Exercícios de

Física Computacional

(Parte 4)

Mestrado em Engenharia Física-Tecnológica (MEFT)

Rui Coelho

Departamento de Física do Instituto Superior Técnico

Ano Lectivo: 2019-20

rui.alves.coelho@tecnico.ulisboa.pt

versão: 1 de Outubro de 2019

4. Representação numérica e Precisão

Exercício 35. Considere o número real de precisão simples e 32 bits, dado por

sinal	expoente	mantissa
0	1000 0001	1100 1100 1100 0000 0000 000

a) Determine o valor do expoente verdadeiro.
b) Mostre que a mantissa vale 1.7998046875
c) Determine o valor do número real.

Exercício 36. (*adaptado de Barão 2016*) Escreva uma funçao em C++ que determine os limites underflow e overflow do seu computador e linguagem de programação, dentro de um factor 2. Comece com um número real igual a 1 e progressivamente imprima os valores multiplicados e divididos por 2 (use pelo menos 15 caracteres para representar os valores).

a) Obtenha os valores limite de underflow e overflow para números reais de precisão simples. *Justifique no código fonte os valores limite.*
b) Obtenha os valores limite de underflow e overflow para números reais de precisão dupla. *Justifique no código fonte os valores limite.*

Exercício 37[*]. Desenvolva uma função que faça a conversão de um número real positivo para um binário usando 32bits (float), sem usar operadores unários para manipulação dos bits que representam números inteiros.

Sugestão 1: pode desenvolver uma função ***int_to_string*** que converta inteiros para binário e usar o método ***reverse*** (ver <u>*algorithm*</u> container) para reverter a ordem de caracteres numa string.
Sugestão 2: use caso necessite o método ***substr*** para extrair sub-strings de strings. Caso necessite de uma sub-string que vá até ao último carácter use string::npos (ver documentação de string::substr).

Considere agora o número real r=7.281. Verifique se o número ***r*** tem representação exacta em binário (seja r_{float} o número real efectivamente representado). Caso não tenha, verifique se a representação em binário obtida é a mais próxima do número ***r*** ou se há outra que minimize o erro absoluto $|r-r_{float}|$.

Exercício 38 : *(Barão 2016)* Escreva uma funçao em C++ que determine a precisão do computador. Por exemplo, implemente um algoritmo em que se adicione ao número 1. um número cada vez mais pequeno até que este seja inferior à precisão e a soma seja 1.

a) Para números reais de precisão simples.
b) Para números reais de precisão dupla.

Exercício 39 : Neste exercício vamos verificar a transição entre erro de discretização e erro de round-off no cálculo da derivada de uma função. Considere a função $f(x)=e^{2x}$. Elabore para o efeito um código ***discretization_and_round.cpp*** onde defina a função f(x) e sua derivada i.e.

```
float f1(float x) {
      return exp(2.0*x);
}
float f1d(float x) {
      return 2.0*exp(2.0*x);
}
```

e faça um gráfico com a diferença entre a aproximação da derivada de $f(x_0)$ dada por $[f(x_0+h)- f(x_0)]/h$ e a derivada real para valores de h entre *5e-7* e *1e-1*. Assuma $x_0=3$.

Exercício 40 : (adaptado de *Barão 2016*) Resolva a equação quadrática $x^2 - 2bx + c = 0,\ \ b^2 > c$ pode ser feito com recurso à formula resolvente dando lugar à seguinte solução:

$$x_{1,2} = b \pm \sqrt{b^2 - c} \quad \textbf{\textit{(A)}}$$

a) Mostre que o produto das duas soluções é dado por: $x_1 x_2 = c$

b) Para evitar o problema de cancelamento da subtração, as soluções da equação podem ser dadas por:

$$\left.\begin{array}{ll} x_1 = b + \sqrt{b^2 - c} \ \ , \ \ x_1 x_2 = c & \text{se b>0} \\ x_1 = b - \sqrt{b^2 - c} \ \ , \ \ x_1 x_2 = c & \text{se b<0} \end{array}\right. \qquad \textbf{\textit{(B)}}$$

Mostre que estas soluções (B) têm menor erro que a solução (A) usando b=0.03 e c=0.0008.