THE MATHEMATICS BEHIND THE FAST INVERSE SQUARE ROOT FUNCTION CODE

Charles McEniry
August 2007

ABSTRACT

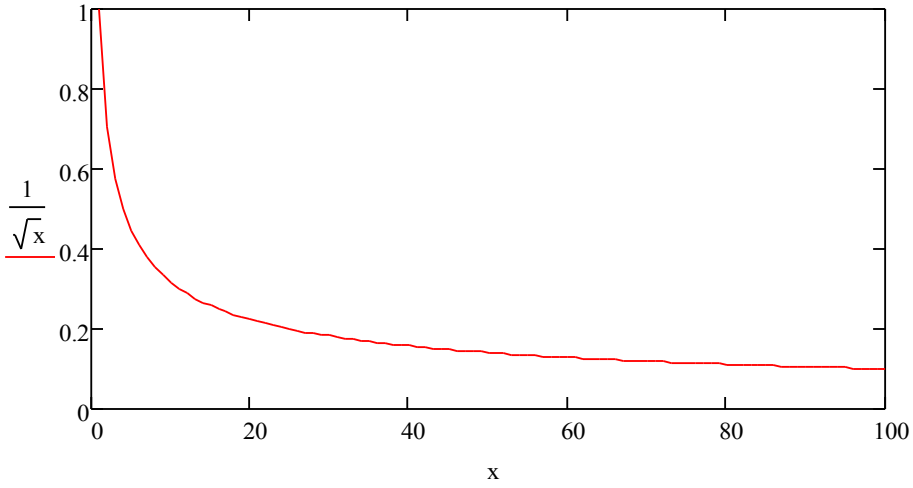
In the recent past, much ado has been made about the origin and operation of the fast inverse of the square root function found in online code libraries. Although plenty of coverage has been given to the use of the Newton-Raphson method for finding roots in the algorithm and the processing cost of the algorithm code, conversely, very little coverage has dealt with the mathematics behind the initial approximation used in the algorithm. Thus, using a mathematical approach, this article deals with the derivation of the code's infamous magic formula and possible methods used to determine an optimal value of the formula's magic number.

INTRODUCTION

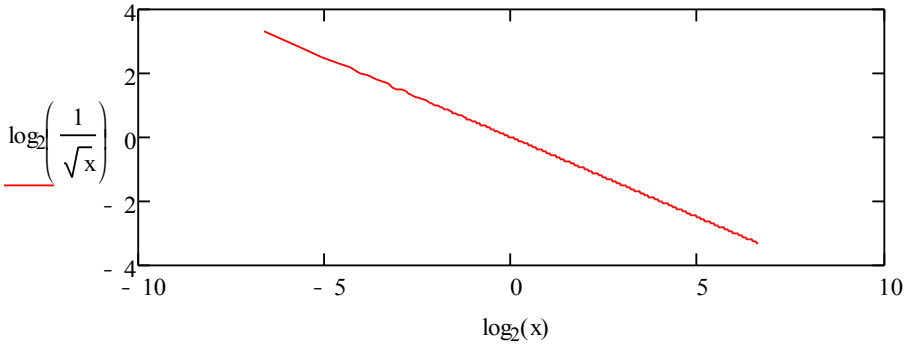
Ryszard Sommefeldt republished an article (http://www.beyond3d.com/content/articles/8/) in late 2006 on the quest for the origin of the fast invsqrt function code. The article traced through the memories of graphic programming luminaries John Carmack, Terje Mathisen, and Gary Tarolli, but brought no solid conclusion as to who originated the code. The article did prompt a Slashdot effect leading to a second article (http://www.beyond3d.com/content/articles/15/) and revealed Greg Walsh, working with Cleve Moler roughly twenty years ago or more, as the most likely origin of the code. Even before the release of the Quake 3 Arena source code in the summer of 2005, discussions about this function code have appeared on various online postings, news groups, and forums over the past several years. Most notably of the postings was the paper (http://www.lomont.org/Math/Papers/2003/InvSqrt.pdf) by Chris Lomont in early 2003.

```
float InvSqrt (float x){
    float xhalf = 0.5f*x;
    int i = *(int*)&x;
    i = 0x5f3759df - (i>>1);
    x = *(float*)&i;
    x = x*(1.5f - xhalf*x*x);
    return x;
}
```

For those who have not wandered upon this topic before, the function code implements a fast evaluation of the inverse or reciprocal of the square root of the input value for floating-point numbers based upon the IEEE standard for 32-bit binary floating-point numbers, in short, by using the Newton-Raphson method with an initial guess determined from the input value. For graphics programming, especially twenty years ago, this code provided a significant advantage in performance over using the nominal method of calling the square root function and performing floating-point division, but at a loss of precision. Ideally, the constant used in the function is such that the maximum relative error for all possible input values is reduced to a minimum and within application tolerances. Today, most graphical processing units and many of the additional instruction sets for processors implement similar, but more complex, methods in the hardware.



Although many have used the term magic to describe what the code does, the magic is really just the clever application of some mathematical methods and techniques along with taking advantage of the IEEE representation format for floating-point number. So, instead of having the classical proof by divine intervention reason of "then a miracle occurs…" to describe the magical line, the magical line is derived mathematically, based upon the linear relationship between the logarithms of the input value and the desired output value. The resulting equation can then be subjected to error analysis and numerical methods for optimization and error reduction.



DERIVATION

Given the IEEE representation format for 32-bit binary floating-point numbers:

S	E	M
1 *bit*	8 *bits*	23 *bits*
bit 31	30 ← *bits* → 23	22 ← *bits* → 0

where S is the sign bit with a 1 for denoting a negative number, E is an 8-bit biased exponent, and M is the remaining 23 bits representing a mantissa; the value represented is:

$$x = (-1)^S (1.M) 2^{E-B} \quad \text{with} \quad B = 127 \text{ as the bias.}$$

And, the integer I corresponding to the 32-bit value is:

$$I = S \cdot 2^{31} + E \cdot 2^{23} + M .$$

Since $x \geq 0$ for $y = 1/\sqrt{x}$, then $y \geq 0$, $S = 0$, and, simply:

$$x = (1.M)2^{E-127} \quad \text{and} \quad I = E \cdot 2^{23} + M .$$

Alternately and correspondingly, with any n-bits binary representation with a b-bits exponent,

S	E	M
1 *bit*	b *bits*	$(n - 1 - b)$ *bits*
n *bits*		

the values of x and I_x are:

$$x = (1 + m_x)2^{e_x} \quad \text{and} \quad I_x = E_x L + M_x$$

$$\text{where} \quad m_x = \frac{M_x}{L} \quad \text{and} \quad e_x = E_x - B \quad \text{with} \quad L = 2^{n-1-b} \quad \text{and} \quad B = 2^{b-1} - 1 .$$

Thereby:

$$y = 1/\sqrt{x}$$

$$\log_2 y = -\tfrac{1}{2}\log_2 x$$

$$\log_2(1 + m_y) + e_y = -\tfrac{1}{2}\log_2(1 + m_x) - \tfrac{1}{2}e_x$$

Given $\log_2(1 + x) \approx x$ for $0 \leq x \leq 1$, let $\log_2(1 + x) \cong x + \sigma$.

$$m_y + \sigma + e_y = -\tfrac{1}{2}m_x - \tfrac{1}{2}\sigma - \tfrac{1}{2}e_x$$

$$M_y + (E_y - B)L = -\tfrac{3}{2}\sigma L - \tfrac{1}{2}M_x - \tfrac{1}{2}(E_x - B)L$$

$$E_y L + M_y = \tfrac{3}{2}(B - \sigma)L - \tfrac{1}{2}(E_x L + M_x)$$

Thus,

$$I_y = R - \tfrac{1}{2}I_x \quad \text{where} \quad R = \tfrac{3}{2}(B - \sigma)L .$$

Hence, the equation $I_y = R - \frac{1}{2}I_x$ corresponds to the magical line in the function code,

```
i = 0x5f3759df - (i>>1);
```

where $R = 0x5F3759DF$, and the resulting value for σ is $\sigma = 0.045046567916870117 1875$.

ANALYSIS

From the equation $I_y = R - \frac{1}{2} I_x$, first, determine separate equations for E_y and m_y in terms of E_x and m_x, directly or indirectly:

$$I_y = R - \tfrac{1}{2} I_x$$

$$E_y L + M_y = \tfrac{3}{2}(B - \sigma)L - \tfrac{1}{2}(E_x L + M_x)$$

Let $\phi = \{0,1\}$ depending on whether E_x is even or odd, and let $m'_x = \frac{1}{2}(\phi + m_x)$.

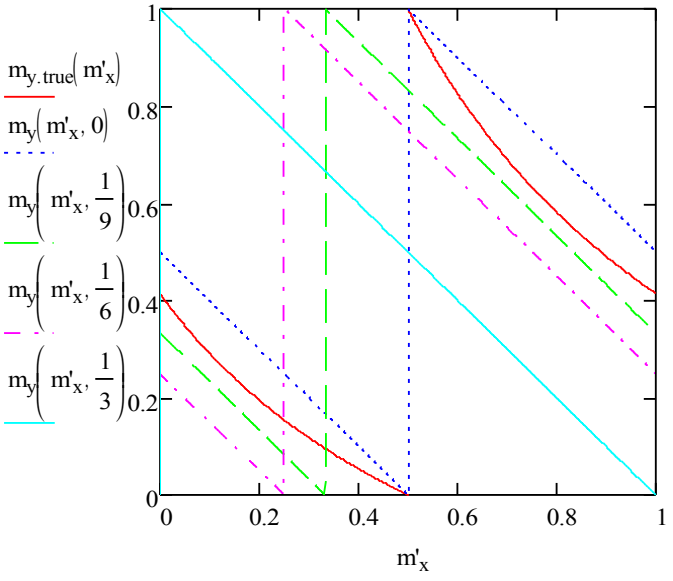
$$E_y + m_y = \tfrac{1}{2}\left(3B - 1 - E_x + \phi\right) + \tfrac{1}{2}(1 - 3\sigma) - m'_x$$

Thus, $$E_y = \tfrac{1}{2}\left(3B - 1 - E_x + \phi\right) \text{ and } m_y = \tfrac{1}{2}(1 - 3\sigma) - m'_x .$$

From the IEEE representation format, constraints on E_y and m_y are (1) $E_y \equiv \lfloor E_y \rfloor$ or E_y is an integer, and (2) $m_y \in [0,1)$. Determining the equations for E_y and m_y according:

$$E_y = \begin{cases} \frac{1}{2}(3B - 1 - E_x) \\ \frac{1}{2}(3B - 3 - E_x) \\ \frac{1}{2}(3B - 2 - E_x) \end{cases} \text{ and } m_y = \begin{cases} \frac{1}{2}(1 - 3\sigma) - m'_x \\ \frac{3}{2}(1 - \sigma) - m'_x \\ \frac{3}{2}(1 - \sigma) - m'_x \end{cases} \text{ for } m'_x : \begin{cases} 0 \le m'_x \le \frac{1}{2}(1 - 3\sigma) \\ \frac{1}{2}(1 - 3\sigma) < m' < \frac{1}{2} \\ \frac{1}{2} \le m'_x < 1 \end{cases} \text{ and } \phi : \begin{cases} 0 \\ 0 \\ 1 \end{cases}$$

Solving for σ in $0 \le \frac{1}{2}(1 - 3\sigma) \le \frac{1}{2}$ returns $0 \le \sigma \le \frac{1}{3}$ as the bounds of σ. The following figure plots the true value of m_y and the equation $m_y(m'_x, \sigma)$ for $0 \le m'_x < 1$ and $\sigma \in \{0, \frac{1}{9}, \frac{1}{6}, \frac{1}{3}\}$.



Let y_0 represent y determined by $I_y = R - \frac{1}{2} I_x$, and let $\varepsilon_n = y - y_n$. Therefore,

$$|E_n| = \left|\frac{\varepsilon_n}{y}\right| = \left|1 - \frac{y_n}{y}\right|.$$

Solving for E_0:

$$|E_0| = \left|1 - \frac{\left[1 + \frac{1}{2}(k - 3\sigma) - m'_x\right] \cdot 2^{\left[\frac{1}{2}(3B - K - E_x) - B\right]}}{\left(1 + m_y\right) \cdot 2^{(E_y - B)}}\right|, k \in \{1,3\}, K \in \{1,2,3\}$$

Letting $E_y \cong \frac{1}{2}(3B - K - E_x)$,

$$|E_0| \approx \left|\frac{m_y - \frac{1}{2}(k - 3\sigma) + m'_x}{\left(1 + m_y\right)}\right|$$

Thus, $E_0 \propto m'_x$, and E_0 is not directly dependent upon E_x, but only if E_x is even or odd, which is accounted for by $m'_x = \frac{1}{2}(\phi + m_x)$, however, E_0 is discontinuous at the transition point of $m'_x = \frac{1}{2}(1 - 3\sigma)$. This relationship implies that only a continuous subset of x is needed to determine the maximum relative error for the approximation, and, likewise, only a corresponding range of values for I is needed. The advantage is doing fewer iterations and calculations to determine the maximum relative error per value of σ tested, especially if those operations are being performed on a system from roughly twenty years ago.

The resulting value of y from I_y is an approximation of $1/\sqrt{x}$ with an error related to R and the value of σ used to calculate R. By using an iterative method where $y_n = F(x, y_{n-1})$, the error of the approximation is reduced. However, generally, instead of iterating the function directly, the equation is rewritten in a form of

$$g(y) = x - f^{-1}(y)$$

where the desired value of y is a root of $g(y)$, and a root-finding method is implored to determine a value for y which is the approximate solution to $y = f(x)$ for the given x. Since an initial value for y is known, a trivial method to use is the Newton-Raphson method for root-finding. The equation for $g(y)$ is

$$g(y) = x - \frac{1}{y^2} = x - y^{-2}$$

Equation for $g'(y)$: $$g'(y) = 2y^{-3}$$

Newton-Raphson formula: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Resulting equation for y_{n+1}: $y_{n+1} = \frac{1}{2} y_n (3 - x y_n^2)$

Applying $\varepsilon_n = y - y_n$: $y - \varepsilon_{n+1} = \frac{1}{2}(y - \varepsilon_n)\left(3 - x(y - \varepsilon_n)^2\right)$

Solving for ε_{n+1}: $\varepsilon_{n+1} = \frac{1}{2} x \varepsilon_n^2 (3y - \varepsilon_n)$

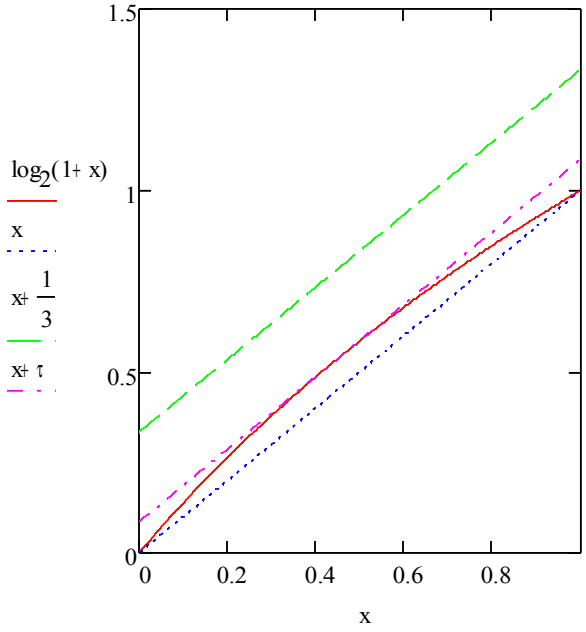
Solving for E_{n+1}: $E_{n+1} = \frac{1}{2} E_n^2 (3 - E_n)$

Thus, $E_1 = \frac{1}{2} E_0^2 (3 - E_0)$

Since, $E_0 \propto m'_x$, and E_0 is not directly dependent upon E_x, therefore, $E_1 \propto m'_x$, and E_1 is not directly dependent upon E_x. Again, the benefit is doing fewer iterations and calculations to determine the maximum relative error per value of σ tested.

CALCULATIONS

Although σ is bounded by $[0, \frac{1}{3}]$, for any value $\sigma : \sigma > \tau$, where τ corresponds to $(x + \tau)$ tangent to $\log_2(1 + x)$, produces exceeding larger errors. Thus, σ is better bounded by $[0, \tau]$. The following figure displays the plots of $f(x) = \log_2(1 + x)$, $f(x) = x$, $f(x) = x + \frac{1}{3}$, and $f(x) = x + \tau$.



For the equation:

$$\log_2(1+x) = x+\sigma \;,\; x \in [0,1)\,,$$

let $\sigma \in [0,\tau]$ where τ is the value of σ such that $\log_2(1+x) \perp (x+\sigma)$. Calculating for τ using

$$\tau = \log_2(1+x) - x \text{ where } x : \frac{d}{dx}\log_2(1+x) = \frac{d}{dx}(x+\tau)$$

results with $\tau = \alpha \ln(\alpha) - \alpha + 1 = 0.086071332055934206887573098776923$

where $\alpha = 1/\ln 2$, $x = \alpha - 1$, and $\log_2(1+x) = \alpha \ln(\alpha)$.

For $\sigma \in [0,\tau]$, maximum errors of $|\varepsilon(x)| = |\log_2(1+x) - (x+\sigma)|$ occur at $\sigma = 0$ and $\sigma = \tau$. To minimize the maximum errors, let $\eta \in [0,\tau]$ such that

$$|(x+\tau) - (x+\eta)| = |x - (x+\eta)|\,.$$

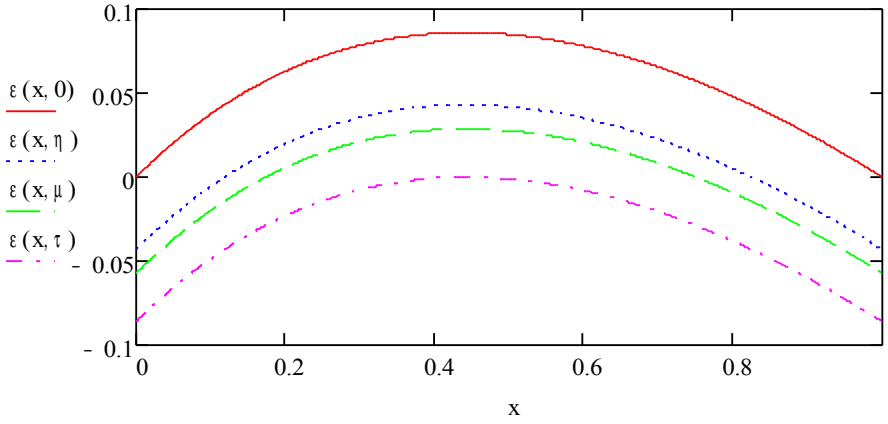
Solving for η gives $\eta = \frac{1}{2}\tau = 0.043035666027967103443786549388461$.

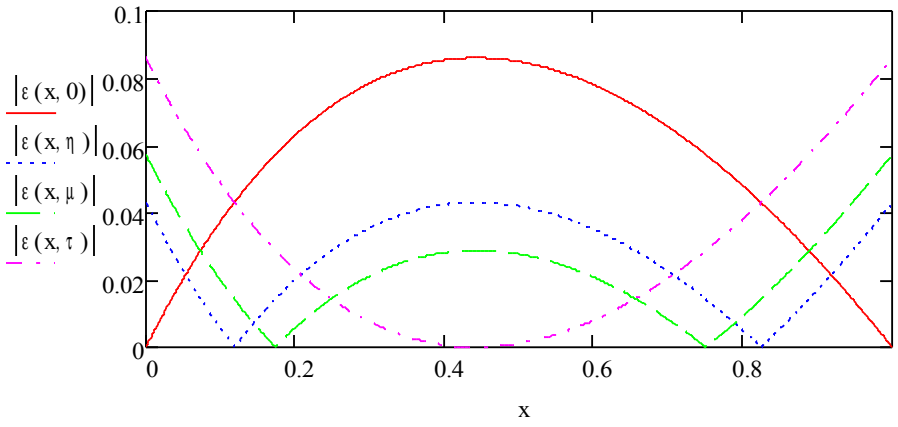
To minimize the total error of $\varepsilon(x) = \log_2(1+x) - (x+\sigma)$ for $\sigma \in [0,\tau]$, let $\mu \in [0,\tau]$ such that

$$\int_0^1 [\log_2(1+x) - (x+\mu)]\,dx = 0\,.$$

Solving for μ gives $\mu = \frac{3}{2} - \alpha = 0.057304959111036592640075318998108$.

The figures below plot $\varepsilon(x,\sigma) = \log_2(1+x) - (x+\sigma)$ and $|\varepsilon(x,\sigma)|$ for $\sigma \in \{0,\eta,\mu,\tau\}$ and $0 \le x \le 1$.





Let $y_0(x,\sigma)$ represent y derived using $I_y = R - \frac{1}{2}I_x$ with $R = \frac{3}{2}(B - \sigma)L$:

$$y_0(x,\sigma) = \left(1 + \frac{M_y}{L}\right)\cdot 2^{(E_y - B)} = \left[1 + \frac{1}{2}\left(k - 3\sigma - \frac{M_x}{L}\right)\right]\cdot 2^{\frac{1}{2}(B - K - E_x)},\ k \in \{1,3\},\ K \in \{1,2,3\}$$

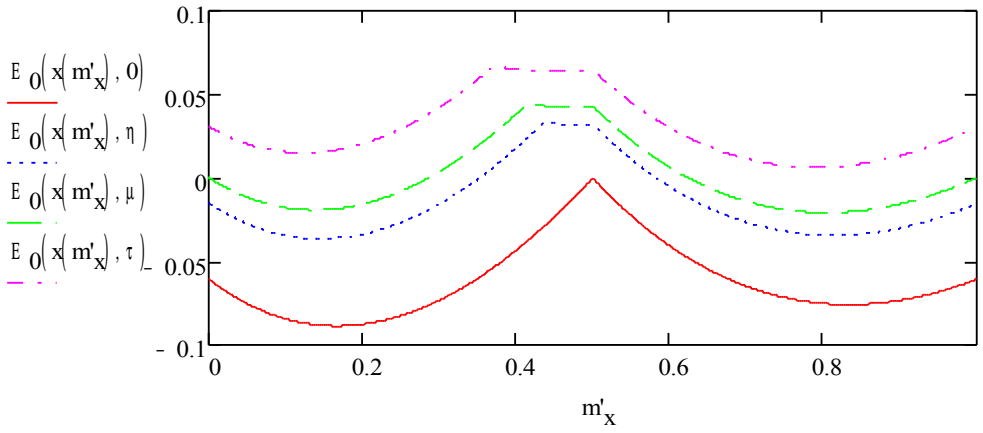
Let $y_1(x,\sigma)$ represent y derived using the Newton-Raphson method with $y_0(x,\sigma)$:

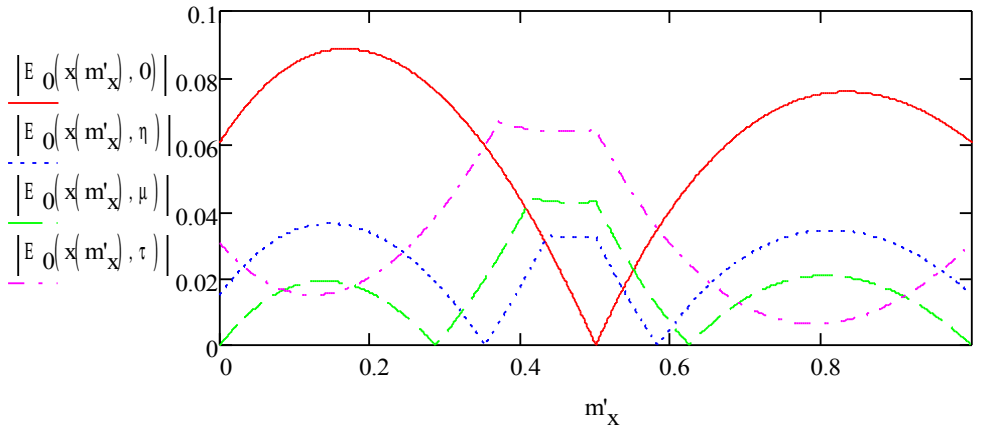
$$y_1(x,\sigma) = \frac{1}{2}y_0(x,\sigma)\left(3 - x\cdot y_0(x,\sigma)^2\right)$$

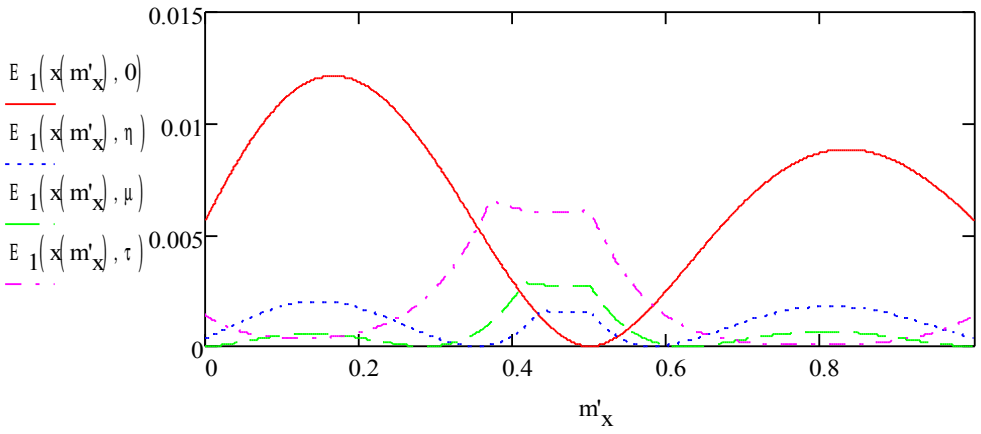
And, let the equations for the relative error of $y_0(x,\sigma)$ and $y_1(x,\sigma)$, respectfully, be

$$|E_0(x,\sigma)| = \left|1 - \frac{y_0(x,\sigma)}{1/\sqrt{x}}\right| \quad \text{and} \quad |E_1(x,\sigma)| = \left|1 - \frac{y_1(x,\sigma)}{1/\sqrt{x}}\right|.$$

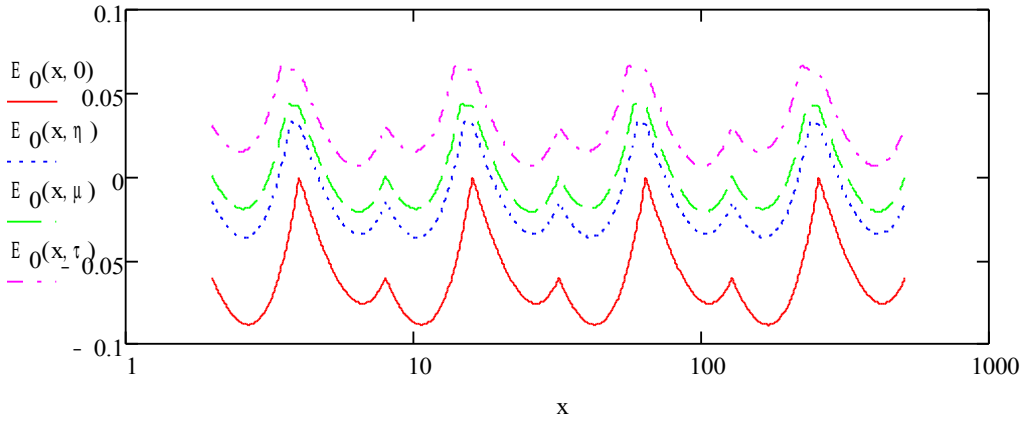
The following figures plot E_0, $|E_0|$, and E_1 for various values of σ against $m'_x : m'_x \in [0,1)$.

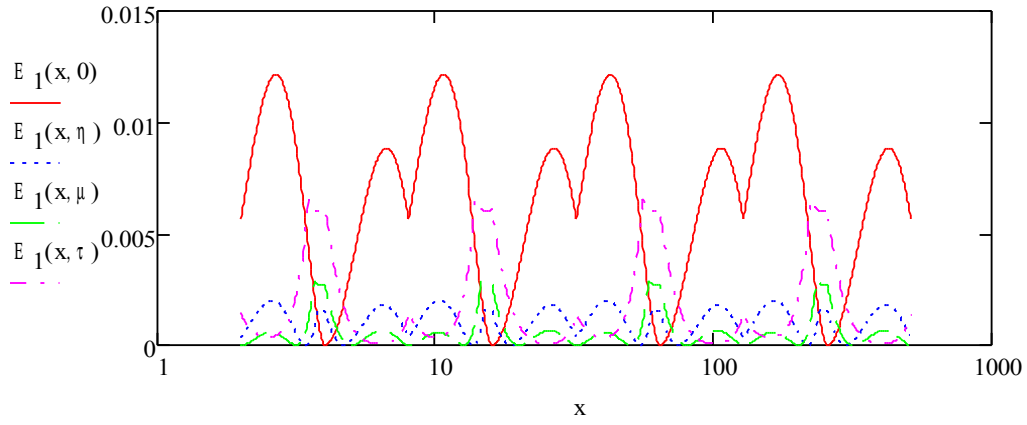




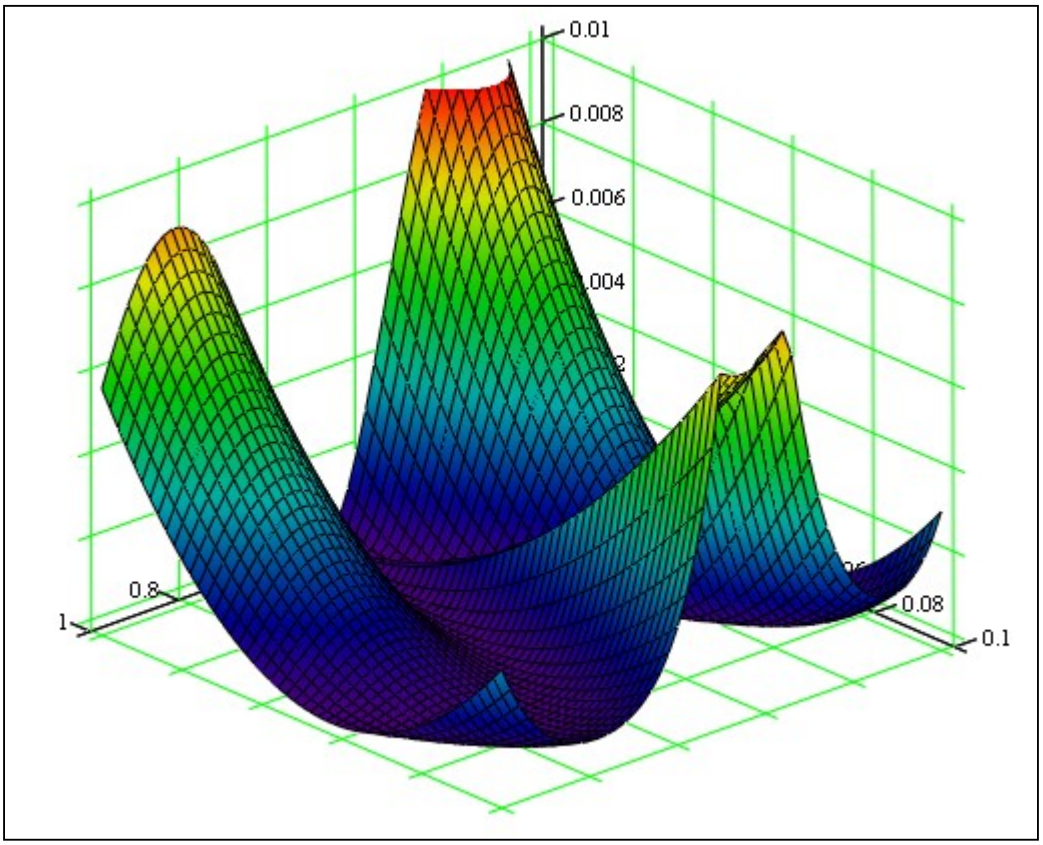


To exemplify E_0, and E_1 over $m'_x : m'_x \in [0,1)$, the following additional figures show the repeating pattern of E_0, and E_1 over a sample range of x.





The following plot better illustrates the response of E_1 to both m'_x and σ with $m'_x \in [0,1]$ and $\sigma \in [0,0.1]$, limiting $E_1 \in [0,0.01]$.

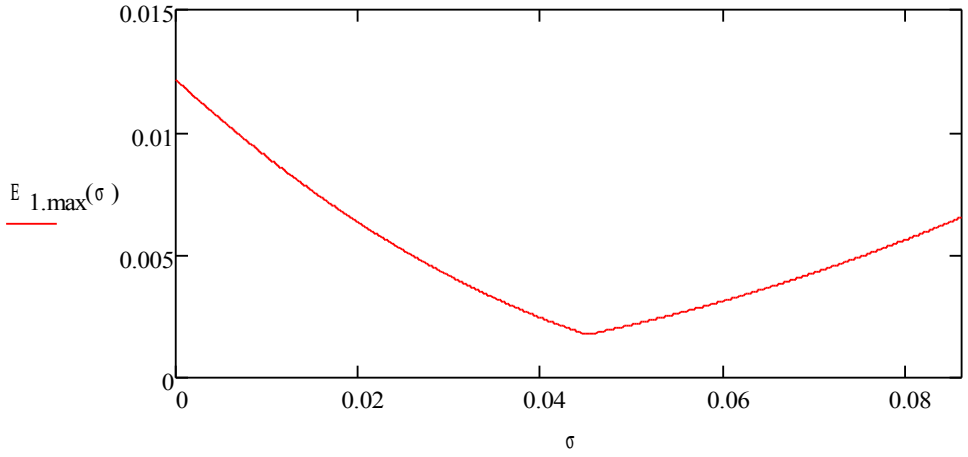


E_1

Given the response of E_1 to m'_x and σ, to minimize the maximum relative error of E_1, the value of σ needs to be found for the following:

$$\max\left(E_1(m'_x,\sigma),0 \le m'_x \le \tfrac{1}{2} - \tfrac{3}{2}\sigma\right) = \max\left(E_1(m'_x,\sigma),\tfrac{1}{2} - \tfrac{3}{2}\sigma < m'_x < \tfrac{1}{2}\right)$$

The following figure illustrates the response of $\max(E_1)$ against σ .



OPTIMIZATION

Part of optimizing the function code is determining the value of R that minimizes the maximum relative error over all possible inputs. Although not very practical, the basic method to determine R is brute force iteration over all possible inputs with all possible values of R for $\sigma \in [0,\frac{1}{3}]$, but this would result in a considerable number of calculations and conditional evaluations, costing considerable amount of time for processing, especially, on a system from roughly twenty years ago. Given the repeating pattern of the error evaluation, E_1, as shown earlier, the range of input values can be considerably reduces, to less than 1% of the input range for the IEEE 32-bit float. Additionally, the range of possible values for R can be reduced to the range corresponding to $\sigma \in [0,\tau]$, $\tau : \log_2(1+x) \perp (x+\tau)$, as determined previously, to roughly 26% of the initial range of values for R. Presumably, the optimal value for R would exist between $R:\sigma = \eta$, the σ minimizing the maximum error of $|\varepsilon(x)| = |\log_2(1+x) - (x+\sigma)|$, and $R:\sigma = \mu$, the σ minimizing the total error of $\varepsilon(x,\sigma) = \log_2(1+x) - (x+\sigma)$, closer to the former. This reduces the range for values of R to roughly 5% of the initial range. These combine to reduce a brute force approach to iterating over less than 0.04% of all possible inputs and all possible values of R. Finally, instead of using straight linear brute force and given the response of E_1 to m'_x and σ, a bisecting method, similar to finding the minimum or maximum of a polynomial segment, may be applied to determining the optimal value of R. The following table gives the maximum number of iterations possibly performed, with 8388608 iterations per each value of E_x and $R(\sigma)$

Iterations	#	$\sigma \in [0,\frac{1}{3}]$ Linear (4194304)	$\sigma \in [0,\tau]$ Linear (1083028)	$\sigma \in [\eta,\mu]$ Linear (179549)	$\sigma \in [\eta,\mu]$ Bisecting (18)
$\forall E_x$	(254)	8936830510563328	2307614719206848	382566761366852	38352715776
m'_x	(2)	70368744177664	18170194639424	3012336703676	301989888

The corresponding percentages for the previous table are listed below.

Iterations %	$\sigma \in [0,\frac{1}{3}]$ Linear	$\sigma \in [0,\tau]$ Linear	$\sigma \in [\eta,\mu]$ Linear	$\sigma \in [\eta,\mu]$ Bisecting
$\forall E_x$	100	25.8	4.28	0.00429
m'_x	0.787	0.203	0.0337	0.00000338

Depending upon the precision used in the error analysis, using brute force iteration results in $R = 0x5F375A85$ or $R = 0x5F375A86$, as previously presented by Chris Lomoth (http://www.lomont.org/Math/Papers/2003/InvSqrt.pdf) after having reduced the range to iterate over. Using the bisecting method provides a quicker means to find a solution, but the bisecting method is not without pitfalls. Because the bisecting method proceeds with a subinterval based upon the evaluation at the midpoint, the method may inadvertently proceed with the wrong subinterval if the appropriate endpoint has a higher associated maximum relative error than the other endpoint. In this case, the bisecting has "stepped over" the actual minimum. The new endpoint or the other endpoint is the minimum on selected subinterval, and the bisecting converges to that endpoint. Using a weighted or adaptive bisecting method may avoid this case, but may require more bisecting iterations, as shown in the following list:

Result, Max. Rel. Error, Bisecting Iterations Total Iterations ($\delta = \frac{1}{4}(\mu - \eta)$) #	Average (Fair) Bisecting	Heavily Weighted Bisecting	Lightly Weighted Bisecting
$\sigma = 0$,0x5F400000 $\sigma = \tau$,0x5F2F796C	0x5F37598F 0.00175281170385233 21 176160768	0x5F373C65 0.00179984232160280 13 109051904	0x5F375A86 0.00175127001276110 39 327155712
$\sigma = \eta$,0x5F37BCB6 $\sigma = \mu$,0x5F34FF59	0x5F375A16 0.00175195107385662 18 150994944	0x5F375895 0.00175435792917256 11 92274688	0x5F375A86 0.00175127001276110 34 285212672
$\sigma = \eta - \delta$,0x5F386C0D $\sigma = \mu + \delta$,0x5F345001	0x5F375A83 0.00175127879737136 19 159383552	0x5F36F819 0.00191248533062094 12 100663296	0x5F375A86 0.00175127001276110 35 293601280
$\sigma = \eta$,0x5F37BCB6 $\sigma = \mu - \delta$,0x5F35AEB0	0x5F375A16 0.00175195107385662 18 150994944	0x5F376FAD 0.00180539383568146 11 92274688	0x5F375A86 0.00175127001276110 33 276824064

To avoid possible pitfalls with bisecting, alternate approaches may have been used in the past. Previously, the computing cost of iterating for a single value of both E_x and $R(\sigma)$ was

significant. To further reduce the total number of iterations, a more mathematical approach could have been used, such as a predictive method. Given that $\max(E_1(m'_x,\sigma)), \forall m'_x \in [0,1)$, as $\sigma \to \sigma_{optimal}$, is decreasing towards $\sigma = \sigma_{optimal}$, from both the right and the left, let the functions of $E_1(m'_x,\sigma)$ for both the left and right be in the quadratic form of:

$$\max(E_1(m'_x,\sigma))\Big|_{\forall m'_x \in [0,1)}^{\sigma \to \sigma_{optimal}} = A\sigma^2 + B\sigma + C$$

Let point $(u,v) = (\sigma, \max(E_1(m'_x,\sigma)))$, $\forall m'_x \in [0,1)$. With additional points (u,v) iterated, the Lagrange interpolating polynomials for the both the left and right polynomials can be determined with a minimum of six points—three points used to determine the left-side polynomial, and three points for the right-side polynomial—and, thus, six iteration over $m'_x : \forall m'_x \in [0,1)$. The calculated intersection of the two polynomial equations would approximate $\sigma = \sigma_{optimal}$.

The Lagrange interpolating polynomial is defined by:

$$P(x) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

The polynomial for the three points (u,v) on either side:

$$P(u) = \frac{(u-u_2)(u-u_3)}{(u_1-u_2)(u_1-u_3)}v_1 + \frac{(u-u_1)(u-u_3)}{(u_2-u_1)(u_2-u_3)}v_2 + \frac{(u-u_1)(u-u_2)}{(u_3-u_1)(u_3-u_2)}v_3$$

To further reduce, and simplify, the polynomial equations, also reducing the computing cost, let $u_1 = a - \delta$, $u_2 = a$, and $u_3 = a + \delta$, thus:

$$P(u) = \frac{1}{\delta^2}\left(\left(\tfrac{1}{2}v_1 - v_2 + \tfrac{1}{2}v_3\right)(u-a)^2 - \tfrac{1}{2}\delta\left(v_1 - v_3\right)(u-a) + \delta^2 v_2\right)$$

Rewritten in the form of $Au^2 + Bu + C$ gives the following coefficients, with simplification:

$$A = \frac{1}{\delta^2}\left(\tfrac{1}{2}v_1 - v_2 + \tfrac{1}{2}v_3\right), \quad B = -2aA - \tfrac{1}{2}(v_1 - v_3)/\delta, \quad C = v_2 - a^2A - aB.$$

Thus, the intersection of the left and right side polynomials, defined by:

$$A_{left}u^2 + B_{left}u + C_{left} = A_{right}u^2 + B_{right}u + C_{right}$$

has a real solution, determined from the quadratic formula, which approximates the value $\sigma = \sigma_{optimal}$ and, therefore, approximates the optimal value $R(\sigma_{optimal})$ for the function code,

dependent upon the iterated data points. Using this predictive method, the table below lists approximation value of $R : \sigma \approx \sigma_{optimal}$ for the given values of a and δ .

a_{left}	a_{right}	δ	R_σ	$\max(E_1(m'_x, \sigma))$
$\eta - \delta$	$\mu - \delta$	$\frac{1}{4}(\mu - \eta)$	0x5F375A91	0.00175137771629619
$\eta - \delta$	μ	$\frac{1}{4}(\mu - \eta)$	0x5F375A9F	0.00175151631478343
$\eta - \delta$	$\mu + \delta$	$\frac{1}{4}(\mu - \eta)$	0x5F375AFB	0.00175242876177617
$\eta - \delta$	μ	$\frac{1}{2}(\mu - \eta)$	0x5F375A88	0.00175128948205849
$\eta - \delta$	$\mu + \delta$	$\frac{1}{2}(\mu - \eta)$	0x5F375B3B	0.00175306525834174
$\eta - \delta$	$\mu + \delta$	$\frac{1}{4}(\tau - \mu)$	0x5F375B39	0.00175304601624759
$\eta - \delta$	$\mu + \delta$	$\frac{1}{2}(\tau - \mu)$	0x5F375BD5	0.00175459372633202

Using the predictive method can reduce the number of iteration runs over all value of m'_x to less then ten runs—six runs to get needed data points for the Lagrange interpolation polynomial—but may require additional runs to refine the value. However, the number of runs, and therefore the total number of iteration, would be less than using than using bisecting. The minimum total number of iterations using a predictive method is 50331648 individual iterations, or 0.0000005632% of the full iteration over all m'_x and σ values.

Going back to the response plot of E_1 and the equation for $|E_0|$, let the value $\sigma = \sigma_{optimal}$ be the solution of:

$$\max(E_1(m'_x, \sigma), 0 \le m'_x \le \tfrac{1}{2} - \tfrac{3}{2}\sigma) = \max(E_1(m'_x, \sigma), \tfrac{1}{2} - \tfrac{3}{2}\sigma < m'_x < \tfrac{1}{2})$$

Now, in addition to using a bisecting method for the iteration over the values of σ , a bisecting method may also be applied to the determination of the maximum value of E_1 for the range $0 \le m'_x \le \frac{1}{2}(1 - 3\sigma)$, but no iteration is needed over the range $\frac{1}{2}(1 - 3\sigma) \le m'_x \le \frac{1}{2}$ because the maximum of E_1 on that segment occurs at $m'_x = \frac{1}{2}(1 - 3\sigma)$. This double bisecting method provides a result of ~525 iterations and $R = 0x5F375A87 \pm 1$ depending upon starting conditions and precision. This is 0.0010431% the number of iterations against the best case using the predicting method above, and, amazingly, 5.8745E-12% or roughly six-trillionth of a percent against the full range.

Finally, instead of iterating to determine an optimal solution, a mathematical approach shall to be used. Returning to the constraint equation from above:

$$\max(E_1(m'_x, \sigma), 0 \le m'_x \le \tfrac{1}{2} - \tfrac{3}{2}\sigma) = \max(E_1(m'_x, \sigma), \tfrac{1}{2} - \tfrac{3}{2}\sigma < m'_x < \tfrac{1}{2})$$

For the right-side of the equation, the maximum value of E_1 occurs at $m'_x = \frac{1}{2}(1 - 3\sigma)$. For the left-side of the equation, the maximum value of E_1 occurs at

$$m'_x : \frac{dE_1}{dm'_x} = 0, 0 \le m'_x \le \tfrac{1}{2} - \tfrac{3}{2}\sigma$$

From $E_1 = \frac{1}{2}E_0^{\,2}(3 - E_0)$, $\quad \frac{dE_1}{dp} = \frac{1}{2}\left[1 - (1 - E_0)^2\right]\frac{dE_0}{dp}$

The maximum occurs at $\quad E_0 = 0$, $E_0 = 2$, and $\frac{dE_0}{dp} = 0$.

Given $m'_x = \frac{1}{2}(\phi + m_x)$ and $\phi = 0$ for $0 \le m'_x < \frac{1}{2}$, thus $m'_x = \frac{1}{2}m_x$, and calculating for E_0:

$$E_0 = 1 - \frac{\frac{1}{2}(3 - 3\sigma - m_x)}{(1 + m_y)}$$

Solving for m_y in terms of m'_x or m_x for $m'_x \in [0, \frac{1}{2}(1 - 3\sigma)]$ gives $(1 + m_y) = \sqrt{2}/\sqrt{1 + 2m'_x}$ or $(1 + m_y) = \sqrt{2}/\sqrt{1 + m_x}$.

$$E_0 = 1 - \tfrac{1}{2\sqrt{2}}(3 - 3\sigma - m_x)\sqrt{1 + m_x}$$

Let $p = 1 + m_x$ and $w = (3 - 3\sigma + m_x)\sqrt{1 + m_x}$ or $w = (4 - 3\sigma - p)\sqrt{p}$.

$$E_0 = 1 - \tfrac{1}{2\sqrt{2}}w \quad \text{and} \quad \frac{dE_0}{dp} = \tfrac{1}{4\sqrt{2}}(3p + 3\sigma - 4)/\sqrt{p}$$

Therefore, $E_0 = \{0 \mid 2\} \rightarrow p^3 + 2p^2(3\sigma - 4) + p(3\sigma - 4)^2 - 8 = 0$ and $\frac{dE_0}{dp} = 0 \rightarrow p = \frac{4}{3} - \sigma$.

Solving for the maximum along $0 \le m'_x \le \frac{1}{2}(1 - 3\sigma)$ gives

$$m_x = \tfrac{1}{3} - \sigma \quad \text{or} \quad m'_x = \tfrac{1}{6} - \tfrac{1}{2}\sigma$$

And, for the right-side, $\quad m_x = 1 - 3\sigma \quad \text{or} \quad m'_x = \tfrac{1}{2} - \tfrac{3}{2}\sigma$

Thus, solving for the $E_1|_{left} = E_1|_{right}$ gives $\sigma = 0.04503327680657146$ and $R = 0\text{x}5\text{F}375\text{A}86$. Calculating for the double-precision floating-point number, where $n = 64$ and $b = 11$, gives $R = 0\text{x}5\text{FE}6\text{EB}50\text{C}7\text{B}537\text{AA}$.

CONCLUSION

Hopefully, this article has answered the questions surrounding the function code for the fast inverse square root function and shown what is behind the magic curtain of the function code. Applying an approximation for the base 2 logarithm resulted in the mathematical derivation of the magic code line, and analyzing the resulting error from the approximation provided for an evaluation of the optimal value for the magic constant in the code. This means of derivation may be applied to more general equations providing for the following:

$y = 1/\sqrt[n]{x}$	$y = \sqrt[n]{x}$	$y = x^a$
$I_y = \left(\frac{n+1}{n}\right)(B - \sigma)L - \frac{1}{n}I_x$	$I_y = \left(\frac{n-1}{n}\right)(B - \sigma)L + \frac{1}{n}I_x$	$I_y = (1 - a)(B - \sigma)L + aI_x$
$y_1 = \frac{1}{n}y_0(n + 1 - xy_0^n)$	$y_1 = \frac{1}{n}y_0(n - 1 - xy_0^{-n})$	$y_1 = y_0(1 - a + axy_0^{-1/a})$

The optimal constant value varies for each root or power function. However, for the great majority of root or power functions, the evaluation will not be able to take advantage of integer operations if implemented in code. Also, remember $\sqrt{x} = x \cdot \left(1/\sqrt{x}\right)$.

Finally, as a last suggestion, the following function code is provided as an alternative:

```
float inv_sqrt( float x )
{
    union { float f; unsigned long ul; } y;
    y.f = x;
    y.ul = ( 0xBE6EB50CUL - y.ul ) >> 1;
    y.f = 0.5f * y.f * ( 3.0f - x * y.f * y.f );
    return y.f;
}
```

Adjusting for double-precision floating-point numbers changes the “float” to “double”, the “unsigned long” to “unsigned long long”, and the constant from above to the unsigned long long value of 0xBFCDD6A18F6A6F55ULL.

ENDNOTE

The primary intent behind this article is to explain the mathematics behind the fast inverse square root function code, and, hopefully, recreate the method or approach that was possibly used for development of the original code and the value $R = 0\text{x}5\text{f}3759\text{df}$. Although none of the various methods discussed to find the optimal value were able to derive the value $R = 0\text{x}5\text{f}3759\text{df}$, the value $R = 0\text{x}5\text{f}3759\text{df}$ did occur in the lightly-weighted bisecting of $\sigma = 0$ and $\sigma = \tau$. This tends to suggest that the value might have been derived at by bisecting to a desired tolerance, but without further information about the original development, seemingly, only speculation remains as to explain the unique value—possibly derived at by a predefined limit for the allowable maximum relative error, constrained by parameters based upon aspects of an application, or, even from unusual starting points used in one of the various methods, such as bisecting.