*n*th root algorithm

From Wikipedia, the free encyclopedia

The principal *n*th root $\sqrt[n]{A}$ of a positive real number *A*, is the positive real solution of the equation

$$x^n = A$$

(for integer *n* there are *n* distinct complex solutions to this equation if $A > 0$, but only one is positive and real).

There is a very fast-converging ***n*th root algorithm** for finding $\sqrt[n]{A}$:

1. Make an initial guess x_0
2. Set $x_{k+1} = \frac{1}{n}\left[(n-1)x_k + \frac{A}{x_k^{n-1}}\right]$. In practice we do $$\Delta x_k = \frac{1}{n}\left[\frac{A}{x_k^{n-1}} - x_k\right]; x_{k+1} = x_k + \Delta x_k.$$
3. Repeat step 2 until the desired precision is reached, i.e. $|\Delta x_k| < \epsilon$.

A special case is the familiar square-root algorithm. By setting *n* = 2, the *iteration rule* in step 2 becomes the square root iteration rule:

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right)$$

Several different derivations of this algorithm are possible. One derivation shows it is a special case of Newton's method (also called the Newton-Raphson method) for finding zeros of a function $f(x)$ beginning with an initial guess. Although Newton's method is iterative, meaning it approaches the solution through a series of increasingly accurate guesses, it converges very quickly. The rate of convergence is quadratic, meaning roughly that the number of bits of accuracy doubles on each iteration (so improving a guess from 1 bit to 64 bits of precision requires only 6 iterations). For this reason, this algorithm is often used in computers as a very fast method to calculate square roots.

For large *n*, the n^{th} root algorithm is somewhat less efficient since it requires the computation of x_k^{n-1} at each step, but can be efficiently implemented with a good exponentiation algorithm.

Derivation from Newton's method [edit]

Newton's method is a method for finding a zero of a function *f(x)*. The general iteration scheme is:

1. Make an initial guess x_0
2. Set $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
3. Repeat step 2 until the desired precision is reached.

The n^{th} root problem can be viewed as searching for a zero of the function

$$f(x) = x^n - A$$

So the derivative is

$$f'(x) = nx^{n-1}$$

and the iteration rule is

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)}{f'(x_k)} \\ &= x_k - \frac{x_k^n - A}{nx_k^{n-1}} \\ &= x_k - \frac{x_k}{n} + \frac{A}{nx_k^{n-1}} \\ &= \frac{1}{n}\left[(n-1)x_k + \frac{A}{x_k^{n-1}}\right] \end{aligned}$$

leading to the general n^{th} root algorithm.

See also [edit]

- Recurrence relation

References [edit]

- Atkinson, Kendall E. (1989), *An introduction to numerical analysis* (2nd ed.), New York: Wiley, ISBN 0-471-62489-6.

Categories: Root-finding algorithms