Deriving formulas used in fastmath.c

This document derives the formulas to approximate floating-point operations, as well as their Newton–Raphson method.

Based on the famous "Fast inverse square root" algorithm.

Additionally, the following resources helped me a great deal:

Lomont, Chris (February 2003). "Fast Inverse Square Root"

YouTube - @Nemean: "Fast Inverse Square Root — A Quake III Algorithm"

Float 32

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Sign bit
- Exponent
- Mantissa

Bit representation = $2^{\mathrm{M_{bits}}} * \mathrm{Exp} + \mathrm{Man.}$

Value = $\left(1 + \frac{\mathrm{Man.}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\mathrm{Exp} - \mathrm{E_{bias}}}$

Note: The mantissa and exponent are treated as unsigned integers.

The exponent needs to represent negative values for number less than one (e.g., $2^{-2} = 0.25$). Instead of using a signed bit/2's complement, a float's exponent is biased so the first half of values are negative.

Taking the binary log of a float

$$\log_2(\mathrm{float}) = \log_2\left(\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\mathrm{Exp} - \mathrm{E_{bias}}}\right)$$

$$= \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \log_2\left(2^{\mathrm{Exp} - \mathrm{E_{bias}}}\right)$$

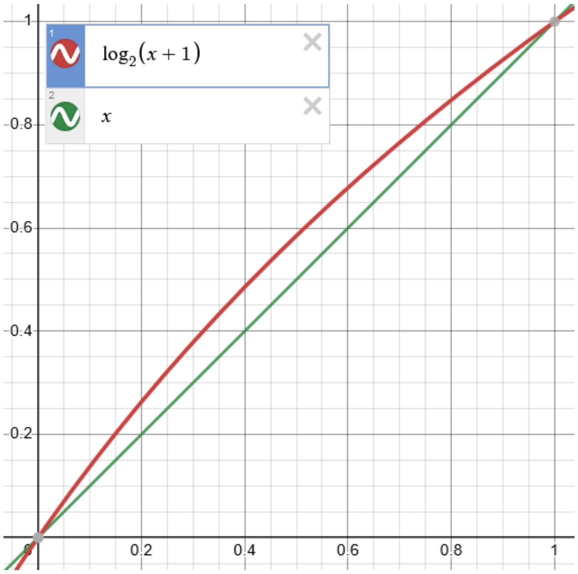
$$= \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}$$

$$\frac{\mathrm{bit\_repr}}{2^{\mathrm{M_{bits}}}} - \mathrm{E_{bias}} = \mathrm{Exp.Man} - \mathrm{E_{bias}}$$

$$= \lfloor \log_2(\mathrm{x}) \rfloor + \frac{1}{2} - \frac{1}{\pi} * \tan^{-1}\left(\cot\left(\pi * x * 2^{1 - \lceil \log_2(\mathrm{x}) \rceil}\right)\right)$$

('x' being the input float's value)

Approximation for $\log_2(x+1),\ x \in \{0,1\}$



As you can see, $\log_2(x+1) \approx x$, notably they are the same at x=0 and 1. We can improve this approximation by reducing the absolute difference between them, so on average they are closer.

$$\int_0^1 (\log_2(x+1) - x)\,dx \approx 0.0573$$

This is the term that will minimize the average difference between our two functions, so our better approximation will be:

$$\log_2(x+1) \approx x + \mu = x + 0.0573$$

Using approximation to remove logarithm.

We can use that approximation to remove the logarithm in our function. Let the error correction term be, $\mu = 0.0573$

$$= \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}$$

$$\approx \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}} + \mu + \mathrm{Exp} - \mathrm{E_{bias}}$$

$$\approx \frac{\mathrm{Man} + \mathrm{Exp} * 2^{\mathrm{M_{bits}}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}}$$

$$\approx \frac{1}{2^{\mathrm{M_{bits}}}} * (\mathrm{Man} + \mathrm{Exp} * 2^{\mathrm{M_{bits}}}) + \mu - \mathrm{E_{bias}}$$

$$\approx \frac{1}{2^{\mathrm{M_{bits}}}} * \mathrm{bit\_repr} + \mu - \mathrm{E_{bias}}$$ {Equal to our bit representation}

We can use this log2 approximation to calculate its inverse square root using some log laws.

$$\log\left(\frac{1}{\sqrt{n}}\right) = \log\left(\mathrm{n}^{-\frac{1}{2}}\right) = -\frac{1}{2} * \log(\mathrm{n})$$

Let $\mathbf{\Gamma}$ be our solution. $(1/\sqrt{y})$

$$\log_2(\mathbf{\Gamma}) = -\frac{1}{2}\log_2(y)$$

Using our log2 approximation of a floating-point number, we get:

$$\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{\Gamma}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}} = -\frac{1}{2}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit\_repr}} + \mu - \mathrm{E_{bias}}\right)$$

Solving for the bit representation of $\mathbf{\Gamma}$.

$$\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{\Gamma}_{\mathrm{bit_{repr}}} = -\frac{1}{2}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit\_repr}} + \mu - \mathrm{E_{bias}}\right) + \mathrm{E_{bias}} - \mu$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \left(-\frac{1}{2}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}}\right) + \mathrm{E_{bias}} - \mu\right) * 2^{\mathrm{M_{bits}}}$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \left(-\frac{\mathbf{y}_{\mathrm{bit_{repr}}}}{2 * 2^{\mathrm{M_{bits}}}} - \frac{\mu}{2} + \frac{\mathrm{E_{bias}}}{2} + \mathrm{E_{bias}} - \mu\right) * 2^{\mathrm{M_{bits}}}$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \left(-\frac{\mathbf{y}_{\mathrm{bit_{repr}}}}{2 * 2^{\mathrm{M_{bits}}}} + \frac{\mathrm{E_{bias}} + 2 * \mathrm{E_{bias}} - \mu - 2\mu}{2}\right) * 2^{\mathrm{M_{bits}}}$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \left(-\frac{\mathbf{y}_{\mathrm{bit_{repr}}}}{2 * 2^{\mathrm{M_{bits}}}} + \frac{3 * \mathrm{E_{bias}} - 3\mu}{2}\right) * 2^{\mathrm{M_{bits}}}$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \left(-\frac{\mathbf{y}_{\mathrm{bit_{repr}}}}{2 * 2^{\mathrm{M_{bits}}}} + \frac{3}{2} * (\mathrm{E_{bias}} - \mu)\right) * 2^{\mathrm{M_{bits}}}$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = -\frac{\mathbf{y}_{\mathrm{bit_{repr}}}}{2} + \frac{3}{2} * 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$$

$$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} = \frac{3}{2} * 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu) - \frac{1}{2} * \mathbf{y}_{\mathrm{bit_{repr}}}$$

The less half of the equation is a constant, which in the original Quake 3 algorithm is the following:

$$\mathrm{0x5f3759df} - (\mathbf{y} \gg 1) \leftrightarrow \frac{3}{2} * 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu) - \frac{1}{2} * \mathbf{y}_{\mathrm{bit\ repr}}$$

(Bit shifting a binary value to the right is equivalent to dividing by 2^n, n being the distance shifted.)

This approximation is pretty good, but we can improve it even further by using Newton's method.

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

This equation recursively finds an 'x', in which $f(x) \to 0$. In our case, we want to minimize the difference between our approximation and the actual value.

On way to measure how accurate an approximation is to a function, is to put that approximation through the inverse of that function and measure how well it matches the original input.

$$f(approx) = \frac{1}{approx^2} - input$$

Using this expression, we can create a Newton's method iteration. But first we need its derivative.

$$f'(approx) = \frac{dy}{d(approx)}(approx^{-2} - input)$$

$$f'(approx) = -2 * approx^{-3}$$

Plugging these into Newton's method, we get:

$$approx_{new} = approx - \frac{\frac{1}{approx^2} - input}{\frac{-2}{approx^3}}$$

$$approx_{new} = approx - \frac{\left(\frac{1}{approx^2} - input\right) * approx^3}{-2}$$

$$approx_{new} = approx - \frac{\left(\frac{1 - approx^2 * input}{approx^2}\right) * approx^3}{-2}$$

$$approx_{new} = approx - \frac{approx - approx^3 * input}{-2}$$

$$approx_{new} = \frac{2 * approx + approx}{2} - \frac{input * approx^3}{2}$$

$$approx_{new} = approx * \frac{3}{2} - \frac{1}{2} * input * approx^3$$

$$approx_{new} = approx * \left(\frac{3}{2} - \frac{1}{2} * input * approx * approx\right)$$

This is equivalent to the line:

$$y \;=\; y \;*\; (\,threehalfs \;-\; (\;0.5F \;*\; number \;*\; y \;*\; y\,)\,)$$

Exponentiation to any power

$$\frac{1}{2^{\mathrm{M_{bits}}}} * \boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \mathrm{pow}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit\_repr}} + \mu - \mathrm{E_{bias}}\right) + \mathrm{E_{bias}} - \mu$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \left(\mathrm{pow}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}}\right) + \mathrm{E_{bias}} - \mu\right) * 2^{\mathrm{M_{bits}}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \mathrm{pow} * 2^{\mathrm{M_{bits}}}\left(\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{y}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}}\right) + 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}} - 2^{\mathrm{M_{bits}}} * \mu$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \mathrm{pow} * \mathbf{y}_{\mathrm{bit_{repr}}} + 2^{\mathrm{M_{bits}}} * \mu * \mathrm{pow} - \mathrm{pow} * 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}} + 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}} - 2^{\mathrm{M_{bits}}} * \mu$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \mathrm{pow} * \mathbf{y}_{\mathrm{bit_{repr}}} + 2^{\mathrm{M_{bits}}} * \mu * (\mathrm{pow} - 1) - 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}} * (\mathrm{pow} - 1)$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} = \mathrm{pow} * \mathbf{y}_{\mathrm{bit_{repr}}} + 2^{\mathrm{M_{bits}}} * (\mathrm{pow} - 1) * (\mu - \mathrm{E_{bias}})$$

Newton’s method for any power

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

$$f(approx) = approx^{\frac{1}{\mathrm{pow}}} - input$$

$$f'(approx) = \frac{dy}{d(approx)}\left(approx^{\frac{1}{\mathrm{pow}}} - input\right)$$

$$f'(approx) = \frac{1}{\mathrm{pow}} * approx^{\frac{1}{\mathrm{pow}}-1}$$

$$\mathrm{approx}_{new} = \mathrm{approx} - \frac{\mathrm{approx}^{\frac{1}{\mathrm{pow}}} - input}{\frac{1}{\mathrm{pow}} * \mathrm{approx}^{\frac{1}{\mathrm{pow}}-1}}$$

$$\mathrm{approx}_{new} = \mathrm{approx} - \mathrm{pow} * \mathrm{approx} * \frac{\mathrm{approx}^{\frac{1}{\mathrm{pow}}} - input}{\mathrm{approx}^{\frac{1}{\mathrm{pow}}}}$$

$$\mathrm{approx}_{new} = \mathrm{approx} * \left(1 - \mathrm{pow} * \left(1 - \frac{input}{\mathrm{approx}^{\frac{1}{\mathrm{pow}}}}\right)\right)$$

This requires using an exponential, which is what this algorithm is trying to avoid.

General forms for $\frac{1}{\sqrt[n]{x}}$

Float approximation

$$\boldsymbol{\Gamma}_{\mathrm{bit}_{\mathrm{repr}}} = \frac{1}{n}\left((n+1) * 2^{\mathrm{M}_{\mathrm{bits}}} * (\mathrm{E}_{\mathrm{bias}} - \mu) - \mathbf{y}_{\mathrm{bit}_{\mathrm{repr}}}\right)$$

Newton's method

$$approx_{new} = approx * \left(1 + \frac{1}{n} * (1 - input * approx^{n})\right)$$

Log2 Newton's method

$$f(approx) = 2^{approx} - \mathrm{input}$$

$$f'(approx) = 2^{approx} * \ln(2)$$

$$\mathrm{approx}_{new} = \mathrm{approx} - \frac{2^{approx} - \mathrm{input}}{2^{approx} * \ln(2)}$$

$$\mathrm{approx}_{new} = \mathrm{approx} - \frac{1}{\ln(2)} * \left(1 - \frac{\mathrm{input}}{2^{approx}}\right)$$

Reciprocal

Float approximation

$$\boldsymbol{\Gamma}_{\mathrm{bit}_{\mathrm{repr}}} = (-1) * \mathbf{y}_{\mathrm{bit}_{\mathrm{repr}}} + 2^{\mathrm{M}_{\mathrm{bits}}} * (-1 - 1) * (\mu - \mathrm{E}_{\mathrm{bias}})$$

$$\boldsymbol{\Gamma}_{\mathrm{bit}_{\mathrm{repr}}} = -2^{\mathrm{M}_{\mathrm{bits}}} * 2 * (\mu - \mathrm{E}_{\mathrm{bias}}) - \mathbf{y}_{\mathrm{bit}_{\mathrm{repr}}}$$

Newton's method

$$\mathrm{approx}_{new} = \mathrm{approx} * \left(1 - (-1) * \left(1 - \frac{input}{\mathrm{approx}^{\frac{1}{-1}}}\right)\right)$$

$$\mathrm{approx}_{new} = \mathrm{approx} * \left(1 + (1 - input * \mathrm{approx})\right)$$

$$\mathrm{approx}_{new} = \mathrm{approx} * (2 - input * \mathrm{approx})$$

Appendix

Natural log

$$\ln(\text{float}) = \ln\left(1 + \frac{\text{Man}}{2^{\text{M}_{\text{bits}}}}\right) + \ln\left(2^{\text{Exp}-\text{E}_{\text{bias}}}\right)$$

$$\ln(\text{float}) = \ln\left(1 + \frac{\text{Man}}{2^{\text{M}_{\text{bits}}}}\right) + (\text{Exp} - \text{E}_{\text{bias}}) * \ln(2)$$

$$\ln(\text{float}) \approx \ln(2) * \frac{\text{Man}}{2^{\text{M}_{\text{bits}}}} + \mu + (\text{Exp} - \text{E}_{\text{bias}}) * \ln(2) \qquad \{\mu = 0.03972\}$$

$$\ln(\text{float}) \approx \ln(2)\left(\frac{\text{Man}}{2^{\text{M}_{\text{bits}}}} + \text{Exp} - \text{E}_{\text{bias}}\right) + \mu$$

$$\ln(\text{float}) \approx \ln(2)\left(\frac{\text{Man}+\text{Exp}*2^{\text{M}_{\text{bits}}}}{2^{\text{M}_{\text{bits}}}} - \text{E}_{\text{bias}}\right) + \mu$$

$$\ln(\text{float}) \approx \ln(2)\left(\frac{\text{bit}_{\text{repr}}}{2^{\text{M}_{\text{bits}}}} - \text{E}_{\text{bias}}\right) + \mu$$

Log gamma function approximations

Lanczos (very accurate)

$$\ln(\mathbf{x}!) \approx \frac{1}{2}\ln(2\pi) + \left(\boldsymbol{x} + \frac{1}{2}\right) * \ln\left(\mathbf{x} + \text{A} + \frac{1}{2}\right) - \left(\boldsymbol{x} + \text{A} + \frac{1}{2}\right) \qquad A = 0.3$$

Stirling's

$$\ln(\mathbf{float}!) \approx \frac{1}{2}\ln(2\pi) + \left(\boldsymbol{x} + \frac{1}{2}\right) * \ln(\mathbf{Z}) - \mathbf{Z} \qquad \mathbf{Z} = \mathbf{x} + \text{A} + \frac{1}{2}$$

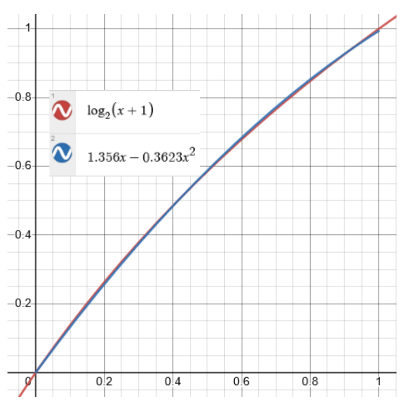
Binary log for x: $(0,1)$

Used for the fractional part in the *fastLog2Bits()* and in *fastLog2Alt()*.

$$\log_2(\text{x} + 1) \approx 1.356x - 0.3623x^2 \qquad \{0 < x < 1\}$$

$$\log_2(\text{x} + 1) \approx 1.4088x - 0.49328x^2 + 0.086x^4 \qquad \{0 < x < 1\}$$

$$\log_2(\text{x} + 1) \approx \frac{3.3739x}{x + 2.3791} \qquad \{0 < x < 1\}$$



Better general power approximation

$\log_2(\mathbf{float}) = \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}$

$$\log_2(\mathbf{float}^{\mathrm{pow}}) = \mathrm{pow}\left(\log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}\right)$$

$$\frac{1}{2^{\mathrm{M_{bits}}}} * \boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}} \approx \mathrm{pow}\left(\log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}\right)$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \left(\mathrm{pow}\left(\log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + \mathrm{Exp} - \mathrm{E_{bias}}\right) - \mu + \mathrm{E_{bias}}\right) * 2^{\mathrm{M_{bits}}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow}\left(2^{\mathrm{M_{bits}}} * \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + 2^{\mathrm{M_{bits}}} * \mathrm{Exp} - 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}}\right) - 2^{\mathrm{M_{bits}}} * \mu + 2^{\mathrm{M_{bits}}} * \mathrm{E_{bias}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow}\left(2^{\mathrm{M_{bits}}} * \log_2\left(1 + \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{Exp} - \mathrm{E_{bias}})\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$$

$\log_2(\mathrm{x} + 1) \approx 1.356x - 0.3623x^2 \quad \{0 < x < 1\}$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow}\left(2^{\mathrm{M_{bits}}} * \frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\left(1.356 - 0.3623\frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{Exp} - \mathrm{E_{bias}})\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow}\left(\mathrm{Man} * \left(1.356 - 0.3623\frac{\mathrm{Man}}{2^{\mathrm{M_{bits}}}}\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{Exp} - \mathrm{E_{bias}})\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow}\left(\mathrm{Man} * \left(1.356 - \left(\frac{0.3623}{2^{\mathrm{M_{bits}}}}\right) * \mathrm{Man}\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{Exp} - \mathrm{E_{bias}})\right) + 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$$

“Fast” multiplication float approximation

$$\mathbf{float}_{\mathrm{value}} = \left(1 + \frac{\mathrm{Man.}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\mathrm{Exp}-\mathrm{E_{bias}}}$$

$$\log_2(\mathbf{a} * \mathbf{b}) = \log_2\left(\left(1 + \frac{\mathbf{a}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\boldsymbol{a}_{\mathrm{Exp}}-\mathrm{E_{bias}}} * \left(1 + \frac{\mathbf{b}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\boldsymbol{b}_{\mathrm{Exp}}-\mathrm{E_{bias}}}\right)$$

$$\log_2(\mathbf{a} * \mathbf{b}) = \log_2\left(\left(1 + \frac{\mathbf{a}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\boldsymbol{a}_{\mathrm{Exp}}-\mathrm{E_{bias}}}\right) + \log_2\left(\left(1 + \frac{\mathbf{b}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) * 2^{\boldsymbol{b}_{\mathrm{Exp}}-\mathrm{E_{bias}}}\right)$$

$$\log_2(\mathbf{a} * \mathbf{b}) = \log_2\left(1 + \frac{\mathbf{a}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) + \boldsymbol{a}_{\mathrm{Exp}} - \mathrm{E_{bias}} + \log_2\left(1 + \frac{\mathbf{b}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}}\right) + \boldsymbol{b}_{\mathrm{Exp}} - \mathrm{E_{bias}}$$

$$\log_2(\mathbf{a} * \mathbf{b}) \approx \frac{\mathbf{a}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}} + \mu + \boldsymbol{a}_{\mathrm{Exp}} - \mathrm{E_{bias}} + \frac{\mathbf{b}_{\mathrm{Man}}}{2^{\mathrm{M_{bits}}}} + \mu + \boldsymbol{b}_{\mathrm{Exp}} - \mathrm{E_{bias}}$$

$$\log_2(\mathbf{a} * \mathbf{b}) \approx \frac{\mathbf{a}_{\mathrm{bit\_repr}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}} + \frac{\mathbf{b}_{\mathrm{bit\_repr}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}}$$

Let $\boldsymbol{\Gamma}$ be our solution.

$$\frac{1}{2^{\mathrm{M_{bits}}}} * \boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}} \approx \frac{\mathbf{a}_{\mathrm{bit\_repr}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}} + \frac{\mathbf{b}_{\mathrm{bit\_repr}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \left(\frac{\mathbf{a}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}} + \frac{\mathbf{b}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}} - \mu + \mathrm{E_{bias}}\right) * 2^{\mathrm{M_{bits}}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathbf{a}_{\mathrm{bit_{repr}}} + \mathbf{b}_{\mathrm{bit_{repr}}} + 2^{\mathrm{M_{bits}}} * (\mu - \mathrm{E_{bias}})$$

Division

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \left(\frac{\mathbf{a}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} + \mu - \mathrm{E_{bias}} - \frac{\mathbf{b}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} - \mu + \mathrm{E_{bias}} - \mu + \mathrm{E_{bias}}\right) * 2^{\mathrm{M_{bits}}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \left(\frac{\mathbf{a}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} - \frac{\mathbf{b}_{\mathrm{bit_{repr}}}}{2^{\mathrm{M_{bits}}}} - \mu + \mathrm{E_{bias}}\right) * 2^{\mathrm{M_{bits}}}$$

$$\boldsymbol{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathbf{a}_{\mathrm{bit_{repr}}} - \mathbf{b}_{\mathrm{bit_{repr}}} - 2^{\mathrm{M_{bits}}} * (\mu - \mathrm{E_{bias}})$$

Constant raised to a variable power

Float approximation for 2^{n}

$\log_2(\mathbf{\Gamma}) = \log_2(2^{\mathrm{pow}})$

$\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{\Gamma}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}} \approx \mathrm{pow}$

$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} \approx (\mathrm{pow} - \mu + \mathrm{E_{bias}}) * 2^{\mathrm{M_{bits}}}$

Float approximation for e^{n}

$\frac{1}{2^{\mathrm{M_{bits}}}} * \mathbf{\Gamma}_{\mathrm{bit_{repr}}} + \mu - \mathrm{E_{bias}} \approx \mathrm{pow} * \log_2(\mathrm{e})$

$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} \approx (\mathrm{pow} * \log_2(\mathrm{e}) - \mu + \mathrm{E_{bias}}) * 2^{\mathrm{M_{bits}}}$

$\mathbf{\Gamma}_{\mathrm{bit_{repr}}} \approx \mathrm{pow} * \log_2(\mathrm{e}) * 2^{\mathrm{M_{bits}}} + 2^{\mathrm{M_{bits}}} * (\mathrm{E_{bias}} - \mu)$

Use 15 for x: $(0, 1)$

$$\ln(x+1) = 2 * \sum_{n=1}^{\infty}\left(\frac{1}{2n-1} * \left(\frac{x}{x+2}\right)^{2n-1}\right)$$

$2\left(\frac{1}{1} * \frac{x}{x+2} + \frac{1}{3} * \frac{x}{x+2} * \frac{x}{x+2} * \frac{x}{x+2} + \frac{1}{5} * \frac{x}{x+2} * \frac{x}{x+2} * \frac{x}{x+2} * \frac{x}{x+2} * \frac{x}{x+2} \ldots\right)$

$= \quad \frac{1}{1} * c \qquad c = \frac{x}{x+2}$

$+= \frac{1}{3} * c \qquad c *= \frac{x}{x+2} * \frac{x}{x+2}$

$+= \frac{1}{5} * c \qquad c *= \frac{x}{x+2} * \frac{x}{x+2}$

$+= \frac{1}{7} * c \qquad c *= \frac{x}{x+2} * \frac{x}{x+2}$