Square Roots via Newton's Method

S. G. Johnson, MIT Course 18.335

February 4, 2015

1 Overview

Numerical methods can be distinguished from other branches of analysis and computer science by three characteristics:

- They work with arbitrary **real** numbers (and vector spaces/extensions thereof): the desired results are not restricted to integers or exact rationals (although in practice we only ever compute *rational approximations* of irrational results).
- Like in computer science (= math + time = math + money), we are concerned not only with existence and correctness of the solutions (as in analysis), but with the **time** (and other computational resources, e.g. memory) required to compute the result.
- We are also concerned with **accuracy** of the results, because in practice we only ever have **approximate** answers:
  - Some algorithms may be intrinsically approximate—like the Newton's-method example shown below, they converge *towards* the desired result but never *reach* it in a finite number of steps. *How fast they converge* is a key question.
  - *Arithmetic with real numbers is approximate* on a computer, because we approximate the set $\mathbb{R}$ of real numbers by the set $\mathbb{F}$ of floating-point numbers, and the result of every elementary operation $(+,-,\times,\div)$ is **rounded** to the nearest element of $\mathbb{F}$. We need to understand $\mathbb{F}$ and how *accumulation* of these rounding errors affects different algorithms.

2 Square roots

A classic algorithm that illustrates many of these concerns is "Newton's" method to compute square roots $x = \sqrt{a}$ for $a > 0$, i.e. to solve $x^2 = a$. The algorithm starts with some guess $x_1 > 0$ and computes the sequence of improved guesses

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right).$$

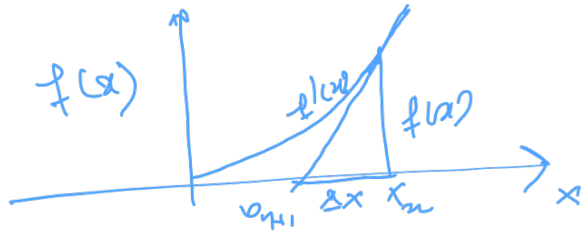
The intuition is very simple: if x_n is too big ($> \sqrt{a}$), then a/x_n will be too small ($< \sqrt{a}$), and so their arithmetic mean x_{n+1} will be closer to $\sqrt{a}$. It turns out that this algorithm is very old, dating at least to the ancient Babylonians circa 1000 BCE.[1] In modern times, this was seen to

[1] See e.g. Boyer, *A History of Mathematics*, ch. 3; the Babylonians used base 60 and a famous tablet (YBC 7289) shows $\sqrt{2}$ to about six decimal digits.

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

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

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

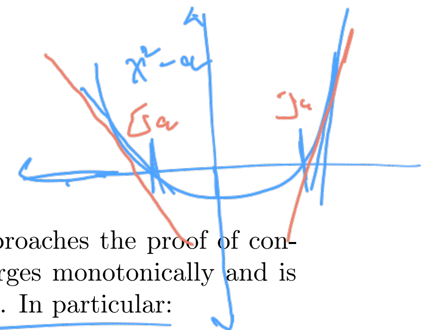


Babylon

$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n}$

be equivalent to Newton's method to find a root of $f(x) = x^2 - a$. Recall that Newton's method finds an approximate root of $f(x) = 0$ from a guess x_n by approximating $f(x)$ as its tangent line $f(x_n) + f'(x_n)(x - x_n)$, leading to an improved guess x_{n+1} from the root of the tangent:

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



and for $f(x) = x^2 - a$ this yields the Babylonian formula above.

2.1 Convergence proof

A classic analysis text (Rudin, *Principles of Mathematical Analysis*) approaches the proof of convergence of this algorithm as follows: we prove that the sequence converges monotonically and is bounded, and hence it has a limit; we then easily see that the limit is $\sqrt{a}$. In particular:

1. Suppose $x_n > \sqrt{a}$, then it follows $\sqrt{a} < x_{n+1} < x_n$:
   (a) $x_{n+1} - x_n = \frac{1}{2}(\frac{a}{x_n} - x_n) = \frac{a - x_n^2}{2x_n} < 0$.
   (b) $x_{n+1}^2 - a = \frac{1}{4}(x_n^2 + 2a + \frac{a^2}{x_n^2}) - a = \frac{1}{4}(x_n^2 - 2a + \frac{a^2}{x_n^2}) = \frac{1}{4}(x_n - \frac{a}{x_n})^2 = \frac{(x_n^2 - a)^2}{4x_n^2} > 0$ (regardless of whether $x_n > \sqrt{a}$).
2. A monotonic-decreasing sequence that is bounded below converges (Rudin theorem 3.14). If $x_1 < \sqrt{a}$, the second property above means that $x_2 > \sqrt{a}$; then for $n > 2$ it is monotonically decreasing and bounded below by $\sqrt{a}$.
3. The limit $x = \lim_{n\to\infty} x_n$ satisfies $x = \frac{1}{2}(x + \frac{a}{x})$, which is easily solved to show that $x^2 = a$.

However, this proof by itself tells us nothing about *how fast* the sequence converges

2.2 Convergence example

Using the accompanying Julia notebook, we will apply this method to compute the most famous root of all, $\sqrt{2}$. (Supposedly, the Greek who discovered that $\sqrt{2}$ is irrational was thrown off a cliff by his Pythagorean colleagues.). As a starting guess, we will use $x_1 = 1$, producing the following sequence when computed with about 60 digits of accuracy, where the correct digits are shown in boldface:

1
1.5
1.416666666666666666666666666666666666666666666666666666666675
1.41421568627450980392156862745098039215686274509803921568627 45
1.4142135623746899106262955788901349101165596221157440445849057
1.414213562373095048801689623502530243614981925776197428498289 0
1.4142135623730950488016887242096980785696718753772340015610125
1.41421356237309504880168872420969807856967187537694807317667 96

Looking carefully, we see that the **number of accurate digits approximately doubles on each iteration**. This fantastic convergence rate means that we only need seven Newton iterations to obtain more than 60 accurate digits—the accuracy is quickly limited only by the precision of our floating-point numbers, a topic we will discuss in more detail later on.

$\frac{1}{4}\left(x_n^2 + 2a + \frac{a^2}{x_n^2}\right) - a$

$x_n^2 - 2a + \frac{a^2}{x_n^2}$

$= \frac{1}{4}\left(x_n - \frac{a}{x_n}\right)^2 = \frac{(x_n^2 - a)^2}{(2x_n)^2}$

$$\sum_{n=0}^{\infty} \frac{f^{n}(a)}{n!}(x-a)^n \rightarrow \text{Taylor}$$

2.3 Convergence rate

Let us analyze the convergence rate quantitatively—given a small error δ_n on the n-th iteration, we will determine how much smaller the error δ_{n+1} is in the next iteration.

In particular, let us define $x_n = x\,(1+\delta_n)$, where $x = \sqrt{a}$ is the exact solution. This corresponds to defining $|\delta_n|$ as the **relative error**:

$$|\delta_n| = \frac{|x_n - x|}{|x|},$$

what powers of 10.

also called the **fractional error** (the error as a fraction of the exact value). Relative error is typically the most useful way to quantify the error because it is a *dimensionless* quantity (independent of the units or overal scalling of x). The logarithm $(-\log_{10}\delta_n)$ of the relative error is roughly the number of **accurate significant digits** in the answer x_n.

We can plug this definition of x_n (and x_{n+1}) in terms of δ_n (and δ_{n+1}) into our Newton iteration formula to solve for the iteration of δ_n, using the fact that $a/x = x$ to divide both sides by x:

$$1 + \delta_{n+1} = \frac{1}{2}\left(1 + \delta_n + \frac{1}{1+\delta_n}\right) = \frac{1}{2}\left[1 + \cancel{\delta_n} + 1 \cancel{- \delta_n} + \delta_n^2 + O(\delta_n^3)\right],$$

where we have Taylor-expanded $(1-\delta_n)^{-1}$. The $O(\delta_n^3)$ means roughly "terms of order δ_n^3 or smaller;" we will define it more precisely later on. Because the sequence converges, we are entitled to assume that $|\delta_n|^3 \ll 1$ for sufficiently large n, and so the δ_n^3 and higher-order terms are eventually negligible compared to δ_n^2. We obtain:

$$\boxed{\delta_{n+1} = \frac{\delta_n^2}{2} + O(\delta_n^3),}$$

$-\log_{10}\delta_n^2$ also

which means the **error roughly squares** (and halves) on each iteration once we are close to the solution. Squaring the relative error corresponds precisely to doubling the number of significant digits, and hence explains the phenomenon above. This is known as **quadratic convergence** (not to be confused with "second-order" convergence, which unfortunately refers to an entirely different concept).

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

$x = \sqrt{a} \rightarrow$ exact solution

$$\Rightarrow \boxed{x = \frac{a}{x}}$$

$$\frac{x_{n+1}}{x} = \frac{1}{2}\left(\frac{x_n}{x} + \frac{a}{x_n x}\right)$$

$\frac{x}{x_n}$

$$1 + \delta_{n+1} = \frac{1}{2}\left(1 + \delta_n + \frac{1}{1+\delta_n}\right).$$