Padrão IEEE 754 para Aritmética Binária de Ponto Flutuante

Gerardo Valdisio Rodrigues Viana

Universidade Estadual do Ceará
Departamento de Estatística e Computação
e-mail: valdisio@uece.br

Resumo - O padrão IEEE 754 [1], recomendado pelos institutos ANSI (*American National Standard Institute*) e IEEE (*Institute of Electrical and Eletronic Engineers*), refere-se às normas a serem seguidas pelos fabricantes de computadores e construtores de compiladores de linguagens científicas, ou de bibliotecas de funções matemáticas, na utilização da aritmética binária para números de ponto flutuante. As recomendações são relativas ao armazenamento de dados numéricos, métodos de arredondamento, tratamento de casos de *underflow* e *overflow*, formas de realização das quatro operações aritméticas básicas e implementação de funções nas linguagens de programação. A grande vantagem destes procedimentos é permitir uma maior portabilidade dos *softwares* numéricos e criação de novos tipos de dados a fim de resolver problemas conhecidos como "mal condicionados" que contribuem para que a computação numérica forneça resultados errôneos, mesmo quando são utilizados métodos comprovadamente eficazes. Neste trabalho conceituamos como os números "reais" são armazenados, quais seus limites de grandeza e precisão; de que forma é verificada e tratada uma ocorrência de *underflow* ou *overflow*; quais os métodos de arredondamento utilizados; qual a configuração especial de *bits* recomendada e como devem ser realizadas as operações aritméticas com números de ponto flutuante. Apresentamos estas informações, acompanhadas de exemplos, a fim de dar subsídios a todos interessados na área de matemática computacional.

Palavras-Chave: Aritmética Binária. Forma de Armazenamento de Números. Métodos de Arredondamento. Normalização. Ponto Flutuante. Precisão e Grandeza.

1 INTRODUÇÃO

O padrão IEEE 754 (ANSI /IEEE Std 754-1985, New York, 1985 - *IEEE Standard for Binary Floating-Point Arithmetic*) contém normas [1,5] a serem seguidas pelos fabricantes de computadores e construtores de *Softwares* no tratamento da aritmética binária para números de ponto flutuante relativo ao armazenamento, métodos de arredondamento, ocorrência de *underflow* e *overflow*, além, é claro, da realização das operações aritméticas básicas.

Existe também o padrão IEEE 854 (ANSI/IEEE Std 854-1987, New York, 1987 - *IEEE Standard for Radix-Independent Floating-Point Arithmetic*) que tem os mesmos objetivos [2], porém, relativos a uma padronização para uma base independente genérica, para aqueles que utilizam outra base diferente da binária.

Entre as recomendações dos padrões IEEE 754 e IEEE 854 existem considerações sobre a implementação de funções nas linguagens de programação, de modo que a maioria dos fabricantes atuais de computadores (*Hardware*) e construtores de compiladores de linguagens científicas ou de bibliotecas de funções matemáticas (*Software*) seguem estas recomendações. Um tratamento especial também é citado para o cálculo do produto escalar de dois vetores feito através da multiplicação, seguida imediatamente da adição com um único arredondamento feito no fim da sequência de operações [3].

A grande vantagem deste procedimento é permitir uma maior portabilidade dos softwares numéricos e criação de novos tipos de dados a fim de tratar problemas conhecidos como “mal condicionados” [8], descritos a seguir.

2 JUSTIFICATIVA

Problemas de mal-condicionamento contribuem para que a computação numérica forneça resultados errôneos, mesmo quando são utilizados métodos de comprovada eficácia [6]. Por exemplo, a soma $\mathbf{(1 + 10^{50}) - 10^{50}}$ retorna o resultado errado **0** (zero) em qualquer máquina programada para trabalhar com tipos convencionais de precisão simples, dupla ou estendida, isto porque haveria necessidade de uma precisão equivalente a no mínimo cinqüenta casas decimais para que o resultado correto **1** (um) fosse obtido (no tópico adiante, relativo às operações com números de ponto flutuante, é indicado como a adição é realizada para um melhor entendimento deste problema).

Para superar algumas destas dificuldades a computação moderna combina recursos de *hardware*, como utilização de dígitos de guarda, implementados nos tradutores de linguagem, além da técnica conhecida como verificação dos resultados que consiste basicamente em [6]:

i) computação do resíduo, substituindo a solução encontrada na expressão matemática do problema; se o resíduo for pequeno a solução é considerada boa;

ii) repetir a mesma operação em precisão dupla, esperando que os resultados obtidos coincidam com os de precisão simples e

iii) repetir a computação com um ou mais dados ligeiramente modificados, esperando que uma pequena variação na solução indique a estabilidade do algoritmo e bom condicionamento do problema.

3 ARITMÉTICA DE PONTO FLUTUANTE

3.1 CONCEITO

Seja F (b, t, m, M) um sistema de ponto flutuante (*Floating-Point System*) que representa um subconjunto dos números reais, onde **b** $\geq$ 2 é a base de representação, **t** $\geq$ 1 é a precisão, e **m** e **M** são respectivamente o menor e maior expoente. O subconjunto F$\subset$ R contém números reais, de ponto flutuante, da forma $\mathbf{X = (-1)^s . b^e.(0.d_1d_2d_3.....d_t)}$, onde **s** é sinal (**0** se positivo e **1** se negativo), **e** o expoente ($m \leq e \leq M$, com $m < 0$, $M > 0$ e $|m| \approx M$) e $\mathbf{d_i}$ são dígitos da mantissa na base-b ($0 \leq d_i \leq b-1, \forall\ i, 1 \leq i \leq t$).

$f = 0.d_1d_2d_3.....d_t$ é chamada de mantissa, sendo que o dígito inicial **0** (zero) e o ponto decimal não são armazenados. Na representação IEEE 754 o ponto decimal, implicitamente, está entre os dígitos d_1 e d_2 , o que permite o armazenamento de mais um dígito da mantissa.

Exemplo: Considerando F (2, 8, –4, 3) e sendo X e Y, dois números da forma:

X= | 0 | 010 | 11100110 | Y= | 0 | 010 | 11100111 |

$$X = (-1)^0 . 2^2.(0.11100110) = (11.100110)_2 = (3.59375)_{10}$$

$$Y = (-1)^0 . 2^2.(0.11100111) = (11.100111)_2 = (3.609375)_{10}$$

Observe que X e Y são dois números consecutivos neste sistema de representação, sendo que o número decimal 3.6 não teria representação exata.

3.2 NORMALIZAÇÃO

O primeiro dígito (d_1) é diferente de zero para assegurar a unicidade de representação, e manter sempre a precisão máxima suportada pela mantissa. Esta forma é a chamada forma normalizada.

O valor zero não pode ser normalizado e tem representação especial, com mantissa nula (todos dígitos iguais a zero) e expoente o menor possível (m–1).

3.3 RELAÇÕES

Para todo sistema da forma **F(b,t,m,M)** são válidas as seguintes relações [7]:

i) Variação da mantissa: $1/b \leq f < 1$

ii) Menor número positivo: $\lambda = b^{m-1}$

iii) Maior número positivo: $\Lambda = (1 - b^{-t}) \cdot b^{M}$

iv) Número de elementos: $|F| = 2.(b-1).b^{t-1}.(M-m+1) + 1$

v) Épsilon da máquina: $\varepsilon = (0.5).b^{1-t}$

ε (épsilon) é um valor tal que $(1 + \varepsilon) = 1$; e normalmente é chamado de "zero" da máquina, apesar de ser um valor positivo, porém, muito pequeno. Este valor pode ser obtido através do algoritmo:

```
EPS = 1
repita
    EPS = EPS/2
    EPS1 = EPS + 1
até EPS1 = 1
```

Resultados obtidos num microcomputador 486DX (processador Intel 80x87, 66 Mhz), com utilização dos compiladores *Fortran 90* e *Turbo Pascal 7.0*:

i) Em FORTRAN

Precisão simples: $\varepsilon = 0.5421011\ E{-}19$

Precisão dupla: $\varepsilon = 0.5421010862427522\ E{-}19$

ii) Em PASCAL

Tipo REAL: $\varepsilon = 0.45474735089\ E{-}12$

Tipo DOUBLE: $\varepsilon = 0.111022302462516\ E{-}15$

Tipo EXTENDED $\varepsilon = 0.5421010862427522\ E{-}19$

Observe que somente para o tipo EXTENDED em PASCAL se consegue encontrar a ordem de grandeza correta do zero da máquina, o que não ocorre para a linguagem FORTRAN que é a linguagem científica mais recomendada para a computação numérica.

3.4 OCORRÊNCIA DE UNDERFLOW E OVERFLOW

Seja X um número real e X' uma aproximação de X, então, sabendo que λ é o menor número positivo do sistema de representação e Λ o maior número positivo, tem-se:

i) Se $|X| > \Lambda$; um caso de *overflow,* que é considerado uma situação irremediável, e, conseqüentemente, o programa deve ser abortado;

ii) Se $0 < |X| < \lambda$, um caso de *underflow* que é considerado como X' = 0. Esta prática pode invalidar resultados, por isso é chamado de *underflow* destrutivo.

iii) Se $\lambda < |X| < \Lambda$, com X necessitando de **d** dígitos de mantissa, causa um arredondamento para X' contendo **t** dígitos, sendo descartados (d – t) dígitos usando um dos métodos de arredondamento descritos a seguir.

3.5 MÉTODOS DE ARREDONDAMENTO

Os métodos de arredondamento [7] são utilizados para padronizar a forma de truncamento dos dígitos, em função do tamanho fixo da mantissa, por ocasião de seu aramazenamento. Para cada um dos métodos descritos abaixo apresentamos um exemplo onde foi considerado que d=5 e t=3.

i) Em direção ao zero (para o valor imediatamente anterior, ou menor, para positivos e imediatamente posterior, ou maior, para negativos) ;

Exemplos: $X = +0.34521 \Rightarrow X' = +0.345$

$Y = -0.34521 \Rightarrow Y' = -0.345$

ii) Em direção a $+\infty$ (para o maior valor relativo);

Exemplos: $X = +0.34521 \Rightarrow X' = +0.346$

$Y = -0.34521 \Rightarrow Y' = -0.345$

iii) Em direção a $-\infty$ (para o menor valor relativo);

Exemplos: $X = +0.34521 \Rightarrow X' = +0.345$

$Y = -0.34521 \Rightarrow Y' = -0.346$

iv) Para o mais próximo, com opção para a aproximação próximo par em caso de empate (este método é o mais recomendado pela IEEE 754).

Exemplos: $X = \pm 0.3452 \Rightarrow X' = \pm 0.345$

$Y = \pm 0.3458 \Rightarrow Y' = \pm 0.346$

$Z = \pm 0.3455 \Rightarrow Z' = \pm 0.346$

$T = \pm 0.3445 \Rightarrow T' = \pm 0.344$

4 PADRÃO IEEE 754 PARA PONTO FLUTUANTE

A base de representação recomendada é a binária (b = 2).

As operações devem ser executadas em precisão estendida com uso de dígitos de guarda e expoente deslocado (δ).

O uso do expoente deslocado, também chamado característica, tem por objetivo eliminar o sinal do expoente, por exemplo, se m= –127 e M=127, (δ deve ser igual a 127, de forma que a variação de expoente seria de 0 a 254.

Considerando o formato padrão para números de ponto flutuante:

s	e	$f = d_1d_2d_3.....d_t$
sinal	expoente	Mantissa

O padrão IEEE 754 recomenda os seguintes números de bits, de acordo com a precisão usada. A expressão (N = s + e + t), corresponde ao tamanho da palavra em bits.

i) Precisão Simples: s = 1, e = 8, f = 23 (+ 1 escondido), N = 32 bits.

$$X = (1.f * 2^{e-\delta}) \qquad \delta = 127$$

ii) Precisão Simples Estendida: s = 1, e ≥11, f ≥ 32 , N ≥ 43 bits.

$$X = (0.f * 2^{e-\delta}) \qquad \delta = 127$$

iii) Precisão Dupla: s = 1, e = 11, f = 52 (+ 1 escondido), N = 64 bits.

$$X = (1.f * 2^{e-\delta}) \qquad \delta = 1023$$

iv) Precisão Dupla Estendida: s = 1, e ≥ 15, f ≥ 64 , N ≥ 79 bits.

$$X = (0.f * 2^{e-\delta}) \qquad \delta = 1023$$

A maioria dos atuais processadores (Intel, Motorola, Sparc, entre outros) possui uma pastilha co-processadora de ponto flutuante [10] obedecendo este padrão. Eles

utilizam 3 formatos: precisão simples (32 bits), precisão dupla (64 bits) e estendida (80 bits).

5 CONFIGURAÇÃO ESPECIAL DE BITS - PADRÃO IEEE 754

Alguns resultados de operações aritméticas não são suportadas pela máquina, em virtude de divisão por zero, *overflow* etc; desta forma, eles são representadas de forma especial. Nos casos seguintes, o item **i** corresponde a uma representação especial para o **0** (zero); o item **ii** é um número com menor expoente possível, e os itens **iii** e **iv** são exceções, invariavelmente, intratáveis.

	Expoente	Fração	Significado
i)	e = m–1	f = 0	± 0
ii)	e = m–1	f $\neq$ 0	$\pm$ 0.f * 2^m
iii)	e = M+1	f = 0	$\pm \infty$
iv)	e = M+1	f $\neq$ 0	NaN (*Not a Number*)

O padrão IEEE 754 permite números desnormalizados, que são números com expoente igual a (m – 1) e zeros no início da mantissa.

Interpretação:

Número Normalizado		Número Desnormalizado
$1.f_1f_2f_3 \ldots f_t \times 2^e$	$\equiv$	$0.f_1f_2f_3 \ldots f_t \times 2^m$

Obs: $f_i = 0$ ou $f_i = 1$

6 OPERAÇÕES ARITMÉTICAS

Dados dois números X e Y representados na forma: $(-1)^s . b^e.(0.d_1d_2d_3.....d_t)$ são definidas as seguintes regras para as operações aritméticas:

6.1 ADIÇÃO E SUBTRAÇÃO

i) Escolher o número com menor expoente entre X e Y e deslocar sua mantissa para a direita um número de dígitos igual à diferença absoluta entre os respectivos expoente;

ii) Colocar o expoente do resultado igual ao maior expoente entre X e Y;

iii) Executar a adição/subtração das mantissas e determinar o sinal do resultado;

iv) Normalizar o valor do resultado, se necessário;

v) Arredondar o valor do resultado, se necessário e

vi) Verificar se houve overflow/underflow.

Exemplo: Seja F(10, 4, –50, 49), $\delta = 50$, com um dígito de guarda.

Se X = 436.7 e Y = 7.595 obter a soma (X+Y).

Obs.: Para todas operações, faremos uma análise da precisão do resultado obtido, através dos erros absoluto e relativo [4].

X = | 0 | 5 3 | 4 3 6 7 | Y = | 0 | 5 1 | 7 5 9 5 |

i) $e_1 - e_2 = 53 - 51 = 2$ (deslocamento de dois dígitos do menor , no caso Y)

Y = | 0 | 5 1 | 7 5 9 5 | ≡ Y = | 0 | 5 3 | 0 0 7 5 | 9

ii) Expoente do resultado: e = 5 3

iii) Adição das mantissas: 4 3 6 7 (0) + 0 0 7 5 (9) = 4 4 4 2 (9) , onde o valor entre parênteses é o dígito de guarda.

iv) Normaliza o resultado: Não há necessidade, pois $d_1 = 4 \neq 0$.

v) Arredonda: 4 4 4 2 (9) $\Rightarrow$ f = 4 4 4 3

vi) Verifica underflow/overflow: e – 50 = 53 –50 = 3 < 49. Não há.

Resultado:

X + Y = | 0 | 5 3 | 4 4 4 3 | ou X + Y = $0.4443 * 10^{53-50}$ = 444.3

Análise do resultado:

Erro absoluto = | 444.295 – 444.3 | = 0.01

Erro Relativo = 0.01 / 444.295 = 1.13×10^{-5}

6.2 MULTIPLICAÇÃO

i) Colocar o expoente do resultado igual à soma dos expoentes de X e Y;

ii) Executar a multiplicação das mantissas e determinar o sinal do resultado;

iii) Normalizar o valor do resultado, se necessário;

iv) Arredondar o valor do resultado, se necessário e

v) Verificar se houve overflow/underflow.

Exemplo: Para os mesmos dados do exemplo anterior obter o produto (X * Y).

i) Expoente: e = 53 + 51 = 104

ii) Multiplicação das mantissas: (4 3 6 7) * (7 5 9 5) = (3 3 1 6 7 3 6 5), considerando quatro dígitos de guarda.

iii) Normalizar o valor do resultado. Ajustar expoente (e–δ) = 104 – 50 = 54.

iv) Arredondar o valor do resultado: (3 3 1 6 7 3 6 5) $\Rightarrow$ (3 3 1 7).

v) Verifica underflow/overflow: e –50 = 54 –50 = 4 < 49. Não há.

Resultado:

X * Y = | 0 | 5 4 | 3 3 1 7 | ou X * Y = $0.3317 * 10^{54-50}$ = 3317

Análise do resultado:

Erro absoluto = | 3316.7365 – 3317 | = 0.2635

Erro Relativo = 0.2635 / 3316.7365 = 7.9445×10^{-5} (mesma precisão da adição).

6.3 DIVISÃO

i) Colocar o expoente do resultado igual à diferença dos expoentes de X (dividendo) e de Y (divisor);

ii) Executar a divisão das mantissas e determinar o sinal do resultado;

iii) Normalizar o valor do resultado, se necessário;

iv) Arredondar o valor do resultado, se necessário e

v) Verificar se houve overflow/underflow.

Exemplo: Para os mesmos dados do exemplo anterior obter a divisão (X / Y).

i) Expoente: e = 53 – 51 = 2

ii) Divisão das mantissas: (4 3 6 7) * (7 5 9 5) = (5 7 4 9 8 3 5 4), considerando quatro dígitos de guarda.

iii) Normalizar o valor do resultado. Ajustar expoente (e+δ) = 2 + 50 = 52.

iv) Arredondar o valor do resultado: (5 7 4 9 8 3 5 4) $\Rightarrow$ (5 7 5 0).

v) Verifica underflow/overflow: e –50 = 52 –50 = 2 < 49. Não há.

Resultado:

X / Y = | 0 | 5 2 | 5 7 5 0 | ou X / Y = $0.5750 * 10^{52-50}$ = 57.50

Análise do resultado:

Erro absoluto = | 57.49835418 – 57.50 | = 0.001645819

Erro Relativo = 0.001645819 / 57.49835418 = 2.8623×10^{-5}

7 CONCLUSÃO

Esperamos que este trabalho tenha atingido todos seus objetivos. Como pretendíamos, mostramos as normas de cálculo com números de ponto flutuante e suas formas de armazenamento, de acordo com as recomendações do padrão IEEE 754.

Como leitura complementar indicamos especialmente as referências [3,5,7,8]. Em relação ao tratamento da precisão dos métodos numéricos, decorrentes, ou não, das

operações aritméticas é conveniente consultar livros de Cálculo Numérico [4,6,9], bem como, para observar de que forma são implementadas a potenciação (x^y) e funções matemáticas básicas (sen, cos, log, arctg, e^x). Estas funções são resolvidas por séries de potências, as quais utilizam apenas as operações aritméticas básicas na forma descritas no item 6.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] American National Standards Institute / Institute of Electrical and Electronics Engineers: *IEEE Standard for Binary Floating-Point Arithmetic,* ANSI/IEEE Std 754-1985, New York, 1985.

[2] American National Standards Institute / Institute of Electrical and Electronics Engineers: *IEEE Standard for Radix-Independent Floating-Point Arithmetic,* ANSI/IEEE Std 854-1985, New York, 1987.

[3] BOHLENDER, G., *What do we need beyond IEEE Arithmetic ?*, Computer Arithmetic and Self-Validating, Numerical Methods, Academic Press Inc., 1990.

[4] DORN, W.S. e D.D. MCCRACKEN, *Cálculo Numérico com estudo de casos em Fortran IV,* Editora Campus e Editora USP, Rio de Janeiro, 1968.

[5] FIGUEIRÊDO, M.A.B., *Um pacote de Aritmética de Múltipla Precisão*, Tese de Mestrado em Informática na UFPB, área de concentração: Ciência da Computação, Campina Grande, 1989.

[6] FORSYTHE, G.E., M.A. MALCOLM e C.B. MOLER, *Computer Methods for Mathematic Computations*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1977.

[7] HATTORI, M.T. e B.C.N QUEIROZ, *Acredita no Computador ?,* Departamento de Sistemas e Computação, UFPB, Campina Grande, 1994, trabalho não publicado.

[8] HATTORI, M.T. e B.C.N QUEIROZ, *Métodos e Software Numéricos*, Departamento de Sistemas e Computação, UFPB, Campina Grande, 1994, trabalho não publicado.

[9] HUMES, A.F.P.C., I.S.H. de MELO, L.K. YOSHIDA e W.T. MARTINS, *Noções de Cálculo Numérico,* Editora McGraw-Hill, São Paulo, 1984.

[10] TANENBAUM, A. S., *Organização Estruturada de Computadores,* 3ª Edição, Prentice/Hall do Brasil, Rio de Janeiro, 1992.