Métodos Computacionais em Física

Aula 01

Representação Numérica e Erros

Um número qualquer

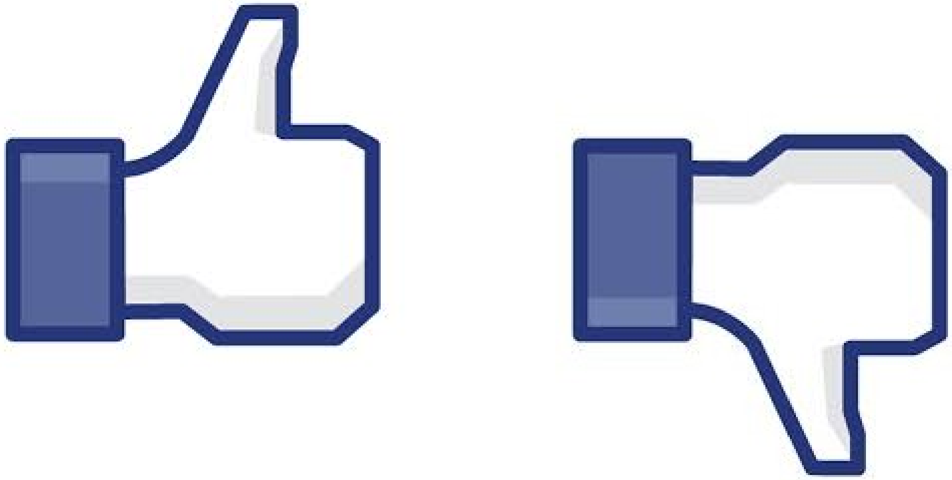
- 127,8254
- Mantissa → 1278254 (7 algarismos)
- Expoente -4 (no caso da base 10)
- Podemos escrever

$$s \times b^e$$

Bases

• Base 10:	• Base 16:		• Base 2:
1	1	16	1
2	2	17	10
3	3	18	11
4	4	19	100
5	5	1A	101
6	6	1B	110
7	7	1C	111
8	8	1D	1000
9	9	1E	1001
10	A	1F	1010
11	B	20	1011
12	C	...	1100
13	D		1101
14	E		1110
15	F		1111
16	10		10000
17	11		10001
18	12		10010
19	13		10011
20	14		10100

Spin e Memória



1 bit

Qual o intervalo de inteiros representáveis por N bits?

- N=1 → 0,1
- N=2 → 00, 01, 10, 11
- N=3 → 000, 001 010, 011, 100, 101, 110, 111
- ...
- Em geral → $\mathbf{[0, 2^{N}-1]}$
- → ERRADO!!!
- Usa-se um bit para representar o sinal
- Conclusão: $\mathbf{[-2^{N-1}+1, 2^{N-1}-1]}$

- Normalmente usa-se 8 bits para se representar um caractere, como “a” ou “b”;
- 8 bits = 1 B = 1 byte
- 1KB = 1024 B
- 1MB = 1024 KB
- 1GB = 1024 MB
- 1 TB = 1024 GB, ...

- Computadores antigos utilizavam 8 bits para um inteiro: máximo inteiro representável =127;
- Hoje, a maioria dos computadores utiliza 64 bits: máximo inteiro = $2^{63}-1 \approx 10^{19}$;
- Bom o suficiente?
- Razão tamanho do universo/tamanho de um próton = 10^{41}

Quem gosta de números decimais?

Quem gosta de números binários?

Conversão da base 10 para a base 2

- Converter 25_{10} para base binária:
- Divide por 2 → resultado=12, resto 1;
- Divide por 2 → resultado=6, resto 0;
- Divide por 2 → resultado=3, resto 0;
- Divide por 2 → resultado=1, resto 1;
- Divide por 2 → resultado=0, resto 1;

$$25_{10} = 11001_2$$

Conversão da base 2 para a base 10

$$11001_2 = 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + \underline{1}*2^0$$

$$= 16_{10} + 8_{10} + 1_{10} = 25_{10}$$

$$= \ (((((0*2+1)*2+1)*2+0)*2+0)*2+\underline{1})$$

Conversão da base 10 para a base 2: números menores que a unidade

- Começa com 0,
- Multiplica o número por 2 a cada passo
- Resultado maior (menor) que 1: adiciona 1 (0)
- A cada passo despreze a parcela maior que 1 do resultado;
- Exemplo:

- $(1/5)_{10}$
- $(1/5)*2=2/5<1 \rightarrow 0$
- $(2/5)*2=4/5<1 \rightarrow 0$
- $(4/5)*2=8/5>1 \rightarrow 1$
- $(3/5)*2=6/5>1 \rightarrow 1$
- $(1/5)*2=2/5<1 \rightarrow 0$
- ...

$$(1/5)_{10}=(0{,}0011001100 11\ldots)_2$$

- $(1/8)_{10}$
- $(1/8)*2=1/4<1 \rightarrow 0$
- $(1/4)*2=1/2<1 \rightarrow 0$
- $(1/2)*2=1/1=1 \rightarrow 1$
- $(0)*2=0<1 \rightarrow 0$
- ...

$$(1/8)_{10}=(0,001)_2$$

- $(1/10)_{10}$
- $(1/10)*2=1/5<1 \rightarrow 0$
- $(2/5)*2=4/5<1 \rightarrow 0$
- $(4/5)*2=8/5>1 \rightarrow 1$
- $(3/5)*2=6/5>1 \rightarrow 1$
- $(1/5)*2=2/5<1 \rightarrow 0$
- $(2/5)*2=4/5<1 \rightarrow 0$
- $(4/5)*2=8/5<1 \rightarrow 1$
- ...

$$(1/10)_{10}=(0,0011001100110\ldots)_2$$

Representando Números reais

$$x = (-1)^{s} \times 1,f \times 2^{e-bias}$$

Single precision → 32 bits=4B

1 bit

8 bits=1B

23 bits

Qual a precisão da mantissa?

- 23 bits?
- Outra resposta?
- Resposta correta: precisão de 24 bits!!!!!
- Por quê?

Representando Números reais

$$x = (-1)^s \times 1,f \times 2^{e-bias}$$

Phantom Bit

Number name	Values of s, e, and f	Value of single
Normal	$0 < e < 255$	$(-1)^s \times 2^{e-127} \times 1.f$
Subnormal	$e = 0, \; f \neq 0$	$(-1)^s \times 2^{-126} \times 0.f$
Signed Zero (± 0)	$e = 0, \; f = 0$	$(-1)^s \times 0.0$
$+\infty$	$s = 0, \; e = 255, \; f = 0$	$+\texttt{INF}$
$-\infty$	$s = 1, \; e = 255, \; f = 0$	$-\texttt{INF}$
Not a number	$s = u, \; e = 255, \; f \neq 0$	$\texttt{NaN}$

$$1,4 \times 10^{-45} \leq singleprecision \leq 3.4 \times 10^{38}$$

Representando Números reais

$$x = (-1)^{s} \times 1,f \times 2^{e-bias}$$

Double precision → 64 bits=8B

1 bit

10 bits

53 bits

Qual a precisão da mantissa?

- 53 bits?
- Outra resposta?
- Resposta correta: precisão de 54 bits!!!!!
- Por quê?

Number name	Values of s, e, and f	Value of double
Normal	$0 \leq e \leq 2047$	$(-1)^s \times 2^{e-1023} \times 1.f$
Subnormal	$e = 0, \ \ f \neq 0$	$(-1)^s \times 2^{-1022} \times 0.f$
Signed zero	$e = 0, \ \ f = 0$	$(-1)^s \times 0.0$
$+\infty$	$s = 0, \ \ e = 2047, \ \ f = 0$	`+INF`
$-\infty$	$s = 1, \ \ e = 2047, \ \ f = 0$	`-INF`
Not a number	$s = u, \ \ e = 2047, \ \ f \neq 0$	`NaN`

$$4,9 \times 10^{-324} \leq doubleprecision \leq 1.8 \times 10^{308}$$

Overflow e underflow

- Para singles:

  - Overflow: x é maior que 2^{128};

  - Underflow: x é menor que 2^{-128};

- Resultado de overflow é normalmente um NAN, mas isso depende da máquina;
- Normalmente um underflow é convertido para o valor zero.

Terminações

- $(1/10)_{10}=0,1$
- $(1/3)_{10}=0,3333...$
- $(1/2)_2=0,1$
- $(1/4)_2=0,01$
- $(1/8)_2=0,001$
- $(1/10)_2=0,0011001100110011...$

Caso do 1/10

- Convertendo $(0,1)_{10}$ para binário será:
- $(1/10)_2=0,0011001100110011001100110 0...$
- Mas no caso single precision, o computador trunca para:
- $(1/10)_2=0,00110011001100110011001100$
- Que quando transformado de volta para decimal fica:
- $(1/10)_{10}=0.100000001490116119384765625$

Caso do (1/10) ao quadrado

- Para você: x=0,1
- Para o PC: x=0.100000001490116119384765625
- Se VOCÊ quadrar o último resultado, obterá exatamente: x*x=0,010000000298023226097399174250313080847263336181640625
- O PC truncará para:

  x*x=0.010000000707805156707763671875
- Mas o número representável mais próximo de 0,01 é 0.009999999776482582092285156250
-

Precisão da máquina

$$1_c + \epsilon_m = 1_c$$

Singles possuem erro na sexta casa decimal
Doubles possuem erro na décima quinta casa decimal

Exemplo: $7+10^{-7}$

Erros

- Teoria ruim e vacilos...

  - Erros tipográficos

  - Utilizar o arquivo errado ou dados errados

  - Teoria com “furos”

- Erros aleatórios

  - Flutuações na rede elétrica

  - Raios cósmicos ou alguém puxando a tomada

Erros

- Erros de aproximação

  - Truncando uma série

  - elementos finitos

  - Trocar variáveis por constantes

- Erros de arredondamento

  - O computador é discreto, enquanto o mundo é (muitas vezes) contínuo

Modelo para o desastre

$$x_c = x(1 + \epsilon_x)$$

a=*b*-*c* → *b* e *c* próximos

$$\frac{a_c}{a} = 1 + \frac{b}{a} MAX(|\epsilon_b|, |\epsilon_c|)$$

Acumulação de erros

a=bxc

$$\frac{a_c}{a} \approx 1 + \epsilon_b + \epsilon_c$$

Para várias operações seguidas, argumentos estatísticos podem mostrar que:

$$\epsilon_{round\ off} \approx \sqrt{N}$$

- Um computador rápido realiza cerca de 10^{10} operações por segundo;
- Em três horas teremos 10^{14} operações;
- O erro será de cerca de $10^{7}\varepsilon_m$
- Então o erro tem de ser bem menor que 10^{-7};
- Moral da história: use doubles.

Minimizando erros

- Conhecimento da análise numérica;
- Cuidado com o compilador;
- Binários podem descrever uma vasta coleção de escalas, mas pode falhar em dar resultados decimais exatos;
- Cuidado com operações iterativas (matrizes);
- Aspirações de matemáticos pode ser frustradas pelo computador;
- Evite o teste “se x=y”.

Alguns passos rumo ao sucesso

- 1: Testar;
- 2: Testar;
- 3: Testar;
- 4: Testar;
- 5: Testar;
- 6: Testar;
- 7: Testar;
- ...