Computing Square Roots

1 June 2005

James A. Crawford

http://www.am1.us/

1 Introduction

Aside from the basic trigonometric and logarithm functions, the square-root (SQRT) function is one of the first functions that we encounter in school that forces us to deal with irrational numbers. As a student of computation, greater insight into mathematics can be had by knowing more about what is behind the √x function on a $7.99 Casio calculator. This memorandum takes a brief look at methods for calculating the square root of a positive real value.

One of the primary motivations for studying the square root function here is to uncover methods that can be easily implemented in semi-custom VLSI circuitry. For simplicity sake, attention within this paper will be limited to base-10 floating-point numbers even though VLSI implementations are normally binary in nature. Some computational techniques are better suited to the base-2 number system than others, but we defer such considerations to other texts that deal with this subject in depth.

The square root computational methods that will be examined are:

Sums of odd integers
Continued fractions
Long-hand division
Newton's type-1
  Double-step Newton's type-1
Newton's type-2
Secant
Muir's
Ladder (Theon)
Taylor series
  Chebyshev economized series
Rational series
Cordic
Bisection

2 A Touch of History

It is always valuable to have some awareness of history when it comes to long-standing concepts such as the square root.

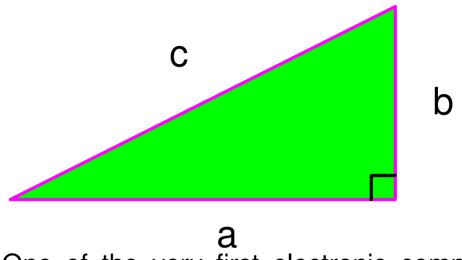
"It[1] would seem that [Archimedes] had some (at present unknown) method of extracting the square root of numbers approximately." "His [2] method for computing square roots was similar to that used by the Babylonians." One of the Babylonian methods[3] used was essentially the same as the method known as "Newton's Method" in more modern times, and we will examine this technique in Section 4.2. A second method attributed to the Babylonians is known as the "ladder method" and we will examine that method in Section 4.6.

In the first century AD, the Greek mathematician Heron of Alexandria also made use[4] of Newton's Method for computing square roots. There can be no question but that the ancient Pythagoreans were keenly aware of square roots since

(1) $$c = \sqrt{a^2 + b^2}$$

in which a,b, and c are the sides of the right-triangle shown in Figure 1.

Figure 1 Basic Right-Triangle Used for Pythagorean Theorem



One of the very first electronic computers in the world named the ENIAC used the "sum of odd integers" method to compute[5] square root as described in Section 3.1. Numbers with a fractional portion were simply pre-multiplied by an even power of 10 in order to circumvent the technique's limitation to purely integer values.

A brief search of the Internet reveals that many societies like India and China also have a rich history of computation regarding the square root function, and interested readers are encouraged to utilize that resource to do their own research on this topic.

[1] W.W Rouse Ball, Short Account of The History of Mathematics, 1908

[2] C. B. Boyer, *A History of Mathematics*, 1968

[3] "Archimedes and the Square Root of 3", http://www.mathpages.com/home/kmath038.htm

[4] http://www.merriampark.com/bigsqrt.htm

[5] http://www4.wittenberg.edu/academics/mathcomp/bjsdir/ENIACSquareRoot.htm

3 Interesting Facts

Before we jump into serious computation of the SQRT function, it is interesting to look at a few examples that surprisingly pertain to square roots.

3.1 Sums of Odd Integers

The sum of the first N odd integers produces a sum equal to N^2. On the surface, this is a very surprising result but nevertheless true. The first few cases of

(2) $$\sum_{ii=1}^{N}(2ii-1)=N^2$$

follow as shown below:

1 + 3 = 4
1 + 3 + 5 = 9
1 + 3 + 5 + 7 = 16
1 + 3 + 5 + 7 + 9 = 25

The underlying mathematics responsible for this result[6] is based upon the simple arithmetic series result in which

(3) $$\sum_{n=1}^{N}(a+nd)=Na+\frac{N}{2}(N+1)d$$

Selecting a= -1 and d=2 provides the result given by (2). Although this result does not directly lend itself to an algorithm for computing non-integer square roots, it is still an intriguing result, and as noted in Section 2, it was the basis for computing square roots in the ENIAC computer.

3.2 Continued Fractions

A second interesting result[7] for the square root of a value **A** is provided by the continued fraction form where **√A = 1 + r** in which

(4) $$r=\cfrac{1}{A+\cfrac{1}{A+\cfrac{1}{A+\cfrac{1}{A+\dots}}}}$$

[6] [1], page 30
[7] [2], page 104

In the case of **A** = 2, this formula converges to within 2 10^{-9} after 10 iterations.

Another simple recursion for the square root can be found by starting with the identify

(5) $$(1+x)^2=A$$

Upon expanding this out we obtain

(6) $$x(2+x)=A-1$$

If we "boldly" assume that we can divide the right-hand side by (2+x) in order to refine our estimate for x, we obtain the recursion

(7) $$x_{i+1}=\frac{A-1}{2+x_i}$$

and upon repeated application of this recursion, we can formulate the continued fraction expression given by

(8) $$x=\cfrac{A-1}{2+\cfrac{A-1}{2+\cfrac{A-1}{2+\cfrac{A-1}{2+\cfrac{A-1}{2+\cfrac{A-1}{2+\dots}}}}}}$$

with the final result given as √**A**= 1 + x.

This ad-hoc recursion is interesting, but the heavy reliance upon division makes it fairly unattractive for square root computation because it converges rather slowly as shown in Figure 2 and Figure 3.

It is also fun to note that if we had started with a slightly different identity of

(9) $$\left(x+\frac{1}{2}\right)^2=A$$

the appropriate recursion would have been

(10) $$x_{i+1}=\frac{A-\frac{1}{4}}{1+x_i}$$

and the final root given by √**A**= x + 0.50.

It is very interesting that this simple continued fraction method works as well as it does, and it seems plausible that other forms are also possible.

Figure 2 Convergence of Continued Fraction Method (8) Versus A and Number of Iterations

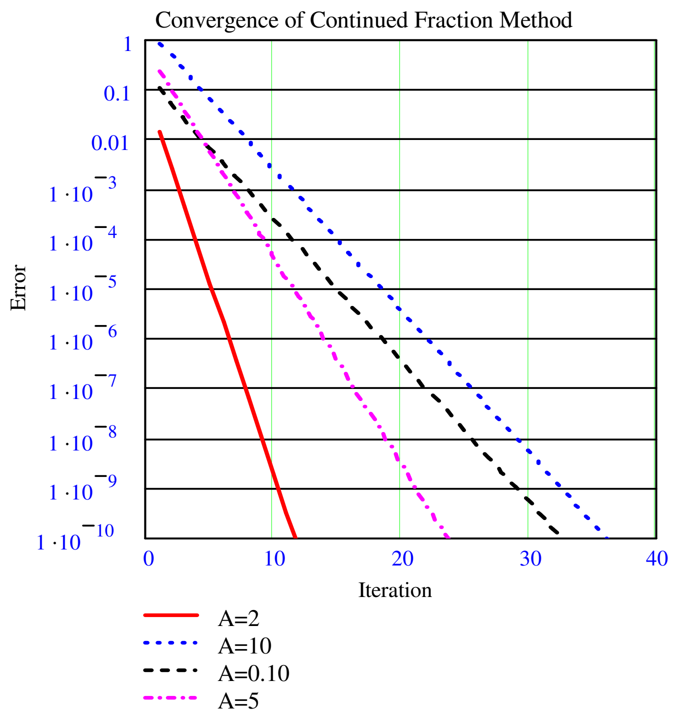
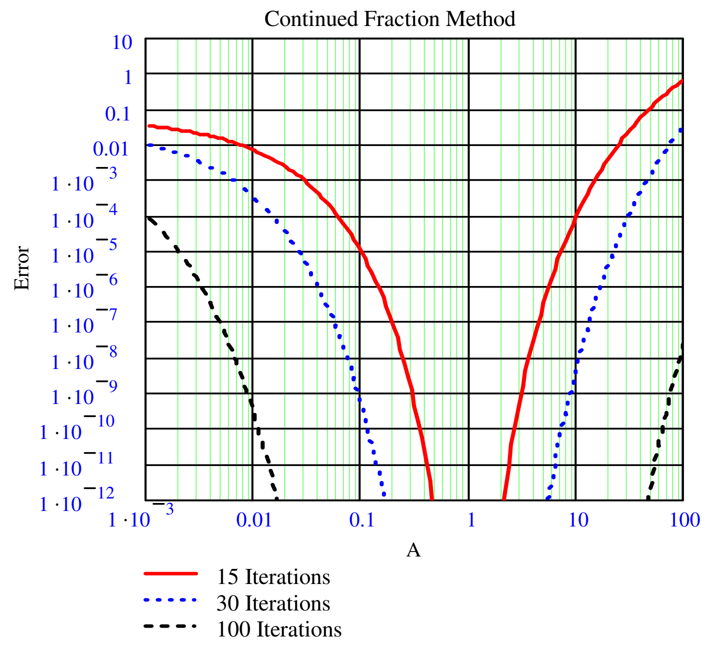


Figure 3 Convergence Range for Continued Fraction Method 8)



4 Computational Methods for Square Root

A wide range of computational methods for SQRT are provided in the sections that follow. Non-rigorous[8] attention is also provided to the convergence question for each method as well.

4.1 Long-Hand Grade School Method

Most of us learned a long-hand method for computing the square root function at one point or another in school, but with little or no explanation for why the method worked. The method shares many visual similarities with long-hand division. First we will present the method, and then justification will be provided for why this method works.

4.1.1 Long-Hand SQRT Method[9]

Assume that we wish to compute the square root of a positive base-10 value represented by $n_1n_2n_3.n_4n_5n_6$ where each n_i represents an integer between 0 and 9. For the case where a= 652.341, n_2= 5 for example.

The long-hand method consists of the following steps:

- Step 1: Group the number in "twos" from the decimal place. (If you have a number with an odd number of digits, the group to the far left will only be a group of 1 digit.)
- Step 2: Start with the first group of two (the group on the left). This group may be a group of only one number if your number has an odd number of digits. Find the greatest square less than or equal to that group of digits and its square root will be your first approximation of the entire square root.
- Step 3: Subtract the current approximation squared and bring down the next group of numbers behind it. This is your next number to work with.
- Step 4: Double the current approximation of the root.
- Step 5: Find the "ones" digit of the doubled number that would result in a number which divides into the number you are currently working with- with the smallest possible remainder. This is the next number in your approximation of the square root.

[8] See Section 3.2 of [6] for additional discussion

[9] http://jwilson.coe.uga.edu/EMT668/EMAT6680.F99/Challen/squareroot/sqrt.html

- Step 6: Multiply the "ones" digit by the doubled number plus the "ones" digit. Subtract this number from the number you are currently working with and bring down the next group of numbers behind it. This is your next group of numbers to work with.
- Step 7: Repeat steps 4 through 6 until you get an approximation with an acceptable number of significant digits.

In the case where a= 652.341, the first several steps of this method are shown in Figure 4 through Figure 7. The iterations are continued until sufficient accuracy is obtained. At any given iteration, the "quotient" obtained when squared will always be < **A** because this method provides a ***truncated*** result for √**A** rather than a rounded one.

Figure 4 Grouping of Integers and First Iteration

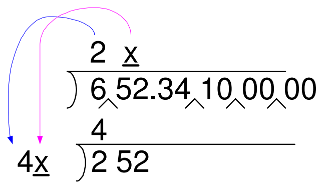


Figure 5 Second Iteration

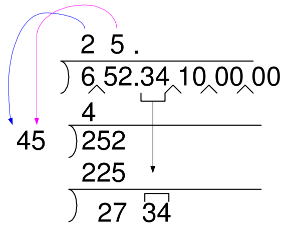


Figure 6 Third Iteration

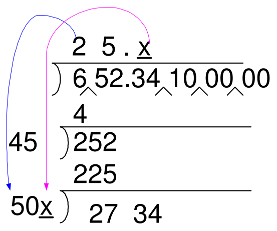
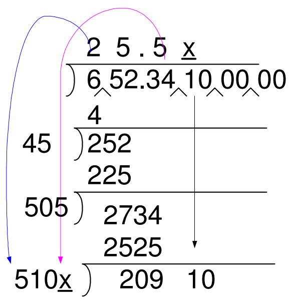


Figure 7 Fourth Iteration



4.1.2 Justification for Long-Hand SQRT Method[10]

The square root of a number **A** is the number M, so that $M^2 = \mathbf{A}$. The square root algorithm is set up so that we take the square root of a number in the form of $(X + R)^2$. The square root of this number is obviously (X + R). X represents the current approximation for the square root, and R represents the remainder of the number left over from the approximation. Our approximation will always be the correct square root of the number **truncated** (not rounded) to the number of digits in our approximation. If we expand our number $(X + R)^2$ it will equal $X^2 + 2RX + R^2$. This gives us the basis for our derivation of the square root algorithm.

Step 1: The square root of a number between 1 and 100 is a number between 1 and 10. Furthermore, the square root of a number between 100 and 10000 is a number between 10 and 100, and so on. Therefore, if a number is divided into groups of two digits each from the decimal point to the left, the number of digits in the integer part of the square root of that number will be equal to the number of groups of figures.

Step 2: Only the first group of two (or one) digits determine the first digit of the square root. If the first two (or one) digits are a perfect square, then there is nothing left over and the process can be repeated on the next two digits of the number. This is usually not the case, which means there is a part of the first two (or one) digits which has not been accounted for and our approximation is not perfect. This leads into Step 3.

Step 3: Take the expanded value of our number: $X^2 + 2RX + R^2$. We subtract off the current approximation, X^2, which results in $2RX + R^2$, the part of the original number which is not accounted for in our approximation. This gives us our next value to work with.

Step 4: Rewriting $2RX + R^2$ gives us R(2X + R). We see that our current approximation, X, must be doubled, resulting in 2X, which are the first digits of the number we will be working with.

Step 5: In order to find the next approximation, we need the R value. This number must divide into the next grouping with the smallest remainder, as shown by R(2X + R). (R obviously divides this number.)

Step 6: Since you've found an approximation based on this number, you must subtract of R(2X + R) so you take into account any remainder from your previous group of numbers.

Step 7: The procedure can be repeated as many times as necessary until either no remainder is found, meaning you have the actual square root, or until an acceptable number of decimal places are estimated.

Given a little practice, long-hand computation

[10] Ibid.

of a square root will be as natural as long-hand division.

4.2 Newton's Method- Type I

We will look at Newton's Method first because it was referenced first in the context of Babylonian mathematics in Section 2. It is unknown what process the Babylonians followed to come up with this powerful computational method.

We will look at two different paths to arrive at the same Newton's Method formula, and will call them (a) the polynomial method and (b) the derivative method.

4.2.1 Polynomial Method

The polynomial method will be most comfortable with those who prefer a non-calculus approach to the formula.

Assume that we wish to compute the SQRT of the quantity **A**, and that we start with an initial guess for the solution of x_0. It turns out that the initial guess can be any positive real number although a better initial guess will lead to faster convergence. Since x_0 is an initial guess, the true square root value can be represented by $\sqrt{\mathbf{A}} = x_0 + \delta$. Using this simple starting point, we can write

(11) $$(x_0+\delta)^2 = A$$

Expanding the binomial leads to

(12) $$x_0^2 + 2x_0\delta + \delta^2 = A$$

We now make the "bold" assumption that $|2x_0\delta| >> \delta^2$ and we can therefore drop this second-order term and approximate δ as

(13) $$\delta \approx \frac{a - x_0^2}{2x_0}$$

It is a simple matter to use this estimate for δ to form a new estimate for $\sqrt{\mathbf{A}}$ as $x_1 = x_0 + \delta$, and this process can be repeated as many times as desired. The recursive formula for x is given by

(14) $$x_{i+1} = \frac{1}{2}\left[x_i + \frac{A}{x_i}\right]$$

This result is known as "Newton's Method" for computing SQRT. We note that it is very simple to use this same approach for the nth root of the value **A** by simply starting with a nth order polynomial in (11) and following through the same steps leading to the recursive formula

(15) $$x_{i+1} = \frac{(n-1)x_i^n + A}{nx_i^{n-1}}$$

4.2.2 Derivative Method

The derivative method makes direct use of the Newton-Raphson formula[11] to compute the recursive square root formula given by (14). For a function F(x), the recursive formula for the solution of F(x)= 0 takes the form

(16) $$x_{i+1} = x_i - \frac{F(x_i)}{F'(x_i)}$$

in which the prime indicates differentiation. For $F(x)= x^2 - \mathbf{A}$, the derivative is simply given by $F'(x)= 2x$ and upon substitution into (16), we obtain the desired result identical with (14).

4.2.3 Convergence Properties of Newton's Method

A non-rigorous proof that (14) always converges to the correct answer can be obtained by computing the ratio between successive estimates for x_i as

(17) $$\frac{x_{i+1}}{x_i} = \frac{1}{2}\left(1 + \frac{A}{x_i^2}\right) = \gamma_i$$

Inspection of this ratio shows that γ is a strictly positive quantity and

(18) $$\gamma_i = \begin{cases} >1 : x_i^2 < A \\ <1 : x_i^2 > A \end{cases}$$

which guarantees convergence. The normalized error versus iteration index and normalized initial estimate is shown in Figure 8 for several different cases. Two distinct convergence rates are self-evident; the first corresponds to a geometric error reduction rate for which the error is reduced by approximately a factor of 2 for each iteration, and a second error reduction rate corresponding to a quadratic convergence rate once

[11] [3], pages 657-9

the root estimate is reasonably close to the true value. Aside from possibly the first iteration step, the error reduces monotonically for each iteration.

4.2.4 Double-Steps with Newton Type I

Most of the recursion formulas presented in this paper can be algebraically re-organized to do two or more iterations in just one step. This is particularly attractive to do if multiplications are relatively inexpensive to perform whereas division operations are not.

For example, if (14) is applied two times in a row, the recurrence relationship that results is

$$\textbf{(19)}\ x_{i+1} = \frac{1}{4}\frac{x_i^4 + 2Ax_i^2 + A^2}{x_i\left(x_i^2 + A\right)}$$

More multiplications are involved even if synthetic division methods are used, but only one division is required per iteration and the convergence rate per iteration is effectively doubled.

If the original recursion (14) is applied three times in a row, the recurrence relationship that results is

$$\textbf{(20)}\ x_{i+1} = \frac{\frac{1}{16}\left(x_i^4 + 6ax_i^2 + a^2\right)^2 + ax_i^2\left(x_i^2 + a\right)^2}{\frac{1}{2}\left(x_i^5 + 6ax_i^3 + a^2x_i\right)\left(x_i^2 + a\right)}$$

Additional re-organization can make this result more attractive for actual hardware implementation. This construction technique can be used to create rational approximations for the square root that involve only one division step as shown.

4.3 Newton's Method- Type II

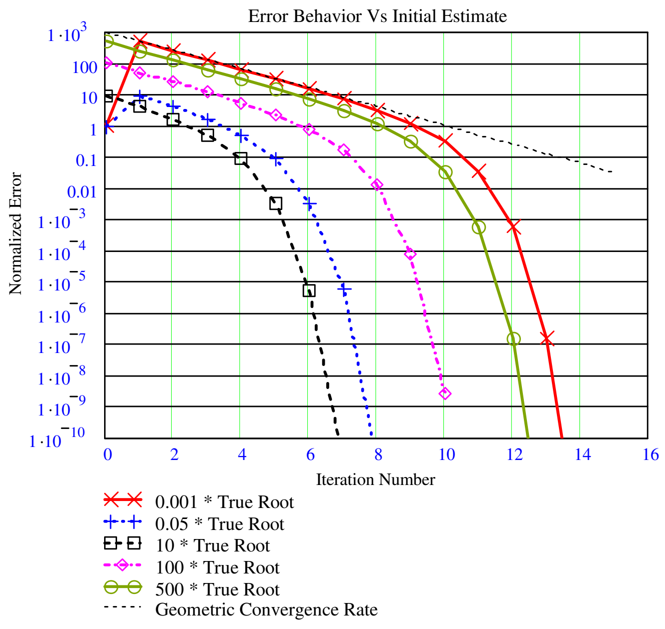
Newton's method can be used with an alternative problem formulation to arrive at a recursive formula similar to (14) that does not require division. Although this aspect is very attractive, the convergence range of this method is limited.

In this approach, the function that is to be solved is given by

$$\textbf{(21)}\ F(u) = \frac{1}{u^2} - A$$

Once the value of u is found in (21), the square root can be computed as x = **A** u. This method is attractive because it does not require any explicit division operations, but the range of convergence is limited as discussed in Section 4.3.2.

Figure 8 Normalized Error Versus Iteration Number and Initial Normalized Initial Guess



4.3.1 Derivation of the Type II Method

Beginning with (21) and making use of the first term in its Taylor series expansion, we can write

$$\textbf{(22)}\ F(u+\delta) = \frac{1}{u^2} - A + \left(\frac{2}{u^3}\right)\delta = 0$$

Solving this equation for δ, we find that

$$\textbf{(23)}\ \delta = \frac{u^3}{2}\left(\frac{1}{u^2} - A\right) = \frac{u}{2} - \frac{A\,u^3}{2}$$

from which a refined value for u can be written as

$$\textbf{(24)}\ u' = u + \frac{u}{2} - \frac{A\,u^3}{2} = \frac{u}{2}\left(3 - A\,u^2\right)$$

This formula can be applied recursively until the desired degree of precision is achieved.

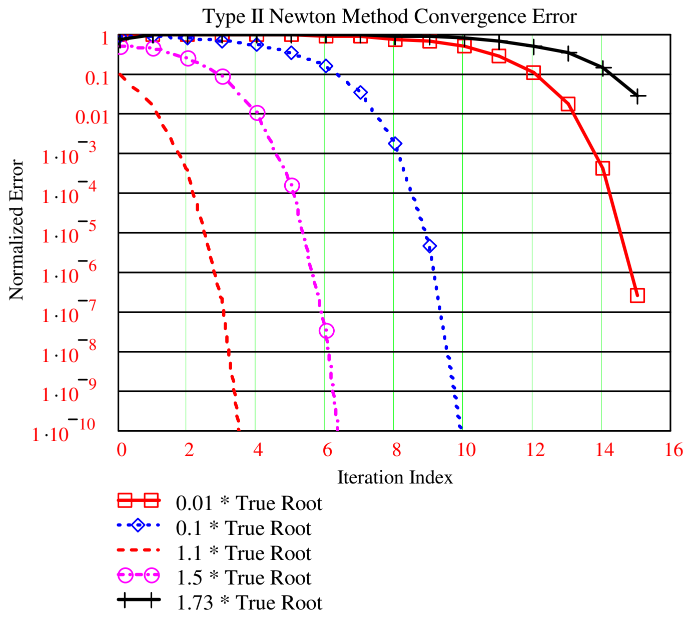
4.3.2 Type II Method Convergence

The initial solution estimate for u must be constrained as mentioned earlier in order to guarantee that this method properly converges. If we look at the ratio of u_{i+1} to u_i by using (24), we easily find that this ratio is given by

(25) $$\gamma_i = \frac{u_{i+1}}{u_i} = \frac{1}{2}\left(3 - A\,u^2\right)$$

It turns out that sufficient conditions to insure proper convergence are that (a) $u_0 > 0$ and (b) $u_0 < \sqrt{(3/\mathbf{A})}$ in order to insure that $\gamma_i > 0$ is always true. Choosing u_0 extremely small is one way to insure that convergence is achieved, but this also leads to a large number of iterations being required in order to obtain good precision. A better alternative is to adopt a crude initial

Figure 9 Convergence Error Behavior of Type II Method Versus Initial Estimate and Iteration Index



estimator for u_0 such that u_0 strictly less than $\sqrt{(3/\mathbf{A})}$ is achieved but also insures that u_0 is not overly small compared to the end-solution. In the case of using 10 iterations, a uniformly good initial estimate for u_0 given by 1.64/**A** delivers 15-place decimal accuracy for $1 \le \mathbf{A} \le 100$. Different initial estimates can be easily determined for other ranges of **A** and the number of iterations employed.

Figure 9 also shows different rates of convergence depending upon the quality of the initial estimate used. The initial convergence rate is not geometric as for the Type I Newton's Method, but it does become quadratic once the estimation error becomes sufficiently small.

4.4 Secant Method

The Secant Method[12] is closely related to the Newton-Raphson method that was used in Section 4.2.2. Rather than employ the exact derivative of F(x), the derivative is approximated at each iteration step as

(26) $$F'(x_i) \approx \frac{F(x_i) - F(x_{i-1})}{x_i - x_{i-1}}$$

In the case where $F(x) = x^2 - \mathbf{A}$, this leads to the recursive formula

(27) $$x_{i+1} = \frac{A + x_i x_{i-1}}{x_i + x_{i-1}}$$

Since this recursive formula involves a multiplication as well as a division, its utility compared to the exact formula given by (14) is questionable.

4.4.1 Convergence Behavior of Secant Method

The convergence behavior of the Secant Method is very similar to that of the Type I Newton's Method because they start from the same conceptual perspective. The algorithm's convergence properties versus iteration number and initial estimate are shown in Figure 10. Since two initial values for x are required in the Secant Method, it can be assumed that both x_0 and x_1 are set equal to the same initial estimate.

The ratio of x_{i+1} to x_i is easily found to be

(28) $$\gamma = \frac{A + x_i x_{i-1}}{x_i^2 + x_i x_{i-1}} = \frac{1 + \dfrac{A}{x_i x_{i-1}}}{1 + \dfrac{x_i^2}{x_i x_{i-1}}}$$

If we now assume that the error at iteration n is given by

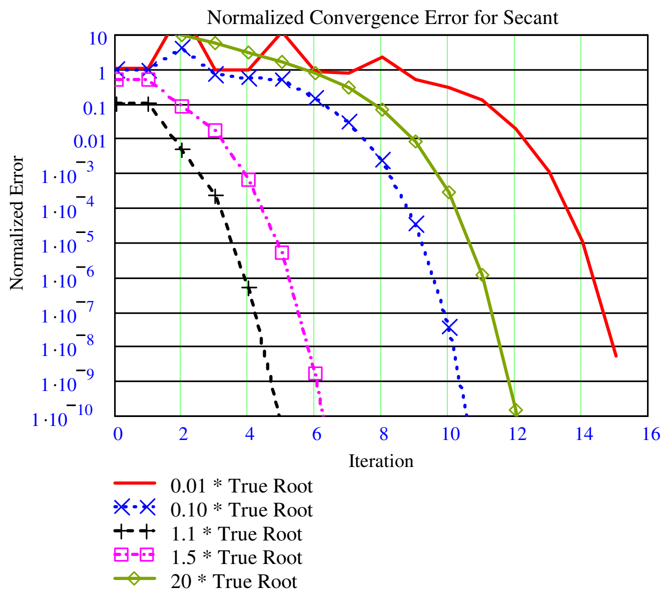
(29) $$\Delta_n = A - x_n^2$$

and substitute back into (28), we can show that when $x_i x_{i-1} \ll \mathbf{A}$ that $\gamma \to \mathbf{A}/(\mathbf{A}-\Delta)$ which is a potentially large but always positive quantity (since $\Delta < \mathbf{A}$), and when $x_i x_{i-1} \gg \mathbf{A}$ that $\gamma \to 1/[\ 1 + (\mathbf{A}-\Delta)/(x_i x_{i-1})\ ]$ which tends

12 http://sepwww.stanford.edu/oldsep/sergey/sepsergey/specfac/paper_html/node3.html

toward the value of ½ just as for the Type I Newton's Method. Consequently, the initial convergence can be more erratic for the Secant Method than for the Type I Newton's Method, but convergence eventually occurs regardless.

Figure 10 Convergence Behavior of Secant Method



4.5 *Muir's Method*[13]

This method is a rather obscure method that is based upon choosing a special function for F(x). The recursion formula is attractive in that it does not require any multiplications to be performed, but a division step is required for each iteration. The recursion is given by

(30) $$x_{i+1} = \frac{x_i + A}{x_i + 1}$$

The special function used for Muir's Method is

(31) $$F\left(x\right) = \left|x + \sqrt{A}\right|^{\frac{\sqrt{A}-1}{2\sqrt{A}}} \left|x - \sqrt{A}\right|^{\frac{\sqrt{A}+1}{2\sqrt{A}}}$$

4.5.1 Convergence Behavior of Muir's Method

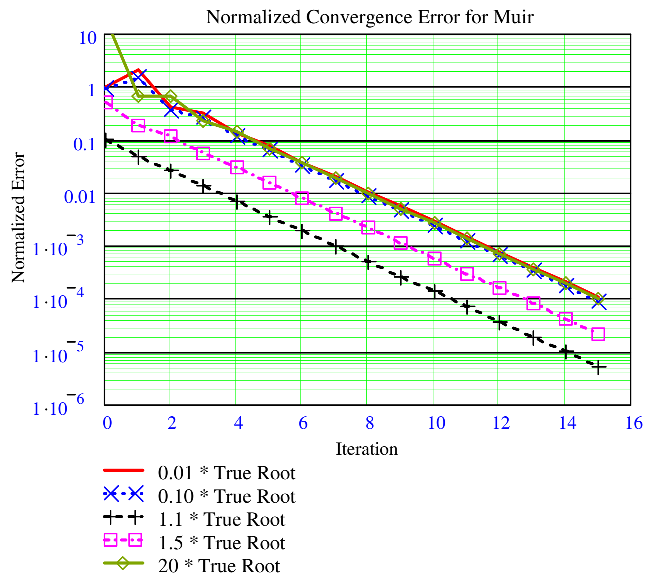
The convergence rate of Muir's Method is substantially slower than the previous methods that have been examined thus far, and it is primarily due to the denominator construction shown in (30). The convergence is shown for several initial estimate cases in Figure 11.

It is insightful to examine the ratio x_{i+1} to x_i as we have done with the previous methods resulting in

(32) $$\gamma = \frac{1 + \frac{A}{x_i}}{1 + x_i}$$

So long as $x_0 > 0$ is used, this ratio remains positive as desired. For **A** and x_i both >> 1, $\gamma \rightarrow$ **A**/x_i^2 which is properly larger than unity when the root estimate is too small, and smaller than unity when the estimate is too large. In situations where x_i << 1, $\gamma \rightarrow 1 +$ **A**/x_i which can be quite large (but always positive). The constant 1 factors in (32) lead to a different error behavior versus iteration index than seen thus far as shown in Figure 12.

Figure 11 Convergence Behavior for Muir's Method Versus Iteration Index and Initial Estimate



4.6 *Ladder Method (Theon's)*

One of the most attractive features of this method is that it only depends upon the addition and subtraction of integers until the last step at which time a final ratio is computed.

"It[14] seems plausible that the Greeks may have proceeded as follows: The square root of **A** can be broken into an integer part and a remainder, i.e.,

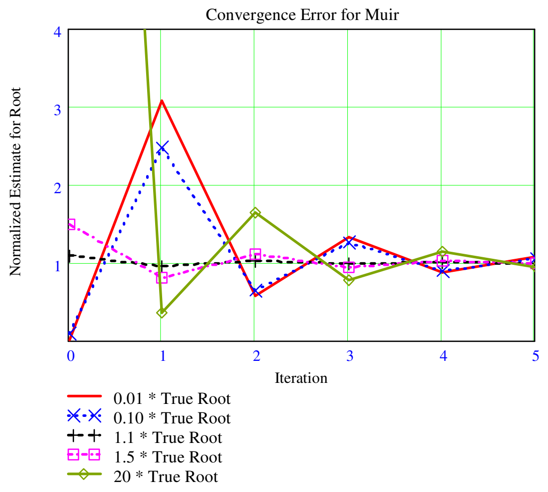
[13] http://sepwww.stanford.edu/oldsep/sergey/sepsergey/specfac/paper_html/node3.html

[14] "Archimedes and the Square Root of 3", http://www.mathpages.com/home/kmath038.htm Theon of Smyrna, circa 140 AD

sqrt(**A**) = N + r where N is the largest integer such that N^2 is less than **A**. The value of r can be approximated to any desired degree of precision using only integer additions and multiplications based on the recurrence formula

(33) $s_i = 2Ns_{i-1} + \left(A - N^2\right)s_{i-2}$

Figure 12 Convergence Error for Muir's Method Showing Oscillatory Behavior



It's easy to see that the value of (**A**-N^2)(s_i / s_{i+1}) approaches r as n goes to infinity. This is a form of the so-called "ladder arithmetic", of which some examples from ancient Babylonia have survived."

A plausible explanation for this recursion formula can be obtained by dividing both sides of (33) by s_{i-1} which produces

(34) $\frac{s_i}{s_{i-1}} = 2N + \left(A - N^2\right)\frac{s_{i-2}}{s_{i-1}}$

If we identify the quantity

(35) $r_{i-1} = \left(A - N^2\right)\frac{s_{i-2}}{s_{i-1}}$

then we can re-write (34) as

(36) $\frac{A - N^2}{r_i} = 2N + r_{i-1}$

and upon removing the denominator r_i, this becomes

(37) $r_{i-1}r_i + 2Nr_i + N^2 = A$

In the limit as i → ∞, $r_i = r_{i-1}$ = r and (37) takes the form of a perfect square as $(N+r)^2$ = **A**.

Although N is to be chosen as the largest integer such that $N^2 \le$ **A**, it turns out that the recursion formula converges for other values of N albeit less quickly, but it nonetheless converges. This is due to the definition of r being proportional to the error (**A**-N^2) which can be positive or negative.

As an example, to find sqrt(3) we have **A**=3 and N=1, so the recurrence formula is simply $s_i = 2s_{i-1} + 2s_{i-2}$ with $s_0 = 0$ and $s_1 = 1$, and the subsequent values in the sequence are

2, 6, 16, 44, 120, 328, 896, 2448, 6688, 18272, 49920,...

Using the last two results in the series, r= (3-1)(18272 / 49920)= 0.732051282 making the estimate for the root equal to 1 + r = 1.732051282 which is accurate to within 0.5 10^{-6}.

Reference [7] provides a slightly different recursion formula for the ladder that supports the finding of any root, and it is given by

(38)
$$x_i = x_{i-1} + y_{i-1}$$
$$y_i = x_i + \left(A - 1\right)x_{i-1}$$

with the starting "rungs" being $x_0 = y_0 = 1$. In this formulation, it is further assumed that **A** > 1. This recursion results in the following sets of value for the **A**=3 case:

Table 1 Theon's Ladder Construction from [7]

x_i	y_i	y_i/x_i
1	1	1
2	4	2
6	10	1.66666666666666
16	28	1.75
44	76	1.72727272727272
120	208	1.73333333333333
328	568	1.7317007317…
896	1552	1.732142857…
2448	4240	1.732026144…
6688	11584	1.732057416…
18272	31648	1.732049037…
	Exact	1.732050808…

4.6.1 Convergence of Ladder Method

If we initially assume that the limit $\lim_{n\to\infty} y_n/x_n$ exists as done in [7], the ratio y_n/x_n can be written as

(39) $$\frac{y_i}{x_i} = \frac{x_i + (A-1)x_{i-1}}{x_{i-1} + y_{i-1}}$$

By further dividing the numerator and denominator of the right-hand side by x_{i-1} we obtain

(40) $$\frac{y_i}{x_i} = \frac{\frac{x_i}{x_{i-1}} + (A-1)}{1 + \frac{y_{i-1}}{x_{i-1}}}$$

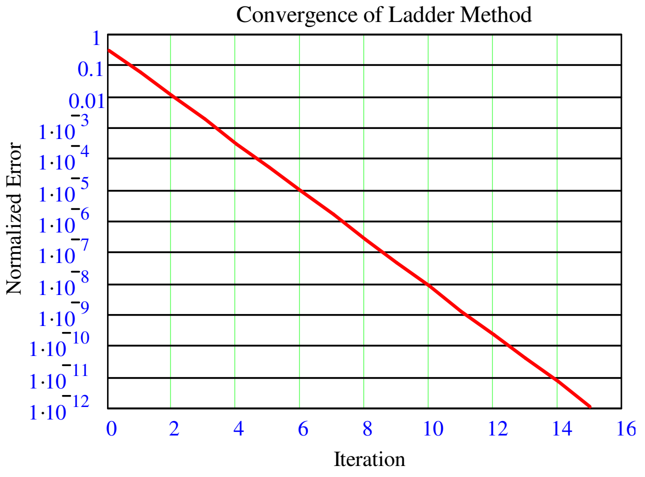
From (38) we can express x_i in (40) in terms of x_{i-1} and y_{i-1} which leads to

(41) $$\frac{y_i}{x_i} = \frac{A + \frac{y_{i-1}}{x_{i-1}}}{1 + \frac{y_{i-1}}{x_{i-1}}} = \frac{A+r}{1+r}$$

where r= $\lim_{i \to \infty} y_i / x_i$. This is equivalent in the limit to writing r = (**A**+r) / (1 + **A**) which reduces to r^2 = **A**.

Convergence of the recursion (38) for √2 and √7 are shown in Figure 13 and Figure 14 respectively. Since x_0= y_0= 1 in both cases, convergence for √7 is less rapid as shown.

Figure 13 Convergence of (38) for √2



4.7 Series Expansions

It is only natural that we examine what series expansions are available for the square root computation. The expansions that we will look at in this section encompass the Taylor, Chebyshev, and rational varieties.

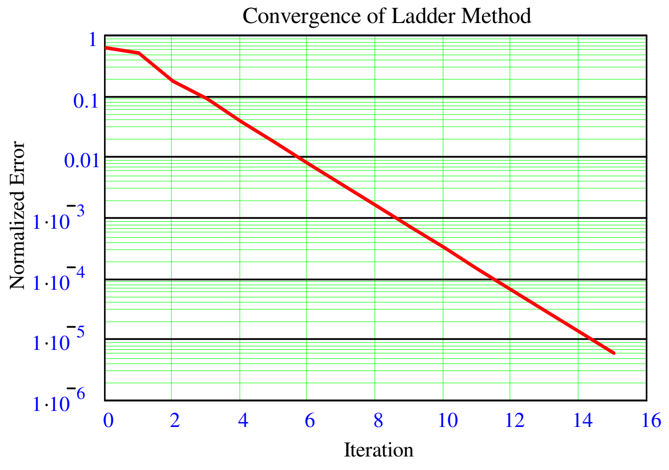
4.7.1 Taylor Series

The Taylor series expansion for √(1+**a**) is given[15] by

(42) $$\sqrt{1+a} = 1 + \frac{a}{2} - \frac{a^2}{8} + \frac{a^3}{16} - \frac{5a^4}{128} + \frac{7a^5}{256} - \frac{105a^6}{5120} + \frac{165a^7}{10240} + \ldots$$

This type of formula must be used with a limited range for parameter **a** that is based upon the accuracy desired and the number of terms used in the series. Normally, |**a**| < 0.50 is recommended for reasonable performance.

Figure 14 Convergence of (38) for √7



4.7.2 Chebyshev Series

The Taylor series of the previous section can be made much more computationally efficient by "economizing" the series expansion [16] by using Chebyshev-based polynomials. The recursive relationship for the polynomials is given by $T_0(x)$= 1, $T_1(x)$= x, and $T_{n+1}(x)$= $2T_n(x)$ – $T_{n-1}(x)$. The first few such Chebyshev polynomials are

[15] [1] page 40
[16] [8] Section 10.4

$$T_0(x)=1$$
$$T_1(x)=x$$
(43) $$T_2(x)=2x^2-1$$
$$T_3(x)=4x^3-3x$$
$$T_4(x)=8x^4-8x^2+1$$

These Chebyshev polynomials can be rearranged to express different powers of x as given by (44).

The economization is done by starting with a Taylor series expansion that achieves the desired accuracy over a specified parameter range of [-α, +α]. Each power of x in the Taylor series is then replaced by the corresponding Chebyshev polynomial from (44), like $T_n(x)$ terms collected, and then the $T_n(x)$ replaced by the original polynomials in x/α given by (43). The key to the economization is that only m Chebyshev polynomials are used, m being such that the desired accuracy if obtained. The resultant series expansion can be considerably shorter than the original Taylor series expansion while providing nearly the same or better precision over an extended range of x.

$$1=T_0$$
$$x=T_1$$
$$x^2=\frac{1}{2}(T_0+T_2)$$
$$x^3=\frac{1}{4}(3T_1+T_3)$$
(44) $$x^4=\frac{1}{8}(3T_0+4T_2+T_4)$$
$$x^5=\frac{1}{16}(10T_1+5T_3+T_5)$$
$$x^6=\frac{1}{32}(10T_0+15T_2+6T_4+T_6)$$
$$x^7=\frac{1}{64}(35T_1+21T_3+7T_5+T_7)$$
$$etc.$$

In general, if we have a Taylor series of interest with its coefficients given by b_k, the coefficients of the Chebyshev-based series are given by

(45) $$C_n=\varepsilon_n\sum_{p=0}^{\infty}\binom{2p+n}{p}\frac{b_{2p+n}}{2^{2p+n}}$$

where ε_n= 1 for n=0 but otherwise 2, and

(46) $$f(x)=\sum_{n=0}^{\infty}C_nT_n(x)$$

As an example, assume that we start with (42), but in the end, we only wish to retain up to 4th order powers of x. We also stipulate that the range of **A** of greatest interest is 0.5 < **A** < 1.5. Upon replacing the powers of x in (42) with the corresponding Chebyshev polynomials series from (44), we obtain

(47) $$f(x)=0.983343T_0(x)+0.256475T_1(x)-0.017023T_2(x)+ \\ 0.002270T_3(x)-0.000379T_4(x)$$

It only remains to substitute in the appropriate polynomials from (43) for each respective $T_n(x)$ and collect like powers of x up to the fourth power.

Several sample results are provided below. In this formulation, the Chebyshev economized series has the form

(48) $$f(A)=\sum_{ii=0}^{Nuse}Cx_{ii}2^{ii}(A-1)^{ii}$$

and the Cx coefficients are provided in Table 2. The error performance for the 4th order and 5th order series are shown in Figure 15 and Figure 16 respectively. The equal-ripple behavior that is characteristic of the Chebyshev fit is clearly visible across the range 0.50 ≤ **A** ≤ 1.50.

Table 2 Economized Cx Coefficients for Equation (48)

Power of x	N_{Use}= 4	N_{User}= 5	N_{User}= 6
0	0.999986389	0.999986389	1.000000632
1	0.249664888	0.250019730	0.250019730
2	-0.031012944	-0.031012944	-0.031269310
3	0.009080022	0.007660657	0.007660657
4	-0.003032666	-0.003032666	-0.002349022
5	-	0.001135492	0.001135492
6	-	-	-0.000455762

Figure 15 Chebyshev Series Economized for 0.50 < x < 1.5, 4th Order Polynomials Used for Taylor Series and Chebyshev Series

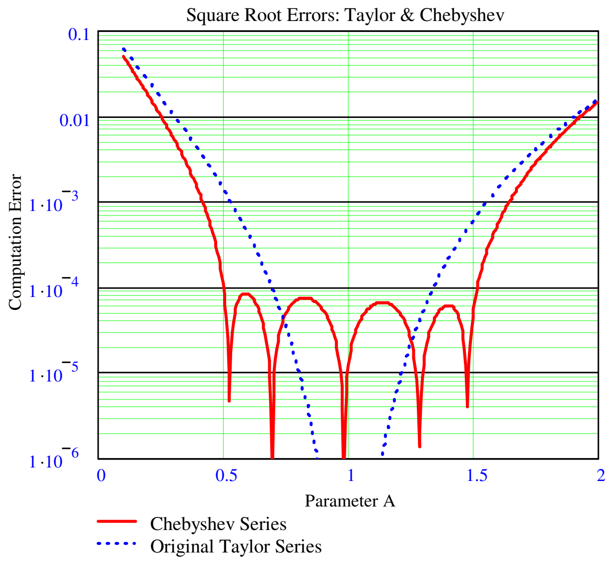
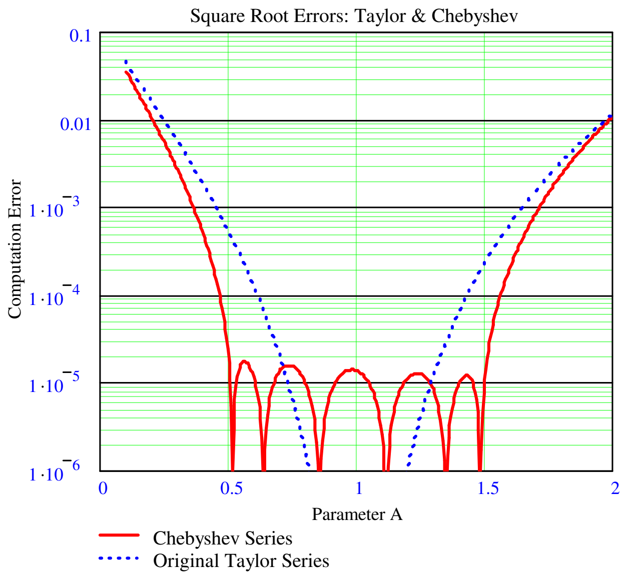


Figure 16 Chebyshev Series Economized for 0.50 < x < 1.5, 5th Order Polynomials Used for Taylor Series and Chebyshev Series



4.7.3 Rational Series

Rational series by definition always involve at least one division operation. An optimally designed rational series can exhibit exceptional precision while incurring limited complexity. Many of the approximation formulas provided in the classic text [9] are based upon rational series for example.

The Ladder Method of Section 4.6 leads to a ratio of two terms and can therefore be considered to be a rational series method. Similarly, the two-step and three-step accelerated Newton's Methods of Section 4.2.4 also lead to a ratio of polynomials and can therefore be considered to be a rational series method.

4.8 Cordic Method

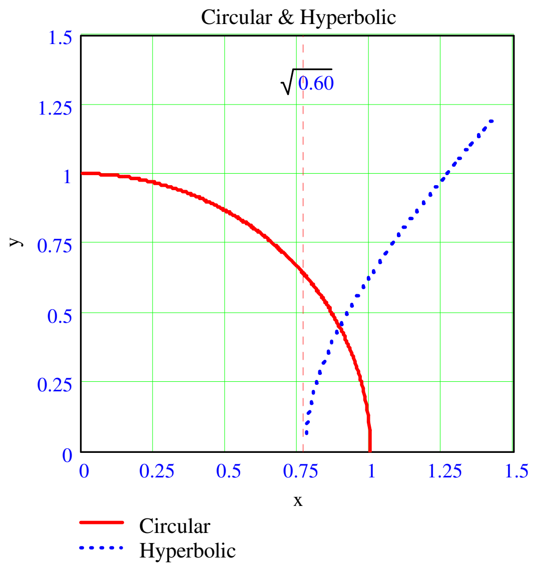
The COordinate Rotation DIgital Computer (CORDIC) method is widely used to compute trigonometric as well as other transcendental functions in its circular form as developed at length in the classic references [10,11]. A hyperbolic variant of this technique can be used to compute other transcendental quantities as well as the square root. An example of the circular and hyperbolic cases is shown in Figure 17 in which the hyperbolic equation is given by $x^2-y^2= 0.60$.

A transformation must be performed before using the hyperbolic CORDIC to compute the square root. If the square root of **A** is sought, we first must compute

(49)
$$x = A + \frac{1}{4}$$
$$y = A - \frac{1}{4}$$

Note that in this form, the underlying square root computation is clear since

Figure 17 Example Circular & Hyperbolic Curves



(50) $x_0^2 - y_0^2 = \left(A + \frac{1}{4}\right)^2 - \left(A - \frac{1}{4}\right)^2 = A$

Pseudo-rotations are done using these starting values of x and y with

(51)
$$x_{i+1} = x_i + y_i d_i 2^{-i}$$
$$y_{i+1} = y_i + x_i d_i 2^{-i}$$
$$z_{i+1} = z_i - d_i \tanh^{-1}\left(2^{-i}\right)$$
$$d_i = 1\; if \;\; y_i < 0, otherwise -1$$

At the conclusion of n iterations, we have

(52)
$$x_n = R_n \sqrt{x_0^2 - y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tanh^{-1}\left(\frac{y_0}{x_0}\right)$$
$$R_n = \prod_{ii=1}^{n} \sqrt{1 - 2^{-2ii}}$$

The convergence range of the CORDIC is limited as discussed in the next section. Unlike the circular CORDIC methods, every (3k+1)th iteration must be re-done (using the same 2^{-i} value as the previous iteration) in order to guarantee convergence.

4.8.1 Convergence of the CORDIC SQRT Function

The convergence region of the CORDIC SQRT function is approximately 0.03 < **A** < e^1. This can be clearly seen from the approximation error plots provided below in Figure 18 through Figure 20. Aside from its fairly large convergence region and good error performance, the CORDIC method only requires simple addition, subtraction, and shifts-by-1 (binary numbers) which makes it very attractive for hardware implementation. This fact makes the CORDIC family of algorithms very appealing to VLSI designers.

Figure 18 CORDIC Error Versus A for 10 Iterations

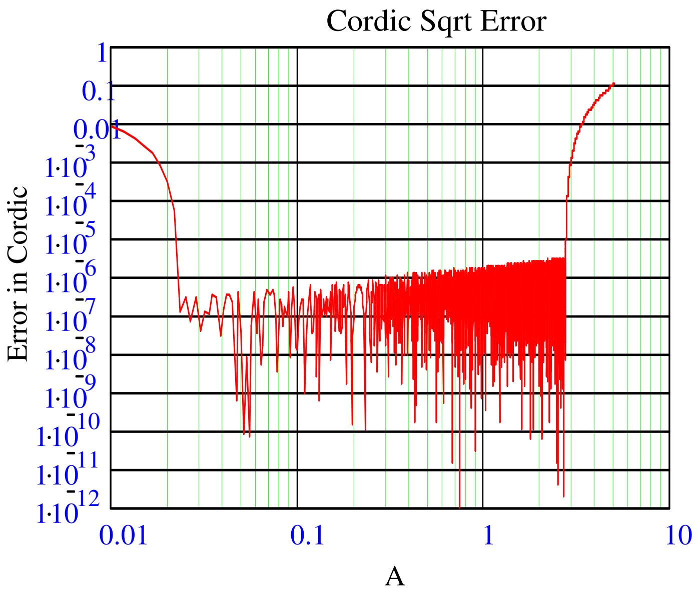


Figure 19 CORDIC Error Versus A for 15 Iterations

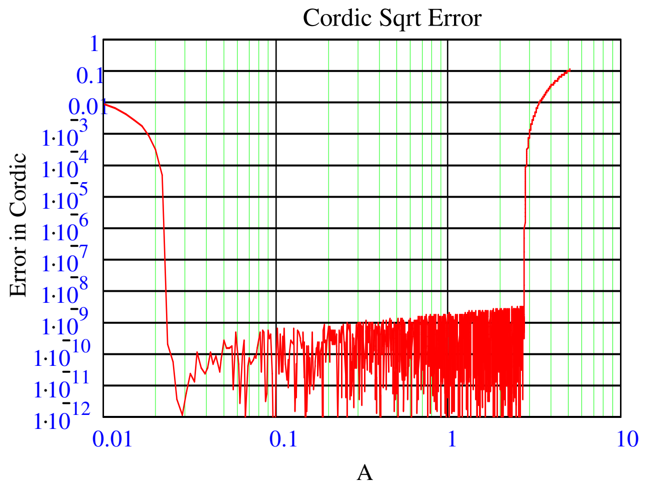
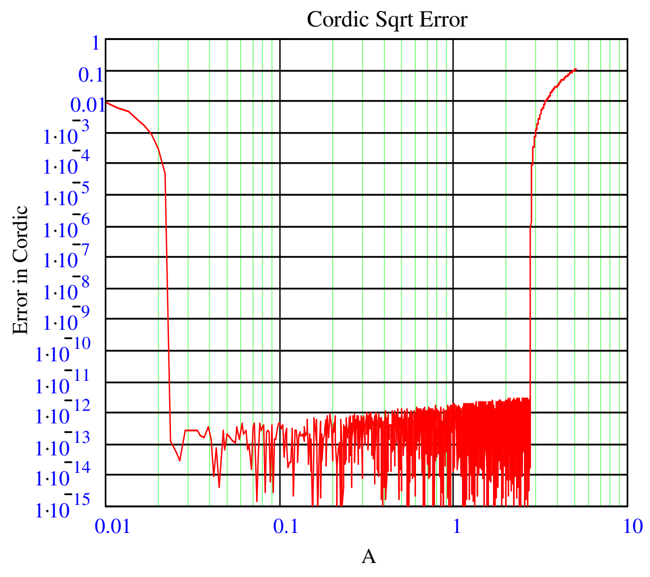


Figure 20 CORDIC Error Versus A for 20 Iterations



4.9 *Bisection Method*

The bisection method is a well-known brute force root-solving method for real quantities that works very well when the (single) zero of a function is known to lie between two values of x, $x_L < x < x_H$. Normally, the method involves the following steps:

1. Select the minimum and maximum values of permissible x, and do whatever pre-processing is required in order to insure that the (single) root to the equation $x^2 - \mathbf{A} = 0$ lies within that range.
2. Compute $F_L = x_L^2 - \mathbf{A}$ and $F_H = x_H^2 - \mathbf{A}$.
3. Compute $x_M = 0.50(x_L + x_H)$ and $F_M = x_M^2 - \mathbf{A}$
4. If the sign of F_L and F_M differ, set $x_H = x_M$ and $F_H = F_M$; otherwise, set $x_L = x_M$ and $F_L = F_M$.
5. If $|F_M|$ > allowable error, go to step 2 and repeat.

This method essentially halves the range of uncertainty for the x solution with every iteration thereby making the convergence rate geometric in nature. So long as the true square root value is initially located between x_L and x_H, this method is guaranteed to converge.

5 Summary of Results

A tabular summary of the different methods considered in this paper is provided below for easy comparison. The grading is very subjective. Normally, division operations are very undesirable and unless a given method's accuracy or convergence rate was exceptional, the use of division was considered very undesirable.

Table 3 Summary of Square Root Methods

Method	Range	Convergence Rate	Divisions per Iteration	Grade
Sum of Integers	Unlimited	Slow	None	D
Cont. Frac.	Limited	Slow	1	D
Long-Hand	Unlimited	Medium	1+	C
Newton Type-1	Unlimited	Geometric then quadratic	1	A
Newton Type-1 Double	Unlimited	Doubly geometric then quadratic	1	A
Newton Type-2	Limited	Geometric then quadratic	0	A
Secant	Unlimited	Geometric then quadratic	1	B
Muir	Unlimited	Geometric	1	C
Ladder	Unlimited	Geometric	0	B
Taylor	Limited		0	C+
Cheby	Limited		0	B+
Cordic	Limited	Geometric	0	A
Bisect	Limited	Geometric	0	B

6 References

1. Zwillinger, D., ***CRC Standard Mathematical Tables and Formulae***, 30th ed., CRC Press, Boca Raton, 1996
2. Hogben, L., ***Mathematics for the Million***, W.W. Norton & Co., New York, 1983
3. Sokolnikoff, I.S., Redheffer, R.M., ***Mathematics of Physics and Modern Engineering***, ISBN 07-059625-5, McGraw-Hill Book Co., New York, 1966
4. Andrews, A., "Mathematical Microprocessor Software: A √x Comparison", Byte Magazine, May 1982
5. Chu, W., Li, Y., "Cost/Performance Tradeoffs of n-Select Square Root Implementations", Computer Laboratory, University of Aizu, Japan
6. Kreyszig, E., ***Advanced Engineering Mathematics***, 3rd ed., John Wiley & Sons, New York, 1972
7. Giberson, S., Osler, T.J., "Extending Theon's Ladder to Any Square Root", *The College Mathematics Journal*, May 2004
8. Gerald, C.F., Wheatley, P.O., ***Applied Numerical***

Analysis, 3rd ed.,ISBN 0-201-11577-8, Addison-Wesley Publishing Co., Reading, MA, 1984

9. Abramowitz, M., Stegun, I.A., ***Handbood of Mathematical Functions***, Dover Publications, New York, 1972
10. Volder, J.E., “The CORDIC Trigonometric Computing Technique”, IRE Trans. Electronic Computers, Sept. 1959
11. Walther, J.S., “A Unified Algorithm for Elementary Functions”, Spring Joint Computer Conference, 1971
12. Andrews, M., “Mathematical Microprocessor Software: A √x Comparison”, IEEE Micro, May 1982
13. “Cordic Methods”, http://www.voidware.com/cordic.htm
14. Andraka, R., “A Survey of CORDIC Algorithms for FPGA Based Computers”, FPGA 98 Monterey CA, 1998