2º trabalho de Física Computacional

Fernando Barao (Dep. Física, IST)
versão: 2 Dez 2014, 9H50

Instruções

A resolução dos exercícios deve ser feita:

- na zona de trabalho *svn* de cada grupo, criando para tal o directório **FC/2014/<groupname>/Trab2**

```
cd FC/2014/<groupname>
svn mkdir Trab2
svn ci -m "directorio Trab2 criado"
```

- deve ser acompanhada de uma **pequena** memória descritiva com no máximo 5 páginas e escrita num ficheiro latex e convertida para pdf, com as respostas pedidas em cada exercício (veja o sumário no topo de cada exercício). A memória descriptiva deve ser inserida no mesmo directório.

```
ficheiro tex: groupXXTrab2.tex
ficheiro pdf: groupXXTrab2.pdf
```

- o código implementado deve conter os comentários necessários e suficientes à sua compreensão
- no directório **FC/LIBs** existe a classe *cFCGraphics* que deve ser usada para se obterem os gráficos do trabalho. Os métodos implementados na classe podem ser consultados no *cFCgraphics.h*.
  Atenção: o código desta classe poder ter evolução durante o trabalho, daí ser importante usar sempre a versão que está neste directório (e não copiar a classe para um directório pessoal).

Este trabalho deve ser entregue, isto é, **svn commited**, até às 20H00 do dia 6 de Dezembro de 2014.

[Exercício 2.]

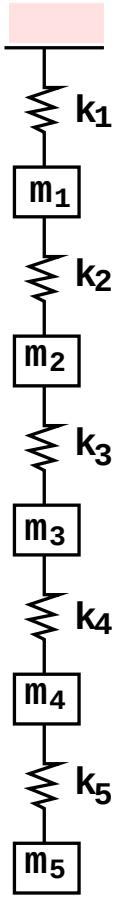
sumário

```
No final deste exercício, os seguintes códigos devem existir
no directório FC/2014/<groupname>/Trab2:
-> EqSolver.h, EqSolver.C    ................ class
-> Springs.C   .............................. programa
-> Makefile  ................................ makefile
   - para compilar o programa: make Springs

A memória descriptiva deve conter para o exercício 2:
-> o sistema de equações a resolver
-> a solução
```

Considere o sistema de 5 molas cujas constantes de restituição são $k_1, k_2, \cdots, k_5$ cujas massas são desprezáveis. Essas molas encontram-se penduradas, sujeitas à força Peso (P) e ligadas a cinco massas $m_1, m_2, \cdots, m_5$, tal como se mostra na figura junta. Na situação de equilíbrio, as molas estão deformadas sendo $x_1, x_2, \cdots, x_5$ os deslocamentos em relação à sua posição de equilíbrio. Definindo,

$$k_1 = k_2 = k_3 = 10\text{N/mm} \qquad k_4 = k_5 = 5\text{N/mm}$$
$$P_1 = P_3 = P_5 = 100\text{N} \qquad P_2 = P_4 = 50\text{N}$$



a) Determine o sistema de equações que relacionam as massas e os deslocamentos das molas.

b) Resolva o sistema de equações usando a classe **EqSolver** já usada no âmbito da 2a série de exercícios e implemente os métodos para resolver matrizes tridiagonais,

```
// pass the matrix diagonals and the matrix dimension
// return the decomposed arrays

void LUdecomposition3(float*, float*, float*, int);
void LUsolve3(...);
```

Escreva um programa **Springs.C** que apresente a solução.

[Exercício 3.]

sumário

```
No final deste exercício, os seguintes códigos devem existir
no directório FC/2014/<groupname>/Trab2:
-> DataInterpolator.h, DataInterpolator.C.................. class
-> mainInterpolator.C ..................................... programa
-> Makefile ............................................... makefile
   - para compilar o programa: make Interpolator

A memória descriptiva deve conter para o exercício 3:
-> os resultados (valores e plots) pedidos abaixo
```

As secções eficazes de interacção (medem a probabilidade de interacção) de um neutrão com um núcleo, em função da energia do neutrão, são dadas pela seguinte tabela:

i	1	2	3	4	5	6	7	8	9
E_i (MeV)	0	25	50	75	100	125	150	175	200
σ_i (mbarn)	10.6	16.0	45.0	83.5	52.8	19.9	10.8	8.25	4.7

O objectivo deste exercício é realizar a interpolação dos dados adquiridos. Construa para isso a classe **DataInterpolator** e construa os seguintes métodos:

- **CubicSplineCurvatures:** obter os valores das segundas derivadas nos pontos (knots)
- **CubicSpline:** obter a interpolação por Cubic Spline e retornando um objecto TF1* de root
- **CubicSplineSegment:** obter a funcão que descreve o segmento a que pertence o ponto x
- **CubicSplineDeriv:** obter a derivada da interpolação Cubic Spline
- **CubicSplineDerivN:** obter a derivada numérica ($\mathcal{O}(h^2)$) da interpolação Cubic Spline
- **Polynomial:** obter o polinómio interpolador que passe pela totalidade dos pontos
- **PolynomialDerivN:** obter a derivada numérica ($\mathcal{O}(h^2)$) do polinómio interpolador

```
class DataInterpolator {
  public:
  // N data points
  DataInterpolator(int N, double* x, double* y);
  TGraph* Draw(); //draw points
  // cubic spline (N knots)
  double* CubicSplineCurvatures(); // return k's
  TF1* CubicSpline(double* k); // k=input curvatures
```

```
  TF1* CubicSplineSegment(double* k, double x); // k=input curvatures
  double CubicSplineDeriv(double* k, double a); // at point a
  double CubicSplineDerivN(double* k, double a);
  //polynomial
  ... (métodos cálculo do polinómio)
  TF1* Polynomial();
  double PolynomialDerivN(double a);

  private:
  ....
};
```

Obtenha os seguintes resultados num programa **mainInterpolator.C**:

a) Qual o valor da interpolação por cubic spline e por polinómio no ponto de energia $E = 57.3$ MeV?

b) Desenhe as duas funções interpoladores num mesmo gráfico, conjuntamente com os pontos da tabela.

c) Desenhe a função diferença dos dois interpoladores.

d) Desenhe a diferença entre a derivada numérica cubic spline e a derivada calculada analiticamente.

e) Desenhe a função diferença entre as derivadas numéricas.

[Exercício 4.]

sumário

No final deste exercício, os seguintes códigos devem existir
no directório FC/2014/<groupname>/Trab2:
-> Element.h,C PhysProcess.h,C BetaDecay.h,C ..... classes
-> Decay.C ..... programa
-> Makefile ..... makefile
      - para compilar o programa: make decay

A memória descriptiva deve conter para o exercício 4:
-> o sistema de equações diferenciais e a solução analítica
-> os resultados pedidos na alínea c)

O decaimento radioactivo é um processo estocástico. O momento em que o núcleo instável se desintegra não está predeterminado e ocorre com uma dada probabilidade p_i. Neste exercício pretende-se estudar a seguinte cadeia de desintegração,

$$^{210}Bi \underset{\beta^-}{\rightarrow} {}^{210}Po \underset{\alpha}{\rightarrow} {}^{206}Pb$$

onde:

- o elemento bismuto desintegra-se através da conversão de um neutrão em protão e a respectiva emissão de um electrão (com neutrinos claro!) - **beta decay**, cujo espectro de energia cinética do electrão (T_e) é dado por:

  $$N(T_e) = \left(T_e^2 + 2T_e m_e c^2\right)^{1/2} \left(Q - T_e\right)^{1/2} \left(T_e + m_e c^2\right) \quad \text{com} \quad [T_e] = MeV$$

  formando o elemento Polónio. Q é a energia libertada na desintegração.

- o elemento Polónio desintegra-se através da emissão de uma partícula α

A tabela que se segue sumariza as características dos núcleos:

Elemento	Z	A	Q	$T_{1/2}$
^{210}Bi	83	210	1.1612 MeV	5.012 dias
^{210}Po	84	210	5.40746 MeV	138.376 dias

a) Obtenha o sistema de equações diferenciais que definem a evolução no tempo das populações de Bismuto e Polónio. Apresente as soluções analíticas do problema.

b) Para a resolução numérica do problema, usando o método de Monte-Carlo, construa as seguinte classes:

- **Element:** classe que armazena as características de um elemento incluindo o nome (ex: "210Bi"para o Bismuto 210)

```
class Element {
  public:
  Element(string fname, int fZ, int fA, int fN);
  void SetPhysProcess(PhysProcess*);
  void Print();
  ...

private:
  int N; //number of elements
  int Z; //atomic number
  int A; //mass number
  string name; //element name
  vector<PhysProcess*> v; //phys processes
};
```

- **PhysProcess:** classe base puramente virtual que define os processos físicos

```
class PhysProcess {
public:
  PhysProcess() {;}
  virtual double Spectrum(double Te) = 0; //input: kinetic energy electron
  void SetName(string);
  string GetName();
  ...
protected:
  string name; //nome do processo fisico
};
```

- **BetaDecay:** classe *BetaDecay* que implementa o processo físico *beta decay* e que herda da classe *PhysProcess*. Note que terá que implementar o espectro de emissão do electrão.

```
class ...   {
public:

private:
 double T12; //half-time in sec
 double Q; //Te_max MeV
};
```

c) Realize agora um programa **Decay.C** onde vai realizar as tarefas que se seguem. O programa deve mostrar os resultados também no ecrã.

```
int main() {
  // Este programa exemplifica as etapas de construção de um elemento
```

```
  // e a definição do processo físico que ele terá

  // make element "bla"!
  Element E("bla",...);

  // instantiate physics decay process that element undergoes
  ...

  // add physics process to element
  ...

  // numerical solution

}
```

1. Realize a simulação numérica da cadeia de desintegração usando números aleatórios. Qual o intervalo de tempo que vai usar? Qual a probabilidade de desintegração por unidade de de tempo, do Bismuto e do Polónio?
2. Desenhe o gráfico das populações de $\boldsymbol{Bi}$ e $\boldsymbol{Po}$ em função do tempo $\boldsymbol{t}$, para as condições iniciais: $\boldsymbol{N(Po)_{t=0} = 0}$ e $\boldsymbol{N(Bi)_{t=0} = 1000, 10000, 100000}$ e compare graficamente com as soluções analíticas. Qual o número de elementos Bismuto e Polónio ao fim de $\boldsymbol{t = 2\ dias}$.
3. Determine o instante no qual é máxima a produção de partículas $\boldsymbol{\alpha}$.
4. Gere aleatoriamente 10 000 energias do electrão e armazene-as num histograma, a partir da função $\boldsymbol{N(T_e)}$
   Nota: o histograma deve ter xmin=0.0, xmax=1.2, Nbins=100
5. Admita agora que os electrões são colectados por um detector cuja eficiência é dada por:

   $$\varepsilon(\boldsymbol{T_e}) = \frac{\mathbf{1}}{\boldsymbol{e}^{\mathbf{6-15(T_e/MeV)}} + \mathbf{1}}$$

   Determine numericamente, quer pelo método de Simpson, quer usando o método MC, o seguinte integral, com precisão 0.001

   $$\int_0^\infty \boldsymbol{N(T_e)}\ \varepsilon(\boldsymbol{T_e})\ \boldsymbol{dT_e}$$

   Descreva sumariamente quantas tiragens MC teve que fazer e qual o número de intervalos Simpson (e largura) que teve que usar. Nota: defina no interior do ficheiro **Decay.C** as funções auxiliares **integrationSimpson(...)** e **integrationMC(...)**

6. Determine numericamente o valor médio da energia do electrão detectável.

Fernando Barao
Dep. Fisica IST
29 de Novembro de 2014