FP Function Approximations for log₂(), 2ˣ, & powf(x, r) Routines

First a review of how a single precision (32-bit) floating point number is represented in memory:

<table>
<tr><td colspan="32">Bit Position</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>s_x</td><td colspan="8">$e_x = e[7:0] = \sum_{i=0}^{7} e[i] \cdot 2^i$</td><td colspan="23">$m_x = m[22:0] = \sum_{i=0}^{22} m[i] \cdot 2^i$</td></tr>
<tr><td colspan="9">Integer Interpretation</td><td colspan="23">$X = s_x \cdot 2^{31} + e_x \cdot 2^{23} + m_x$</td></tr>
<tr><td colspan="9">Floating Point Interpretation</td><td colspan="23">$x = (-1)^s \left(1 + \frac{m_x}{2^{23}}\right) \cdot 2^{e_x - 127} = (-1)^s \cdot m'_x 2^{e'_x}$
$m'_x = \left(1 + \frac{m_x}{2^{23}}\right), \quad e'_x = (e_x - 127), \quad 1 \le m'_x < 2, \quad -127 \le e'_x \le 128$</td></tr>
<tr><td colspan="9">Hexadecimal Floating-Point Representation</td><td colspan="23">Is a compact method of representing floating point values that prevents improper decimal truncation. Supported by C99 spec.

$\pi = 3.1415927410125732421875 = 0x3.243F6Cp0$
$e^1 = 2.71828174591064453125 = 0x2.B7E15p0$
$\frac{1}{\sqrt{2 \cdot \pi}} = 0.398942291736602783203125 =$ 0x6.62115p-4</td></tr>
</table>

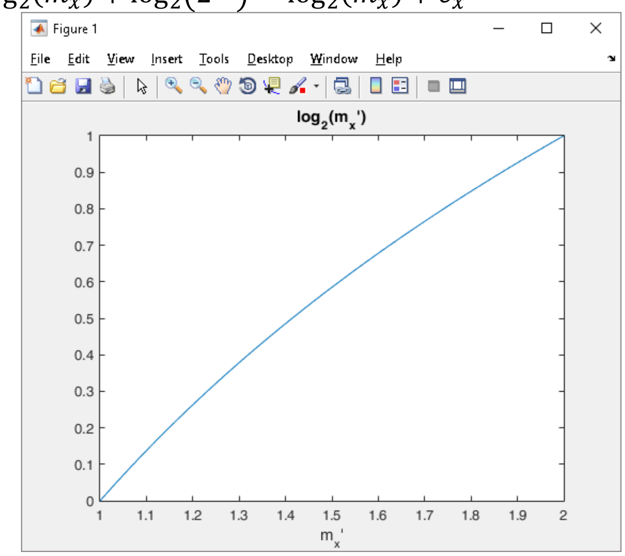
There are a couple caveats to representing particular numbers:

x	s_x	e_x	m_x
0	X	0	0
$+\infty$	0	255	0
$-\infty$	1	255	0

<u>Approximating the logarithm base 2 function</u>

$$y = \log_2|x|, \qquad y = \log_2\left(m'_x \cdot 2^{e'_x}\right) = \log_2(m'_x) + \log_2\left(2^{e'_x}\right) = \log_2(m'_x) + e'_x$$

Calculating the function approximation via range reduction allows for much higher precision polynomial fit over such a short interval. (i.e. $1 \le m'_x < 2$)

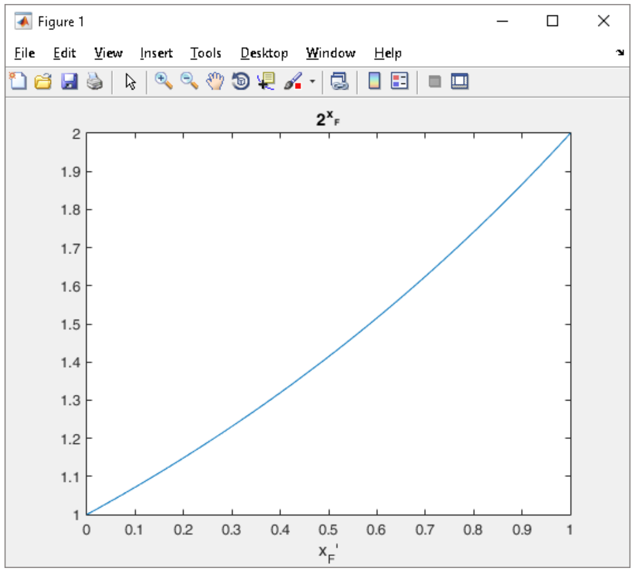


Using an open source Linux package Sollya to calculate a bounded N-th order polynomial

FP Function Approximations for $\log_2()$, 2^x, & powf(x, r) Routines

Approximating the exponent base 2 function

$$y = 2^x, \quad m'_y \cdot 2^{e'_y} = 2^x = 2^{x_I + x_F}, \; where\; x_I = \lfloor x \rfloor, \quad x_F = x - \lfloor x \rfloor \; then$$
$$m'_y \cdot 2^{e'_y} = 2^{x_I + x_F} = 2^{x_F} \cdot 2^{x_I}, \; e'_y = x_I, \; m'_y = 2^{x_F}, \; 0 \le x_F < 1, \; 1 \le m'_y < 2$$



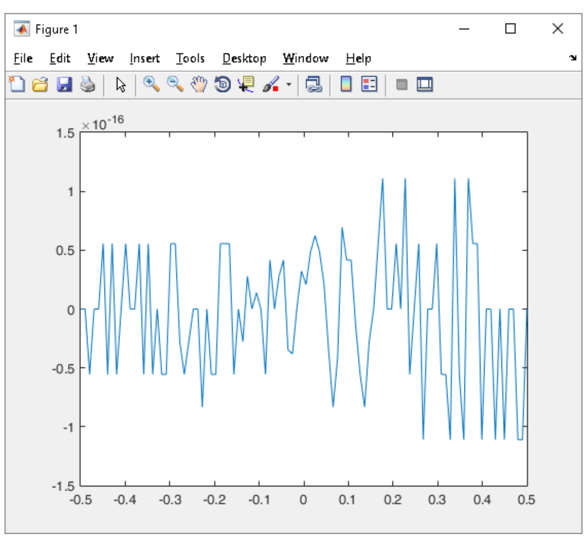
It's important to note that the approximation nodes must be exact at 0 & 1 which requires a hand tuned polynomial approximation using Netwon's Divided Difference or Lagragange Interpolation method to force both end-points to be exact. **Caution** this method has an inherent flaw, the final error from the bounded polynomial grows with the size of the input x exponentially. The only way to counter this would be to use a pade approximation such as the one used in the boost library:

$$y = x\left(c_0 + \frac{c_{n0} + c_{n1}\cdot x^1 + c_{n2}\cdot x^2 + c_{n3}\cdot x^3 + c_{n4}\cdot x^4 + c_{n5}\cdot x^5}{c_{d0} + c_{d1}\cdot x^1 + c_{d2}\cdot x^2 + c_{d3}\cdot x^3 + c_{d4}\cdot x^4 + c_{d5}\cdot x^5}\right), \qquad -\frac{1}{2} < x < \frac{1}{2}$$

$c_0 = 0.10281276702880859e1$

$c_n = \begin{Bmatrix} \text{-0.28127670288085937e-1} & \text{0.51278186299064534e0} & \text{-0.63100290693501 98e-1} & \text{0.11638457975729296e-1} \\ & \text{-0.52143390687521003e-3} & \text{0.21491399776965688e-4} & \end{Bmatrix}$

$c_d = \begin{Bmatrix} \text{1.0} & \text{-0.45442309511354755e0} & \text{0.90850389570911714e-1} & \text{-0.10088963629815502e-1} \\ & \text{0.63003407478692265e-3} & \text{-0.17976570003654402e-4} & \end{Bmatrix}$



This pade approximation introduces increased complexity as well as division which is bad news bears for decreasing functional latency. It is also not necessary as we will see the accuracy is adequate if we restrict the input $0 < x \le 1$

FP Function Approximations for $\log_2()$, 2^x, & powf(x, r) Routines

Approximating the power function

$$y = x^p, \qquad \log_2(y) = \log_2(x^p) = p \cdot \log_2(x), \qquad y = 2^{p \cdot \log_2(x)}, \qquad for\ \ 0 < x \leq 1, \qquad -\infty < p \cdot \log_2(x) \leq 0$$

So the exponent will always be negative. A special restriction must be applied to prevent $\log_2(0)$ and $-127 \leq p \cdot \log_2(x)$. Below is functional error for all combinations of possible ordered polynomials for both the $\exp 2(x)$ and $log2(x)$.