Newton’s Method

Find root of $f(x) = 0$ through successive approximation e.g., $f(x) = x^2 - a$

Figure 1: Newton’s Method.

Tangent at $(x_i, f(x_i))$ is line $y = f(x_i) + f'(x_i) \cdot (x - x_i)$ where $f'(x_i)$ is the derivative.

$x_{i+1} = \text{intercept on x-axis}$

\[
x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}
\]

Square Roots

We want to find the square root of a number $a$ through successive approximation using Newton’s Method, so that our answer $x$ will converge on $a^{1/2}$. It looks like $f(x) = a^{1/2} - x$ has a root at $x = a^{1/2}$. Let’s try to use it with Newton’s Method.

\[
\begin{align*}
f(x) &= a^{1/2} - x \\
f'(x) &= -1 \\
x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\
&= x_i - \frac{a^{1/2} - x}{-1} \\
&= x_i + a^{1/2} - x_i \\
&= a^{1/2}
\end{align*}
\]

The choice of this function was unsuccessful, since we’re unable to compute $x_{i+1}$ using $x_i$. Let’s try squaring both sides of $x = a^{1/2}$. This time, it looks like we get something usable:
\[ f(x) = x^2 - a \]

\[ x_{i+1} = x_i - \frac{x_i^2 - a}{2x_i} = \frac{x_i + a}{2} \]

**Error Analysis of Newton’s Method**

Suppose \( x_n = \sqrt{a} \cdot (1 + \epsilon_n) \) \( \epsilon_n \) may be + or -

Then,

\[ x_{n+1} = \frac{x_n + a/x_n}{2} \]

\[ = \frac{\sqrt{a}(1 + \epsilon_n) + \frac{a}{\sqrt{a}(1 + \epsilon_n)}}{2} \]

\[ = \sqrt{a} \left( 1 + \frac{(1 + \epsilon_n) + \frac{1}{1 + \epsilon_n}}{2} \right) \]

\[ = \sqrt{a} \left( \frac{2 + 2\epsilon_n + \epsilon_n^2}{2(1 + \epsilon_n)} \right) \]

\[ = \sqrt{a} \left( 1 + \frac{\epsilon_n^2}{2(1 + \epsilon_n)} \right) \]

Therefore,

\[ \epsilon_{n+1} = \frac{\epsilon_n^2}{2(1 + \epsilon_n)} \]

Quadratic convergence, as number of correct digits doubles each step.

Newton’s method requires high-precision division. We covered multiplication in the previous lecture.

**Multiplication Algorithms:**

1. Naive Divide & Conquer method: \( \Theta(d^2) \) time

2. Karatsuba: \( \Theta(d \log_2 3) = \Theta(d^{1.584...}) \)

3. Toom-Cook generalizes Karatsuba (break into \( k \geq 2 \) parts)

   \[ T(d) = 5T(d/3) + \Theta(d) = \Theta \left( d \log_3 5 \right) = \Theta \left( d^{1.465...} \right) \]

4. Schönhage-Strassen - almost linear! \( \Theta(d \ lg \ d \ lg \ lg \ d) \) using FFT. All of these are in gmpy package
5. Furer (2007): $\Theta \left( n \log n \ 2^{O(\log^* n)} \right)$ where $\log^* n$ is iterated logarithm (number times log needs to be applied to get a number that is less than or equal to 1).

**High Precision Division**

We want high precision rep of $\frac{a}{b}$

- Compute high-precision rep of $\frac{1}{b}$ first

- High-precision rep of $\frac{1}{b}$ means $\lfloor \frac{R}{b} \rfloor$ where $R$ is large value s.t. it is easy to divide by $R$

  Ex: $R = 2^k$ for binary representations

**Division**

Newton’s Method for computing $\frac{R}{b}$

$$f(x) = \frac{1}{x} - \frac{b}{R} \quad \text{(zero at } x = \frac{R}{b})$$

$$f'(x) = -\frac{1}{x^2}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{1}{x_i} - \frac{b}{R}$$

$$x_{i+1} = x_i + x_i^2 \left( \frac{1}{x_i} - \frac{b}{R} \right) = 2x_i - \frac{bx_i^2}{R} \rightarrow \text{multiply } \frac{R}{b} \rightarrow \text{easy div}$$

**Example**

Want $\frac{R}{b} = \frac{2^{16}}{5} = \frac{65536}{5} = 13107.2$

Try initial guess $\frac{2^{16}}{4} = 2^{14}$

$$x_0 = 2^{14} = 16384$$

$$x_1 = 2 \cdot (16384) - 5(16384)^2/65536 = 12288$$

$$x_2 = 2 \cdot (12288) - 5(12288)^2/65536 = 13056$$

$$x_3 = 2 \cdot (13056) - 5(13056)^2/65536 = 13107$$
Complexity of Computing Square Roots

We apply a first level of Newton’s method to solve $f(x) = x^2 - a$. Each iteration of this first level requires a division. If we set the precision to $d$ digits right from the beginning, then convergence at the first level will require $\log d$ iterations. This means the complexity of computing a square root will be $\Theta(d^\alpha \log d)$ if the complexity of multiplication is $\Theta(d^\alpha)$, given that we have shown that the complexity of division is the same as the complexity of multiplication.

However, we can do better, if we recognize that the number of digits of precision we need at beginning of the first level of Newton’s method starts out small and then grows. If the complexity of a $d$-digit division is $\Theta(d^\alpha)$, then a similar summation to the one above tells us that the complexity of computing square roots is $\Theta(d^\alpha)$.

Termination

Iteration: $x_{i+1} = \left\lfloor x_i + \frac{a}{x_i} \right\rfloor - \left\lfloor \left\lfloor \frac{a}{x_i} \right\rfloor \right\rfloor$  

Do floors hurt? Does program terminate? ($\alpha$ and $\beta$ are the fractional parts below.)

Iteration is

$$x_{i+1} = \frac{x_i + \frac{a}{x_i} - \alpha}{2} - \beta = \frac{x_i + \frac{a}{x_i} - \gamma}{2}$$  where $\gamma = \frac{\alpha}{2} + \beta$ and $0 \leq \gamma < 1$

Since $\frac{a + b}{2} \geq \sqrt{ab}$, $\frac{x_i + \frac{a}{x_i}}{2} \geq \sqrt{a}$, so subtracting $\gamma$ always leaves us $\geq \lfloor \sqrt{a} \rfloor$. This won’t stay stuck above if $\epsilon_i < \frac{1}{2}$ (good initial guess).

Cube Roots

Now that we’ve seen square roots, let’s calculate cube roots. This time, $x$ will converge on $a^{1/3}$, instead. We can use $f(x) = a - x^3$:

$$f(x) = x^3 - a$$  $$f'(x) = 3x^2$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{1}{3} \left( x_i - \frac{a}{x_i^2} \right)$$  $$= \frac{2}{3} x_i + \frac{1}{3} \frac{a}{x_i^2}$$
Suppose \( x_{i+1} = a^{1/3}(1 + \epsilon_i) \) \( \epsilon_i \) may be \(+\) or \(-\). Then,

\[
x_{i+1} = \frac{2}{3}x_i + \frac{1}{3} a \frac{a}{x_i^2}
\]

\[
= \frac{2}{3} a^{1/3}(1 + \epsilon_i) + \frac{1}{3} a \frac{(a^{1/3}(1 + \epsilon_i))^2}{(1 + \epsilon_i)^2}
\]

\[
= \frac{a^{1/3}}{3} \left( 2(1 + \epsilon_i) + \frac{1}{(1 + \epsilon_i)^2} \right)
\]

\[
= \frac{a^{1/3}}{3} \left( \frac{1}{(1 + \epsilon_i)^2} \right) (2(1 + 3\epsilon_i + 3\epsilon_i^2 + \epsilon_i^3) + 1)
\]

\[
= \frac{a^{1/3}}{3} \frac{3 + 6\epsilon_i + 6\epsilon_i^2 + 2\epsilon_i^3}{1 + 2\epsilon_i + \epsilon_i^2}
\]

\[
= \frac{a^{1/3}}{3} \left( 3 + \frac{3\epsilon_i^2 + 2\epsilon_i^3}{1 + 2\epsilon_i + \epsilon_i^2} \right)
\]

\[
= \frac{a^{1/3}}{3} \left( 1 + \frac{\epsilon_i^2 + \frac{2}{3}\epsilon_i^3}{(1 + \epsilon_i)^2} \right)
\]

Notice that the cubic term gets dominated by the quadratic term in the numerator, and that the denominator is very close to 1. Therefore, we have quadratic convergence here, just like in the square root case. The running time analysis is very similar to the square root case as well: here we have multiplies and divides, and since we’ve shown that divides have the same running time as multiplies, the running time is the same as in the square root case.