2º Trabalho de Física Computacional (MEFT/IST)

1º semestre 2020-21

Fernando Barão, Jorge Vieira, Miguel Orcinha

Entrega do trabalho

1. Entrega do trabalho até às 15H de dia 6 de Dezembro (sábado) através do svn.
   Não se esqueçam de fazer `commit` de todos ficheiros com excepção dos ficheiros *.o
   Nota: A operação `svn status` permite identificar os ficheiro ainda não `commited` ou ainda não sob controlo de svn.
2. Salvaguarda dos trabalhos no svn e entrega do trabalho para além da hora
   - às 15H00 será feita uma cópia (`svn copy`) de todos os trabalhos dos alunos
     nota: somente os ficheiros comitados no svn por cada grupo são guardados
   - às 15H30 será feita uma nova e última cópia, que só será considerada pelos grupos que o solicitem através de mail com o assunto "solicito avaliação do trab02 do grupo [group_ID] revisão 15H30". *Neste caso, a nota máxima do trabalho será reduzida para 15 valores*.
   - relembramos que somente os trabalhos submetidos em svn serão considerados

Correcção do exercício

Em cada grupo, foi criada a pasta **trab02** que contém as seguintes pastas (não se esqueçam de fazer `svn update` para os obter!):

```
trab02/src ......................... [user classes: header e source files]
      /main ........................ [main program: contém programa principal]
      /bin ......................... [object files: ficheiros .o e .exe]
      /lib ......................... [user library: libFC.a]
      /rootANA ..................... [analysis macros: eventuais macros de análise]
      Makefile .....................
```

O programa principal de desenvolver deve possuir o nome `main_trab02.C` e estar localizado em `trab02/main`.

A correcção deste exercício será feita usando o Makefile de cada grupo localizada em `trab02/Makefile`. As seguintes acções devem estar definidas:

- regra `make trab02`: deve permitir produzir o ficheiro executável `trab02/bin/main_trab02.exe`
- regra `make lib`: criação da biblioteca `trab02/lib/libFC.a`
- regra `make clean`: apagar todos os ficheiros `trab02/bin/*.o` e `trab02/bin/*.exe` e ainda a biblioteca `trab02/lib/libFC.a`

Para a resolução deste trabalho é necessário desenvolver uma classe em C++ cujo nome é `Vandermonde`; os ficheiros header `trab02/src/Vandermone.h` e código `trab02/src/Vandermonde.C`.
Em relação ao programa principal não existe um guia de realização; este deve realizado para resolver as alíneas abaixo.
Em relação à classe `Vandermonde`, fornece-se de seguida o ficheiro `header` que contém todas os métodos e `data members` que classe deve conter.

```
1  class Vandermonde {
2
3  public:
4
5    // construtors, destructor
6
7    Vandermonde();
8    Vandermonde(unsigned, double, double, const TF1&);
9    ~Vandermonde();
10
11   // getters
12
13   const Vec& GetCoefficients();
14   const FCmatrixFull& GetMatrix();
15   const double* GetX();
16   const double* GetY();
17   double GetCoeffError();
18   void GetInverseMatrix(FCmatrixFull&);
19
20   // graphics getters
21   TGraph& GetGraphPoints(); // return data points graph
22   TF1& GetPolyFunc(); // return polynomial function
23
24  private:
25
26   // setters
27   void SetGraphicsPoints(); // set graphics with data points
28   void SetPolynomialFunction(); // set polynomial TF1 function
29
30   // data members
31   FCmatrixFull MatrixCoefs; //coeff matrix (C)
```

```
32    Vec PolCoefs; // polynomial coeffs (a)
33
34    TF1* fPoly; // polynomial function
35    TGraph gPoints; // point's graphics
36
37    int n; // number of points
38    double* x;
39    double* y;
40
41  };
```

Na implementação da classe `Vandermonde` deve recorrer às classes próprias desenvolvidas por cada grupo como sejam, `FCmatrix`, `FCmatrixFull`, `Vec`, ...

Na correção do trabalho, utilizar-se-á um programa principal realizado pelos docentes, que fará apelo aos métodos da classe `Vandermonde`. O programa será linkado com a biblioteca `lib/libFC.a` de cada grupo, onde devem estar todas as classes necessárias à resolução do problema.

Quotação

alíneas	quotação	observações
	4	- Makefile: teste das regras `trab02 clean lib` - teste da produção da biblioteca libFC.a - teste de compilação do programa realizado pelos docentes usando os métodos implementados na classe Vandermonde
a)	8	
b)	4	
c)	4	
		nota: Na avaliação das alíneas a,b,c) ter-se-ão em conta os resultados obtidos e a qualidade da implementação (comentários ao código e metodologia).

Enunciado

Consideremos o conjunto de n pontos (x_i, y_i) com $i = 0, 1, 2, \cdots, n-1$. É possível fazer passar por todos os pontos do conjunto um polinómio de grau $n-1$,

$$p(x) = a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 + \cdots + a_{n-1}\, x^{n-1}$$

fazendo este obedecer à equação,

$$p(x_i) = y_i$$

Assim a determinação dos coeficientes do polinómio pode ser feita resolvendo o sistema linear,

C a $=$ **b**

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^{n-1} \\ & & & & \vdots & \\ 1 & x_{n-1} & x_{n-1}^2 & x_{n-1}^3 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

A matriz de coeficientes **C** acima, de dimensão $n \times n$, é conhecida como uma matriz de Vandermonde.

a) [8 val]
Suponha que temos um conjunto de $n = 10$ pontos igualmente espaçados entre $[0, 1]$, $\{(x_i, y_i)\}_{i=0}^{n-1}$, obtidos usando a função,

$$y(x) = \sin(2\pi\, x) + 0.002\, \cos(100\, x)$$

que corresponde a uma função seno com uma pequena modulação de ruído.

- Determine o vector de coeficientes do polinómio, **a**
- Determine os gráficos dos pontos e o polinómio interpolado, através respectivamente de, `GetGraphPoints()` e `GetPolyFunc()`.

```
características do gráfico:
- o gráfico deve somente desenhar os pontos (opção "P")
- estilo do ponto: 20
- côr do ponto: 38
- tamanho do ponto: 2

características da função:
- tipo de linha: traço contínuo
- côr da linha do polinómio: kRed+2
- largura da linha do polinómio: 2
```

- Salve no ficheiro ROOT `trab02.root` (que deve ser criado no directório trab02/):
  - o gráfico com os pontos e cujo nome seja `gPoints`
  - a função TF1 interpoladora cujo nome seja `fPoly`
  - o `TCanvas` onde estejam os pontos e a curva interpolada sobrepostos e cujo nome deverá ser `cPoly`,

> O código que será usado para testar o que se pede fará o seguinte:
> - instancia o objecto `Vandermonde` de acordo com o construtor acima definido `Vandermonde(...)`
> - obtém o vector de coeficientes do polinómio usando o método `const Vec& GetCoefficients()` e fará

> a sua impressão no ecran usando o `cout << ...`
> - obtém o gráfico dos pontos usando o método `GetGraphPoints()`
> - obtém a curva interpolada usando o método `GetPolyFunc()`

b) [4 val]
Para se obter uma estimativa do erro associado à determinação dos coeficientes, podemos determinar o majorante do erro ΔE,

$$\Delta E = max\{|D_i|\} \quad \text{com} \quad \mathbf{D} = \mathbf{C}\,\mathbf{a} - \mathbf{b}$$

Este erro deve calculado através do método `GetCoeffError()` onde deve fazer o cáculo do vector **D** recorrendo aos operadores desenvolvidos nas classes `FCmatrix` e `FCmatrixFull` e `Vec`.

> O código que será usado para testar a determinação do erro fará um ciclo em i e executará para cada n o seguinte:
> - Instancia o objecto `Vandermonde` para o conjunto de pontos n
> - Obtém a estimativa do erro dos coeficientes para esse conjunto através de uma chamada ao método `GetCoeffError()`

Realize um gráfico com a evolução do logaritmo (base 10) do erro em função do número de pontos da amostra n, i.e., $log_{10}(\Delta E).vs.n$, com $n = 4(1+i)$ e fazendo i variar de, $i = 0, 1, 2, \cdots, 15$.
Salve o gráfico no ficheiro `trab02.root` sob o nome `gErrors`.

```
características do gráfico:
- deve desenhar os pontos (opção "P") e possuir uma linha a ligá-los (opção "L")
- estilo do ponto: 21
- cor do ponto: 39
- tamanho do ponto: 2
- tipo de linha: traço contínuo
- côr da linha: kGreen+2
- largura da linha: 2
```

c) [4 val]
Determine a matriz inversa dos coeficientes, $\mathbf{C^{-1}}$ para o caso do conjunto de pontos da alínea a), ou seja, 10 pontos igualmente espaçados entre $[0, 1]$.

> O código que será usado para testar fará o seguinte:
> - Obtém a matriz inversa usando o método da classe `void GetInverseMatrix(FCmatrixFull&)`
> - Imprime a matrix inversa no ecran com `cout << ...`

Fim do enunciado do 2º trabalho de Física Computacional