On Computing General Root Algorithms Based on Binomial Expansion, Bernoulli's Method of Continuous Compound Interest, Series Expansion and Other Modification Methods.

Mr. K. Dikomang[1], R. Tshelametse[2], T. Yane[3]

University of Botswana

Abstract
Most of the Pth root algorithms exhibit high latency or small convergence rates when iteratively computing the roots. Here we present a slew of algorithms based on series expansions of binomial form, and exponential terms that show low latency or small computational cost for finding the Pth root. The latency decreases if the family of series taken are truncated at higher terms. We show that Babylonian method converges quadratically while the binomial series show an increase in convergence rate from quadratic, cubic and higher order $O(t^N)$ convergence rate as the series is sequentially truncated at higher order terms.

Keywords: Binomial Expansion, Newton Rhapson Method, Babylonian Method, Computational Cost.

Introduction

Square rooting technology is ubiquitously implemented in IEEE, science, and mathematics platforms such as in computing for root mean square, solving linear equations, in vision apparatus and computer graphics. Square rooting algorithms are also used in engineering platforms like field programmable gates arrays and spectrum analysers [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

The Babylonians were able to formulate a remarkable iterative loop [11, 12], for computation of positive square roots in circa 1500 BC this primitive mathematical program emanating from the ancient Babylonian epoch is still implemented today in many computing models. Kosheleva [13], explains how the Babylonian method emerges naturally from somehow loose and archaic methods, he shows that by just taking the first term of the binomial series and making a slight mathematical tweak will generate a method that links well to the primitive Babylonian algorithm. Osler [14], somehow did extend the Babylonian method to high order roots not just square root using a brute force method where he introduces well-conditioned coefficients to fit the wanted algorithm using a technology the is quasi graph fitting or brute force through trial and error and observation. In addition, Johnson [15], provided an iterative procedure for computing square root. Knill [16], on the other hand developed a modified Babylonian method for calculating square root and Dubeau [17], used a double iteration method to calculate general roots. Bagala et. al [18], showed using binomial expansion that square root can be computed from the first term of the series which is analogous to the Babylonian method. The Babylonian method which was later refined by Heron of Alexandria is still in use today in calculators and other computing units [16].

The Heron method furnishes the average of a two guess roots. In the past, a lot of mathematicians have shown a proficient approximation technique for square roots, and the general pth roots using binomial expansion, and brute force [16, 18] approach while generating generic computational algorithms which are analogous to Newton-Rhapson and the Babylonian method. These methods show case the Babylonian method when the binomial series is truncated at the second term, or when authors furnish a brute force method to approximate the Babylonian iterative loop.

Several methods for computing square roots exist in the literature, including a paleo technique that was formulated by an ancient Indian mathematician Aryabhata using a long division algorithm known as digit-by-digit calculation to compute square roots, and cube roots documented in his manuscript aryabhatiya [19]. Another paleo document with origins from India was written by Bhakshali that gives an algorithm for computing square roots which was shown to have quartic convergence because it's an envelope of two iterations of Newton-Rhapson method [20, 21]. This method is also equivalent to two iterations of Heron's algorithm which converges quadratically when two iterations are taken the convergence rate is quartic. Taylor expansion is yet another technique that can be furnished to compute square roots which is like binomial expansion.

In digital signal processing and field programmable gate array robust square rooting techniques that surpass the need to use division in calculations are employed as division tends to serve as a bottle neck in code execution time leading to increased computational cost, or high network latency. Most practitioners want to use cheaper calculation units that have efficient software rather than spend money on hard wares that centralize expensive divisors [22]. It's true that most computer hardware has effectively powerful multipliers that's why algorithms like, single clock and reciprocal of square root methods are employed in digital signal processing and field programmable gate array. In our work we focus on algorithms that have small latency thus they converge rapidly rather than rely on the hardware specifications.

Johnson et al. [23] discussed convergence rate of the Babylonian algorithm quantitatively using relative error of the tth iteration showing that the error will diminish quickly following subsequent iterative terms. Given $A_n = A(1+t_n)$ where $A = \sqrt{a}$, is the exact solution, A relative error expression $t_n = \frac{|A_n - A|}{A}$ given the Babylonian algorithm as viz:

$$A_{n+1} = \frac{A_n}{2} + \frac{A^2}{2A_n}. \tag{1}$$

Set A = 1 and $A_n = 1 + t_n$.

It can be shown that.

$$1 + t_{n+1} = \frac{1+t_n}{2} + \frac{1}{2(1+t_{n)}}. \tag{2}$$

On expanding the term $\frac{1}{2(1+t_{n)}}$ as a geometric series, one obtains,

$$1 + t_{n+1} = \frac{1+t_n}{2} + \frac{1}{2}(1 - t_n + t_n^2 + O(t_n^3)), \tag{3}$$

where, t_n^3 is negligibly small. After some arithmetic manipulations we get the following:

$$t_{n+1} = t_n^2 + O(t_n^3) \tag{4}$$

so, it has been shown that the Babylonian algorithm converges quadratically since higher terms of order 3 and beyond are negligibly small.

Rubin et al [24] provided a proof of convergence of monotonic sequence when it is bounded both above and below converging to a limit $\sqrt{a}$,. He detailed in his book principles of mathematical analysis 3 axioms to confirm the convergence of a sequence.

1) Suppose $A_n > \sqrt{a}$, it follows that $\sqrt{a} < A_n < A_{n+1}$.
2) A monotonically decreasing series that is bounded below converges if $A_1 < \sqrt{a}$, and above if $A_2 > \sqrt{a}$, for $n > 2$. This is a monotonically decreasing sequence.
3) The limit $A = \lim_{N\to\infty} A_N$ is a Babylonian algorithm given by,

$$A = \frac{A_N}{2} + \frac{A^2}{2A_N}. \tag{5}$$

Whiteside et al. [7] gave an outline of binomial series for the P^{th} power crediting Briggs, partial discovery, the 17th century English mathematician, in anticipation of Newton of the general binomial expansion.

$$(1+A)^P = \lim_{N\to\infty} \sum_{0\le\lambda<N}[\binom{P}{\lambda} * A^{\lambda}], A < 1. \tag{6}$$

$$(1+A)^P = 1 + PA + \frac{P(P-1)}{2!}A^2 + \frac{P(P-1)(P-2)}{3!}A^3 + \cdots \tag{7}$$

Johnson et al [13] proposed a technique that shows that by extracting the first two terms of the binomial series yields a Babylonian algorithm as viz, [25]:

$$(A^2 + \mathrm{h})^{\frac{1}{2}} \sim \mathrm{A} + \frac{h}{2A}, 0 < \mathrm{h} < A^2 \tag{8}$$

When employing the first two terms of the binomial series, they gave a crude approximation of the Babylonian algorithm as vide infra trying to justify the discovery or mechanics of the algorithm from conventional mathematical technique. This is a method that was discovered by the Babylonians before Heron of Alexandria modified it to what is currently known formula for the Babylonian algorithm.

$$(1 + \frac{h}{A^2})^{\frac{1}{2}} \sim 1 + \frac{h}{2A^2} \tag{9}$$

Square rooting algorithm is often employed in field programmable gate array applications of image processing [20], spectrum analyser [26] and many others. Bagala et al [17, 27] proposed a single clock square root algorithm applicable in field programmable gate array.

Methods and Procedures

Proposed Methods

In this section, we derive Binomial Expansion of Pth root and different algorithms as subsequently extracted by truncating the series at different levels of performance or the Nth term.

We want to solve for A_0, we start by adding and subtracting a dummy term A_1^p

$$A_0 = (A_O^P + A_1^P - A_1^P)^{\frac{1}{P}} \tag{10}$$

$$A_0 = \left(\frac{A_0^P}{A_1^P} - 1 + 1\right)^{\frac{1}{p}} A_1. \tag{11}$$

Let $x = \frac{A_0^P}{A_1^P} - 1$, then.

$$A_0 = (x+1)^{\frac{1}{p}} A_1 \tag{12}$$

We then expand the binomial series in terms of x.

$$A_0 = (1 + \frac{x}{P} + \frac{\frac{1}{P}\left(\frac{1}{P}-1\right)}{2!} x^2 + \frac{\frac{1}{P}\left(\frac{1}{P}-1\right)\left(\frac{1}{P}-2\right)}{3!} x^3 + \cdots) A_1 \tag{13}$$

If we truncate the binomial expansion at the first term, we get the general Babylonian algorithm (or the first-generation algorithm) of the P^{th} root.

$$A_0 = A_1\left(1 - \frac{1}{P}\right) + \frac{A_1}{P}\frac{A_0^P}{A_1^P} \tag{14}$$

If we set P = 2, we consequently derive the Babylonian loop for square roots

$$A_0 = A_1\left(1 - \frac{1}{2}\right) + \frac{A_1}{2}\frac{A_0^2}{A_1^2} \quad = \frac{A_1}{2} + \frac{A_0^2}{2A_1} \tag{15}$$

Second generation algorithm is extracted from truncating the binomial series at the second term.

$$A_0 = A_1 + \frac{xA_1}{P} + \frac{\frac{1}{P}\left(\frac{1}{P}-1\right)}{2!} A_1 x^2 \tag{16}$$

The series when truncated at consecutively higher terms yields even more robust iterative algorithms that perform better than their predecessors and it is noteworthy that the series converges to the P^{th} root when all the terms are taken.

Rate of convergence of the second-generation algorithm as it was derived from binomial expansion and compared to quadratically converging Babylonian method for square roots is shown as viz:

The second-generation algorithm is depicted below:

$$A_{n+1} = \frac{A_n}{2} + \frac{A_0^2}{2A_n} - \frac{1}{8}\left(\frac{A_0^2}{A_n^2} - 1\right)^2 A_n \tag{17}$$

Substitute $A_0 = 1$ and $A_n = t_n + 1$ into the above equation, where t_n is a relative error term.

$$1 + t_{n+1} = \frac{t_n + 1}{2} + \frac{1}{2(1+t_{n)}} - \frac{1}{8}\left(\frac{1}{(1+t)^2} - 1\right)^2 (1+t)^1 \tag{18}$$

By taking geometric series of reciprocal terms and simplifying the equation we get

$$t_{N+1} = t_N^3 + O(t_N^4) \tag{19}$$

We have shown that our algorithm converges cubically and thus subsequent algorithms in the binomial expansion have high order convergence rate better than the Babylonian algorithm.

A Rooting algorithm derived from the mathematics of compound interest (Bernoulli's method)

Here we are going to derive the modified Babylonian iterative method for the p^th^ root using a formula for continuous compound interest. In addition, we also derive an extended second-generation iterative method for the P^th^ root.

We want to approximate A_0 by adding a fudge term to a guess function $A_{1.}$ A mathematical technique analogous to the method of gradient descent where the step size is 1 and a 1-dimensional gradient is denoted by N. we first introduce a thought process or a heuristic argument of how the method works. This is a simple mental algorithm or mathematical thought process to show how we arrive at the square root of A_0^2.

- Choose Trial root function, $A_1 = 5$

- Compute $A^2_1 = 25$
- If $A_1^2 > A_0^2$ Lower the value of $A_1 = 5$ by $\lambda = 0.5$
- A_2 becomes 4.5
- Compute $A^2_2 = 20.25$
- If $A_2^2 > A_0^2$, Note $20.25 > 16$
- Lower$A_2 = 4.5$ by $\lambda = 0.5$
- $A_1 = 4$
- Q.E.D, our algorithm has converged,
- congratulations!!!

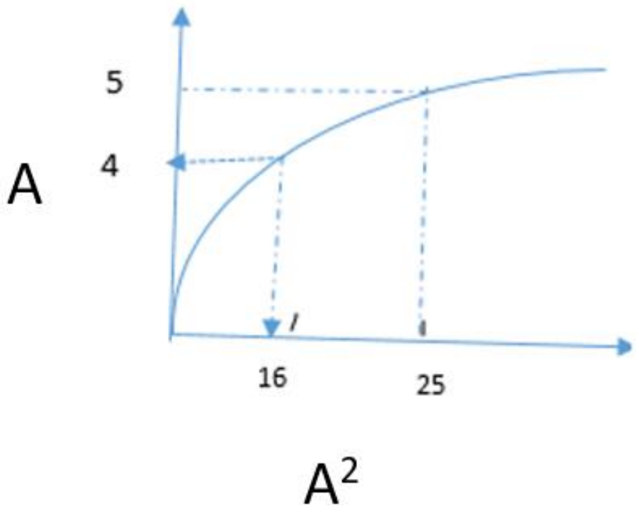

Figure 1. A map of square root A against square A^2

The length (gradient) of the line is:

$$N = A_1 - A_0 \tag{20}$$

Which leads to:

$$A_0^P = (A_1 - N)^p \tag{21}$$

We take the limiting case of compound interest that applies when the power is infinite not just 2 for square roots but for general positive root of real numbers.

$$A_0^P = A_1^P \left(1 - \frac{N}{A_1}\right) P \tag{22}$$

Then we excerpt Bernoulli formula for compound interest or exponentials

$$A_O^s = A_1^P \, e^{-\frac{PN}{A_1}} \tag{23}$$

power series expansion of N is derived by expressing $\ln\left(\frac{A_0^P}{A_1^P}\right)$ as a power series.

$$N \sim -\frac{A_1 \ln\left(\frac{A_0^P}{A_1^P}\right)}{P} \tag{24}$$

We only show the truncated power series of $\ln\left(\frac{A_0^P}{A_1^P}\right)$ taken at the first term.

$$N \sim -\frac{A_1}{P}\left(\frac{A_O^P}{A_1^P} - 1\right) + \cdots \tag{25}$$

$$N \sim \frac{A_O^P}{PA_1^{P-1}} + \frac{A_1}{P} \tag{26}$$

Substitute N into equation [20]

$$A_0 \sim A_1 + \frac{A_O^P}{PA_1^{P-1}} - \frac{A_1}{P} \tag{27}$$

$A_0 \sim \frac{A_1(P-1)}{P} \frac{A_O^P}{PA_1^{P-1}}$, this is the general Babylonian iterative algorithm. Both the binomial expansion and power series expansion of $\ln\left(\frac{A_0^P}{A_1^P}\right)$ from the method of compound interest give the general Babylonian algorithm when truncated at the first term.

Second generation Modified Babylonian method.

The second-generation Babylonian method is derived by truncating the $\ln\left(\frac{A_0^P}{A_1^P}\right)$ at the second term. We obtain this using equation [24] to get:

$$N \sim -\frac{A_1}{P}\left(\frac{A_O^P}{A_1^P} - 1\right) + \frac{A_1}{2P}\left(\frac{A_O^P}{A_1^P} - 1\right)^2 + \cdots \tag{28}$$

Substitute N into $A_0 = A_1 - N$, from gradient descent algorithm (20), we obtain:

$$A_0 = A_1 + \frac{A_1}{P}\left(\frac{A_O^P}{A_1^P} - 1\right) - \frac{A_1}{2P}\left(\frac{A_O^P}{A_1^P} - 1\right)^2 + \cdots. \tag{29}$$

We prove that A_N is a monotonically converging function and is bounded.
Prove that $A_{N+1} \leq A_n$ or $A_0 \leq A_{N+1}$

The following is a second-generation Babylonian algorithm for square roots extracted from the Bernoulli method of compound interest power series.

$$A_N - \frac{3A_N}{4} + \frac{A_0^2}{A_N} - \frac{A_0^4}{4A_N^3} \leq A_N \tag{30}$$

$$-3A_N^4 + 4A_N^2A_0^2 - A_0^4 \leq 0 \tag{31}$$

Set $H = A_N^2$

$$0 \leq 3H^2 - 4A_0^2H + A_0^4 \tag{32}$$

$$0 \leq (A_0^2)(H - A_0^2) \tag{33}$$

replace H by A_N^2

$$A_0 \leq A_N \quad or \quad \sqrt{3}A_0 \leq A_N \tag{34}$$

Hence shown that A_0 is bounded above.
The only wrinkle is the insight that this method (Bernoulli series of continuous compound interest) does not converge to A_0 but to some numerical value close to A_0. Each truncation at subsequent term is an iterative loop but it's not necessarily true that successive terms perform better than their predecessors like with the method of binomial expansion. There is a sweet spot somewhere in the series progression when it comes to a more robust iterative algorithm.

General roots here are obtained by extracting only the odd terms of the Bernoulli series.

$$A_0^p = (A_1 - N)^p \tag{35}$$

$$\left(\frac{A_0}{A_1}\right)^p = \left(1 - \frac{N}{A_1}\right)^p \tag{36}$$

Again, we employ the Bernoulli formula of compound interest.

$$A_0^P = A_1^P e^{\frac{-NP}{A_1}} \tag{37}$$

The root is expressed as vide infra.

$$A_0 = A_1 e^{\frac{-N}{A_1}} \tag{38}$$

Note that N represents the number of iterations and P is the power of A
Let us make N the subject of the argument.

$$N = -\frac{A_1}{p} \ln\left(\frac{A_0}{A_1}\right)^P \tag{39}$$

Substitute N into equation [38]

$$A_0 = A_1 e^{\frac{1}{P}\ln\left(\frac{A_0}{A_1}\right)^P} \tag{40}$$

Power series expansion of

$$\log\left(\frac{A_0}{A_1}\right)^P = \left[\left(\frac{A_0}{A_1}\right)^P - 1\right] - \frac{1}{2}\left[\left(\frac{A_0}{A_1}\right)^P - 1\right]^2 + \frac{1}{3}\left[\left(\frac{A_0}{A_1}\right)^P - 1\right]^3 - \cdots \tag{41}$$

Let $x = \left[\left(\frac{A_0}{A_1}\right)^p - 1\right]$ (42)

We take all the odd terms of the series and drop the even powers:

$$A_0 = A_1 e^{\frac{1}{P}\log\left(\frac{A_0}{A_1}\right)^P} \tag{43}$$

Power series expansion of equation [43]

$$A_0 = A_1 \left\{1 + \frac{1}{P}\left(\left[\left(\frac{A_0}{A_1}\right)^2 - 1\right] - \frac{1}{2}\left[\left(\frac{A_0}{A_1}\right)^P - 1\right]^2 + \frac{1}{3}\left[\left(\frac{A_0}{A_1}\right)^P - 1\right]^3 \ldots -\right)\right\} \tag{44}$$

$$A_0 = A_1 \left\{1 + \frac{1}{P}\left[x - \frac{1}{2}x^2 + \frac{1}{3}x^3 \ldots\right] -\right\} \tag{45}$$

We show that equation [45] of odd terms is numerically expressed as below.

$$\sim A_1\left\{1 + \frac{1}{\mathrm{P}}\ln(1 + \mathrm{x})\right\} \tag{46}$$

Let's extract the odd terms only.

$$A_0 = A_1\left\{1 + \frac{1}{P}\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 \ldots\right)\right\} \tag{47}$$

Let

$$y = x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \cdots \tag{48}$$

$$\frac{dy}{dx} = 1 + x^2 + x^4 + x^6 + x^8 \tag{49}$$

Note that equation [49] is a geometric series of the form:

$$\frac{dy}{dx} = \frac{1}{1 - x^2} \tag{50}$$

We then solve for y numerically as an approximate value of the sum by taking an integration instead of the sum.

$$y = \int \frac{1}{1 - x^2} dx \tag{51}$$

$$y = 0.5\left(\ln(x+1) - \ln(1-x)\right) \tag{52}$$

This is the upper bound of y which shows that the odd terms of the series converge to.

$$A_0 = A_1 + \frac{A_1}{2P}\left(\ln(x+1) - \ln(1-x)\right) = A_1\left\{1 + \frac{1}{P}\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 \dots\right)\right\} \tag{53}$$

When we excerpt the first term of the series, we succinctly derive the general Babylonian iteration loop for the roots.

$$A_0 = A_1 + \frac{A_1}{P}x \tag{54}$$

Substituting an expression of x into equation [54]

$$A_0 = A_1 + \frac{A_1}{P}\left(\frac{A_0}{A_1}\right)^P - \frac{A_1}{P} \tag{55}$$

The general Babylonian iteration loop is typified by the formula above for A_0.

Now let us truncate the series at the second odd term so that we get yet another iterative algorithm.

$$A_0 = A_1\left\{1 + \frac{1}{P}\left(x + \frac{1}{3}x^3\right)\right\} \tag{56}$$

Substituting the expression for x into equation [56] gives:

$$= A_1 - \frac{4A_1}{3P} + \frac{2A_1A_0^P}{PA_1^P} + \frac{A_1A_0^P}{3PA_1^{3P}} - \frac{A_1A_0^{2P}}{PA_1^{2P}} \tag{57}$$

We use binomial expansion to evaluate the 3rd term of the series of continuous compound interest.

$$\frac{A_1}{5P}\left[\left(\frac{A_0}{A_1}\right)^P - 1\right]^5$$

$$= \frac{A_1}{5P}\left[\left(\frac{A_0}{A_1}\right)^{5P} - 5\left(\frac{A_0}{A_1}\right)^{4P} + 10\left(\frac{A_0}{A_1}\right)^{3P} - 10\left(\frac{A_0}{A_1}\right)^{2P} + 5\left(\frac{A_0}{A_1}\right)^{P} - 1\right] \tag{58}$$

Now we show the expression of Babylonian for the 3rd generation from the Bernoulli series of continuous compound interest.

$$A_0 = \frac{A_1}{5P}\left(\frac{A_0}{A_1}\right)^{5P} - \frac{A_1}{P}\left(\frac{A_0}{A_1}\right)^{4P} + \frac{2A_1}{P}\left(\frac{A_0}{A_1}\right)^{3P} - \frac{2A_1}{P}\left(\frac{A_0}{A_1}\right)^{2P} + \frac{A_1}{P}\left(\frac{A_0}{A_1}\right)^{P} - \frac{A_1}{5P} + A_1 - \frac{4A_1}{3P} + \frac{2A_1}{P}\left(\frac{A_0}{A_1}\right)^{P} + \frac{A_1}{3P}\left(\frac{A_0}{A_1}\right)^{3P} - \frac{A_1}{P}\left(\frac{A_0}{A_1}\right)^{2P} \tag{59}$$

General methodology when considering all the terms, odd and even of the series for Bernoulli method of continuous compound interest.

$$A_1\left\{1+\frac{1}{p}\left(x-\frac{1}{2}x^2+\frac{1}{3}x^3\dots\right)\right\} \tag{60}$$

The odd terms only method

$$A_{N+1}=A_N\left\{1+\frac{1}{P}(x+\frac{1}{3}x^3+\frac{1}{5}x^5\dots)\right\}=A_N+\frac{\mathrm{A_N}}{2\mathrm{P}}\left(\ln(\mathrm{x}+1)-\ln(1-\mathrm{x})\right) \tag{61}$$

$x=\left[\left(\frac{A_0}{A_N}\right)^p-1\right]$

This algorithm performs better than taking all terms or even terms only.

The even terms only method

$$A_0=A_1\left\{1+\frac{1}{P}\left(\frac{1}{2}x^2+\frac{1}{4}x^4\dots\right)\right\}=A_1(1+\frac{1}{2P}\left(\ln(1-\mathrm{x}^2)\right) \tag{62}$$

This algorithm performs poorly, so it shall not be considered as it is a tortoise when it comes to iteratively computing P^th roots.

Surrogate general algorithm as a modification of the general root algorithm

By using a quasi-curve fitting method we successfully derive a surrogate yet more robust algorithm that is better than the three Bernoulli algorithms, by modifying equation [63] we get [64] or the surrogate algorithm,

$$A_0=A_1\left\{1+\frac{1}{p}(\ln(1+x))\right\} \tag{63}$$

Is

$$A_0=A_1\{1+(\ln(1+x'))\} \tag{64}$$

Where:

$$x'=\left(\frac{1}{P}\frac{A_0^P}{A_1^P}-\frac{1}{P}\right) \tag{65}$$

Experimental Results

When the Binomial series is truncated at terms 1, 2, 3 … etc. a progression trend of convergence rate is observed.

When the series is truncated at term 1 Babylonian algorithm is obtained which converges quadratically, term 2, term 3, … result in $O(t^3)$, $O(t^4)$, $O(t^N)$ convergence rates, respectively. Here we take the case of square roots to illustrate our main ideas.

First generation algorithm or Babylonian iteration loop:

$$A_{N+1}=\frac{A_N}{2}+\frac{A_0^2}{2A_N} \tag{66}$$

Set $A_0=1$ and $A_N=t_N+1$

$$A_{N+1} = t_{N+1} + 1 = \frac{t_N+1}{2} + \frac{1}{2(t_N+1)} \qquad (67)$$

We employ geometric series to represent the reciprocal terms.

Convergence rate is $t_N^2 + O(t_N^3)$, higher order terms disappear because: $0 < t_n < 1$

Second-generation binomial algorithm,

$$A_{N+1} = \frac{A_N}{2} + \frac{A_0^2}{2A_N} - \frac{A_N}{8}\left(\frac{A_0^2}{A_N^2} - 1\right)^2 \qquad (68) \qquad t_N + 1 =$$

$$A_{N+1} = \frac{t^3}{2} - O(t^4) \qquad (69)$$

The algorithm converges cubically.

Third-generation binomial series:

$$A_{N+1} = \frac{A_N}{2} + \frac{A_0^2}{2A_N} - \frac{A_N}{8}\left(\frac{A_0^2}{A_N^2} - 1\right)^2 + \frac{A_N}{16}\left(\frac{A_0^2}{A_N^2} - 1\right)^3 \qquad (70)$$

$$t_N + 1 = A_{N+1} = \frac{5t^4}{8} + O(t^5) \qquad (71)$$

The third-generation binomial algorithm converges quartically.

This trend continues for higher order terms.

Iteration	Babylonian $\frac{A_N}{2} + \frac{A_0^2}{2A_N}$	Even terms only $A_N(1 + \frac{1}{4}[\ln(1 - (\frac{A_0^2}{2A_N^2} - \frac{1}{2})^2)])$	Odd terms only $A_N + \frac{A_N}{4}(ln(x+1) - ln(1-x))$ $x = \left[\left(\frac{A_0}{A_N}\right)^2 - 1\right]$	Surrogate Algorithm $A_1\{1 + (\ln(1 + x'))\}$ $x' = \left(\frac{1}{P}\frac{A_0^P}{A_1^P} - \frac{1}{P}\right)$	General terms $A_N - \frac{A_N}{2}\ln\left(\frac{A_0^2}{A_N^2}\right)$	Second-generation Binomial algorithm $A_{N+1}\frac{A_N}{2} + \frac{A_0^2}{2A_N} - \frac{A_N}{8}\left(\frac{A_0^2}{A_N^2} - 1\right)^2$
1	4.1	4.826 496 996	4.057 785 247	4.007 745 306	3.884 282 243	4.019
2	4.001 219 512	4.701 952 728	4.000 396 147	4. 000 000 005	3.998 309 797	4.000 044 411
3	4.000 000 186	4.608 609 337	4.000 000 002	4	3.999 999 643	4

Discussion

The computational cost of selected algorithms increases in the following fashion: surrogate method algorithm, second generation binomial series derived algorithm,, odd terms only Bernoulli algorithm, general Bernoulli algorithm, Babylonian algorithm, even terms only Bernoulli algorithm. But the study of their convergence rates show that higher generation binomial terms have higher Order rate of convergence rates. Babylonian series converges quadratically, while second, third and Nth -generation binomial algorithms have progressive higher convergence rates as O (t^3), (t^4) …(t^N) respectively. The Babylonian method is simpler and has satisfactory convergence rate and computational cost. Other methods are sophisticated and more robust in computing roots. But the even only terms algorithm is poor and has high latency, so it is less useful in computing roots, it is a tortoise algorithm while higher order binomial algorithm and surrogate method have low latency. The methods furnished here can be used to compute root mean square and solve linear equations or serve as go to methods in IEEE platforms that require small computational costs.

Conclusion

Several methods have been applied to calculate the roots of positive real numbers. Classical methods include the Babylonian and Newton Raphson techniques. The current paper has employed series expansion as a method of generating algorithms for roots truncated at different levels of accuracy. The work has deduced several formulations that calculates the roots. The binomial expansion is somehow more superior to other series expansion methods. This paper wishes to bridge the gap in which classical methods did not exhaust.

References

1. Dikomang, K. (2023). In the computation of square roots using trigonometric exact method, International journal of Advances in Engineering and Management, vol 5, 680 -683
2. Dianov, A, Anuchin A. (2020). review of fast square root calculation methods for fixed microcontroller-based control systems of power electronica, international journal of power electronics and drive system, vol 1, pp1153- 1164
3. Walczyk, C. J, Moroz, L.V, and. Cieslinki, J. L. (2021). improving the accuracy of the fast inverse square root by modifying Newton-Rhapson corrections, Entropy, vol 13, 25(1), 86
4. Raspberry Pi 3 Model B. RS Components: Corby, UK. Available online: https://www.alliedelec.com/m/d/4252b1ec d92888dbb9d8a39b536e7bf2.pdf (accessed on 08 March 20Raspberry Pi 3 Model B. RS Components: Corby, UK. Available online: https://www.alliedelec.com/m/d/4252b1ec d92888dbb9d8a39b536e7bf2.pdf (accessed on 27 May 2020).
5. Pardeshi, M. A, Jadhav, S. A, Nair, N. A. 2023. Comparative study of calculating square roots using Hero's formula and a novel method discovered, international journal of advances in engineering and management, Vol. 5, Issue 2 Jan2023, pp:366-370,
6. Kwon, T. J, Draper, J. 2019. Floating point division and square root implementation using a Taylor-series expansion algorithm with reduced lookup table, computation, , 7(3),
7. Lomont, C. (2003). Fast inverse square root, Purdue university, technical report,
8. Parhami, B. Computer Arithmetic: Algorithms and Hardware Designs; Oxford University Press: Oxford, UK, 2010; ISBN 9780195328486. [Google Scholar]
9. Dubeau, F. (1998). nth root extraction: double iteration process and Newton method, Journal of computational and applied mathematics vol 91, 191-198

10. Hagara, M, Stojanovic R, Kubinec P, Ondracek, O. (2011). "Localization of moving edge with sub-pixel accuracy in 1-D images and its FPGA implementation", in Microprocessors and Microsystems, Volume 51, pp. 1-7,
11. Burton, D. M. (1991). A history of mathematics: An introduction, McGraw Hill, (3rd ed), P79
12. Heath, T. (1921). A history of Greek mathematics, Clarendon press, oxford, , vol 2
13. Kosheleva. O. Babylonian method of computing the square root justification based on fuzzy techniques and on computational complexity.
14. Osler, T. J. (1999). Extending the Babylonian algorithm, mathematics and computer education vol 33, No. 2, pp 120-128
15. Kenneth, R. J. An iterative method for approximating square roots, mathematics magazine 62:4, 253-299
16. Ronald J. K. A modified Babylonian algorithm, The American mathematical monthly, 99:8; 734-737
17. Bagala, T. Fibich, A. Kubinec, M. H, P. Štofanik,V. single clock square root algorithm based on binomial series and its FPGA implementation, 2018 7th Mediterranean conference on embedded computing (MECO), 11-14 JUNE 2018, Budva, Montenegro
18. Clark. W. (1930). translation and commentary, the Aryabhatina of Aryabhata: an ancient Indian work on mathematics and astronomy, university of Chicago, Chicago, 11,
19. Hagashi. T. (1995). The Bhakshali manuscript: an ancient Indian mathematical treatise, John Benjamin's publishing company, Amsterdam
20. Plofker. K. (2007). Mathematics in India, in vector J. Katz, ed, mathematics of Egypt, Mesopotamia, China, India and Islam, A source book, Princeton university press, Princeton, N. J,
21. Dianov. A, Anuchin. A. (2020). review of fast square root calculation methods for fixed point micro-controller-based control systems of power electronics, international of power electronics and drive systems, vol 11, no.3, 1153-1164, https://math.mit.edu/~stevenj/18.335/newton-sqrt.pdf
22. Rubin, W. Principles of mathematical analysis, 3rd edition, pp 55
23. Eves, H. (1969). An Introduction to the History of Mathematics, Holt, Rinehart and Winston,
24. Feldhaus G. and Roth A. (2016). "A 1MHz to 50 GHz direct down-conversion phase noise analyzer with cross-correlation," in 2016 European Frequency and Time Forum (EFTF), pp. 1–4
25. Piromsopa J., Aporntewan C., Chongsatitvatana P. (2001). "An FPGA implementation of a fixed-point square root operation," in Proceedings of the IEEE Symposium on communication and Information Technologies, pp. 587-589.