Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 11, Number 3 (2015), pp. 1541-1555

http://www.ripublication.com

A Novel Way to Compute Square Root of A Real Number

Snehanshu Saha and Mushtaque Ahamed A

Department of Computer Science,
PES Institute of Technology-South Campus,
Bangalore, Karnataka, India. Pincode: 560100
E-mail: snehanshusaha@pes.edu
mushtaqueahamed41@yahoo.com

Abstract

Most of the methods and algorithms to find square root of a real number are inefficient and/or slow when implemented as a computer program. Some of the methods commonly used are too cumbersome to implement.In an effort to solve this problem, an algorithm is designed and analyzed in this work,which is both computationally efficient and easily implementable.

AMS subject classification: 41A20 (Approximation by rational function), 65D15 (Algorithm for functional approximation)
Keywords: Order of Convergence, Accuracy, Stability, Iterative method.

1. Introduction

Let N be a real number whose square root has to be evaluated. Let x be the initial approximate solution. Then a better approximate solution to $\sqrt{N}$ is $g(x)$. Where $g(x)$ is,

$$g(x) = \frac{Nx}{N + x^2} + \frac{N + x^2}{4x}$$

In order to achieve even better approximation ,recursively apply g by taking $g(x)$ as new value of x. i.e.,

$$g(g(x)) = g^2(x)$$

In effect when a sequence $\{x_n\}_{n=0}^{\infty}$ is generated using the relation $x_{n+1} = g(x_n)$ with x_0 being initial guess, the sequence converges to the exact solution, $\sqrt{N}$.Therefore we have our solution as,

$$\sqrt{N} = \lim_{i \to \infty} g^i(x)$$

Here, i denotes number of iterations. We claim that our method has an order of convergence = 4.

2. Related work

We mention some of the related work done by others in this respect. The *Taylor series expansion* [4] of $\sqrt{1+x}$ is given by

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} - \frac{5x^4}{128} + \frac{7x^5}{256} + \cdots$$

The series converges for $x \leq 1, x \in R^+$ but diverges for $x > 1$.

Newton–Raphson's formula [11] for $\sqrt{N}$ is

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

This formula generates approximate solutions iteratively. The method converges quadratically. We claim that our method is faster compared to NR as will be shown later.

Secant method [14] gives a formula to iteratively evaluate $\sqrt{N}$ as

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

It is known to have order of convergence equal to ≈ 1.618 (golden ratio).

Logarithmic method It can be shown that

$$\sqrt{N} = 10^{\frac{\log_{10} N}{2}}$$

Evaluation of logarithms and powers are computationally expensive,unless we can exploit large look-up tables of desired accuracy. Moreover manually evaluating logarithm is a computationally intensive task.

The *Babylonian Method* [17] can be described as follows. Let $x_0 \approx \sqrt{N}$ be the initial approximate solution. Then $\sqrt{N} = \lim_{n\to\infty} x_n$, where,

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

This method is endowed with quadratic convergence.

Digit-by-digit extraction (aka: Division Method) This is one of the most common methods taught at high schools. It is used to obtain the solution digit-by-digit. It involves a lot of trial and error to get the solution. Also as the number of digits to extract increases,the steps become cumbersome. It has linear convergence.

Vedic Method This is similar to the division method[10]. But, this procedure aims at obtaining solution in less number of steps but with greater sophistication.

Bisection Method In this method the solution is worked out iteratively similar to binary search. The error reduces by half in each iteration and converges linearly.

The methods dicussed are not as fast the method proposed here. Division method and Vedic method are non iterative algorithms. The implementation as a computer program is not easy. Other methods are iterative in nature. They can be easily implemented as computer programs.But they are not as efficient as our method. The next few sections would illustrate the accuracy, speed of convergence and stability of the proposed algorithm.

3. Saha-Ahmad (SA) Algorithm

Description:To find Square root of a given Real number N.
Input: $N \rightarrow$ *the real number whose square root has to be found,*
$x_0 \rightarrow$ *initial approximate solution*
$P \rightarrow$ *the desired number of decimal places to which to the solution needs to be accurate.*
Output: Approximate value of $\sqrt{N}$ *accurate to P decimal places.*
while $\parallel N - t \parallel \geq 10^{-P}$ *do*
$t \leftarrow g(t)$
return t

3.1. Proof of correctness

Consider the equation

$$g(x) = \frac{Nx}{N + x^2} + \frac{N + x^2}{4x}$$

The algorithm is correct if and only if

$$\sqrt{N} = \lim_{i \to \infty} g^i(x).$$

This is true if $g(x)$ is a more accurate solution than x itself $\forall\, i$, where i denotes number of iteration.

Let $x = \sqrt{M}$ and $N = M + h$. For small values of h, we have $x \approx \sqrt{N}$. Let x be

the initial solution. Now, consider

$$\begin{aligned}\sqrt{N}-\sqrt{M} &= \sqrt{M+h}-\sqrt{M}\\ &= \frac{(\sqrt{M+h})^2-(\sqrt{M})^2}{\sqrt{M+h}+\sqrt{M}}\\ &= \frac{h}{\sqrt{M+h}+\sqrt{M}}\\ &\approx \frac{h}{2\sqrt{M}}\end{aligned}$$

since h is small. Therefore,

$$\begin{aligned}\sqrt{N} &\approx \sqrt{M}+\frac{h}{2\sqrt{M}}\\ &= x+\frac{h}{2x}\\ &= \frac{x^2+h}{2x}\end{aligned}$$

The value obtained in the above expression is a better solution than x. Now, replacing x in the RHS of above equation by itself will yeild an even better solution.

$$\begin{aligned}\sqrt{N} &\approx \left(\frac{x^2+N}{2x}\right)_{x\to\left(\frac{x^2+N}{2x}\right)}\\ &= \frac{\left(\frac{x^2+N}{2x}\right)^2+N}{2\left(\frac{x^2+N}{2x}\right)}\\ &= \frac{Nx}{N+x^2}+\frac{N+x^2}{4x}\end{aligned}$$

Since g is the square root finding algorithm,we obtain

$$g(x)=\frac{Nx}{N+x^2}+\frac{N+x^2}{4x}.$$

This completes the proof. ■

3.2. Initial value of Solution

We can start with any positive real number as an initial approximate solution. For instance, let us start with $x_0 = N$ itself, or $x_0 = 0$. But, the number of iterations i may differ to get accurate solution. To reduce the number of iterations i, we have to start with a fairly good initial approximate solution. Such an initialization would be

$$x_0 \approx 2^{\log_4(N)}$$

This is due to the fact that

$$\sqrt{N} = 2^{\log_4(N)}$$

3.3. Pseudocode

We can implement the above algorithm as a computer program using the following pseudocode.

//To obtain initial approximate solution

$x \leftarrow 1$
$t \leftarrow \frac{N}{4}$
while $t > 1$
  $t \leftarrow \frac{t}{4}$
  $x \leftarrow 2x$
end while

//To obtain solution accurate to P decimal places

while $\| t - x \| \leq 10^{-P}$
  $t \leftarrow x$
  $x \leftarrow \frac{Nx}{N+x^2} + \frac{N+x^2}{4x}$
end while
return x

4. Order of Convergence

Consider the function

$$g(x) = \frac{Nx}{N+x^2} + \frac{N+x^2}{4x}.$$

The *Taylor series* expansion of the above function at the point $x = \sqrt{N}$ will be

$$\begin{aligned} g(x) &= g(\sqrt{N}) + (x-\sqrt{N})g^I(\sqrt{N}) + \frac{(x-\sqrt{N})^2}{2!}g^{II}(\sqrt{N}) \\ &\quad + \frac{(x-\sqrt{N})^3}{3!}g^{III}(\sqrt{N}) \\ &\quad + \frac{(x-\sqrt{N})^4}{4!}g^{IV}(\sqrt{N}) + \cdots \end{aligned}$$

Let us evaluate

$$g(\sqrt{N}), g^I(\sqrt{N}), g^{II}(\sqrt{N}), g^{III}(\sqrt{N}), g^{IV}(\sqrt{N}),$$

and $g^V(\sqrt{N})$. We have $g(\sqrt{N}) = \sqrt{N}$

$$g^I(x) = -\frac{2Nx^2}{(N+x^2)^2} + \frac{N}{N+x^2} - \frac{N}{4x^2} + \frac{1}{4}.$$

This implies,

$$g^I(\sqrt{N}) = 0$$

Now,

$$g^{II}(x) = \frac{8Nx^3}{(N+x^2)^3} - \frac{6Nx}{(N+x^2)^2} + \frac{N}{2x^3}.$$

This implies,

$$g^{II}(\sqrt{N}) = 0$$

Now,

$$g^{III}(x) = \frac{48Nx^2}{(N+x^2)^3} - \frac{48Nx^4}{(N+x^2)^4} - \frac{6N}{(N+x^2)^2} - \frac{3N}{2x^4}$$

This implies,

$$g^{III}(\sqrt{N}) = 0$$

Now,

$$g^{IV}(x) = \frac{384Nx^5}{(N+x^2)^5} - \frac{480Nx^2}{(N+x^2)^4} + \frac{120Nx}{(N+x^2)^3} + \frac{6N}{x^5}$$

This implies,

$$g^{IV}(\sqrt{N}) = \frac{3}{N\sqrt{N}}$$

Lastly,

$$g^{V}(x) = \frac{5760Nx^4}{(N+x^2)^5} - \frac{2160Nx^2}{(N+x^2)^4} - \frac{3840Nx^6}{(N+x^2)^6} + \frac{120N}{(N+x^2)^3} - \frac{30N}{x^6}$$

This implies,

$$g^{V}(\sqrt{N}) = -\frac{30}{N^2}$$

Substituting these values in the above *Taylor series* expansion, we obtain

$$g(x) = \sqrt{N} + \frac{(x-\sqrt{N})^4}{4!} \cdot \frac{3}{N\sqrt{N}} - \frac{(x-\sqrt{N})^5}{5!} \cdot \frac{30}{N^2} + \cdots$$

Further simplification yields

$$\frac{g(x)-\sqrt{N}}{(x-\sqrt{N})^4} = \frac{1}{8N\sqrt{N}} - \frac{x-\sqrt{N}}{4N^2} + O((x-\sqrt{N})^2)$$

Applying limit as $x \to \sqrt{N}$, we get

$$\lim_{x\to\sqrt{N}} \frac{g(x)-\sqrt{N}}{(x-\sqrt{N})^4}$$

$$= \lim_{x\to\sqrt{N}} \left(\frac{1}{8N\sqrt{N}} - \frac{x-\sqrt{N}}{4N^2} + O((x-\sqrt{N})^2) \right) \qquad = \frac{1}{8N\sqrt{N}}$$

By definition of order of convergence [11], if

$$\lim_{x\to\sqrt{N}} \frac{g(x)-\sqrt{N}}{(x-\sqrt{N})^{\alpha}} = \lambda$$

where λ is some constant. Then α is the order of convergence of the function $g(x)$. Therefore $\alpha = 4$. Thus *Order of Convergence* of our algorithm (SA) is 4.

5. Time Complexity

The time required for the complete execution of the SA algorithm in asymptotic [6] case is considered here. Consider the equation

$$\begin{aligned} g(x) &= \frac{Nx}{N+x^2} + \frac{N+x^2}{4x} \\ &= \frac{x}{4} + \frac{N}{4x^2} + \frac{N}{N+x^2} \\ &= \frac{x}{4} + \frac{N}{4x^2} + \left(\frac{N}{x^2} - \frac{N^2}{x^4} + \frac{N^3}{x^6} + \cdots\right) \\ &= \frac{x}{4} + \frac{5N}{4x^2} - \frac{N^2}{x^4} + \frac{N^3}{x^6} + \cdots \end{aligned}$$

For large value of x, i.e., $x \gg N$, value of $g(x) \approx \frac{x}{4}$. Thus for large value of x_n, the next approximate solution is $x_{n+1} = \frac{x_n}{4}$. That is the asymptotic behaviour of our algorithm is linear. The algorithm exhibits linear convergence, when $x \gg N$. Thus the number of iterations $i \propto x$ until $x \sim \sqrt{N}$.

5.1. Proof of Linear Convergence

Consider the function

$$g(x) = \frac{Nx}{N+x^2} + \frac{N+x^2}{4x}$$

The *Taylor series* expansion of above function as $x \to \infty$ will be

$$\begin{aligned} g(x) &= \lim_{a\to\infty} \left(g(a) + (x-a)g^{I}(a) + \frac{(x-a)^2}{2!} g^{II}(a) \right. \\ &\quad \left. + \frac{(x-a)^3}{3!} g^{III}(a) + \cdots \right) \end{aligned}$$

Consider the equation

$$g^{I}(a) = -\frac{2Na^2}{(N+a^2)^2} + \frac{N}{N+a^2} - \frac{N}{4a^2} + \frac{1}{4}.$$

This implies,

$$\lim_{a\to\infty} g^I(a) = \lim_{a\to\infty}\left(-\frac{2Na^2}{(N+a^2)^2} + \frac{N}{N+a^2} - \frac{N}{4a^2} + \frac{1}{4}\right) = \frac{1}{4}$$

Now,

$$g^{II}(a) = \frac{8Na^3}{(N+a^2)^3} - \frac{6Na}{(N+a^2)^2} + \frac{N}{2a^3}$$

This implies,

$$g^{II}(\infty) = 0$$

Now,

$$g^{III}(a) = \frac{48Na^2}{(N+a^2)^3} - \frac{48Na^4}{(N+a^2)^4} - \frac{6N}{(N+a^2)^2} - \frac{3N}{2a^4}$$

This implies,

$$g^{III}(\infty) = 0$$

and the iterations may continue. Thus

$$\begin{aligned} g(x) &= \lim_{a\to\infty}\left(g(a) + (x-a)g^I(a) + \frac{(x-a)^2}{2!}g^{II}(a) + \frac{(x-a)^3}{3!}g^{III}(a) + \cdots\right) \\ &= g(a) + \frac{x-a}{4} \end{aligned}$$

Thus we obtain

$$\lim_{x\to a}\frac{g(x)-g(a)}{x-a} = \frac{1}{4}$$

By definition of order of convergence, it is 1. Thus the SA algorithm is linearly convergent. This completes the proof. ■

6. Accuracy

The number of decimal places after decimal period serves as a parameter for accuracy[7]. If P is the number of decimal places to which the solution is accurate, the absolute error is meant to be less than or equal to 10^{-P}. When g is applied i times the absolute error would be given by $\parallel g^i(x) - \sqrt{N} \parallel$. This implies

$$10^{-P} \geq \parallel g^i(x) - \sqrt{N} \parallel .$$

In worse case,

$$10^{-P} = \parallel g^i(x) - \sqrt{N} \parallel$$

implying

$$P = -\log_{10} \parallel g^i(x) - \sqrt{N} \parallel .$$

P is a function of $N,\ x\ and\ i$.

Let P be,

$$P(N, x, i) = -\log_{10} \parallel g^i(x) - \sqrt{N} \parallel .$$

Neglecting higher order terms of

$$g(x) = \sqrt{N} + \frac{h^4}{8N^{3/2}} - \frac{h^5}{4N^2} + \frac{17h^6}{48N^{5/2}} - \frac{5h^7}{16N^3} + \frac{9h^8}{32N^{7/2}} + O(h^9)$$

where $h = x - \sqrt{N}$, we get

$$g(x) = M + \frac{h^4}{8M^3}$$

where $M = \sqrt{N}$. This implies,

$$\log_{10}(g(x) - M) = 4\log_{10} h + c$$

where $c = -\log_{10}(8M^3) = -3\log_{10}(2M)$. Now,

$$g^2(x) = M + \frac{h_1^4}{8M^3}$$

where $h_1 = g(x) - \sqrt{N}$. i.e,

$$g^2(x) = M + \frac{(g(x) - M)^4}{8M^3}.$$

This implies,

$$\log_{10}(g^2(x) - M) = 4\log_{10}(g(x) - M) + c$$

where

$$c = -\log_{10}(8M^3) = -3\log_{10}(2M).$$

But

$$\log_{10}(g(x) - M) = 4\log_{10} h + c.$$

Using the above two equations, we get

$$\begin{aligned} \log_{10}(g^2(x) - M) &= 4(4\log_{10} h + c) + c \\ &= 4^2 \log_{10} h + 4c + c \end{aligned}$$

Similarly, we get

$$\log_{10}(g^3(x) - M) = 4^3 \log_{10} h + 4^2 c + 4c + c$$

Continuing i times, we get

$$\begin{aligned}\log_{10}(g^i(x)-\sqrt{N}) &= \log_{10}(g^i(x)-M)\\ &= 4^i\log_{10}h+4^{i-1}c+4^{i-2}c+\cdots+c\\ &= 4^i\log_{10}h+\frac{4^i-1}{3}c\\ &= 4^i\log_{10}h-(4^i-1)\log_{10}(2M)\end{aligned}$$

since $c=-3\log_{10}(2M)$. Using the formula for P in above equation, we obtain

$$P(N,x,i)=-4^i\log_{10}h+(4^i-1)\log_{10}(2M).$$

Here, $h=x-\sqrt{N}$ and $M=\sqrt{N}$. In $g(x)$ as $x\to\sqrt{N}$, the value of $h\to 0$. This implies $\log_{10}h\to-\infty$. From the above formula for P, it is clear that $P\to\infty$, as $x\to\sqrt{N}$. Further more,

$$\frac{\partial P}{\partial\log_{10}h}=-4^i.$$

That is a given value of N and i, as $x\to\sqrt{N}$, $h\to 0$, and $\log_{10}h\to-\infty$, the value of $P\to\infty$ linearly with a slope of -4^i. This implies that accuracy increases at rate of 4^i, for each iteration i. Therefore our method accounts for *accuracy* with each iteration.

7. Stability

The rate of change in the absolute numerical error with each iteration of an iterative method is the numerical stability of that method.

That is, fluctuations in the absolute value of error with each iteration, gives a measure of stability cite7 of the algorithm. If the fluctuations are random, then the method is unstable. In the case of the fluctuation steadily decreasing, absolute errors tend towards zero and the method is stable.

In this section,we show that our method is stable by considering the variation of P with respect to i. Re-arranging the formula for P, we get

$$P(N,x,i)=4^i\left(-\log_{10}h+\log_{10}(2M)\right)-\log_{10}(2M).$$

This implies,

$$\frac{\partial P}{\partial i}=4^i\ln 4\left(-\log_{10}h+\log_{10}(2M)\right).$$

This implies

$$\frac{\partial P}{\partial i}\propto 4^i$$

or,

$$\log_4\left(\frac{\partial P}{\partial i}\right)\propto i$$

That is for a given value of N and x, as the value of $i \to \infty$ the value of $P \to \infty$ at an exponential rate. The increase in the value of P implies the absolute error $\leq 10^{-P}$. Thus upon each iteration the value of absolute error reduces *exponentially*. Therefore our procedure is *stable*.

8. Graphical Analysis

A sample example is studied in this section. Let us find square root of 1000. The result is approximately 31.62277. Let the tolerance be 10^{-6} and values of initial approximations vary from 10 to 10000. The number of iterations required to get the exact solution within tolerance limit is shown in the figure 1 below.

It is clear from the figure 1 that number of iterations grow logarithmically to initial approximation.

The figure 2 shows the Number of iterations for different methods.

The green line is plot for Secant method.The red line plots Newtons method, while the blue line is for Our method.

Clearly our method takes less number of iterations when compared to secant and newtons method, to find square root of a number to the same desired accuracy.

The figure 3 shows the Number of Iterations required when our algorithm for initial approximation is used.We see that it takes about 3 iterations here. Clearly the time Complexity is $O(1)$.

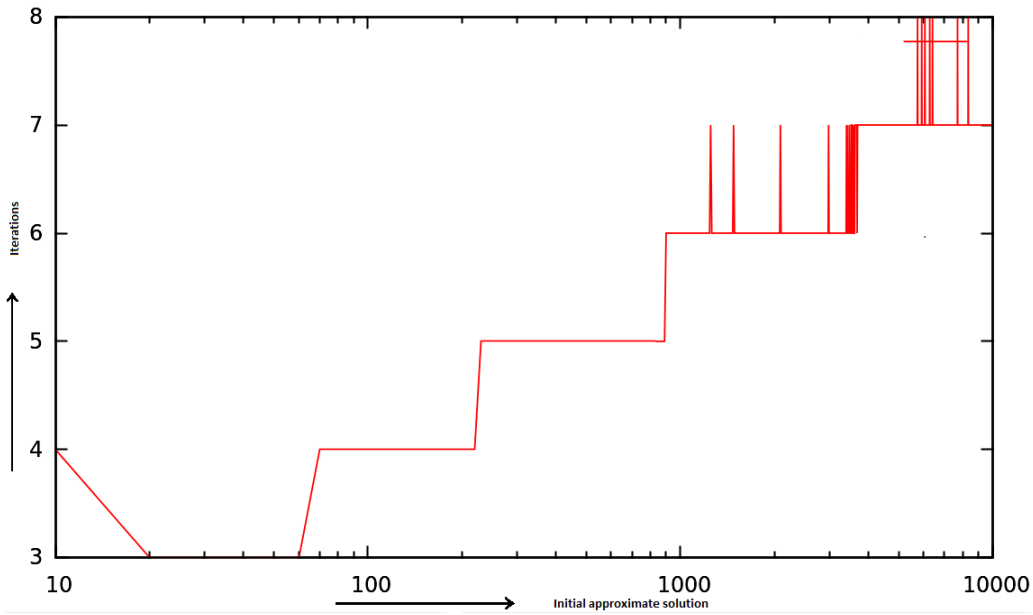

Figure 1: Number of iterations v/s Initial approximate solution

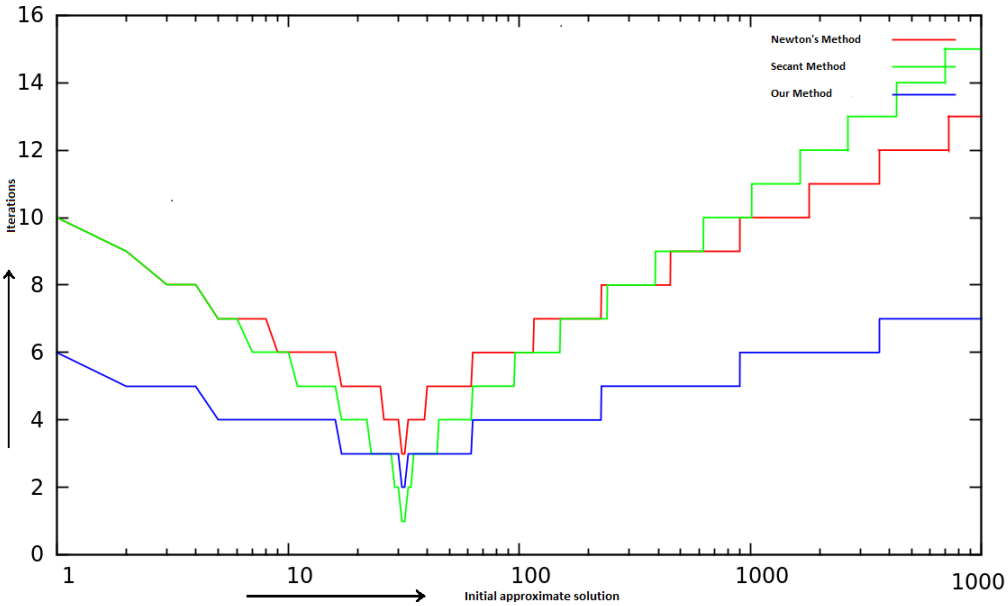

Figure 2: Order of Convergence

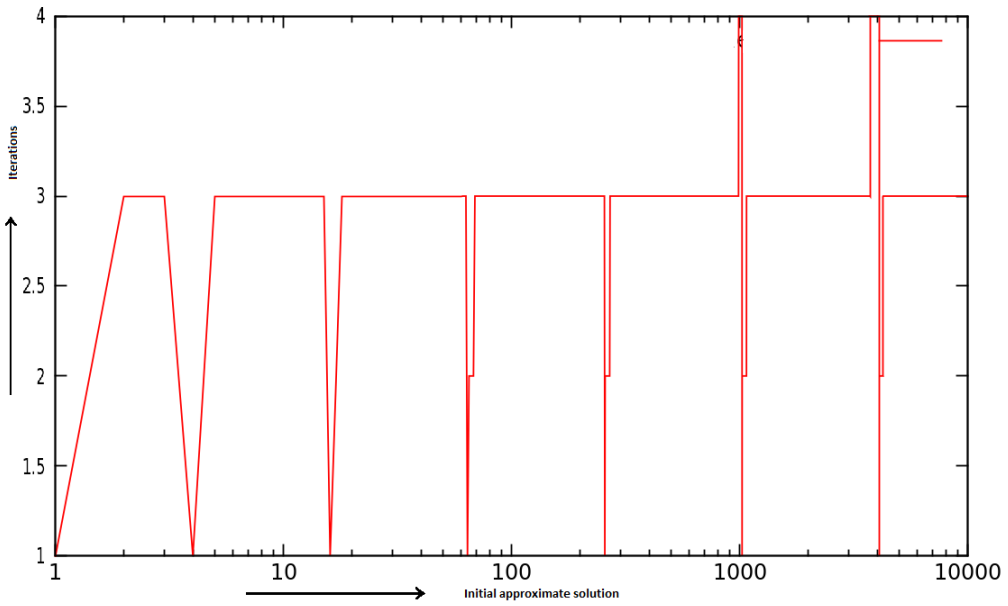

Figure 3: Constant time complexity

9. Higher Order Expansion of $g(x)$

It can be shown that the *Taylor series* expansion $g(x)$ upto 8th power term is

$$g(x) = \sqrt{N} + \frac{h^4}{8N^{3/2}} - \frac{h^5}{4N^2} + \frac{17h^6}{48N^{5/2}} - \frac{5h^7}{16N^3} + \frac{9h^8}{32N^{7/2}} + O(h^9)$$

where $h = x - \sqrt{N}$ This expansion can be used to evaluate $g(x)$ when x is close to $\sqrt{N}$.

10. Results and Discussion

1. When compared to other existing methods,the SA method takes half or lesser number of iterations to get the same amount of accuracy. The order of convergence of our method is twice that of other methods.

2. The formula,
$$g(x) = \frac{Nx}{N + x^2} + \frac{N + x^2}{4x}$$
can be implemented in as
$$g(x) = \frac{N}{Q} + \frac{Q}{4}$$
where $Q = N/x + x$ reducing computational time.

3. The *division-by-4* operation which is required in evaluation of $g(x)$ can be performed by right-shifting twice,which is easy to implement in binary architecture.

4. The formula is simple enough to learn and could be implemented in less complicated steps.

5. We get a simple fraction as a good approximation for the exact solution, when applied to integers.

6. As the initial guess is evaluated as $x_0 = 2^{\log_4 N}$, the number of iterations to be performed remains constant for a given (desired) accuracy.

7. Evaluation of $x_0 = 2^{\log_4 N}$ is simple. We don‘t really have to use logarithm and exponentiation. Instead, we can use the pseudo-code mentioned in section 6.

8. The *division-by-4* and *multiplication-by-2* operations required in evaluation of initial approximate solution can be easily performed as *right-shifting twice* and *left-shifting once* in a binary number system, rendering our method binary architecture friendly.

9. Comparison with other methods

Method	Order of convergence	Accuracy	Comment
Bisection	1	$O(1)$	Linear
Secant	1.618	$O(1.618^i)$	Superlinear
Newton	2	$O(2^i)$	Quadratic
SA	4	$O(4^i)$	Biquadratic

Note: Here i denotes number of iterations. In the above table, the order of dependency of accuracy on i is considered.

References

[1] Bahman Kalantari (2008) Approximation of Square-Roots and Their Visualizations: The Genesis. Polynomial Root-Finding and Polynomiography: pp. 13–38.

[2] Bahman Kalantari (2008) Approximation of Square-Roots Revisited: Basic Family, Continued Fractions and Factorization. Polynomial Root-Finding and Polynomiography: pp. 429–441.

[3] A.E. Bashagha and M.K. Ibrahim, J Circuit Syst Comp 06, 267 (1996). DOI: 10.1142/S0218126696000200 Nonrestoring Radix-2k Square Rooting Algorithm.

[4] Fundamentals of Digital Circuits, Kumar, A.A., isbn:9788120336797, 2009, PHI Learning.

[5] The Taylor Series: An Introduction to the Theory of Functions of a Complex Variable, Dienes, P., lc32012617, 1931, The Clarendon Press.

[6] Theories of Computational Complexity, Calude, C., isbn:9780080867755, Annals of Discrete Mathematics, 2011, Elsevier Science.

[7] Accuracy and Stability of Numerical Algorithms: Second Edition, Higham, N.J., isbn:9780898715217, 2002075848, 2002, Society for Industrial and Applied Mathematics.

[8] Numerical computation, Williams, P.W., 73161341, 1973, Barnes & Noble.

[9] Numerical Methods with Worked Examples, Woodford, C. and Phillips, C., isbn: 9780412721502, gb98066597, 1997, Springer.

[10] Vedic Mathematics: Or, Sixteen Simple Mathematical Formulae from the Vedas (for One-line Answers to All Mathematical Problems), Tirtha, B.K. and Agrawala, V.S. and Sankaracarya, J., 75853415, Hindu Vishvavidyalaya Nepal Rajya Sanskrit series, 1970, Motilal Banarsidass.

[11] Numerical Analysis, Burden, R. and Faires, J., isbn:9781133169338, 2010, Cengage Learning.

[12] Mathematical Time Capsules: Historical Modules for the Mathematics Classroom, Jardine, D. and Shell-Gellasch, A., isbn: 9780883859841, 2011926092, MAA notes, 2011, Mathematical Association of America.

[13] Computational Complexity: A Modern Approach, Arora, S. and Barak, B., isbn:9780521424264, 2009002789, 2009, Cambridge University Press.

[14] Introduction to numerical analysis, Wood, A., International mathematics series, 1999, Addison-Wesley.

[15] Applied Numerical Analysis, isbn:9788131717400, Always learning Pearson, 2007, Pearson Education.

[16] IAR Application Note AN-G002. 1. IAR Application Note G-002. Fast square root in C, netstorage.iar.com/SuppDB/Public/SUPPORT/000419/AN-G-002.p

[17] Amazing Traces of a Babylonian Origin in Greek Mathematics, https://www.worldscibooks.com/etextbook/6338/6338-chap01.pdf

[18] On the efficiency of algorithms of analysis - Project Euclid, projecteuclid.org/euclid.bams/1183552689

[19] A family of root finding methods, folk.uib.no/ssu029/Pdf-file/Hansen77.pdf

[20] Fast Inverse Square Root - LOMONT.org www.lomont.org/Math/Papers/ 2003/InvSqrt.pdf

[21] On the guaranteed convergence of the square-root iteration ... www.miodragpetkovic.com/download/ostrov-jcam-rev.pdf