AULAS 7 E 8: FIS271 - Física Computacional I

Exercício 1. Além do *xmgrace*, uma outra ferramenta computacional para produção de gráficos e visualização de dados é o *gnuplot* [1]. Uma das grandes vantagens do *gnuplot* é que ele permite a elaboração de gráficos em 3 dimensões (veja os exemplos no site). Ele pode ser utilizado tanto no modo iterativo (i.e., simplesmente digitando *gnuplot* no terminal), quanto no modo via script, onde a sequência de comandos é incluida em um arquivo *.gnp*. Neste caso, o script pode ser chamado pelo *gnuplot* através do comando:

```
gnuplot seu_script.gnp
```

Utilize os dados produzidos nos items (d) e (e) do Exercício 1 (vide Roteiro das Aulas 5 e 6) para produzir gráficos utilizando o *gnuplot*, conforme descrito abaixo:

a) Crie um script *ex01a.gnp* para fazer um único gráfico com as cinco curvas correspondentes à função $f(t)$ obtidas com parâmetros A, B e C diferentes. Utilize comandos para (i) imprimir o gráfico para um arquivo *ex01a.png*; (ii) alterar o "label x" para t e o "label y" para $f(t)$; (iii) fazer com que o "eixo x" fique no intervalo $t \in [0.0, 4.0]$.

b) Crie um script *ex01b.gnp* para fazer um único gráfico com as cinco curvas correspondentes à função $r(t)$ obtidas com parâmetros δ, τ e p diferentes. Utilize comandos para (i) imprimir o gráfico para um arquivo *ex01b.png*; (ii) alterar os "label x" para t e o "label y" para $r(t)$; (iii) fazer com que o "eixo x" fique no intervalo $t \in [10^{-5}, 10^{2}]$; (iv) fazer com que ambos os eixos sejam mostrados na escala logarítmica.

Exercício 2. Números decimais em um computador podem ser definidos como [2]

$$(-1)^s \times 2^{c-c^*} \times (1+f) \quad ,$$

onde os intervalos e valores assumidos por s, c, f e c^* são determinados dependendo de como a memória do computador é ocupada, isto é, 16-bits, 32-bits ou 64-bits. Por exemplo, representações em 32-bits (4 bytes) possuem 1 bit para o *sinal* (s), 8 bits para o *expoente* ($c = c_7 2^7 + c_6 2^6 + \ldots c_1 2^1 + c_0 2^0$) e 23 bits para a *mantissa* ($f = f_1 2^{-1} + f_2 2^{-2} + \cdots + f_{23} 2^{-23}$), com o *bias* dado por $c^* = 127$.

a) Escreva o número binário (32-bits) $\boxed{1}$ $\boxed{01101111}$ $\boxed{01001100000000000000000}$ na representação decimal.

b) Escreva o número decimal 1227,86 na representação binária. Verifique se o número encontrado é uma dízima periódica.

Exercício 3. Números decimais podem ser escritos como $0, d_1 d_2 \ldots d_{k-1} d_k \times 10^n$, onde d_k representa o último digito significativo devido ao truncamento. Por exemplo: $fl_5(\pi) = 0,31415 \times 10^1$ para $k = 5$. Considere a seguinte operação de subtração $\Delta = x_1 - x_2$ com $x_1 = 22/7$ e $x_2 = 60/19$. Assumindo $\Delta_5 = fl_5(x_1) - fl_5(x_2)$ e o valor "exato" da subtração como Δ^*, calcule o erro absoluto $|\Delta_5 - \Delta^*|$ e o erro relativo $|\Delta_5 - \Delta^*|/|\Delta^*|$ cometido pelo truncamento.

Exercício 4. Implemente um programa para calcular numericamente o valor do seguinte somatório:

$$S = \sum_{i=k_i}^{k_f} (-1)^i i^5 \quad ,$$

com $k_i = 1$ e $k_f = 100$. Considerando diferentes precisões para a variável S, isto é, `real*4` e `real*8`. Verifique o que ocorre quando o cálculo é feito assumindo $k_i = 100$ e $k_f = 1$.

Referências:

[1] http://www.gnuplot.info

[2] J. D. Faires e R. L. Burden. Numerical Methods (3rd ed.)