Lecture 12: Numerics II

Lecture Overview

- Review:
  - high precision arithmetic
  - multiplication
- Division
  - Algorithm
  - Error Analysis
- Termination

Review:

Want millionth digit of $\sqrt{2}$:

$$\lfloor \sqrt{2 \cdot 10^{2d}} \rfloor \qquad d = 10^6$$

Compute $\lfloor \sqrt{a} \rfloor$ via Newton's Method

$$\begin{aligned} \chi_0 &= 1 \quad \text{(initial guess)} \\ \chi_{i+1} &= \frac{\chi_i + a/\chi_i}{2} \quad \leftarrow \text{division!} \end{aligned}$$

Error Analysis of Newton's Method

Suppose $X_n = \sqrt{a} \cdot (1 + \epsilon_n)$ ϵ_n may be + or -
Then,

$$\begin{aligned} X_{n+1} &= \frac{X_n + a/X_n}{2} \\ &= \frac{\sqrt{a}(1+\epsilon_n) + \frac{a}{\sqrt{a}(1+\epsilon_n)}}{2} \\ &= \sqrt{(a)}\frac{\left((1+\epsilon_n) + \frac{1}{(1+\epsilon_n)}\right)}{2} \\ &= \sqrt{(a)}\left(\frac{2 + 2\epsilon_n + {\epsilon_n}^2}{2(1+\epsilon_n)}\right) \\ &= \sqrt{(a)}\left(1 + \frac{{\epsilon_n}^2}{2(1+\epsilon_n)}\right) \end{aligned}$$

Therefore,

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

Quadratic convergence, as ♯ correct digits doubles each step.

Newton's method requires high-precision division. We covered multiplication in Lecture 12.

Multiplication Algorithms:

1. Naive Divide & Conquer method: $\Theta(d^2)$ time
2. Karatsuba: $\Theta(d^{\log_2 3}) = \Theta(d^{1.584\ldots})$
3. Toom-Cook generalizes Karatsuba (break into $k \geq 2$ parts)

$$T(d) = 5T(d/3) + \Theta(d) = \Theta\left(d^{\log_3 5}\right) = \Theta\left(d^{1.465\ldots}\right)$$

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. ♯ 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 $\frac{R}{b}$

$$\begin{aligned} f(x) &= \frac{1}{x} - \frac{b}{R} \quad \left(\text{zero at } x = \frac{R}{b}\right) \\ f'(x) &= \frac{-1}{x^2} \\ \chi_{i+1} &= \chi_i - \frac{f(\chi_i)}{f'(\chi_i)} = \chi_i - \frac{\left(\frac{1}{\chi_i} - \frac{b}{R}\right)}{-1/{\chi_i}^2} \\ \chi_{i+1} &= \chi_i + {\chi_i}^2\left(\frac{1}{\chi_i} - \frac{b}{R}\right) = 2\chi_i - \frac{b{\chi_i}^2 \to \text{ multiply}}{R \to \text{ easy div}} \end{aligned}$$

Example

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

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

$$\begin{aligned} \chi_0 &= 2^{14} = 16384 \\ \chi_1 &= 2 \cdot (16384) - 5(16384)^2/65536 = \underline{12}288 \\ \chi_2 &= 2 \cdot (12288) - 5(12288)^2/65536 = \underline{13}056 \\ \chi_3 &= 2 \cdot (13056) - 5(13056)^2/65536 = \underline{13107} \end{aligned}$$

Error Analysis

$$\begin{aligned} \chi_{i+1} &= 2\chi_i - \frac{b{\chi_i}^2}{R} \qquad \text{Assume } \chi_i = \frac{R}{b}(1+\epsilon_i) \\ &= 2\frac{R}{b}(1+\epsilon_i) - \frac{b}{R}\left(\frac{R}{b}\right)^2(1+\epsilon_i)^2 \\ &= \frac{R}{b}\left((2+2\epsilon_i) - (1+2\epsilon_i+{\epsilon_i}^2)\right) \\ &= \frac{R}{b}\left(1-{\epsilon_i}^2\right) = \frac{R}{b}(1+\epsilon_{i+1}) \text{ where } \epsilon_{i+1} = -{\epsilon_i}^2 \end{aligned}$$

Quadratic convergence; ♯ digits doubles at each step

One might think that the complexity of division is $\lg d$ times the complexity of multiplication given that we will have $\lg d$ multiplications in the $\lg d$ iterations required

to reach precision d. However, the complexity of division equals the complexity of multiplication.

To understand this, assume that the complexity of multiplication is $\Theta(n^\alpha)$ for n-digit numbers, with $\alpha \geq 1$. Division requires multiplication of *different-sized* numbers at each iteration. Initially the numbers are small, and then they grow to d digits. The number of operations in division are:

$$c \cdot 1^\alpha + c \cdot 2^\alpha + c \cdot 4^\alpha + \cdots + c \cdot \left(\frac{d}{4}\right)^\alpha + c \cdot \left(\frac{d}{2}\right)^\alpha + c \cdot d^\alpha < 2c \cdot d^\alpha$$

S=c.(d)^alpha
(1+1/(2)^alpha
+1/(4)^alpha+..)
=c.(d)^alpha.K

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[1] requires a division. If we set the precision to d digits right from the beginning, then convergence at the first level will require $\lg d$ iterations. This means the complexity of computing a square root will be $\Theta(d^\alpha \lg d)$ 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: $\chi_{i+1} = \lfloor \frac{\chi_i + \lfloor a/\chi_i \rfloor}{2} \rfloor$
Do floors hurt? Does program terminate? (α and β are the fractional parts below.)
Iteration is

$$\begin{aligned} \chi_{i+1} &= \frac{\chi_i + \frac{a}{\chi_i} - \alpha}{2} - \beta \\ &= \frac{\chi_i + \frac{a}{\chi_i}}{2} - \gamma \quad \text{where } \gamma = \frac{\alpha}{2} + \beta \text{ and } 0 \leq \gamma < 1 \end{aligned}$$

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

[1] We are calling this the first level, since Newton's method is used within division, which would be the second level of applying it when we are computing square roots.

6.006 Introduction to Algorithms
Fall 2011