Note Title 13-09-2012

Sept. 14

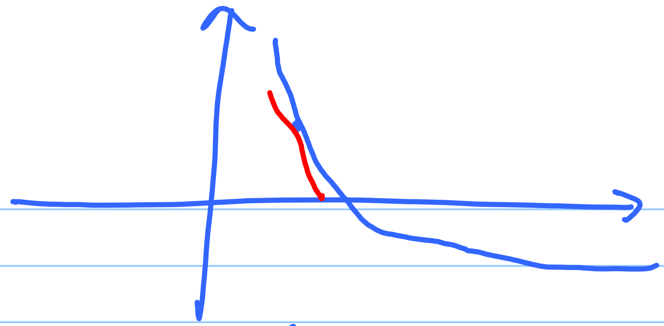
Floating Point Division [$O(\log(n))$]

- Much faster than integer division. [$O(n\log(n))$]

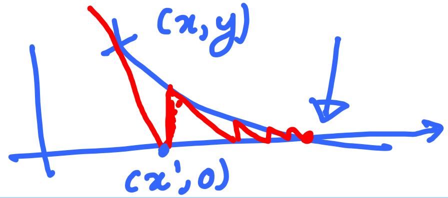
1) Newton - Raphson method.

$$\frac{a}{b} = a \times \left(\frac{1}{b}\right)$$

The main challenge is to compute $\left(\frac{1}{b}\right)$



$(b>0)$



$$f(x) = \frac{1}{x} - b \qquad \{ f(x) = 0 \ , \ x = 1/b \}$$

$$\frac{y-0}{x-x'} = f'(x) = \frac{-1}{x^2}$$

$$\Rightarrow -x^2 y = x - x'$$

$$\Rightarrow -x^2\left(\frac{1}{x} - b\right) = -x + bx^2 = x - x'$$

$$\Rightarrow x' = 2x - bx^2$$

Error function : $E(x) = bx - 1$

$$E\left(\frac{1}{b}\right) = 0$$

At the root (reciprocal), the error is equal to zero

$$E(x) = bx - 1$$

$$E(x') = b(2x - bx^2) - 1$$

$$= 2bx - b^2x^2 - 1$$
$$= -(bx - 1)^2 = -E(x)^2$$

Conclusion:

In every step, the error gets squared

If we can ensure that for the starting

value of x, $E(x) < 1$

Convergence is guaranteed

Let the first value of x be x_0

$$E(x_0) = bx_0 - 1$$

$$E(x_0) < 1$$

$$\Rightarrow bx_0 - 1 < 1$$

$$\Rightarrow bx_0 < 2$$

$$\frac{1}{1.5 \times 2^{30}} = \frac{1}{1.5} \times 2^{-30}$$

Let us consider the significand part of b

b is of the form $1.\overbrace{-------}^{23}$

$[1 \leq b < 2]$ (ignore the exponent)

$$\Rightarrow x_0 < \frac{2}{b}$$

$b = 1$ $\qquad\qquad$ $b = 2$

$x_0 < 2$ $\qquad\qquad$ $\underline{\underline{x_0}} < 1$

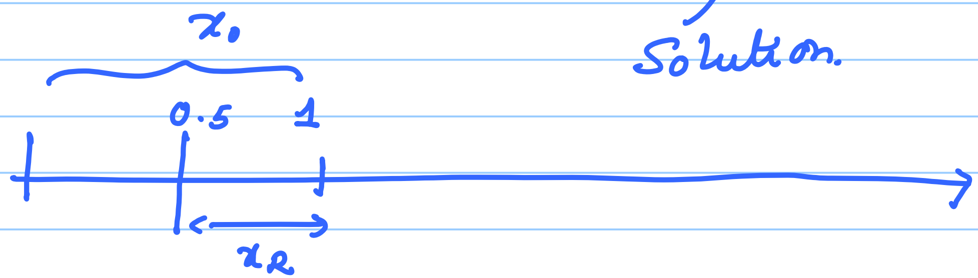
If I take any value of $(x_0 < 1)$ my convergence is guaranteed.

At the root: $\qquad\qquad$ $1 \leq b < 2$

$$b x_R = 1$$

$$[0.5 < x_R \leq 1]$$

Solution.



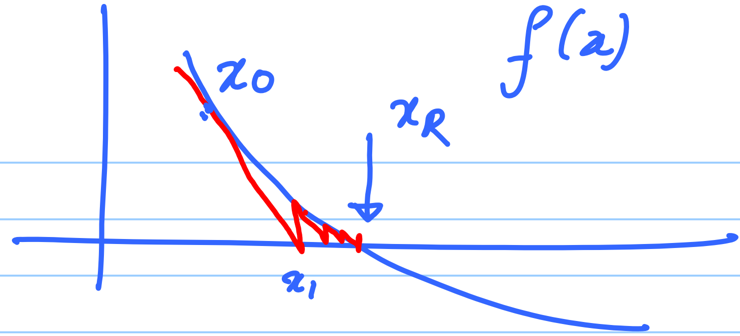
To optimize even further, choose the starting value (x_0) of x to be between 0.5 and 1

Summary

Division is tantamount to computing the reciprocal of the denominator and multiplying it with the numerator $\left(\frac{a}{b}\right)$

$\begin{bmatrix} a > 0 \\ b > 0 \end{bmatrix}$ To compute reciprocal, we define a function,

$$f(x) = \frac{1}{x} - b$$



Results: The process will converge if $x_0 < 1$

How fast is this operation? $O(\log(n))$

Every step: $x' = 2x - bx^2$ $\quad O(\log(n))$

How many steps: constant

Example.

$E(x_0) = 2^{-1}$

$E(x_1) = -2^{-2}$

$E(x_2) = 2^{-4}$

$E(x_3) = -2^{-8}$

$E(x_4) = 2^{-16}$

$E(x_5) = -2^{-32}$

In five steps, the error has become 2^{-32}

When the error is less than the precision the process terminates

In this case, in (5) steps

This algorithm is used in AMD processors.

IBM algorithm

$[a>0, b>0]$ (exponents are 0)

$[1 \le a < 2, 1 \le b < 2]$ (Normal form with no exponents)

$$R = \frac{a}{b}$$

$$b = 2 - y \;\; (y>0) \;\; = 2(1 - y/2) = 2(1-x) \;\; (0 \le x < 1)$$

$$R = \frac{a}{b} = \frac{a/2}{(1-x)} \;\; (0 \le x < 1)$$

$$= \frac{a/2 \; (1+x)}{(1-x)(1+x)} = \frac{a/2 \; (1+x)}{1 - x^2} = \frac{a/2 \, (1+x)(1+x^2)}{(1-x^4)}$$

$$= \frac{a/2 \, (1+x)(1+x^2) \; \cdots \; (1+x^{(2^n)})}{1 - x^{(2^n)}} \quad [x<1]$$

$$R \simeq \frac{a/_2 (1+x) \cdot \cdot (1+x^{(2^n)})}{1}$$

[chain of additions & multiplications]

Because of precision constraints :

$n =$ constant (5)

IBM Algo. $(O(\log(n))$

↑ MidTerm

↓ Next.

Midterm: 2 hrs
End term: 3 hrs.

CSE 40 + EE (20+)

30% Easy ✓, 40% Medium ✓, 20% Hard, 10% Research

ARM