x0 = 1.0
N = 10
#
x = x0
print(0, x)
for i in range(1, N+1):
   x = x - f(x)/fp(x)
   print(i, x)
```

Newton's method in Maxima

Maxima is a free, open-source, programmable computer algebra system. It has some major advantages over Wolfram Alpha, which we have already been using. Maxima is available for download at `http://maxima.sourceforge.net/`.

Here is the Maxima code for implementing a high-precision version of Newton's method. User input is in lines 2–5. `fpprec` defines the precision, i.e. the number of significant digits displayed and used in the computations.

```
/* Newton's Method */
f: x-cos(x)  $
x0: 1  $
N: 10  $
fpprec: 32  $

set_display(ascii)$
fp: diff(f,x,1)$
xn: bfloat(x0)$
print("x", 0, "=", xn)$
for i: 1 thru N do
(  xn: bfloat( xn - at(f,x=xn) / at(fp,x=xn) ),
   print("x", i, "=", xn)
);
```