MÉTODOS NUMÉRICOS PARA AJUSTE DE CURVAS

Evandro Pedro Alves de Mendonça[a], **Marcelino José de Lima Andrade**[a].

[a] *Núcleo de Tecnologia (NTI), Universidade Federal de Pernambuco (UFPE), Campus Acadêmico do Agreste (CAA), Rodovia BR-104, km 59, S/N, Nova Caruaru, CEP. 55.014-900, Caruaru-PE, Brasil, http://www.ufpe.br/caa*

Palavras Chave: ajuste de curvas, resíduos, polinômios, aproximação, dados experimentais, regressão linear.

Resumo: Em diversas situações no cotidiano científico, é necessário encontrar uma fórmula matemática que represente um conjunto finito de pontos. Este procedimento pode ser realizado através do ajuste de curvas, nele é possível realizar aproximações para um conjunto de dados ou uma função através de diversas funções simples. Neste trabalho serão exploradas as aproximações lineares através de uma reta (polinômio de primeiro grau), linearização de funções não lineares e, por fim, de um polinômio de segundo grau (cujo procedimento pode ser estendido para polinômios de graus superiores).

1 INTRODUÇÃO

Muitos estudos e observações são feitos a partir de experimentos, onde os resultados obtidos são analisados e guardados. Algumas vezes, esses dados são armazenados quase que continuamente fazendo com que seja fácil a dedução de alguma função que represente bem aquele conjunto de dados. Porém, geralmente não há espaço suficiente para guardar dados contínuos ou simplesmente não é possível obter tais dados de forma contínua. Por isso é necessário gravar um conjunto de pontos discretos.

Portanto, é preciso definir uma função, ou seja, uma curva que defina esses pontos da melhor forma possível, pois, a partir de uma equação geral fica mais fácil determinar a solução de determinado problema. Existem diversas formas de determinar uma curva para certo conjunto de dados como a interpolação e o ajuste de curvas.

Nesse trabalho é usado um método de ajuste de curvas, o método dos mínimos quadrados, que busca diminuir a diferença entre os valores reais e os valores aproximados obtidos por uma função ajustada aos pontos. Os erros, ou resíduos, são definidos como uma função, que é minimizada por meio de derivadas parciais. Mais detalhes sobre o método serão abordados na discussão dos resultados.

2 EXERCÍCIOS PROPOSTOS

Segue, abaixo, a solução dos exercícios propostos sobre o tema.

2.1 1ª questão

Dados:

```
x =

    0.2000    0.3000    0.6000    0.9000    1.1000    1.3000    1.4000    1.6000
```

Figura 1: Vetor linha que contém todos os pontos no eixo x usados para realizar as aproximações por funções.

```
y =

    0.0504    0.0984    0.3328    0.7266    1.0972    1.5697    1.8487    2.5015
```

Figura 2: Vetor linha que contém os valores de Y para cada respectivo X.

1.1.1 1.a)

- Aproximando os pontos dados nos vetores X e Y para uma função do tipo:

$$F(x) = a + bx$$

Usando o MMQ (Método dos Mínimos Quadrados – Anexo 1), temos:

```
>> [coef_a erro_a] = ajuste(x,y,1)

coef_a =

  -0.512456823999999
   1.665540079999999


erro_a =

   0.335589855759488
```

Figura 3: coeficientes a e b (de cima para baixo) da curva linear e o erro calculado.

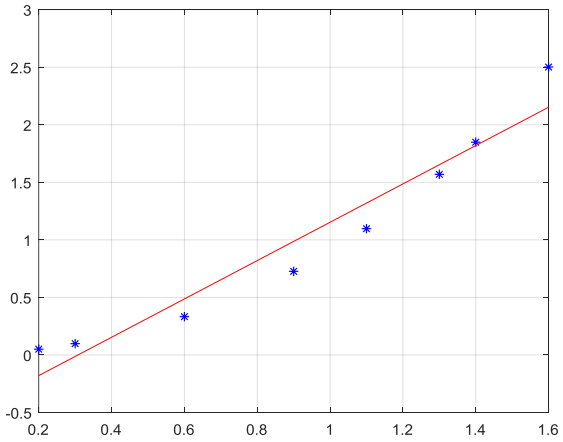

Figura 4: gráfico contendo os pontos (x,y) e a curva linear obtida que se ajusta a esses pontos.

❖ Aproximando os pontos dados nos vetores X e Y para uma função do tipo:

$$\boldsymbol{F(x) = a + bx + cx^2}$$

Usando o MMQ, temos:

```
>> [coef_b erro_b] = ajuste(x,y,2)

coef_b =

   0.085143932517382
  -0.311403456831008
   1.129423867019542


erro_b =

   0.002419914929427
```

Figura 5: coeficientes a, b e c (de cima para baixo) da curva quadrática e o erro calculado.

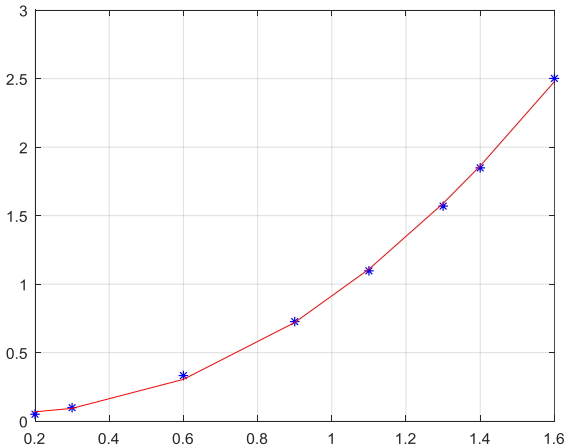

Figura 6: gráfico contendo os pontos (x,y) e a curva quadrática obtida que se ajusta a esses pontos.

❖ Aproximando os pontos dados nos vetores X e Y para uma função do tipo:

$$\boldsymbol{F(x) = a + bx + cx^2 + dx^3}$$

Usando o MMQ, temos:

```
>> [coef_c erro_c] = ajuste(x,y,3)

coef_c =

  -0.018401399296137
   0.248385784190107
   0.402932213098215
   0.266208097739326


erro_c =

    5.074671555628365e-06
```

Figura 7: coeficientes a, b, c e d (de cima para baixo) da curva cúbica e o erro calculado.

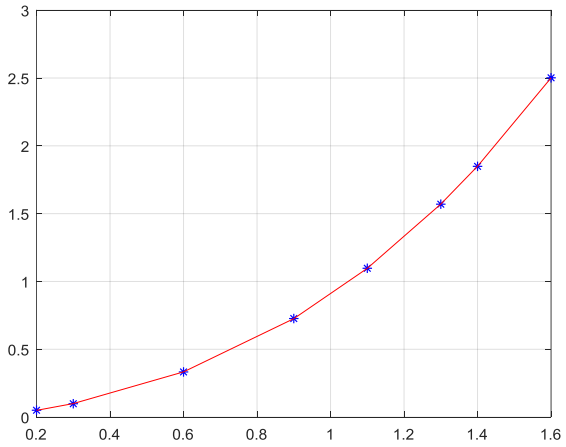

Figura 8: gráfico contendo os pontos (x,y) e a curva cúbica obtida que se ajusta a esses pontos.

❖ Aproximando os pontos dados para uma função do tipo:

$$\boldsymbol{F(x) = be^{ax}}$$

Para poder utilizar o MMQ nesta equação, que não é linear, precisamos linearizá-la. Para isso, fazemos:

$$F(x) = be^{ax} \therefore ln[F(x)] = \ln(be^{ax}) = \ln(b) + \ln(e^{ax}) = \ln(b) + ax$$

Assim, temos uma função linear

$$y = b_1 + ax$$

Onde

$$y = ln[F(x)];\ b_1 = \ln(b) \therefore b = e^{b_1}$$

Usando o MMQ, temos:

```
>> [coef_d erro_d] = ajuste(x,log(y),1)

coef_d =

  -3.085493303114047
   2.707294686913417


erro_d =

   0.593864483463528
```

Figura 9: coeficientes b_1 e a (de cima para baixo) da curva $y = b_1 + ax$ e o erro calculado.

Para descobrir o valor do coeficiente b, usamos a equação $b = e^{b_1}$.

```
>> b = exp(coef_d(1))

b =

   0.045707480695330
```

Figura 10: valor do coeficiente b.

A seguir, será mostrado o gráfico da curva original com os coeficientes encontrados.

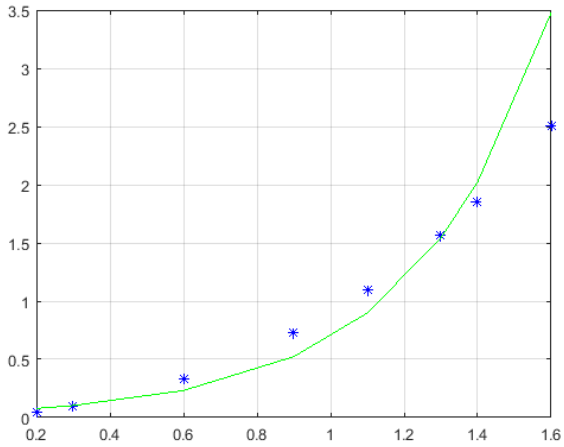

Figura 11: gráfico contendo os pontos (x,y) fornecidos (em azul) e a função não linear que se ajusta aos pontos fornecidos (em verde).

- Aproximando os pontos dados para uma função do tipo:

$$\boldsymbol{F(x) = bx^a}$$

Aqui também realizamos a linearização:

$$F(x) = bx^a \therefore ln[F(x)] = \ln(bx^a) = \ln(b) + \ln(x^a) = \ln(b) + aln(x)$$

Assim, temos uma função linear

$$y = b_1 + ax_1$$

Onde

$$y = ln[F(x)];\ b_1 = \ln(b) \therefore b = e^{b_1};\ x_1 = \ln(x)$$

Usando o MMQ, temos:

```
>> [coef_e erro_e] = ajuste(log(x),log(y),1)

coef_e =

  -0.051128596801404
   1.872009284326521


erro_e =

   0.030170356786041
```

Figura 12: coeficientes b_1 e a (de cima para baixo) da curva $y = b_1 + ax_1$ e o erro calculado.

Para descobrir o valor do coeficiente b, usamos a equação $b = e^{b_1}$.

```
>> b = exp(coef_e(1))

b =

   0.950156475592062
```

Figura 13: valor do coeficiente b.

A seguir, será mostrado o gráfico da curva original com os coeficientes encontrados.

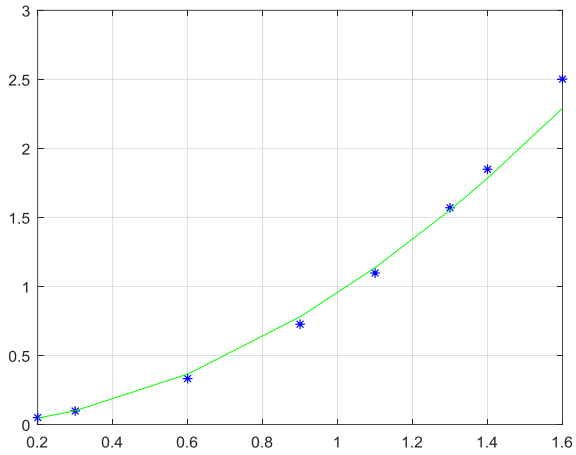

Figura 14: gráfico contendo os pontos (x,y) fornecidos (em azul) e a função não linear que se ajusta aos pontos fornecidos (em verde).

1.1.2 1.b)

Conforme o item anterior, vimos que o valor do erro foi mostrado para cada caso. O caso que apresentou menor erro foi o da aproximação por função polinomial cúbica, com erro da ordem de 10^{-6}. O erro é calculado pelo quadrado da diferença entre o valor da imagem de um ponto e a imagem do mesmo ponto feito pela função de aproximação. Detalhes sobre como esse cálculo foi feito no código, vide Anexo 1.

1.1.3 1.c)

Conforme o item anterior, vimos que o valor do erro foi mostrado para cada caso. O caso que apresentou menor erro foi o da aproximação por função polinomial cúbica, com erro da ordem de 10^{-6}. O erro é calculado pelo quadrado da diferença entre o valor da imagem de um ponto e a imagem do mesmo ponto feito pela função de aproximação. Detalhes sobre como esse cálculo foi feito no código, vide Anexo 1.

1.1.4 1.d)

Usando a função *polyfit* do MATLAB®, obtivemos resultados semelhantes aos encontrados com a função *ajustes*. A seguir, vemos uma tabela que compara os tempos de processamento das duas funções.

Função	***ajustes***	***polyfit***
Linear	0,102323	0,016814
Quadrática	0,105399	0,007898
Cúbica	0,116782	0,051067
Exponencial em e	0,108034	0,078759
Exponencial em x	0,183830	0,005055

Tabela 1: tempo de processamento das funções de ajustes de curvas, em segundos.

Os valores obtidos pela função *polyfit* podem ser vistos no Anexo 4. Quanto ao tempo de processamento, devemos levar em conta que a função *ajustes* realizava também a plotagem a curva gerada pelo algoritmo, o que não acontece na função *polyfit*. Por isso, tem maior tempo de processamento. Porém, mesmo assim, a função *polyfit* tem tempo muito menor que *ajustes*. O que já era esperado, por ser uma função bastante otimizada pela equipe de desenvolvimento do MATLAB®. É claro que, para essa situação, não faz a menor diferença, esse ganho de tempo de processamento, mas para um volume de dados maior, fará sim toda a diferença.

2.2 2ª questão

Foram coletados 30 preços de motos com as características exigidas pelo comprador. Os preços, anos e quilometragens dos carros pesquisados estão descritos na tabela 2 do anexo 5.

Deseja-se ter uma estimativa do preço de uma moto em um ano aleatório e com quilometragem a ser definida por meio de uma função que represente tais valores. Pode-se fazer isso pelo método dos mínimos quadrados com duas variáveis, são elas, o ano e a quilometragem. Para isso, colocou- se em um vetor, chamado km, a quilometragem de cada moto pesquisada. Em outro vetor, chamado ano, colocou-se o ano correspondente a cada moto pesquisada. A função preço fica da seguinte forma:

$$\boldsymbol{Preço = a + b \times Ano + c \times Km}$$

Para determinar os coeficientes a, b e c, utiliza-se o método dos mínimos quadrados, que fica definido no sistema linear, na forma matricial abaixo:

$$\boldsymbol{C \times I = R}$$

- Onde C é a matriz dos coeficientes, dada por:

$$\begin{bmatrix} \langle n,n\rangle & \langle n,Ano\rangle & \langle n,Km\rangle \\ \langle Ano,n\rangle & \langle Ano,Ano\rangle & \langle Ano,Km\rangle \\ \langle Km,n\rangle & \langle Km,Ano\rangle & \langle Km,Km\rangle \end{bmatrix}$$

n - Número de motos (Vetor unitário).

Fazendo todos os produtos internos, a matriz dos coeficientes fica:

$$\begin{bmatrix} 30 & 60365 & 1144814 \\ 60365 & 121464565 & 2.3026 \times 10^{9} \\ 1144814 & 2{,}3026 \times 10^{9} & 6{,}0634 \times 10^{10} \end{bmatrix}$$

- I é a matriz das incógnitas, ou seja, as constantes a, b e c que se quer descobrir.
- R é o vetor das constantes, dado por:

$$\begin{bmatrix} \langle Preço,n\rangle \\ \langle Preço,Ano\rangle \\ \langle Preço,Km\rangle \end{bmatrix}$$

preço - vetor com todos os preços.

Fazendo os produtos internos, a matriz das respostas fica:

$$\begin{bmatrix} 270150 \\ 543669630 \\ 9{,}3330 \times 10^{9} \end{bmatrix}$$

O algoritmo para o cálculo dos produtos internos (discretos) está no anexo 3.

Substituindo os valores encontrados e aplicando no algoritmo da Eliminação de Gauss, obtemos os valores de a,b e c, que são mostrados abaixo como X1,X2 e X3 respectivamente:

X1 = -784291.1102713822
X2 = 394.9192073860
X3 = -0.0353023685

O algoritmo para o cálculo da Eliminação de Gauss está no anexo 2.

Desta forma, tem-se a função preço dada da seguinte forma:

Preço = -784291,1102713822 + 394,9192073860 * Ano – 0,0353023685 * Km

2.2.1 2.a)

Aplicando parâmetros da moto na fórmula encontrada acima, encontra-se o seguinte resultado:

```
(-784291.1102713822)+(394.9192073860*2009)-(0.0353023685*87000)

ans =

     6.030271307591784e+03
```

Isto é, o valor da moto é: R$6030,27

2.2.2 2.b)

Com o valor da venda calculada no item anterior mais os R$5.000,00 fornecidos, totaliza R$11.030,27, assim, as opções de motos para comprar com menor tempo de uso estão listadas na tabela 3 do anexo 5.

Em relação aos anos, ele teria uma quantidade considerável de escolhas para comprar, porém, algumas mesmo sendo mais novas têm mais quilômetros rodados, deve-se levar em conta isso também.

2.2.3 2.c)

Para descobrir a depreciação a cada 10.000 km rodados, basta multiplicar esse número pela derivada parcial da função Preço com respeito a Q, que significa a taxa de variação do preço a cada quilômetro rodado:

$$\frac{\partial P(A,Q)}{\partial Q} = -0.0353023685$$

Depreciação_km = (-0.0353023685) x 10.000 = -353,023

Ou seja, o valor da moto perde R$353,023 a cada 10.000km rodados.

2.2.4 2.d)

O fator que multiplica o ano da moto tem por base a diferença entre o ano da moto e o ano atual. Assim, para saber a depreciação em um ano, pode ser feito o seguinte cálculo:

$$\frac{\partial P(A,Q)}{\partial A} = 394{,}9192073860$$

Depreciação_ano = (394,9192073860) x (ano - (ano + 1))
Depreciação_ano = (394,9192073860) x (ano - ano - 1)
Depreciação_ano = (394,9192073860) x (-1)
Depreciação_ano = -394,9192073860

Portanto, a moto se desvaloriza cerca de R$394,91 por ano.

2.3 3ª questão

2.3.1– 3.a)

Aproximando por uma reta. Pelo Método dos Mínimos Quadrados, na aproximação por um polinômio do primeiro grau, temos um sistema linear da seguinte forma.

$$\begin{bmatrix} \langle \boldsymbol{\phi_0}, \boldsymbol{\phi_0} \rangle & \langle \boldsymbol{\phi_0}, \boldsymbol{\phi_1} \rangle \\ \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_0} \rangle & \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_1} \rangle \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \langle \boldsymbol{f}, \boldsymbol{\phi_0} \rangle \\ \langle \boldsymbol{f}, \boldsymbol{\phi_1} \rangle \end{bmatrix}$$

Onde

$$\boldsymbol{\phi_0} = 1;\ \boldsymbol{\phi_1} = x; \boldsymbol{f} = f(x) = (x^3 - x^2 - 1)^2, x \in [0,1]$$

Calculando os produtos internos, temos:

$$\langle \boldsymbol{\phi_0}, \boldsymbol{\phi_0} \rangle = \langle 1,1 \rangle = \int_0^1 (1 \times 1)\, dx = x|_0^1 = 1 - 0 = 1$$

$$\langle \boldsymbol{\phi_0}, \boldsymbol{\phi_1} \rangle = \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_0} \rangle = \langle 1, x \rangle = \int_0^1 (1 \times x)\, dx = \frac{x^2}{2}\bigg|_0^1 = \frac{1^2 - 0^2}{2} = \frac{1}{2}$$

$$\langle \boldsymbol{\phi_1}, \boldsymbol{\phi_1} \rangle = \langle x, x \rangle = \int_0^1 (x \times x)\, dx = \int_0^1 x^2\, dx = \frac{x^3}{3}\bigg|_0^1 = \frac{1^3 - 0^3}{3} = \frac{1}{3}$$

$$\langle \boldsymbol{f}, \boldsymbol{\phi_0} \rangle = \int_0^1 (x^3 - x^2 - 1)^2\, dx = \frac{247}{210}$$

$$\langle \boldsymbol{f}, \boldsymbol{\phi_1} \rangle = \int_0^1 (x^3 - x^2 - 1)^2 \times x\, dx = \frac{509}{840}$$

Substituindo na equação matricial, temos:

$$\begin{bmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 247/210 \\ 509/840 \end{bmatrix}$$

Resolvendo o sistema no MATLAB®:

```
>> A = [1 1/2;1/2 1/3]

A =

   1.000000000000000   0.500000000000000
   0.500000000000000   0.333333333333333

>> b = [247/210;509/840]

b =

   1.176190476190476
   0.605952380952381

>> c = A\b

c =

   1.069047619047620
   0.214285714285714
```

Figura 15: solução do sistema linear por matriz inversa.

Logo, a função linear que procuramos é

$$g(x) = c_0 + c_1 x = 1{,}069047619047620 + 0{,}214285714285714x$$

Cujo gráfico é

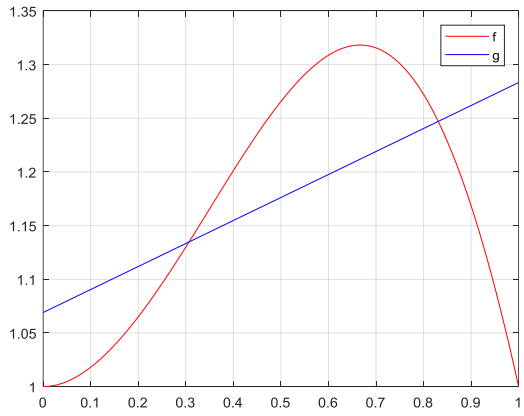

Figura 16: gráfico da função linear g(x) que é ajustada à função f(x) no intervalo de 0 a 1.

2.3.2– 3.b)

Aproximando por um polinômio do segundo grau. Pelo Método dos Mínimos Quadrados, na aproximação por um polinômio do segundo grau, temos um sistema linear da seguinte forma.

$$\begin{bmatrix} \langle \boldsymbol{\phi_0}, \boldsymbol{\phi_0} \rangle & \langle \boldsymbol{\phi_0}, \boldsymbol{\phi_1} \rangle & \langle \boldsymbol{\phi_0}, \boldsymbol{\phi_2} \rangle \\ \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_0} \rangle & \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_1} \rangle & \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_2} \rangle \\ \langle \boldsymbol{\phi_2}, \boldsymbol{\phi_0} \rangle & \langle \boldsymbol{\phi_2}, \boldsymbol{\phi_1} \rangle & \langle \boldsymbol{\phi_2}, \boldsymbol{\phi_2} \rangle \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \langle \boldsymbol{f}, \boldsymbol{\phi_0} \rangle \\ \langle \boldsymbol{f}, \boldsymbol{\phi_1} \rangle \\ \langle \boldsymbol{f}, \boldsymbol{\phi_2} \rangle \end{bmatrix}$$

Onde

$$\boldsymbol{\phi_0} = 1;\ \boldsymbol{\phi_1} = x; \boldsymbol{\phi_1} = x^2; \boldsymbol{f} = f(x) = (x^3 - x^2 - 1)^2, x \in [0{,}1]$$

Calcularemos apenas os itens que não foram calculados no sistema anterior.

$$\langle \boldsymbol{\phi_0}, \boldsymbol{\phi_2} \rangle = \langle \boldsymbol{\phi_2}, \boldsymbol{\phi_0} \rangle = \langle 1, x^2 \rangle = \int_0^1 (1 \times x^2)\, dx = \frac{x^3}{3}\Big|_0^1 = \frac{1^3 - 0^3}{3} = \frac{1}{3}$$

$$\langle \boldsymbol{\phi_2}, \boldsymbol{\phi_1} \rangle = \langle \boldsymbol{\phi_1}, \boldsymbol{\phi_2} \rangle = \langle x^2, x \rangle = \int_0^1 (x^2 \times x)\, dx = \frac{x^4}{4}\Big|_0^1 = \frac{1^4 - 0^4}{4} = \frac{1}{4}$$

$$\langle \boldsymbol{\phi_2}, \boldsymbol{\phi_2} \rangle = \langle x^2, x^2 \rangle = \int_0^1 (x^2 \times x^2)\, dx = \frac{x^5}{5}\Big|_0^1 = \frac{1^5 - 0^5}{5} = \frac{1}{5}$$

$$\langle \boldsymbol{f}, \boldsymbol{\phi_2} \rangle = \int_0^1 (x^3 - x^2 - 1)^2 \times x^2\, dx = \frac{509}{1260}$$

Substituindo na equação matricial, temos:

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 247/210 \\ 509/840 \\ 509/1260 \end{bmatrix}$$

Resolvendo o sistema no MATLAB®:

```
>> A = [1 1/2 1/3;1/2 1/3 1/4;1/3 1/4 1/5]

A =

   1.000000000000000   0.500000000000000   0.333333333333333
   0.500000000000000   0.333333333333333   0.250000000000000
   0.333333333333333   0.250000000000000   0.200000000000000

>> b = [247/210; 509/840; 509/1260]

b =

   1.176190476190476
   0.605952380952381
   0.403968253968254

>> c = A\b

c =

   0.890476190476193
   1.285714285714273
  -1.071428571428559
```

Figura 17: solução do sistema linear por matriz inversa.

Logo, a função quadrática que procuramos é

$$g(x) = c_0 + c_1 x + c_2 x^2$$
$$= 0{,}890476190476193 + 1{,}285714285714273x - 1{,}071428571428559x^2$$

Assim, temos os gráfico:

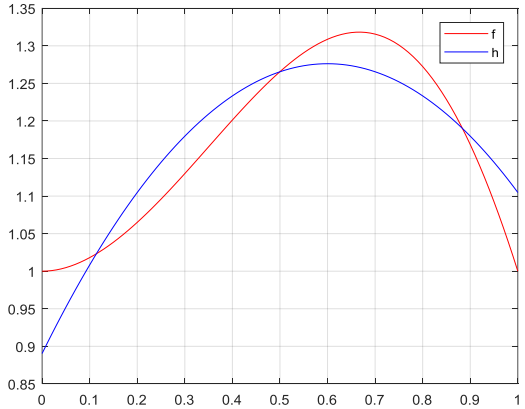

Figura 18: gráfico da função quadrática h(x) que é ajustada à função f(x) no intervalo de 0 a 1.

3 CONCLUSÃO

Ao longo do trabalho, diversas formas de regressão linear foram empregadas para realizar o ajuste de curvas. No seu caso mais simples, aproximando uma função por uma reta (ou polinômio de primeiro grau). Para aproximar por funções que não são lineares, o método pode ser aplicado desde que se reescreva a função em questão numa forma linear, como no caso da função exponencial, que teve que ser expressa como uma função logarítmica para que fosse encontrada a relação linear e, assim, usar a regressão linear. Também foi utilizada a função polinomial de segunda ordem para realizar esse mesmo procedimento, onde foi aplicada uma técnica análoga ao polinômio do primeiro grau e que pode ser estendida para polinômios de ordens superiores. Há que se ter um cuidado, contudo, com polinômios de ordens muito grandes, pois, como foi mencionado no trabalho, estes podem até passar por todos os pontos do conjunto dado, mas o erro relativo entre esses pontos torna-se grande ao ponto de não serem representativos.

O que há de comum entre todos esses métodos é o fato de que eles partem do mesmo princípio: o método dos mínimos quadrados. Isso se faz através do cálculo do erro entre a função de aproximação e a função original (ou conjunto de pontos). Quando o erro é expresso por uma equação quadrática, podemos encontrar valores em que o erro calculado será mínimo. Em termos matemáticos, isso significa que as derivadas parciais dessa função com relação às suas variáveis é nula. Assim, podem ser encontradas expressões que, combinadas, formam um sistema linear que, por sua vez, encontradas as soluções deste, são encontrados os valores dos coeficientes das funções de aproximação. Além disso, como o erro calculado nesses pontos é mínimo, garante-se que esta é a melhor solução.

Portanto, O ajuste de curvas se mostra uma ferramenta muito útil para achar fórmulas matemáticas para um conjunto finito de pontos que seja representativa e com um erro relativo pequeno.

REFERÊNCIAS

Chapra, S. C., e Canale, R. P. *Métodos Numéricos para Engenharia*. 5ª edição. Porto Alegre: AMGH, 2011.

Gilat, A., e Subramaniam, V. *Métodos Numéricos para Engenheiros e Cientistas: uma introdução com aplicações usando o MATLAB*. Porto Alegre: Bookman, 2008.

ANEXO 1

```
function [ c,erro ] = ajuste(x,y,n)
%Esta função realiza o ajuste de curvas de uma função usando uma tabela de
%pontos como referência.
%Parâmetros:
%x = vetor de pontos
%y = vetor de imagens (f(x))
%n = modo de ajuste: 1 - linear, 2 - quadrático, 3 - cúbico, etc.
%Obs: n deve ser menor que o número de elementos em x e maior que zero.

    %validação
    if(((size(x,1)>1)&&(size(x,2)>1))||((size(y,1)>1)&&(size(y,2)>1)))
        disp('Erro! X ou Y não é um vetor.');
        c = 'erro';
        return;
    elseif(length(x)~=length(y))%x e y devem ter mesmo número de elementos
        fprintf('Dimensões de X e Y não são equivalentes.');
        c = 'erro';
        return;
    end
    if n<1||n>=length(x)
        fprintf('N deve ser menor que X ou maior que 0');
        c = 'erro';
        return;
    end
    %#ok<*AGROW>
    %processamento
    %criando vetores phi
    phi = ones(1,length(x));%primeiro vetor: phi_0
    for i=1:n
        phi = [phi; x.^i];
    end
    %montando matriz dos coeficientes
    A = [];
    for i=1:n+1
        for j=1:n+1
            A(i,j) = prod_int(phi(i,:),phi(j,:));
        end
    end
    %montando vetor resposta
    b = [];
    for i=1:n+1
        b = [b; prod_int(y,phi(i,:))];
    end
    %resolvendo sistema linear
    c = eliminacao_gauss(A,b);

    %plotagem
    g = zeros(1,length(x));
    for i=1:n+1
        g = g + c(i)*phi(i,:);
    end
    plot(x,y,'b*'), grid on, hold on;
    plot(x,g,'r-')

    %cálculo do erro
    erro = 0;
    for i=1:length(x)
        erro = erro + (y(i)-g(i))^2;
    end
end
```

Figura 19: função em MATLAB® para ajustes de curvas usando o Método dos Mínimos Quadrados.

ANEXO 2

```
function [ x ] = eliminacao_gauss(A,b)
%Função que calcula a solução de um sistema linear usando o método da
%Eliminação de Gauss.
%Parâmetros: eliminacao_gauss(A,b)
%A = matriz dos coeficientes (deve ser quadrada e det(A)~=0)
%b = vetor resposta ou vetor das constantes do sistema linear (deve ter o
%mesmo número de elementos que a dimensão da matriz A)

%validação dos parâmetros
if size(A,1)~=size(A,2) %testa se o número de linhas é igual ao de colunas.
    disp('Erro. A matriz dos coeficientes não é quadrada.');
    return;
elseif det(A)==0 %testa se o determinante de A é igual a 0.
    disp('Erro. O determinante da matriz dos coeficientes é nulo.');
    return;
end
dim = size(A,1);%determina a dimensão da matriz A e armazena em dim.
if size(b,1)~=1 && size(b,2)~=1%testa se é vetor linha ou coluna
    disp('Erro. O vetor resposta não é um vetor linha ou vetor coluna.');
    return;
elseif size(b,1)~=dim && size(b,2)~=dim %testa equivalência entre A e b.
    fprintf('Erro. O vetor resposta deve ter o mesmo número de elementos');
    fprintf(' que a dimensão da matriz dos coeficientes.\n');
    return;
end
if size(b,1)==1 %se b for um vetor linha,
    b = b';     %é transformado em vetor coluna.
end
%fim da validação dos parâmetros.
%#ok<*NASGU>
m = 0; %constante usada para transformação da matriz em Triangular Superior
temp = [];%vetor temporário auxiliar
temp1 = 0;%variável temporária auxiliar
cont = 0;%contador de pivotações

%Transformando matriz dos coeficientes em uma matriz triangular superior
%através de operações elementares em suas linhas
for i = 1:(dim-1) %for para linhas

    %O if a seguir evita que exista um pivô nulo
    if(A(i,i)==0)
        for k = (i+1):dim %esse for realiza a pivotação
            if(A(k,i)~=0)
                cont = cont + 1; %incrementa contador de pivotações
                %permutando linha da matriz dos coeficientes
                temp = A(i,:);
                A(i,:) = A(k,:);
                A(k,:) = temp;

                %permutando linha do vetor resposta
                temp1 = b(i);
                b(i) = b(k);
                b(k) = temp1;
                break;
            end
        end
    end

    for j = (i+1):dim %for para colunas
        m = A(j,i)/A(i,i);%const. usada p/ zerar elementos abaixo do pivô
        A(j,:) = A(j,:) - m*A(i,:);%transformação em Triangular Superior
        b(j,1) = b(j,1) - m*b(i,1);%Alteração no vetor das constantes
    end
end
x = [];%declaração da variável de saída como um vetor vazio.
%Resolução do sistema linear por retrossubstituição
for i = dim:-1:1 %for para linhas
    soma = b(i);%constante que será somada com os valores já conhecidos
    for j = dim:-1:i %for para colunas
        if(j~=i)
            soma = soma-(A(i,j)*x(j));%valores já descobertos sendo somados
        else
            x(i,1) = soma / A(i,j); %cálculo do valor da incógnita
        end
    end
end
end
```

Figura 20: função que implementa o algoritmo da Eliminação de Gauss (adaptado de T3_CN).

ANEXO 3

```
function [produto] = prod_int(a,b)
%função que calcula o produto interno de dois vetores
%a,b - vetores passados como parÃ¢metro
tamanho_a = size(a);
tamanho_b = size(b);

if(tamanho_a(1)>1)%trasnposição de matriz coluna para matriz linha
    a = a';
end
if(tamanho_b(1)>1)%trasnposição de matriz coluna para matriz linha
    b = b';
end
%validação: checar se entradas são vetores
if(((tamanho_a(1)>1)&&(tamanho_a(2)>1))||((tamanho_b(1)>1)&&(tamanho_b(2)>1)))
    disp('Erro! Ao menos um dos parâmetros não é um vetor.');
    produto = 'erro';
elseif(length(a) ~= length(b))%validação: vetores devem ter mesmo número de elementos
    fprintf('Dimensões não são equivalentes.\nNão é possível realizar o produto interno.\n\n');
    produto = 'erro';
else %passou por todas as validações
    % Inicialização da variável usada para guardar os resultados de cada
    % operação no produto interno entre a e b
    produto = 0;

    for i=1:1:length(a) % vai até o fim da dimensão do vetor de a.
        produto = produto + (a(i)*b(i));
    end
end
end
```

Figura 21: função que implementa o produto interno (discreto) entre dois vetores.

ANEXO 4

```
>> tic,[p S] = polyfit(x,y,1),toc

p =

   1.665540080000000  -0.512456824000000


S =

  struct with fields:

       R: [2×2 double]
      df: 6
   normr: 0.579301178800361

Elapsed time is 0.016814 seconds.
```

Figura 22: resultado de *polyfit* para polinômio linear.

```
>> tic,[p S] = polyfit(x,y,2),toc

p =

   1.129423867019517  -0.311403456830962   0.085143932517366


S =

  struct with fields:

       R: [3×3 double]
      df: 5
   normr: 0.049192630844739

Elapsed time is 0.007898 seconds.
```

Figura 23: resultado de *polyfit* para polinômio quadrático.

```
>> tic,[p S] = polyfit(x,y,3),toc

p =

   0.266208097739523   0.402932213097656   0.248385784190559  -0.018401399296226


S =

  struct with fields:

       R: [4×4 double]
      df: 4
   normr: 0.002252703166338

Elapsed time is 0.051067 seconds.
```

Figura 24: : resultado de *polyfit* para polinômio cúbico.

```
>> tic,[p S] = polyfit(x,log(y),1),toc

p =

   2.707294686913416  -3.085493303114046


S =

  struct with fields:

       R: [2×2 double]
      df: 6
   normr: 0.770626033471183

Elapsed time is 0.078759 seconds.
```

Figura 25: resultado de *polyfit* para função exponencial em e.

```
>> tic,[p S] = polyfit(log(x),log(y),1),toc

p =

   1.872009284326521  -0.051128596801404


S =

  struct with fields:

       R: [2×2 double]
      df: 6
   normr: 0.173696162266301

Elapsed time is 0.005055 seconds.
```

Figura 26: resultado de *polyfit* para função exponencial em x.

ANEXO 5

Tabela 2: Tabela de preço/ano/km das motos pesquisadas.

R$	ANO	KM
7000	2009	48000
6800	2009	63000
7300	2009	81000
7500	2009	62825
7490	2010	41135
11000	2010	35000
7500	2010	38785
6800	2010	60000
7000	2011	66365
6800	2011	44000
7800	2011	46500
7900	2011	48910
8500	2012	32293
7300	2012	108008
7900	2012	50000
9000	2012	50000
10000	2013	5890
8400	2013	36000
10900	2013	13472
9990	2013	18000
9500	2014	31353
10490	2014	27000
12000	2014	7000
9900	2014	29000
10500	2014	18000
11500	2015	6000
11690	2015	8378
10900	2015	4900
9500	2015	44000
11290	2015	20000
TOTAL: 30 MOTOS		

MODELO: HONDA CB 300R
OBS: NO SITE NÃO TINHAM MOTOS NOS ANOS DE 2016/2017.
OBS: AS MOTOS ATINGIRAM O LIMITE MÁXIMO DE TEMPO DA PESQUISA.

Tabela 3: Tabela com as possíveis motos a serem compradas (motos mais novas e com menos quilômetros rodados).

R$	**ANO**	**KM**
7490	2010	41135
11000	2010	35000
7500	2010	38785
6800	2010	60000
7000	2011	66365
6800	2011	44000
7800	2011	46500
7900	2011	48910
8500	2012	32293
7300	2012	108008 (mais quilômetros rodados que a atual)
7900	2012	50000
9000	2012	50000
10000	2013	5890
8400	2013	36000
10900	2013	13472
9990	2013	18000
9500	2014	31353
10490	2014	27000
9900	2014	29000
10500	2014	18000
10900	2015	4900
9500	2015	44000
TOTAL: 22 MOTOS		