# Relatório Trabalho Prático
# Análise Numérica I

Pilar Trindade 54855 e Rita Rodrigues 54859

## 1 Introdução

Neste trabalho elaborámos duas funções Octave para calcular e ilustrar graficamente a aproximação de um conjunto de dados discretos por uma função de um tipo previamente definido.
Os dados são constituídos por uma matriz de tipo $n \times 2$, cuja primeira coluna contém as abcissas dos pontos e que na segunda coluna contém as respetivas ordenadas.

## 2 Desenvolvimento do código

As funções que desenvolvemos em Octave foram as funções `regressao` e `graficos`.
A função `regressao` recebe como parâmetros `X` e `k`. `X` é uma matriz $n \times 2$ cuja primeira coluna contém as abcissas dos pontos e que na segunda coluna contém as respetivas ordenadas. `k` é um inteiro entre 1 e 4 e indica qual o tipo de função a ajustar. Devolve `c`, um vetor que contém os coeficientes $a$ e $b$ da função que melhor se ajusta aos dados.
A função `graficos` recebe `X`, `k` e `c` (caracterizados como na função anterior), e gera, numa janela gráfica, um gráfico com os dados originais representados apenas com um marcador e, sobreposto, um gráfico da função do tipo especificado com os coeficientes contidos no vetor `c`.

## 2.1 Função 1

Primeiro aproximámos os dados por uma função do tipo $y = a + bx$. Dados $(x_i, y_i)\, i = 0, ..., n$, procuramos uma reta da forma $y = a + bx$. Queremos minimizar a soma dos quadrados das distâncias na vertical:

$$\phi(a, b) = \sum_{i=0}^{n} (y_i - (a + bx))^2$$

$\phi$ tem um mínimo que se pode determinar procurando o ponto onde o gradiente de $\phi$ se anula:

$$\frac{\delta\phi}{\delta a}(c_1, c_2) = 0 \qquad \frac{\delta\phi}{\delta b}(c_1, c_2) = 0$$

Vamos então calcular o gradiente de $\phi$.

$$\frac{\delta\phi}{\delta a}(a, b) = \sum_{i=0}^{n} 2(y_i - (a + bx_i))(-1) = 2\sum_{i=0}^{n} (-y_i + a + bx_i)$$

$$\frac{\delta\phi}{\delta b}(a, b) = \sum_{i=0}^{n} 2(y_i - (a + bx_i))(-x_i) = 2\sum_{i=0}^{n} (-x_i y_i + ax_i + bx_i^2)$$

E agora vamos calcular o ponto $(c_1, c_2)$ que minimiza $\phi$.

$$\frac{\delta\phi}{\delta a}(c_1, c_2) = 0 \qquad \frac{\delta\phi}{\delta b}(c_1, c_2) = 0$$

$$\frac{\delta\phi}{\delta a}(c_1, c_2) = -\sum_{i=0}^{n} y_i + c_1 \sum_{i=0}^{n} 1 + c_2 \sum_{i=0}^{n} x_i = 0$$

$$\frac{\delta\phi}{\delta b}(c_1, c_2) = -\sum_{i=0}^{n} x_i y_i + c_1 \sum_{i=0}^{n} x_i + c_2 \sum_{i=0}^{n} x_i^2 = 0$$

Obtemos então um sistema linear:

$$\left(\sum_{i=0}^{n} 1\right) c_1 + \left(\sum_{i=0}^{n} x_i\right) c_2 = \sum_{i=0}^{n} y_i$$

$$\left(\sum_{i=0}^{n} x_i\right) c_1 + \left(\sum_{i=0}^{n} = x_i^2\right) c_2 = \sum_{i=0}^{n} x_i y_i$$

isto é, $\begin{bmatrix} n+1 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix} \quad \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} x_i y_i \end{bmatrix}$

Resolvendo este sistema, obtemos a reta de regressão $y = c_1 + c_2 x$. Foi isso que fizemos no Octave.
Primeiro definimos a matriz: $A = \begin{bmatrix} n+1 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix}$ escrevendo o código:

```
A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)]
```

Nota: `size(X)` corresponde a um vetor cuja primeira coordenada é o número de linhas da matriz `X`, assim, obtemos o número de pontos que a matriz contém.
Depois definimos a matriz: $b = \begin{bmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} x_i y_i \end{bmatrix}$ escrevendo o código:

```
b = [sum(X)(2); sum(prod(X,2))]
```

Para resolvermos o sistema basta introduzir a seguinte linha de código:

```
c = A\b
```

O vetor c tem coordenadas $(a, b)$, logo para definirmos a função $f(x) = a+bx$, basta correr o código:

```
f = @(x) c(1)+c(2)*x
```

## 2.2 Função 2

Para aproximar os dados por uma função do tipo $y = ab^x$, considerámos a transformação gerada pela aplicação do logaritmo natural:

$$\log y = \log a + x \log b$$

Transformámos os dados iniciais em $(x_i, \log y_i)$ através do código:

```
X(:,2) = log(X(:,2))
```

Depois definimos as matrizes $A$ e $b$ e aplicámos a regressão linear que aplicámos na Função 1.

```
A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
b = [sum(X)(2); sum(prod(X,2))];
a = A\b;
```

O vetor a tem coordenadas $(\log a, \log b)$. Como $e^{\log a} = a$ e $e^{\log b} = b$, para obtermos $a$ e $b$, definimos o vetor c como `c = [e^a(1) e^a(2)]`. Assim o vetor c tem coordenadas $(a, b)$, logo para definirmos a função $f(x) = ab^x$, basta correr o código:

```
f = @(x) c(1)*c(2).^x
```

## 2.3 Função 3

Para aproximar os dados por uma função do tipo $y = ax^b$, considerámos a transformação gerada pela aplicação do logaritmo natural:

$$\log y = \log a + b \log x$$

Transformámos os dados iniciais em $(\log x_i, \log y_i)$ através do código:

```
X = log(X)
```

Depois definimos as matrizes $A$ e $B$ e aplicámos a regressão linear que aplicámos na Função 1.

```
A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
b = [sum(X)(2); sum(prod(X,2))];
a = A\b;
```

O vetor a tem coordenadas $(\log a, b)$. Como

$$e^{\log a} = a$$

Para obtermos a e b, definimos o vetor c como `c = [e^a(1) a(2)]`. Assim o vetor c tem coordenadas $(a, b)$, logo para definirmos a função $f(x) = ax^b$, basta correr o código:

```
f = @(x) c(1)*x.^c(2)
```

## 2.4 Função 4

Para aproximar os dados por uma função do tipo $y = \frac{1}{a+bx}$, considerámos a transformação gerada pela inversão:

$$\frac{1}{y} = a + bx$$

Transformámos os dados iniciais em $(x_i, \frac{1}{y_i})$ através do código:

```
X(:,2) = 1./X(:,2)
```

Depois definimos as matrizes $A$ e $B$ e aplicámos a regressão linear que aplicámos na Função 1.

```
A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
b = [sum(X)(2); sum(prod(X,2))];
c = A\b
```

O vetor c tem coordenadas $(a, b)$, logo para definirmos a função $f(x) = \frac{1}{a+bx}$, basta correr o código:

```
f = @(x) 1./(c(1)+c(2)*x)
```

## 2.5 Organização do código

Na função `regressão` construímos um comando condicional para se escolher o tipo de função. Assim, o código final da função `regressão` é:

```
function c = regressao (X, k)
  if k == 1
    A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
    b = [sum(X)(2); sum(prod(X,2))];
    c = A\b;
  elseif k == 2
    X(:,2) = log(X(:,2));
    A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
    b = [sum(X)(2); sum(prod(X,2))];
    a = A\b;
    c = [e^a(1) e^a(2)];
  elseif k == 3
    X = log(X);
    A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
    b = [sum(X)(2); sum(prod(X,2))];
    a = A\b;
    c = [e^a(1) a(2)];
  elseif k == 4
    X(:,2) = 1./X(:,2);
    A = [size(X)(1) sum(X)(1); sum(X)(1) sum(X.^2)(1)];
    b = [sum(X)(2); sum(prod(X,2))];
    c = A\b;
  end
endfunction
```

Na função `graficos`, para os gráficos dos polinómios ajustados gerámos um vetor com bastantes pontos. Para isso executámos:

```
xx = linspace(X(1,1),X(end,1),101)
```

Depois de um comando condicional onde é escolhida e definida a função que se quer desenhar, corre-se a linha

```
plot(X(:,1),X(:,2),"ok","linewidth",2,xx,f(xx),"r","linewidth",2)
```

para realmente desenhar o gráfico. Em conclusão, o código final da função `graficos` é o que se encontra na página seguinte.

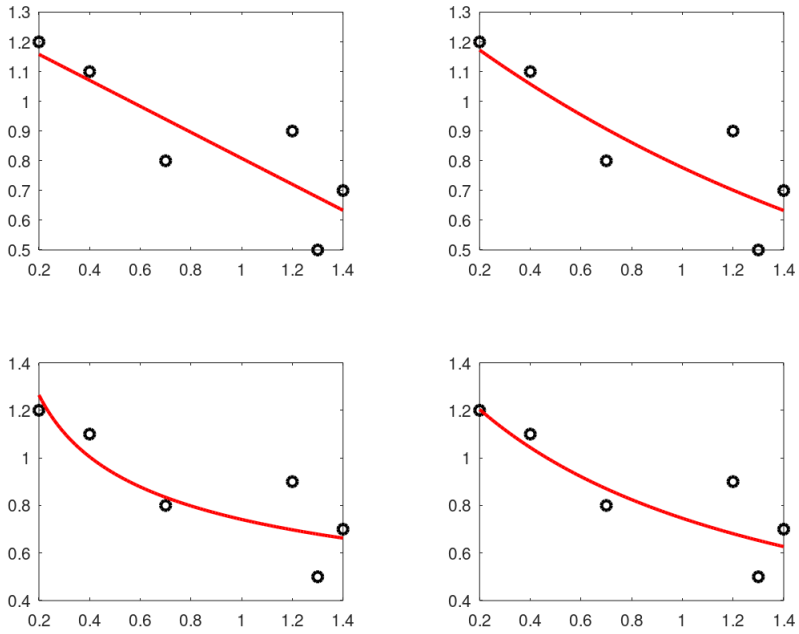
```
function graficos (X,c,k)
  xx = linspace(X(1,1),X(end,1),101);
  if k == 1
    f = @(x) c(1)+c(2)*x;
  elseif k == 2
    f = @(x) c(1)*c(2).^x;
  elseif k == 3
    f = @(x) c(1)*x.^c(2);
  elseif k == 4
    f = @(x) 1./(c(1)+c(2)*x);
  end
  plot(X(:,1),X(:,2),"ok","linewidth",2,xx,f(xx),"r","linewidth",2)
endfunction
```

# 3 Testes desenvolvidos e conclusão

No final do desenvolvimento do código executámos o ficheiro de testes sugerido pelo professor.

```
x=[0.2 0.4 0.7 1.2 1.3 1.4]
y=[1.2 1.1 0.8 0.9 0.5 0.7]
X=[x' y'];

c1=regressao(X,1)
c2=regressao(X,2)
c3=regressao(X,3)
c4=regressao(X,4)

subplot(2,2,1)
graficos(X,c1,1)

subplot(2,2,2)
graficos(X,c2,2)

subplot(2,2,3)
graficos(X,c3,3)

subplot(2,2,4)
graficos(X,c4,4)

print -dpng figura1.png
```

Os gráficos obtidos foram os seguintes:




Os gráficos estão de acordo com os resultados que tínhamos pensado previamente.

# 4 Versão

- O trabalho foi executado na versão 5.2.0 do Octave.

# 5 Bibliografia

- Para desenvolvermos este trabalho recorremos aos pdf disponíveis na página moodle da cadeira.