Cálculo Numérico: Projeto de Ajustamento (Regressão) Linear e Não Linear

Cleber Alberto Cabral Ferreira da Silva, Heloísa Bezerra Neves
Alunos da disciplina de Cálculo Numérico, Turma T7.

Resumo: *Este projeto tem como objetivo observar os comportamentos das funções quando são implementados ajustamentos lineares e não lineares. Para isso, foi elaborado um programa em linguagem C que utiliza o método dos mínimos quadrados.*
Palavras-chave: Ajustamento, Funções, Mínimos Quadrados, Linguagem C.

1. Introdução:

Quando estudamos um fenômeno de forma experimental, é comum termos valores tabelados sobre os quais podemos levantar várias questões, como por exemplo: qual a relação que associa à $\boldsymbol{f(x)}$ a variável $\boldsymbol{x}$? Qual valor de $\boldsymbol{f(x)}$ para um $\boldsymbol{x}$ dado que não está na tabela? (SANTOS; SILVA, 2010, p26).

Neste projeto, será implementado um programa em linguagem C que realizará ajustamentos através do Método dos Mínimos Quadrados (MMQ) em uma coleção de funções descritas abaixo:

$$f1(x) = ax + b$$
$$f2(x) = ax^2 + bx$$
$$f3(x) = ax^3 + bx$$
$$f4(x) = ax^3 + bx^2$$
$$f5(x) = ae^{bx}$$
$$f6(x) = ax^b$$
$$f7(x) = aln(x) + \frac{b}{x}$$
$$f8(x) = ax + \frac{b}{x}$$
$$f9(x) = acos(x) + bx$$
$$f10(x) = \frac{1}{asen(x) + be^x}$$

Cada função apresenta um resíduo dado por $R(x_i) = P(x_i) - f(x_i)$ e dessa maneira o MMQ pode ser descrito como:

$$\sum_{i=0}^{n} R^2(x_i) = \varphi(a_0,\ a_1,\ ...,\ a_m)$$

2. Desenvolvimento:

Sendo estabelecidas as regras descritas na introdução, agora discorreremos sobre as técnicas e o fluxos utilizados para o cálculo dos resultados obtidos.

2.1. Elaboração do Código:

O programa foi desenvolvido na linguagem C utilizando as bibliotecas <stdio.h>, <math.h> e <stdlib.h>. Elas serão necessária para a leitura do arquivo, a alocação dinâmica de memória, conversão dos dados lidos em números decimais e manipulação de funções matemáticas presentes na biblioteca <math.h>.

Todas variáveis reais do projeto são do tipo `double`.

2.2. Fluxo de processamento:

Agora será descrito o fluxo utilizado pelo programa para realizar as operações, seguindo as recomendações do fluxo da operação no livro-texto visto em sala de aula.

O fluxo primeiramente lê todos os dados da tupla *x, f(x)*. Logo após, executa o seguinte sequência descrita:

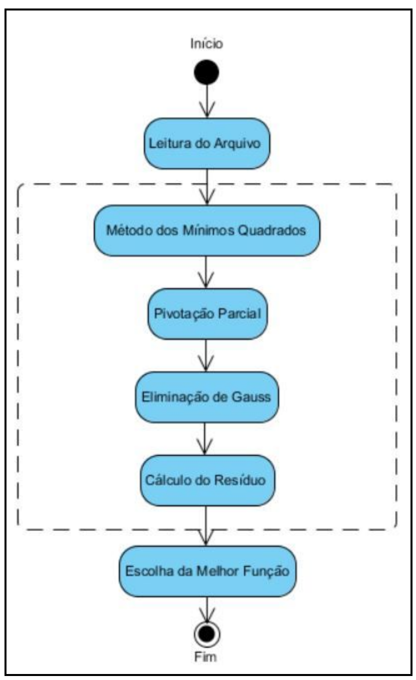

Imagem 1: Fluxo de execução do programa.

A caixa pontilhada representa a sequência utilizada em *loop* para cada função estabelecida.

2.2.1. Método dos Mínimos Quadrados:

Dada a equação de uma reta, como exemplo do método, temos:

$$P(x) = a_0 \times G_0(x) + a_1 \times G_1(x)$$

onde $G_0(x) = 1$ e $G_1(x) = x$, temos para cada par de parâmetro (a_0, a_1) uma reta distinta, $P(x) = a_0 + a_1 x$ distinta.

O objetivo, portanto, é determinar o mínimo de $\varphi(a_0,\ a_1,\ ...,\ a_m)$, no caso:

$$\sum_{i=0}^{n} R^2(x_i) = \varphi(a_0,\ a_1,\ ...,\ a_m)$$

No caso da implementação, percebe-se que nem todas as funções podem ser aplicadas ao método da maneira como estão, sendo necessária uma reescrita. Neste caso, as funções 5 e 6 necessitam ser reescritas utilizando o logaritmo natural e a função 10 necessita ser escrita na forma inversa. Lembrando que os dados lidos não devem conter dados menores ou igual a zero, devido ao uso do logaritmo natural.

Sendo assim, os parâmetros a_0, a_1, G_0 e G_1 de todas as funções seriam:

Função	a_0	G_0	a_1	G_1	**f(x)**
$f1(x) = ax + b$	*a*	*x*	*b*	*1*	*f(x)*
$f2(x) = ax^2 + bx$	*a*	x^2	*b*	*x*	*f(x)*
$f3(x) = ax^3 + bx$	*a*	x^3	*b*	*x*	*f(x)*
$f4(x) = ax^3 + bx^2$	*a*	x^3	*b*	x^2	*f(x)*
$f5(x) = ae^{bx}$	*ln(a)*	*1*	*b*	*x*	*ln(f(x))*
$f6(x) = ax^b$	*ln(a)*	*1*	*b*	*ln(x)*	*ln(f(x))*
$f7(x) = aln(x) + \frac{b}{x}$	*a*	*ln(x)*	*b*	$\frac{1}{x}$	*f(x)*
$f8(x) = ax + \frac{b}{x}$	*a*	*x*	*b*	$\frac{1}{x}$	*f(x)*
$f9(x) = acos(x) + bx$	*a*	*cos(x)*	*b*	*x*	*f(x)*
$f10(x) = \frac{1}{asen(x)+be^x}$	*a*	*sen(x)*	*b*	e^x	$\frac{1}{f(x)}$

Tabela 1: Parametrizações das Funções

Parametrizadas as funções, monta-se o sistema de equações abaixo, a fim de encontrar os parâmetros *a* e *b* através da pivotação parcial e da eliminação de Gauss:

$$\sum_{k=0}^{m} a_k \sum_{i=0}^{n} \varphi_k(x_i)\varphi_j(x_i) = \sum_{i=0}^{n} f(x_i)\varphi_j(x_i)\,,\ j = 0, 1, 2, \ldots, \mathrm{m}\,.$$

Imagem 2: Equação Geral do Sistema de Equação do MMQ.

2.2.2. Pivotação Parcial

Pivotação é o processo usado no método de eliminação de Gauss para trocar, se necessário, as linhas da matriz de modo que o elemento da diagonal principal seja diferente de zero. Estes elementos são chamados de pivô.

A pivotação parcial consiste em escolher o *k-ésimo* pivô e trocar, se preciso, a *k-ésima* linha da matriz de modo que o maior deles, em módulo, entre o restante da *k-ésima* coluna seja usado como pivô.

2.2.3. Método de Eliminação de Gauss:

O livro-texto define o Método de Eliminação de Gauss da seguinte maneira:

Algoritmo de eliminação de Gauss

```
Para k = 1,2, . . . , (n-1) faça
   Para i = (k+1), (k+2), . . . , n faça
      m = a_ik / a_kk  (supondo a_kk ≠ 0)
      a_ik = 0
      Para j = (k+1), (k+2), . . . , n faça
         a_ij = a_ij - m × a_kj
      fim
      b_i = b_i − m × b_k
   fim
fim
x_n = b_n / a_nn
Para k = 1 , 2 , . . . ,(n-1) faça
   x_k = b_k
fim
Para k = (n-1), (n-2), . . . ,1 faça
   Para i = (k+1), (k+2), . . . , n faça
      x_k = x_k − a_ki × x_i
   fim
      x_k = x_k / a_kk
fim.
```

Imagem 2: Algoritmo de Eliminação de Gauss

O método de eliminação de Gauss consiste em transformar o sistema $A\,x = b$ em $T\,x = c$, onde T é uma matriz triangular, tornando assim trivial a determinação da solução x procurada.

2.3. Testes de Execução:

Após uma breve introdução da implementação utilizada, agora aplicam-se os testes para verificação do funcionamento do programa, conforme abaixo:

2.3.1. Função presente na Lista de Funções:

Neste teste, busca-se saber se o programa identifica uma função que esteja presente na lista das dez funções implementadas, dado um tabelamento conhecido.

Função	Arquivo	Saída (função; resíduo; raiz)
$f(x) = acos(x) + bx$ a = 0,1880719166 b = 0,5515511648	10 1.0000E-01,5.0825E+00 2.0000E-01,2.4551E+00 4.0000E-01,1.2065E+00 5.0000E-01,9.7274E-01 8.0000E-01,6.4747E-01 1.1000E+00,5.1934E-01 1.2000E+00,4.9392E-01 1.4000E+00,4.5725E-01 1.5000E+00,4.4396E-01 1.6000E+00,4.3311E-01	melhor funcao de ajustamento: f_{7}(x) = 1.8808e-01*ln(x) + 5.5156e-01/x 3.7359e-09 6.1122e-05

Tabela 2: Teste de Função presente na Lista.

Percebe-se que o programa identificou a função como sendo a de número certo e precisamente encontrou os parâmetros *a* e *b* corretamente.

2.3.2. Função não presente na Lista de Funções:

Neste teste, o programa irá buscar uma aproximação para uma função que não existe na sua lista de funções pré-configuradas.

Função	Arquivo	Saída (função; resíduo; raiz)
$f(x) = ln(\frac{1}{x})^2$	10 1.0000E-01,5.3019E+00 2.0000E-01,2.5903E+00 4.0000E-01,8.3959E-01 5.0000E-01,4.8045E-01 8.0000E-01,4.9793E-02 1.1000E+00,9.0840E-03	melhor funcao de ajustamento: f_{8}(x) = -2.5500e-01*x + 5.1552e-01/x 6.7184e-01 8.1966e-01

	1.2000E+00,3.3241E-02 1.4000E+00,1.1321E-01 1.5000E+00,1.6440E-01 1.6000E+00,2.2090E-01	

Tabela 3: Teste de Função não presente na Lista.

Percebe-se que o programa não identificou a função, já que ela não está presente na lista. Ao final ele apresentou a que possuía o menor ruído comparado ao tabelamento original.

2.3.3. Tabelamento com Pontos em Diferentes Intervalos:

O programa tentará identificar a função para um tabelamento que possui pontos (x, f(x)) com diferentes espaçamentos.

Função	Arquivo	Saída (função; resíduo; raiz)
$f(x) = ax^b$ a = 0,303620002 b = 0,0206907238	10 4.9573E-01,2.9924E-01 8.1656E-01,3.0235E-01 5.5268E-01,2.9992E-01 1.7568E-01,2.9289E-01 5.5605E-01,2.9996E-01 1.7166E-01,2.9275E-01 4.5085E-01,2.9866E-01 1.4104E-01,2.9156E-01 4.4121E-01,2.9852E-01 5.8512E-02,2.8630E-01	melhor funcao de ajustamento: f_{6}(x) = (3.0362e-01)*x^(2.0694e-02) 6.1033e-11 7.8124e-06

Tabela 4: Teste de Tabelamento com Pontos em Diferentes Intervalos.

Percebe-se que o programa identificou a função mesmo com pontos em diferentes espaçamentos.

2.3.4. Tabelamento com Poucos Pontos:

Utilizando apenas três pontos do tabelamento anterior, o programa tentará encontrar a função de ajustamento.

Função	Arquivo	Saída (função; resíduo; raiz)
$f(x) = ax^b$ a = 0,303620002 b = 0,0206907238	3 4.9573E-01,2.9924E-01 8.1656E-01,3.0235E-01 5.5268E-01,2.9992E-01	melhor funcao de ajustamento: f_{6}(x) = (3.0362e-01)*x^(2.0706e-02) 1.5440e-11 3.9294e-06

Tabela 5: Teste de Tabelamento com Poucos Pontos.

Percebe-se que o programa conseguiu identificar a função, porém obteve o parâmetro *b* com uma leve diferença do valor original.

2.3.5. Tabelamento com Muitos Pontos:

Agora utilizando 24 pontos, o programa tentará encontrar a função de ajustamento.

Função	**Arquivo**	**Saída (função; resíduo; raiz)**

$f(x) = \frac{1}{asen(x)+be^x}$
 	
a = 0,303620002
b = 0,0206907238
	`24`
	`7.5054E-02,1.6430E+00`
	`3.2134E-01,1.2196E+00`
	`3.7255E-01,1.1508E+00`
	`7.9293E-01,7.3919E-01`
	`3.2357E-01,1.2164E+00`
	`4.9486E-03,1.8011E+00`
	`1.0716E-01,1.5772E+00`
	`2.1813E-01,1.3761E+00`
	`1.3539E-01,1.5223E+00`
	`4.3484E-01,1.0739E+00`
	`9.4653E-02,1.6024E+00`
	`1.8150E-05,1.8130E+00`
	`8.3501E-02,1.6253E+00`
	`5.0797E-01,9.9196E-01`
	`2.2555E-03,1.8076E+00`
	`4.4189E-01,1.0656E+00`
	`9.1036E-02,1.6097E+00`
	`2.8768E-02,1.7450E+00`
	`3.6911E-04,1.8122E+00`
	`3.8168E-01,1.1390E+00`
	`1.9135E-03,1.8084E+00`
	`8.6584E-02,1.6189E+00`
	`2.6794E-01,1.2973E+00`
	`5.1556E-02,1.6937E+00`
		`melhor funcao de ajustamento:`
		`f_{10}(x) = 1/(1.8805e-01*sen(x) +`
		`5.5155e-01*exp(x))    1.8647e-08      1.3655e-04`

Tabela 6: Teste de Tabelamento com Poucos Pontos.

Percebe-se que o programa conseguiu identificar a função, e há uma tendência de que quanto mais pontos existir no tabelamento, mesmo não equidistantes, a precisão dos parâmetros *a* e *b* tende a ser maior.

2.3.6. Tabelamento com Erros de Medida Experimental:

Neste teste, o programa tentará identificar a função para um tabelamento que possui pontos com erros de medida experimental. Para isso, foi criado um tabelamento com pontos que podem ter um erro de 15% do valor original. Conforme gráfico abaixo:

X	F(X)	F(X) randomizada
1,0000E-01	5,2817E-02	4,5888E-02
2,0000E-01	7,7412E-02	6,9448E-02
4,0000E-01	1,1346E-01	1,1777E-01
5,0000E-01	1,2832E-01	1,1581E-01
8,0000E-01	1,6629E-01	1,7511E-01
1,1000E+00	1,9822E-01	1,8806E-01
1,2000E+00	2,0797E-01	2,3479E-01
1,4000E+00	2,2642E-01	2,4877E-01
1,5000E+00	2,3521E-01	2,5594E-01
1,6000E+00	2,4373E-01	2,7915E-01

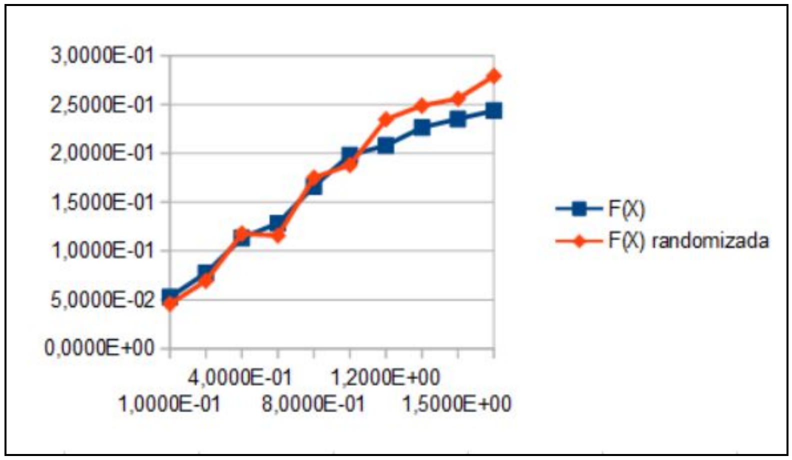

Gráfico 1: Tabelamento com Erros de Medida Experimental.

A mesma função *f(x)* = ax^b será utilizada, desta vez, utilizando os pontos de *f(x) randomizada* e novos parâmetros *a* e *b*.

Função	Arquivo	Saída (função; resíduo; raiz)
$f(x) = ax^b$ a = 0,1880719166 b = 0,5515511648	10 1.0000E-01,4.5888E-02 2.0000E-01,6.9448E-02 4.0000E-01,1.1777E-01 5.0000E-01,1.1581E-01 8.0000E-01,1.7511E-01 1.1000E+00,1.8806E-01 1.2000E+00,2.3479E-01 1.4000E+00,2.4877E-01 1.5000E+00,2.5594E-01 1.6000E+00,2.7915E-01	melhor funcao de ajustamento: f_{6}(x) = (1.9780e-01)*x^(6.4187e-01) 9.9428e-04 3.1532e-02

Tabela 7: Teste de Tabelamento com Erros de Medida Experimental.

Percebe-se que o programa conseguiu identificar a função, porém obteve parâmetros *a* e *b* divergentes dos originais, o que era esperado devido aos erros de medida.

3. Conclusões:

Conclui-se a partir destes testes que o programa de ajustamento de funções implementado respondeu bem às funcionalidades esperadas, tanto na identificação de funções com tabelamento com ou sem erros de medida experimental.

Além disso, verificou-se que a quantidade de pontos do tabelamento tende a satisfazer uma maior precisão dos parâmetros que envolvem as funções.

4. Referências Bibliográficas:

- SANTOS & SILVA, Métodos Numéricos 3ª Edição, Editora Universitária UFPE, 2010.