Computer Science/Mathematics 455
Lecture Notes
Lecture # 16

Today we give a further discussion of Newton’s method and a related method called the secant method. With a “good” initial guess, Newton’s method is given by

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \]

Last time, we discussed a complicated theorem showing that with a good enough initial guess

\[ \lim_{n \to \infty} x_n = \alpha \]
\[ \lim_{n \to \infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = \frac{f''(\alpha)}{2f'(\alpha)}. \]

In the following example, Newton’s method converges for every initial guess except \( x_0 = 0 \).

**Example 1**

\[ f(x) = x^2 - a. \]

We have

\[ f'(x) = 2x, \]
\[ f''(x) = 2. \]

For all \( x_0 > 0 \), Newton’s method converges to \( \alpha = \sqrt{a} \). If \( x_0 < 0 \), it converges to \( \alpha = -\sqrt{a} \).

Moreover, the iteration simplifies to a really nice algebraic form.

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]
\[ x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right). \]

This is very nice, useful formula. A good guess is important. Suppose I guess \( x_0 = 50 \). Then I get

1
\[ x_1 = 25.02, \quad x_2 = 12.55, \ldots, x_5 = 1.9672, \]
\[ x_6 = 1.4919, \quad x_7 = 1.4162, \quad x_8 = 1.4142 (\text{six digits}) \]

The value \( x_9 \) has 13 digits right. However, the iterations before \( x_6 \) are a waste of time. Quadratic convergence is not visible before that.

If we let \( x_0 = 1.5 \), \( x_2 \) is accurate to six digits, \( x_3 \) is accurate to 12 digits, and \( x_4 \) is accurate to IEEE double precision. We would like for Newton’s method to behave in this fashion as often as possible.

However, with a sensible initial guess, this is an excellent way to compute the square root of a number quickly.

Here is theorem about “special circumstances” for the convergence of Newton’s method. It is Theorem 2, p.39 of your book.

**Theorem 1** Let \( f \) be twice differentiable on \([c,d]\) and let it satisfy

1. \( f(c)f(d) < 0 \)
2. \( f'(x) \neq 0 \) for all \( x \in [c,d] \)
3. \( f'' \) does not change sign in \([c,d]\).
4. \[
\left| \frac{f(c)}{f'(c)} \right| \cdot \left| \frac{f(d)}{f'(d)} \right| < d - c.
\]

Then for some \( \alpha \in (c,d) \), \( f(\alpha) = 0 \) and Newton’s iteration converges from any starting point \( x_0 \in [c,d] \).

For Example 1, let \( c = 1, d = a \), for \( a > 1 \). Then

\[
\begin{align*}
f(1) & = 1^2 - a < 0, \quad f(a) = a^2 - a > 0 \\
f'(x) & = 2x > 0, \quad x \in [1,a] \\
f''(x) & = 2 > 0 \\
\left| \frac{f(c)}{f'(c)} \right| & = \left| \frac{1^2 - a}{2} \right| = \frac{a - 1}{2} < d - c = a - 1 \\
\left| \frac{f(d)}{f'(d)} \right| & = \left| \frac{a^2 - a}{2a} \right| = \frac{a - 1}{2} < d - c = a - 1
\end{align*}
\]
So all of the conditions are met and Newton’s iteration will converge for any \( x_0 \in [1,a] \).

Here is another such theorem that I did not have time to give in class. Again these criteria are somewhat easy to verify.

**Theorem 2** If \( f \) is twice continuously differentiable, has a zero, is strictly increasing (that is, \( f'(x) > 0 \), for all \( x \)), and is convex (that is, \( f''(x) > 0 \) for all \( x \)), then the Newton iterates converge for any \( x_0 \).

There are two problems with Newton’s method

1. The conditions for convergence may be hard to meet or hard to verify.

2. The user must supply function for both \( f \) and \( f' \).

The secant method does nothing about the first problem, but much about the second. We approximate \( f'(x_n) \) by

\[
f'(x_n) \approx \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}.
\]

This uses the last two guesses \( x_{n-1} \) and \( x_n \) to get \( x_{n+1} \).

We have then the iteration

\[
x_{n+1} = x_n - f(x_n) / \left[ \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n} \right].
\]

Only one new function evaluation is needed per iteration.

However,

\[
\lim_{n \to \infty} x_n = \frac{\alpha - x_{n+1}}{(\alpha - x_n)^r} = \frac{f''(\alpha)}{2 f'(\alpha)}, \quad r = \frac{1 + \sqrt{5}}{2}.
\]

Here the order of convergence \( r \approx 1.62 \). This is slower than quadratic (like Newton’s method), but faster than linear (like bisection).

**Example 2**

\[
f(x) = \cos x - x = 0.
\]

\[
x_0 = 0, \quad x_1 = \pi/2.
\]
We have
\[
x_2 = 0.61101, \quad x_3 = 0.77153, \quad x_4 = 0.73812
\]
\[
x_5 = 0.73907, \quad x_6 = 0.739085
\]
Here \(x_6\) is correct to eight digits, and \(x_7\) and \(x_8\) are correct to machine precision.

Newton's method requires 4 iterations. However, the secant does 9 function evaluations, while Newton's does 5 function evaluations and 5 derivative evaluations. If derivative evaluations cost the same as function evaluations, Newton's method will be slightly slower.

In fact, if we ignore the cost of implementing the loops and only count function and derivative evaluations, the secant method will be faster than Newton’s method unless \(f'\) can be evaluated in less 44\% of the time required to evaluate \(f\)!