1º Trabalho de Física Computacional (MEFT/IST)

1º semestre 2020-21

Fernando Barão, Jorge Vieira, Miguel Orcinha

Entrega do trabalho

Até às 19H de dia 14 de Novembro (sábado) através do svn.

Não se esqueçam de fazer `commit` de todos ficheiros com excepção dos ficheiros *.o

A operação `svn status` permite identificar os ficheiro ainda não `commited` ou ainda não sob controlo de svn.

Correcção do exercício

Em cada grupo, foi criada a pasta **trab01** que contém as seguintes pastas e ficheiros (não se esqueça de fazer `svn update`):

```
trab01/src ......................... [user classes: Datamanip, DataReader]
      /main ........................ [main program: main_trab01.C]
      /bin ......................... [object files]
      /lib ......................... [user library: libFC.a]
      /rootANA ..................... [analysis macro if needed: mtrab01.C]
      Makefile .....................
      SunspotNumberDATA2020.txt ..... [ficheiro necessário ao executável main_trab01.exe]
```

De notar:

- O programa principal que usarem para testar o vosso código deve estar em main/.
- As classes (ficheiros “header” e “source”) desenvolvidos devem estar na pasta src/.
- O Makefile a usar deve possuir as seguintes tarefas definidas:
  - `make clean`: apagar os ficheiros objecto ”bin/*.o” e a biblioteca “lib/libFC.a”
  - `make trab01`: compilação do programa main **main_trab01.C** existente en main/
  - `make lib`: produção da biblioteca do trabalho `libFC.a`

Passos que serão seguidos na correcção do exercício.

- Usaremos o programa main_trab01.C que se encontra na pasta main/ e cujo conteúdo se lista ao longo das alíneas abaixo.
  Essas linhas de código servem de guião para a construção dos métodos que se pedem no programa e que serão testados através da execução do programa principal.
- Utilizaremos o Makefile de cada grupo existente em trab01 para testar o trabalho, ou seja, criar o executável e a biblioteca
- Correremos o executável de cada grupo situado na pasta do grupo `trab01`, fazendo:
  `./bin/main_trab01.exe`

Quotação

alíneas	quotação
a,b,c,d	15 valores
f,g	5 valores

Enunciado

A actividade solar varia no tempo, possuindo uma correlação directa com um indicador que corresponde ao número de manchas solares que se observam na sua superfície. O número de manchas solares ao longo do tempo é fornecido no ficheiro SunspotNumberDATA2020.txt, que pode ser obtido em sidc[1].

Conteúdo do ficheiro

```
Column 1-3: Gregorian calendar date
- Year
- Month
- Day
Column 4: Date in fraction of year
Column 5: Daily total sunspot number.
          A value of -1 indicates that no number is available for that day (missing value).
Column 6: Daily standard deviation of the input sunspot numbers from individual stations.
Column 7: Number of observations used to compute the daily value.
Column 8: Definitive/provisional indicator.
          - A blank indicates that the value is definitive.
          - A `*` symbol indicates that the value is still provisional and is subject to
        a possible revision (Usually the last 3 to 6 months)
```

Com este trabalho pretende-se fazer a leitura de um sinal que é variável em tempo ($s(t) \equiv$ número de sunspots , ***time series***) e a sua análise.

O trabalho assenta na construção de duas classes C++, *DataReader* e *DataManip*, com a segunda a herdar da primeira.
Os métodos de leitura do ficheiro de dados e os dados devem fazer parte da classe *DataReader*.
Os métodos de análise do sinal devem ser incluídos na classe *DataManip*.

O trabalho será testado usando o programa que se segue (a lista de include files pode não estar completa), e que visa testar a seguinte sequência:

---

a) Implemente a leitura do ficheiro de dados que deve ser realizada na classe **DataReader**, nomeadamente a leitura da coluna 4 do ficheiro (coordenada temporal em anos) e a leitura do número de spots do sol (sunspot number), que se encontra na coluna 5, e armazene a informação ("date in fraction of year", "sunspot number") num *container* `vector<pair<double,double>>` no interior da classe **DataReader**.
Na parte inicial do programa implementamos também um vector de objectos ROOT que permitirá armazenar os objectos ROOT criados no programa. Instanciamos finalmente, após termos testado a existência do ficheiro de dados, a class *DataManip*.
Nota: Existem medidas sem significado físico com o valor -1 nos anos mais antigos (até ao final de 1848). Sugere-se assim que o armazenamento de valores e a análise seja feita a partir do ano de 1849 (inclusive).

```
1  #include "DataReader.h"
2  #include "DataManip.h"
3
4  #include <iostream>
5  #include <iomanip>
6  #include <vector>
7  #include <utility>
```

[1] http://sidc.be/silso/datafiles#total

```
#include <complex>
#include <vector>
#include "TObject.h"
#include "TFile.h"
#include "TGraph.h"
#include "TMultiGraph.h"

int main() {
  // create a ROOT object manager
  // that will allow us to keep track of object created and use it to store them into a file and delete them
  std::vector<TObject*> ROOTmanager;

  // check if data file exists otherwise quit
  if (!DataReader::FileExists("SunspotNumberDATA2020.txt")) exit(1);

  // create DataManip object
  DataManip D("SunspotNumberDATA2020.txt");
```

---

b) Obtenha o vector de dados (coordenada tempo em anos, sunspot number) e o seu gráfico.

```
// get vector with all data read (time, value)
auto V = D.GetDataVector();
std::cout << "number of data points read: " << V.size() << std::endl;

// get a graph (ROOT TGraph) with all data
auto Gdata = D.GetDataGraph(); // use marker style 20, marker size 0.4 and color kBlue+1. the TGraph should be named 'Gdata'
ROOTmanager.emplace_back(Gdata);
```

---

c) Obtenha um novo vector de dados *Vsort* (coordenada tempo em ano fraccionário, sunspot number) que esteja ordenado por ordem decrescente do **sunspot number**.
Imprima para o ecran os 10 valores mais elevados.

```
// make a new vector (value, time) sorted by sunspot number (from higher to lower values)
// vector name: Vsort
auto Vsort = D.GetDataVectorSorted(0); // 0= descending order, 1=ascending order

// print 10 highest values
int c = 0;
for (auto const& x: Vsort ) {
    std::cout << c << " | " << "(" << std::setprecision(7) << x.first << ", " << x.second << ")" << std::endl;
    c++;
    if (c == 10)break;
}
```

---

d) Obtenha as derivadas temporadas temporais em cada ponto, calculando com o auxílio da biblioteca STL as diferenças entre pontos consecutivos.
Obtenha o gráfico das derivadas ao longo do tempo e o histograma dos valores das derivadas.
A fórmula numérica da derivada é:

$$\frac{df}{dt}(x_i) = \frac{f(x_{i+1}) - f(x_i)}{t_{i+1} - t_i}$$

```
46  // get and plot the data time derivatives: df/dx(xi) = [s(i+1) - s(i)] / Delta t_i
47  // use STL to make differences
48  // use again TGraph
49  auto Vderiv = D.GetDataDerivativeVector(); // (time, derivative)
50  auto GdataDeriv = D.GetDataDerivativeGraph(); // use marker style 20, marker size 0.4 and color
        kGreen+2. the TGraph should be named 'GdataDeriv'
51  auto HdataDeriv = D.GetDataDerivativeHisto(); // the Histogram should be named 'HdataDeriv'
52  ROOTmanager.emplace_back(GdataDeriv);
53  ROOTmanager.emplace_back(HdataDeriv);
```

---

e) As variações do sinal com pequeno intervalo de tempo correspondem a variações de alta frequência. A filtragem destas frequências altas pode ser conseguida através da realização da média deslizante com M medidas. Assim, para um sinal amostrado (sampled) em tempo N vezes,

$$\{s_0, s_1, s_2, \cdots, s_{N-3}, s_{N-2}, s_{N-1}\}$$

vamos criar um sinal médio com o mesmo número de N elementos, de acordo com o seguinte algoritmo:

$$\bar{s}_M = \{s_0, s_1, \cdots, s_{\frac{M}{2}-1}, \frac{1}{M}\sum_{i=0}^{M-1} s_i, \frac{1}{M}\sum_{i=1}^{M} s_i, \cdots, s_{N-\frac{M}{2}}, s_{N-\frac{M}{2}+1}, \cdots, s_{N-1}\}$$

No código que se segue deve implementar os métodos que permitam:

- obter os gráficos dos sinais médios com $M = 11, 181$ dias cujas características gráficas devem ser as seguintes:
  usar a opção "L" do `TGraph` e ainda definir as cores de acordo com 11 dias=`kYellow`, 181 dias=`kMagenta`
- obter um gráfico `TMultiGraph` onde o sinal original é representado por `MarkerStyle=20`, `MarkerSize=0.4` e `MarkerColor=kBlue+1` e os sinais médios apareçam sobrepostos com as cores e opções definidas anteriormente
- obter um gráfico `TMultiGraph` onde os sinais médios apareçam sobrepostos com as cores e opções definidas anteriormente

```
55  // A way of smoothing a signal (get rid of very high frequencies) is making a moving average
56  // Get a vector with the moving average time array (time, moving average signals)
57  auto Vsmooth181 = D.GetMovingAverage(181);
58  auto Gsmooth11 = D.GetMovingAverageGraph(11, "g11"); // args: números de dias, nome do objecto.
        LineColor=kYellow
59  auto Gsmooth181 = D.GetMovingAverageGraph(181, "g181"); // args: números de dias, nome do objecto.
        LineColor=kMagenta
60
61  // make a new graph superimposing the two graphs: Gdata and Gsmooth
62  TMultiGraph *G1 = new TMultiGraph("Gmult1", "Gmult1");
63  G1->Add(Gdata, "P"); //only points
64  G1->Add(Gsmooth11, "L"); // lines
65  G1->Add(Gsmooth181, "L"); // lines
66
67  // make a new graph superimposing the two graphs: Gdata and Gsmooth
68  TMultiGraph *G2 = new TMultiGraph("Gmult2", "Gmult2");
69  G2->Add(Gsmooth11, "L"); // lines
70  G2->Add(Gsmooth181, "L"); // lines
71
72  ROOTmanager.emplace_back(G1);
73  ROOTmanager.emplace_back(G2);
```

```
74  ROOTmanager.emplace_back(Gsmooth11);
75  ROOTmanager.emplace_back(Gsmooth181);
```

---

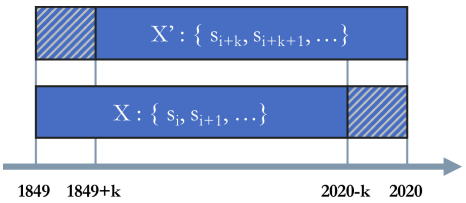
f) Uma outra forma de analisarmos a periodicidade temporal de um sinal é através do estudo das auto-correlações, ou seja, a correlação cruzada com o próprio sinal. Para isso, tomamos uma sub-amostra de dados composta de n elementos

$$X : \{s_i, s_{i+1}, s_{i+2}, \cdots, s_{i+n-2}, s_{i+n-1}\}$$

que consideremos representativos da amostra total (ou seja, que possua a variabilidade temporal que procuramos) e vamos procurar ver a sua relação com uma outra sub-amostra consecutiva com o mesmo número de elementos n e que se inicie num outro instante avançado, ou seja, após um certo "time shift" k

$$X' : \{s_{i+k}, s_{i+k+1}, s_{i+k+2}, \cdots, s_{i+k+n-2}, s_{i+k+n-1}\}$$

A maneira gráfica de ver esta correlação será fazer um gráfico simultâneo dos pontos das duas sub-amostras: $\{(s_i, s_{i+k}), (s_{i+1}, s_{i+k+1}), \cdots, (s_{n-k-1}, s_{n-1})\}$. Caso existisse uma boa correlação linear, veríamos os pontos fortemente alinhados ao longo de uma recta. Quanto mais forte este alinhamento maior a sua correlação.



Em termos estatísticos, a forma de medir esta correlação entre as sub-amostras cuja composição depende do "time shift" k, $X = [1849, 2020 - k]$ e $X' = [1849 + k, 2020]$, faz-se usando o seguinte coeficiente:

$$R = \frac{E[(x_i - \mu_1)(x_{i+k} - \mu_2)]}{\sigma_1 \sigma_2} = \frac{\sum_i [(x_i - \mu_1)(x_{i+k} - \mu_2)]}{\sqrt{\sum_i (x_i - \mu_1)^2 \sum_i (x_{i+k} - \mu_2)^2}}$$

onde:

- μ_1: valor médio da amostra X
- μ_2: valor médio da amostra X'
- σ_1, σ_2: desvios padrão das duas amostras

Procure determinar as componentes periódicas do sinal estudando as correlações de pequeno período (até 1 ano) e de grande período (anos). O factor chave para a identificação das frequências é o passo temporal Δt que irá utilizar na análise do sinal. Note que o passo mínimo de tomada do sinal é 1 dia.

No código que se segue propõe-se assim que determine e faça gráficos:

- do coeficiente de auto-correlação do sinal para grandes períodos de tempo
  utilize um "time shift" k que varie no intervalo $[1, 25]$ anos e com um passo temporal grande ($\Delta t = 1/4$ ano).
- do coeficiente de auto-correlação do sinal para pequenos períodos de tempo
  utilize um "time shift" k que varie no intervalo $[0, 1]$ ano e com um passo temporal pequeno ($\Delta t = 1$ dia, ou seja, $\Delta t = 1/365$ ano).

Faça os gráficos que mostrem a evolução do coeficiente de correlação com o k para o passo temporal grande e pequeno.

Nota: Os objectos TGraph possuem por defeito todos o mesmo nome ("graph"). Por isso para que possam ser armazenados no ficheiro com nomes diferentes, devem ser renomeados, de acordo com os argumentos que se mostram de seguida.

```
77  auto Gcorr = D.GetAutocorrelationGraph(1.,25.,1./4, "Gcorr"); // use marker style 22, marker size 0.4
        and color kGreen+2. the TGraph should be named 'Gcorr'
78  auto Gcorr_detailed = D.GetAutocorrelationGraph(0.,1.,1./365, "Gcorr_detailed"); // use marker style
        22, marker size 0.4 and color kGreen+4. the TGraph should be named 'Gcorr_detailed'
79  ROOTmanager.emplace_back(Gcorr);
80  ROOTmanager.emplace_back(Gcorr_detailed);
```

Com base nos dois gráficos anteriores determine as componentes principais periódicas do sinal. Escreva e justifique a sua resposta no ficheiro `grupoID_trab01.txt`.

Nota final do exercício

No programa main a ser testado que inclui todas as linhas de código aqui listadas, é ainda realizada a salvaguarda de todos os objectos ROOT para um ficheiro *ftrab01.root*.

Fim do enunciado do 1º trabalho de Física Computacional