# Exercícios de

# Física Computacional

(Parte 7)

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

# 7. Derivação, Integração e Raízes

**Exercício 49**. Para o cálculo de derivadas numericamente, pretendemos definir primeiro uma classe de base ***Func1D***. Esta classe ***Func1D***, é definida conforme indicado no *header* file abaixo

```
class Func1D{
public:
       Func1D(TF1* ff=NULL);
       ~Func1D();
       void SetFunc(TF1*);

       TF1* GetFunc() const;
       void Draw();
       double Evaluate(double x);
protected:
       TF1* F;
       static int Nplots;
};
```

a) Implemente os métodos da classe classe. Pode (*e deve*) código *semelhante* ao usado para a classe ***Interpolator*** no que diz respeito ao método *Draw*.

b) Para testar a classe, elabore um programa *main* onde defina um objecto da classe ***Func1D*** com a função ***f(x)=cos(x)*** no intervalo [-2,2], mostre o gráfico no écran e crie o respectivo ficheiro com o gráfico ("***func1d.eps***").

**Exercício 50**. Vamos agora então definir e implementar uma classe ***Derivator***, que deriva da classe ***Func1D***. A definição apresenta-se abaixo.

```
class Derivator: public Func1D {

 public:
  Derivator(TF1 *f=NULL);
  ~Derivator();

  double Deriv_1(double x, double h, int type=0);
  double Deriv_2(double x, double h, int type=0);
  double Deriv_3(double x, double h, int type=0);
  double Deriv_4(double x, double h, int type=0);

};
```

a) Implemente os métodos da classe. Os input *h* e *type* são, respectivamente, o incremento usado no cálculo da derivada e o *tipo* de método usado i.e. central (0), forward (1) e backward (2).
b) Teste agora a sua classe ***Derivator*** usando como função a ***mesma*** usada no Exercício 49. Crie o gráfico da derivada no intervalo [-2,2] e o respectivo ficheiro ***derivator_func1d.eps***. *Não é necessário que a derivada de f(x) seja também ela um objecto da classe Func1D* (pode porém tentar implementar tal funcionalidade)

**Exercício 51** : Dando um passo adiante, pretendemos agora implementar a classe Integrator cuja definição se mostra abaixo:

```
class Integrator: public Func1D {

 public:
  Integrator(double xbeg=0, double xend=0, TF1* func=NULL) :
  x0(xbeg), x1(xend), Func1D(func) {;}
  ~Integrator() {;}
  void SetBoundaries(double x_0,double x_1);
  void TrapezoidalRule(int n, double& result, double& error);
  void SimpsonRule(int n, double& result, double& error);

 protected:
  double x0;
  double x1;
};
```

a) Implemente os ***métodos*** da classe para permitir o ***cálculo*** de ***integrais*** e estimativa do seu respectivo ***erro*** numérico. Os limites de integração são dados pelos membros **x0** e **x1**. Podem ser dados não só no construtor mas também através do método *SetBoundaries*.
b) Para testar a sua classe e métodos, elabore um programa *main* (ou então use e extenda o do exercício anterior) para calcular o integral (*e respectivo erro*)

$$I = \int_0^{\pi/2} \cos(x)\,dx$$

Note que para estimar o erro do integral poderá (*terá*) que utilizar as derivadas implementadas na classe do exercício anterior...

**Exercício 52** : (*adaptado de Barão 2016)*. O *ponto de Lagrange* é o local entre a Terra e a Lua onde um satélite aí colocado, possuirá uma órbita em sincronia total com a Lua. Do ponto de vista da dinâmica, nesse ponto o balanço das forças gravíticas da Terra e da Lua produzem uma força resultante que assegura a manutenção do satélite na órbita. Assumindo órbitas circulares, a distância radial (**r**) a partir do centro da Terra a que se encontra o satélite, obedece à seguinte equação:

$$\frac{GM_{Earth}}{r^2} - \frac{GM_{Moon}}{(R-r)^2} = \omega^2 r$$

onde $\omega$ é a velocidade angular do satélite e da Lua, R distância Terra-Lua e as massas da Terra e Lua dadas por $M_{Earth}$ e $M_{Moon}$ .

Construa um programa em C++, utilizando ***vários métodos de determinação de raízes***, para calcular numericamente o raio orbital **r**.