Fast inverse square root

Nicola Seriani

The Abdus Salam International Centre for Theoretical Physics,
Strada Costiera 11, 34151 Trieste, Italy

Introduction

- We are going to look at a significative example of how to use mathematical tricks and the binary representation of floating point numbers to get a fast evaluation of a function, at a reasonable accuracy
- We are going to approximate $f(x) = x^{-1/2} = 1/\sqrt{x}$
- The trick was developed in the computer graphics industry in the 90's: light reflection of objects
- It is 3-4 times faster than evaluation through square root evaluation

Algorithm overview

- Input: 32-bit floating point number x
- The binary expression of x is interpreted as a (long) integer i
- i is shifted to the right by one position (adding a 0 in the first place of the binary representation)
- i is subtracted from the ‘magic number’ 0x5f3759df
- The resulting binary is re-interpreted as a floating number -> this is our first approximation for $x^{-1/2}$ (usually the error is a few percentage points)
- We refine it by one iteration of the Newton-Raphson method

Algorithm overview

- Input: 32-bit floating point number x
- i = *(long *)&x
- i = i>>1
- i = 0x5f3759df – i
- y = *(float *)&i
- And then Newton-Raphson

First step: representation of the floating point

- 32-bit floating point number x is written as
- $x = \pm 2^{ex}(1+m_x)$, e_x is an integer, $0 \leq m_x < 1$

- Three integers are computed from this:

① S_x is 0 if x>0, 1 if x<0 (1 bit)

② $E_x = e_x + B$ (B = 127, exponent bias) (8 bits)

③ $M_x = m_x * L$, where $L = 2^{23}$ (23 bits)

- These number are then packed in the 32 bits

First element: representation of the floating point

Three integers are computed from this:

① S_x is 0 if x>0, 1 if x<0 (1 bit)

② $E_x = e_x + B$ (B = 127, exponent bias) (8 bits)

③ $M_x = m_x * L$, where $L = 2^{23}$ (23 bits)

- These number are then packed in the 32 bits
- Example: $x = 0.15625 = +2^{-3}(1+0.25)$
- S = 0
- $E = -3 + 127 = 124 = 01111100_2$
- $M = 0.25*2^{23} = 2097152 = 01000000000000000000000_2$
- So the representation is

0 01111100 01000000000000000000000

Second element: aliasing to an integer as an approximation to the logarithm

The main element of the algorithm to calculate $1/\sqrt{x}$ is the relation $\log_2(1/\sqrt{x}) = -1/2 \log_2(x)$; basically the integer operations are devised to yield an approximation for the logarithm of x: $x = 2^{ex}(1+m_x)$

$\log_2(x) = e_x + \log_2(1+m_x) \approx e_x + m_x + \sigma$ (σ free parameter)

$$\log_2(x) \approx e_x + m_x + \sigma$$

On the other hand, interpreting the bit sequence as integer

$I_x = E_x * L + M_x = L(e_x+B+m_x) \approx L \log_2(x) + L (B - \sigma)$

$$\log_2(x) \approx I_x/L - (B - \sigma)$$

Second element: aliasing to an integer as an approximation to the logarithm

Defining $y = 1/\sqrt{x}$, the equation reads

$$\log_2(y) = -1/2 \log_2(x)$$

We can now use the expression derived before on both sides:

$$\mathbf{\log_2(x) \approx I_x/L - (B - \sigma)}$$

$$\mathbf{\log_2(y) \approx I_y/L - (B - \sigma)}$$

The expression becomes

$$I_y/L - (B - \sigma) \approx -1/2 \{I_x/L - (B - \sigma)\}$$

or

$$I_y \approx 3/2\ L\ (B - \sigma) - ½\ I_x$$

Second element: aliasing to an integer as an approximation to the logarithm

$$I_y \approx 3/2\ L\ (B - \sigma) - \tfrac{1}{2}\ I_x$$

This is exactly what is calculated in the algorithm:

```
i = 0x5f3759df – (i>>1)
```

with $\sigma \approx 0.0450466$

hexadecimal constant 0x5f3759df

Hexadecimal constant

Base 16

$\sigma \approx 0.0450466$ (base 10)

hexadecimal constant 0x5f3759df

Remember:

$0_{hex} = 0_{dec}$; $1_{hex} = 1_{dec}$

$2_{hex} = 2_{dec}$; $3_{hex} = 3_{dec}$

$4_{hex} = 4_{dec}$; $5_{hex} = 5_{dec}$

$6_{hex} = 6_{dec}$; $7_{hex} = 7_{dec}$

$8_{hex} = 8_{dec}$; $9_{hex} = 9_{dec}$

$a_{hex} = 10_{dec}$; $b_{hex} = 11_{dec}$

$c_{hex} = 12_{dec}$; $d_{hex} = 13_{dec}$

$e_{hex} = 14_{dec}$; $f_{hex} = 15_{dec}$

The choice of σ

This is exactly what is calculated in the algorithm:

$$i = 0x5f3759df - (i>>1)$$

with $\sigma \approx 0.0450466$

Remember, σ was chosen to appropriately approximate the logarithm:

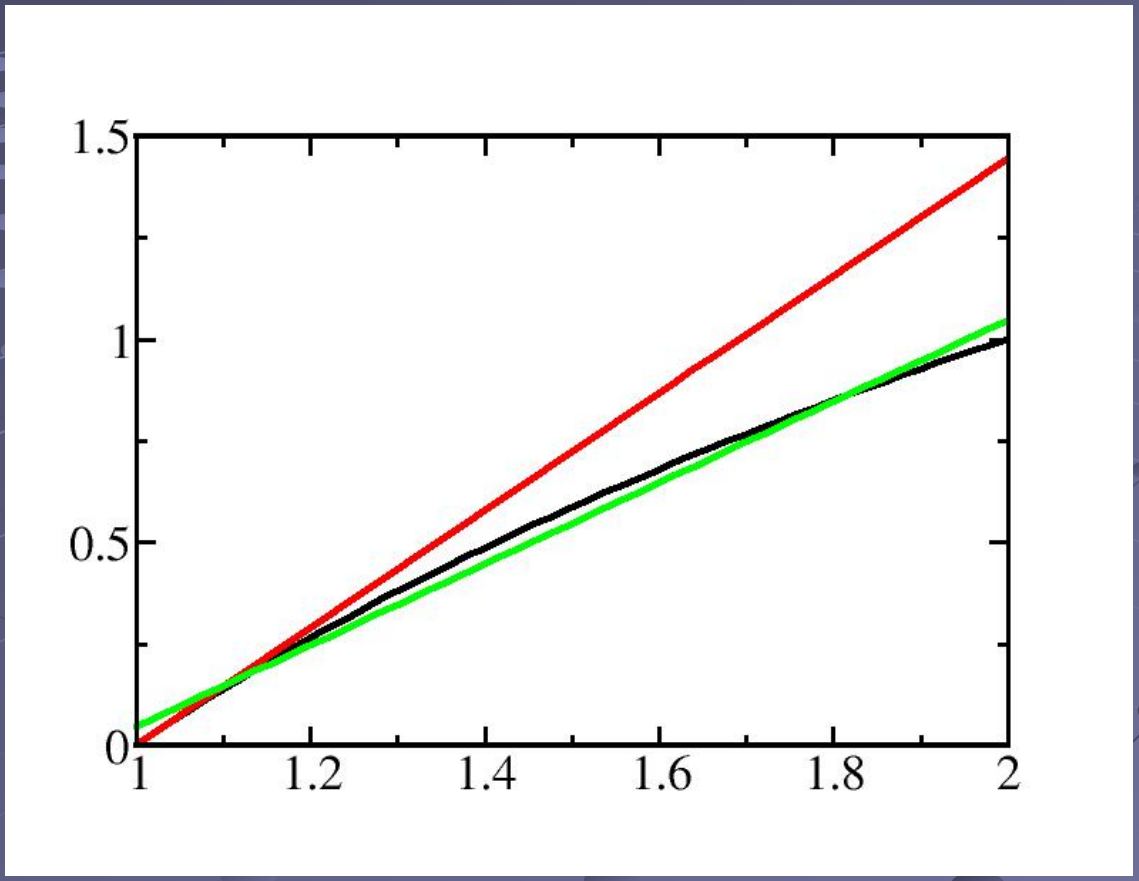
$\log_2(1+m_x) \approx m_x + \sigma$ (σ free parameter)

Taylor expansion to first order:

$\log_2(1+m) \approx \log_2(1) + [\log_2(1+m)]'|_{m=0} * m = m/\ln(2)$

$\ln(2) = 0.6931$; $1/\ln(2) = 1.4427$

The choice of σ



Plot of $\log_2(x)$, $(x-1)/\ln(2)$, and $(x-1)+0.0450466$

The choice of σ

Plot of $\log_2(1+x)$, $x/\ln(2)$, and $x+0.0450466$

Not clear where the constant comes from, maybe trial and error

Minimizing Integral[$(\log_2(x) – x + 1 – A)^2$] with respect to A, I obtain A = 0.05730495

Refining: Newton-Raphson method

We can use one or two NR iterations to refine the results.
Remember: $y = 1/\sqrt{x}$, so function for which to find the zero is:

$f(y) = 1/y^2 - x$

$f'(y) = -2/y^3$

and $y_{n+1} = y_n - f(y_n)/f'(y_n)$