

Cálculo da Raiz Quadrada pelo Método de Newton-Raphson através do Padrão IEEE-754 para Ponto-Flutuante

Ricardo Henrique Brunetto¹

¹Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

ra94182@uem.br

Introdução

Deseja-se obter uma maneira de estimar a raiz quadrada com base no método iterativo de Newton-Raphson, que estimativa de raízes de polinômios e equações transcendentais através da expansão da Série de McLaurin-Taylor.

Tal método permite obter uma precisão específica através de convergência quadrática (se houver).

Este artigo fará uma breve explicação do método de Newton-Raphson e do padrão IEEE 754 de armazenamento de ponto-flutuante que servirão de base para o desenvolvimento da estratégia utilizada. Não serão feitas explanações a respeito da Série de Taylor-McLaurin, visto que foge ao foco do artigo.

Fundamentação Teórica

Aqui serão abordados dois tópicos principais em que se baseará o cálculo da raiz quadrada posteriormente na Seção 3: o método de Newton-Raphson e o padrão IEEE 754.

Método de Newton-Raphson para determinação de raízes

O Método de Newton-Raphson se aplica aos polinômios e às equações transcendentais para determinar soluções de sistemas de equações simultâneas, com várias variáveis. No escopo deste artigo, seu uso será restringido para determinação de raízes em equações polinomiais com uma única variável e, portanto, será feito um breve resumo a respeito de seu funcionamento.

O método de Newton-Raphson se principia com uma raiz aproximada, x_k , de uma equação $f(x) = 0$, e se vale da expansão em série de Taylor para formular um algoritmo computacional que define uma fórmula de recursão para uso.

O desenvolvimento em série de Taylor de uma função nas vizinhanças do ponto x_k tem a forma

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

onde $f'(x_k)$ indica a derivada primeira de $f(x)$ calculada no ponto $(x_k, f(x_k))$.

Dessa forma, dispõe-se da reta tangente ao gráfico da função $f(x)$, traçada no ponto x_k , para determinação da próxima estimativa. Nota-se, que, ao admitir que a função $f(x)$ aproxima-se de sua(s) raiz(es), ocorre $f(x) = 0$ e, portanto, da equação acima:

$$f(x_k) + f'(x_k)(x - x_k) = 0$$

Alternativamente, pode-se expressar tal equação como sendo:

$$f'(x_k) = \frac{f(x_k)}{x_k - x}$$

Admitindo x como a próxima estimativa, sendo esta mais satisfatória que x_k , tem-se a seguinte equação de recorrência, que caracteriza o Método de Newton-Raphson:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (1)$$

Padrão IEEE 754

Aqui será realizada uma curta abordagem ao padrão IEEE 754 para armazenamento e aritmética binária. Não serão realizadas considerações a respeito da quantidade de bits nos diferentes tipos de precisão que o padrão oferece. Nesse caso, o foco torna-se especificamente a maneira como os números são acomodados na máquina para entendimento de como se pode manipular esta estrutura a fim de obter cálculos mais rápidos e precisos.

Qualquer número x pode ser representado por $x = mb^t$, onde m é a mantissa, b é a base e t é o expoente.

Um número decimal x está representado no padrão IEEE 754 da seguinte forma:

$$x_{10} = (-1)^{S_2} (1 + M_{10}) 2^{E_{10} - 1023}$$

onde:

- R_b implica que R está na base b ;
- S é o valor do bit do sinal;
- M é o valor da soma dos bits da mantissa;
- E é o valor do expoente (onde $E \in \mathbb{N}$, pois é envasado);
- 1023 refere-se ao **BIAS** supondo 11 bits de precisão do expoente.

A estrutura permite trabalhar com mantissa M_2 (supondo m bits de precisão) de duas formas: inteira ($v_z(M) \in \mathbb{N}$) ou fracionada ($v_f(M) \in \mathbb{R}$).

$$v_z(M) = \sum_{k=0}^{m-1} M_k 2^k$$

$$v_f(M) = \sum_{k=1}^m M_k 2^{-k}$$

onde b e k indexam o "vetor" de bits da mantissa conforme ilustra a figura abaixo.

Salienta-se que, de qualquer forma, $v_z(M) = v_f(M)$.

Assim, passam a existir duas maneiras de se trabalhar com a recuperação do valor armazenado:

$$x = (-1)^S (1 + v_f(M)) 2^{E-1023}$$

ou

$$x = (-1)^S \left(1 + \frac{v_z(M)}{2^m}\right) 2^{E-1023}$$

	<i>M</i>												
<i>b</i>	51	50	49	48	47	46	45	44	43	42	...	1	0
<i>k</i>	1	2	3	4	5	6	7	8	9	10	...	51	52

Figure 1. Indexação da mantissa

Desenvolvimento

Deseja-se, nesta etapa, desenvolver um método para aplicar Newton-Raphson aproveitando-se do padrão IEEE 754 para calcular a raiz quadrada de um número A em base 10.

Sabe-se que

$$x = \sqrt{A} \implies x^2 = A \implies f(x) = x^2 - A = 0$$

Nota-se que encontrar a raiz quadrada de A significa encontrar o zero da função $f(x)$. Para tal utilizar-se-á Newton-Raphson.

Da Seção 2.2, sabe-se que $A = m_a 2^{e_a}$. Da mesma forma, $x = m_x 2^{e_x}$.

Ao aplicar Newton-Raphson em $f(x)$, tendo x_k como estimativa inicial, tem-se:

$$x_{k+1} = x_k - \frac{x_k^2 - A}{2x_k}$$

que pode ser reescrito como:

$$\begin{aligned} x_{k+1} &= \frac{3x_k - x_k^2 + A}{2x_k} \\ &\equiv x_{k+1} = \frac{3x_k}{2x_k} - \frac{x_k^2}{2x_k} + \frac{A}{2x_k} \\ &\equiv x_{k+1} = \frac{3}{2} - \frac{x_k}{2} + \frac{A}{2x_k} \\ &\equiv x_{k+1} = 1.5 - \frac{x_k}{2} + \frac{A}{2x_k} \end{aligned}$$

Nota-se que $x_k = m_{x_k} 2^{e_{x_k}}$ no padrão IEEE 754. Portanto,

$$\begin{aligned} x_{k+1} &= 1.5 - m_{x_k} 2^{e_{x_k}-1} + \frac{A}{m_{x_k} 2^{e_{x_k}+1}} \\ &\equiv x_{k+1} = 1.5 - m_{x_k} 2^{e_{x_k}-1} + \frac{m_a 2^{e_a}}{m_{x_k} 2^{e_{x_k}+1}} \\ &\equiv x_{k+1} = 1.5 - m_{x_k} 2^{e_{x_k}-1} + \frac{m_a}{m_{x_k}} 2^{e_a - e_{x_k} + 1} \end{aligned}$$

Adotando a extração fracionária do valor da mantissa, tem-se:

$$x_{k+1} = 1.5 - m_{x_k} 2^{e_{x_k}-1} + \frac{m_a}{m_{x_k}} 2^{e_a - e_{x_k} + 1}$$

Nota-se que $\frac{m_a}{m_{x_k}}$ pode ser escrito como

$$\frac{v_f(m_a)}{v_f(m_{x_k})}$$

Substituindo x_k por x_i :

$$\frac{v_f(m_a)}{v_f(m_{x_i})} = \frac{\sum_{k=1}^m (M_a)_k 2^{-k}}{\sum_{k=1}^m (M_{x_i})_k 2^{-k}}$$

Note que a soma e subtração nos expoentes podem ser feitas com *bit shift*.

Algoritmo

Antes de aplicar Newton-Raphson, definem-se algumas operações que podem ser feitas através do padrão IEEE-754. É importante especificar uma devida precisão. Existem duas formas básicas de fazer isso no método: especificar o número de iterações; especificar o erro relativo a ser tolerado (diferença nos resultados entre as iterações). Por motivos de precisão, escolher-se-á a segunda opção. Assim, define-se:

```
#define EPSILON 1e-20
```

Em seguida, iniciar-se-á pela definição da função `doubleToIeee`, converte um valor real de ponto-flutuante na estrutura IEEE-754 definida previamente.

```
void doubleToIeee(double valor, int *s, int *exp, double *mant){
    *exp = floor(log2(valor));
    if(valor < 0){
        valor = valor*(-1);
        *s = 1;
    }else{
        *s = 0;
    }
    *mant = valor/(pow(2, *exp)) - 1.0;
}
```

Deseja-se, neste momento, deseja-se construir uma função que faça o cálculo da raiz quadrada. Para tanto, fazer-se-á uso da técnica de Newton-Raphson, discorrida na Seção ??.

Encontrar a raiz quadrada de um número A significa encontrar a raiz da função $f(x) = x^2 - A$. Nota-se que $f'(x) = 2x$. Deseja-se, também, obter grande precisão. Para tanto, deve-se utilizar um valor reduzido de argumento. Nesse ínterim, uma boa estimativa inicial para o método de Newton-Raphson é a mantissa do valor no padrão IEEE-754, uma vez que a mantissa é sempre maior que zero e menor que 2.

Além do chute inicial, o padrão IEEE-754 pode ser utilizado no cálculo sumário do método Newton-Raphson. Isso significa que, em vez de calcular a raiz da função (valor decimal), calcula-se a mantissa da raiz da função. A manipulação do expoente fica a cargo da seguinte lógica: basta dividi-lo por 2 (raiz quadrada), se for par; caso contrário, incrementa-se o expoente e corrige-se a mantissa (divide-a por 2) e a normaliza, para então poder dividi-lo.

Dessa forma, tem-se, seguindo o método de Newton-Raphson, o cálculo da Raiz Quadrada:

```
double raiz_quadrada (double mantissa, int expoente){
    if(expoente & 1){
        mantissa /= 2.0;
        expoente++;
    }

    expoente /= 2;
    double xk_1 = mantissa/2.0 + 1.0;
    mantissa += 1.0;
    double xk = 0, aux;

    while(fabs(xk_1 - xk) > EPSILON){
        xk = xk_1;
        xk_1 = xk + (mantissa - xk*xk)/(2*xk);
    }
    return xk_1;
}
```

Dessa forma, calcula-se a raiz quadrada através do método Newton-Raphson com o padrão IEEE-754 com precisão de 19 casas decimais.

Casos Teste

Testar-se-á a convergência para o cálculo das seguintes raízes:

$$A = 25$$

```
#1 [xk]: 1.28125000000000000000 [xk+1]: 1.25038109756097570724
#2 [xk]: 1.25038109756097570724 [xk+1]: 1.25000005807643410627
#3 [xk]: 1.25000005807643410627 [xk+1]: 1.250000000000000133227
#4 [xk]: 1.250000000000000133227 [xk+1]: 1.25000000000000000000
#5 [xk]: 1.25000000000000000000 [xk+1]: 1.25000000000000000000
```

Expoente: 2

RAIZ CONVERGIDA: 1.25000000000000000000

$$A = 81$$

```
#1 [xk]: 1.13281250000000000000 [xk+1]: 1.12502693965517241992
#2 [xk]: 1.12502693965517241992 [xk+1]: 1.12500000032254554583
#3 [xk]: 1.12500000032254554583 [xk+1]: 1.12500000000000000000
#4 [xk]: 1.12500000000000000000 [xk+1]: 1.12500000000000000000
```

Expoente: 3

RAIZ CONVERGIDA: 1.12500000000000000000

$$A = 5$$

```
#1 [xk]: 1.12500000000000000000 [xk+1]: 1.11805555555555558023
#2 [xk]: 1.11805555555555558023 [xk+1]: 1.11803398895790206957
#3 [xk]: 1.11803398895790206957 [xk+1]: 1.11803398874989490253
#4 [xk]: 1.11803398874989490253 [xk+1]: 1.11803398874989490253
```

Expoente: 1

RAIZ CONVERGIDA: 1.11803398874989490253

A precisão se mantém mesmo para grandes valores de A . Isso ocorre porque o cálculo da raiz é executado a partir da mantissa do valor de A no padrão IEEE-754 e encontra a mantissa da raiz.

Conclusão

Em relação ao algoritmo desenvolvido, fazer-se-á uma análise a partir de cada iteração do método. Nota-se que a quantidade de iterações depende do tempo de convergência em relação à precisão desejada.

Dessa forma, conclui-se que é possível aplicar Newton-Raphson onde cada iteração necessita de:

- 1 comparação
- 1 adição
- 2 subtrações
- 2 multiplicações
- 1 divisão

As operações de pré-processamento não são contabilizadas nesta análise, pois ocorrem somente uma única vez.

Provavelmente é possível executar melhoras nesta técnica, talvez através do tratamento da mantissa como um número inteiro, para que se possa executar *bit-shift*. Isso substituiria as multiplicações e divisões por 2. Contudo, a conversão requer aplicação de uma fórmula que pode não compensar o uso do deslocamento de bits. Para tanto, é necessário um estudo mais profundo