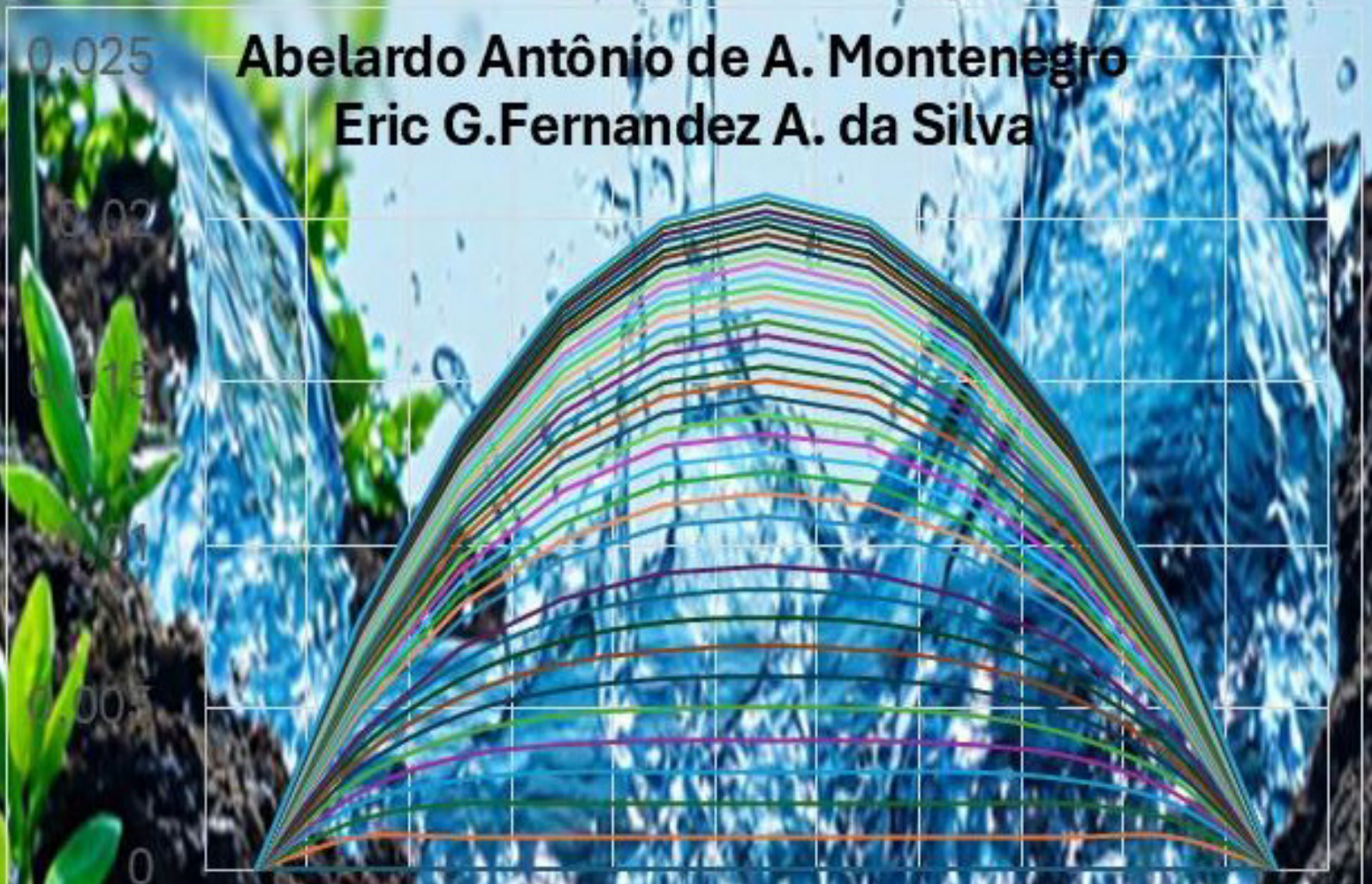
MATEMÁTICA COMPUTACIONAL APLICADA À ENGENHARIA AGRÍCOLA

Abelardo Antônio de A. Montenegro
Eric G.Fernandez A. da Silva



Recife, 2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Bibliotecária Suely Manzi – CRB/4 809

M777m Montenegro, Abelardo Antônio de Assunção
Matemática computacional aplicada à engenharia agrícola / Abelardo Antônio de Assunção Montenegro, Eric G. Fernandez A. da Silva . – 1. ed. - Recife: EDUFRPE, 2025.
67 p.: il.

Inclui bibliografia.

1. Computação – Matemática 2. Análise numérica
3. Modelagem computacional 4. Programação (Matemática)
5. Engenharia agrícola I. Silva, Eric G. Fernandez A. da II. Título

CDD 630

ISBN FÍSICO nº 978-85-7946-549-9
ISBN DIGITAL nº 978-85-7946-548-2

Prefácio

Este material é fruto de duas décadas de ensino na Disciplina de Matemática Computacional aplicada à Engenharia Agrícola, do Curso de Graduação de Engenharia Agrícola e Ambiental, da UFRPE. Com forte influência de experiências em Métodos Numéricos vivenciadas na Escola de Engenharia de São Carlos (EESC-USP), orientadas pelo prof. Antônio Marozzi Righetto, e na Universidade de Newcastle- Reino Unido, sob orientação de prof. Rae Mackay, o material traz aplicações numéricas relacionadas a problemas de engenharia, aproximando o leitor a desafios comumente enfrentados na Engenharia de Recursos Hídricos, na Engenharia de Água e Solo, e nas Ciências Ambientais. Cabe também destacar aqui as motivações advindas das Escolas de Verão do Instituto de Matemática Pura e Aplicada (IMPA), sob supervisão do Dr. Elon Lages Lima, bem como a Iniciação Científica envolvendo pesquisas em Matemática Aplicada com orientação do Dr. Paulo Figueiredo e Engenharia Civil, com os professores José Almir Cirilo e Abelardo Cardoso Montenegro, da UFPE.

O material traz fluxogramas e códigos computacionais cuidadosamente elaborados para cada capítulo, e fornecidos livremente para acesso público.

O material apresentado neste livro recebeu a contribuição de vários monitores da disciplina de Matemática Computacional. Em nome de Aline Chagas, Carolyne Andrade de Farias, Hugo Montenegro, Oto Barbosa, e Robertson Fontes Júnior, prestamos nosso reconhecimento a todos eles, juntamente com o do servidor Rodrigo Costa.

Esperamos que o documento contribua para a formação acadêmica dos estudantes de Engenharias do País.

Sumário

Erros e modelos lineares simples 7
Exercícios numéricos 11
Exercícios computacionais 13
Regressão linear múltipla - Modelos lineares 15
Exercícios numéricos 19
Exercícios computacionais 19
Resolução de sistemas de equações lineares 20
Técnica direta - Eliminação de Gauss 20
Técnica indireta - Iterativa: Método de Jacobi 24
Algoritmo do Método de Jacobi 27
Método de Gauss-Seidel 27
Algoritmo do Método de Gauss-Seidel 28
Exercícios numéricos 32
Raízes de funções implícitas 33
Método de Newton 33
Método das Secantes 34
Aplicação de Engenharia do Método das Secantes 35
Equação de Manning 36
Programação em BASIC - Armazenando Dados 38
Método de Newton: 41
Método das secantes: 41
Exercícios numéricos 42
Interpolação 43
Polinômio interpolador de Lagrange 43
Exercícios numéricos 49
Exercícios computacionais 49
Integração Numérica 50
Método dos Trapézios 50
Discussão a respeito da discretização do intervalo de integração 51
Implementação complementar em Basic 53
Exercícios numéricos 60
Exercícios computacionais 60
Derivação numérica 61
Exercícios numéricos 68
Exercícios computacionais 69

Erros e modelos lineares simples

A modelagem de experimentos ou medições constitui-se em uma importante etapa na análise de comportamento de processos naturais, sendo os erros inerentes à simplificação adotada em tais representações.

Neste capítulo exploraremos os erros associados a ajuste de modelos lineares, largamente utilizados na Engenharia Agrícola.

Estes modelos são representados por polinômios de primeiro grau, com dois parâmetros a serem estimados, conhecidos como coeficiente angular e como coeficiente linear. Temos:

$$\hat{y} = a + bx$$

onde "a" é o coeficiente linear e "b" o coeficiente angular.

A estimativa de tais parâmetros a partir de um conjunto de dados experimentais (x,y) deve ser realizada de modo a minimizar o erro na estimativa $\hat{y}$ de y. Objetivando eliminar o sinal algébrico, então se adota o quadrado da diferença entre a medição e a estimativa. Então busca-se a minimização da soma dos erros quadráticos, dando origem a um processo matemático conhecido como Método dos Mínimos Quadrados.

Tem-se:

$x \rightarrow Variável\ independente$

$y \rightarrow Variável\ dependente$

$a, b \rightarrow Parâmetros$ a serem estimados a partir dos dados experimentais

$b \rightarrow Coeficiente\ angular$

$a \rightarrow Coeficiente\ linear$

Graficamente, tem-se na Figura 1 um modelo linear:

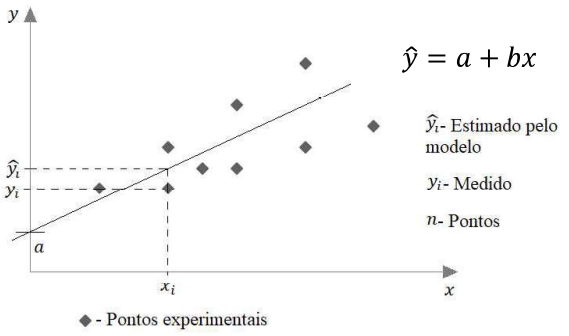


Figura 1- Modelo linear

Para

$x = 0 \rightarrow y = a + b \times 0 = a$

Inclinação do modelo → $\frac{dy}{dx} = y' = b$

É necessário que o modelo represente bem os dados experimentais, ou seja, seja fiel aos dados, minimizando os erros quadráticos:

$$E_i = (\widehat{y_i} - y_i)^2$$

$$E_1 = (\widehat{y_1} - y_1)^2$$

$$E_2 = (\widehat{y_2} - y_2)^2$$

.....

$$E_n = (\widehat{y_n} - y_n)^2$$

A qualidade do ajuste depende da minimização dos erros.

$$E_T = E_1 + E_2 + \cdots + E_n$$

$$\boxed{E_T = \sum_{i=1}^{n} E_i}$$

Derivando-se o erro total e igualando-se a zero, temos:

$\widehat{y_i} = a + bx_i$

$$\boxed{E_T = \sum_{i=1}^{n} \left((a + bx_i) - y_i\right)^2}$$

$$\begin{cases} \frac{\partial E_T}{\partial a} = 0 \\ \frac{\partial E_T}{\partial b} = 0 \end{cases}$$

As derivações acima resultam na estimativa dos valores ótimos de $'a'$ e $'b'$, dados por:

$$\begin{cases} b = \frac{S_{xy}}{S_{xx}} \\ a = \bar{y} - b\bar{x} \end{cases}$$

com

$S_{xy} = \sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}$	$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$
$S_{xx} = \sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}$	$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

Para avaliar a qualidade do ajuste, deve-se verificar se o modelo ajustado representa adequadamente a variabilidade original dos dados. Tal procedimento é realizado a partir da estimativa da variabilidade total dos dados e da variabilidade "explicada" pelo modelo. Podemos escrever:

$$VE = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

$$VT = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

onde:

VE-Variabilidade estimada.

VT-Variabilidade total.

Define-se então o Coeficiente de Determinação R^2, expresso por:

$$R^2 = \frac{VE}{VT}$$

Tal coeficiente varia de 0 a 1. Quanto mais próximo de 1, melhor a qualidade do ajuste.

O leitor deve consultar Martins (2010) para maiores detalhes.

Exercício resolvido

Considere o conjunto de dados abaixo. Estime os parâmetros de um modelo linear para representá-los. Construa o gráfico. Adotar quatro casas decimais.

x	y
1	3
2	7
3	8
4	11
5	14
6	20

Estabelecendo-se um contador i para os dados, e calculando os termos adicionais para estimativa dos parâmetros do modelo, temos:

i	x	y	xy	x^2
1	**1**	**3**	3	1
2	**2**	**7**	14	4
3	**3**	**8**	24	9
4	**4**	**11**	44	16
5	**5**	**14**	70	25
6	**6**	**20**	120	36
Σ	21	63	275	91

Preenchendo as lacunas:

S_{xx}= _____

S_{xy}= _____

b= _____

a= _____

$\hat{y}$= _____

Tem-se:

i	x	y	xy	x^2	$\hat{y}$	$(\hat{y}-\bar{y})^2$	$(y-\bar{y})^2$
1	**1**	**3**	3	1	2,7138	60,61	56,25
2	**2**	**7**	14	4	5,8281	21,82	12,25
3	**3**	**8**	24	9	8,9424	2,424	6,25
4	**4**	**11**	44	16	12,0567	2,434	0,25
5	**5**	**14**	70	25	15,1710	21,81	12,25
6	**6**	**20**	120	36	18,2813	60,53	90,25
Σ	21	63	275	91	$\hat{y} = a + bx$	169,62 (VE)	177,5 (VT)

$$R^2 = \frac{VE}{VT} = \frac{169,62}{177,50}$$

$R^2 = 0,9555$ (bom ajuste)

Graficamente, temos:

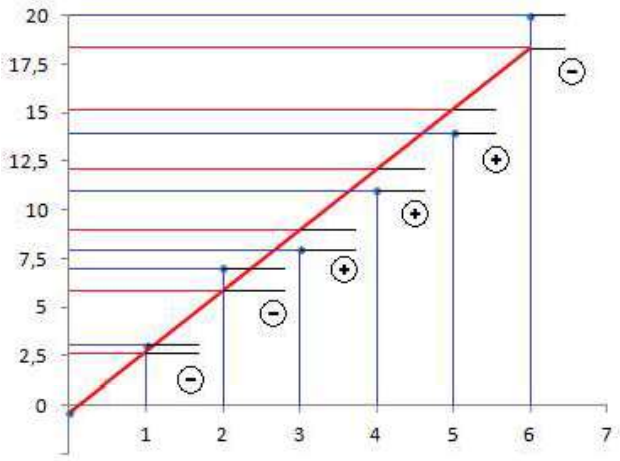


Figura 2: Dados experimentais e valores estimados. Qualidade do ajuste

Exercício resolvido

Realizar a soma dos valores de y do exercício anterior. Utilize um processo iterativo, chamando de i o contador de passos deste processo.

Solução:

Tem-se 6 valores da variável dependente y. Então, o contador i irá variar de 1 a 6, conforme já apresentado na tabela anterior. Chamando a soma de y de SOMAY, a Figura 6 exibe os valores parciais de SOMAY, no processo iterativo.

i	y	Soma_y
1	3	3
2	7	10
3	8	18
4	11	29
5	14	43
6	20	63

Figura 3- Representação gráfica de processo iterativo para a soma de 6 números.
Valores de SOMAY em função da iteração i.

Exercícios numéricos

1. Determine a função linear que modela o seguinte conjunto de dados e determine o Coeficiente de Determinação (R^2) para a função ajustada.

x	Y
2	6.1
4	7.8
7	22.4
12	37.9

2. Considere o seguinte conjunto de dados e a função a seguir:

x	Y
1.2	-1.10
1.5	-1.23
2.6	0.92
4.0	4.04
4.2	4.40
7.0	8.00

Modelo ajustado: $f(x) = 1.67x - 3.2$

Se os dados foram ajustados e obteve-se a equação apresentada, qual será o Coeficiente de Determinação (R^2)?

3. Considere o conjunto de dados abaixo. Estime os parâmetros de um modelo linear para representá-los. Construa o gráfico. Adotar quatro casas decimais. Calcule também o R2 e comente sobre a qualidade do ajuste obtido.

X	Y
1	8
2	13
3	26
4	33
5	38
9	75

4. A tabela abaixo apresenta a leitura da chuva em dois pluviômetros, em uma bacia hidrográfica. Estime os parâmetros de um modelo linear simples para representar a relação entre os equipamentos. Estime o coeficiente de determinação. Obs: Para cálculo do coeficiente de determinação, utilize a expressão R^2= VE/VT, onde VE é a variação explicada e VT a variação total.

Mês (2022)	P1	P2
Jan	3	7
Fev	9	18
Mar	6	14
Abr	4	12

5. Com base no modelo desenvolvido na questão 4, estime a precipitação em P2, correspondendo a uma leitura de 10 mm no pluviômetro P1.

Exercícios computacionais

1. Elaborar um programa em Basic para estimar os coeficientes linear e angular de um modelo linear. O número de valores n deve ser fornecido pelo usuário. Os valores de x e y experimentais devem ser armazenados em vetores.
2. Elaborar um programa em Python para estimar os coeficientes linear e angular de um modelo linear. Os valores de x e y devem ser informados pelo usuário. O coeficiente de determinação deve ser determinado para o modelo ajustado.

A seguir apresenta-se uma possibilidade de solução de modelos de regressão linear simples utilizando a linguagem Python.

```
x, y = [], []
temp = 0

soma_x, soma_y, soma_xy, soma_x2 = 0, 0, 0, 0
sxx, sxy = 0, 0

ve, vt = 0, 0

print('Insira os valores para a regressao. Para parar de inserir, digite o valor -9999')

while temp != -9999:
    temp = float(input('Insira um valor para x: '))
    if temp == -9999:
        break
    x.append(temp)
    temp = float(input('Insira um valor para y: '))
    if temp == -9999:
        x.pop()
        break
    y.append(temp)
    print(f'Valores x: {x}')
    print(f'Valores y: {y}')

n = len(x)

# calculo de somatorio utilizando metodo interno de listas:

soma_x = sum(x)

# calculo de somatorio utilizando laços de repetição:
# usando while
i = 0
while i < len(x):
    soma_x2 += x[i] ** 2
    soma_xy += x[i] * y[i]
    i += 1

# usando for
for i in range(len(x)):
    soma_y += y[i]

sxx = soma_x2 - soma_x ** 2 / n

sxy = soma_xy - soma_x * soma_y / n

a = sxy / sxx

ym = soma_y / n
xm = soma_x / n

b = ym - a * xm

# calculo coeficiente de regressao
for i in range(len(x)):
    ye = a * x[i] + b
    ve += (ye - ym) ** 2
    vt += (y[i] - ym) ** 2

r2 = ve / vt

print(f'A funcao de regressao e: y = {a:.4f}x + {b:.4f} (R2 = {r2:.4f})')
```

Regressão linear múltipla - Modelos lineares

Considerem agora uma situação onde se tem duas variáveis independentes x1 e x2, e que se deseja correlacionar tais variáveis com uma variável y através de um polinômio de primeiro grau. Por exemplo, seja o polinômio:

$$\hat{y} = b_1 x_1 + b_2 x_2 + a$$

A estimativa dos parâmetros do modelo linear b_1, b_2 e a também pode ser realizado aplicando-se o Método dos Mínimos Quadrados, tomando-se por base um conjunto de n dados experimentais (x_i,y_i), com i=1,....,n.

A minimização do erro quadrático médio entre y_i e $\hat{y}_i$ resulta no seguinte sistema matricial linear:

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum {x_{1i}}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum {x_{2i}}^2 \end{bmatrix} . \begin{Bmatrix} a \\ b_1 \\ b_2 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_{1i}\, y_i \\ \sum x_{2i}\, y_i \end{Bmatrix}$$

Este sistema de equações lineares com três incógnitas pode ser facilmente resolvido utilizando procedimentos de álgebra linear.

Por exemplo, recomenda-se revisar o Método da Eliminação de Gauss, também conhecido como pivoteamento, para aplicar ao sistema acima.

O método linear múltiplo é largamente utilizado na Engenharia Agrícola. Por exemplo, na Hidrologia, é muito utilizado para preenchimento de falhas em registros em estações experimentais para medições em bacias hidrográficas.

Recomenda-se aos leitores consultar procedimentos de medição de precipitação pluviométrica em bacias.

Em tais medições, é comum ocorrerem falhas de leituras, seja devido a problemas com operador ou a defeitos de funcionamento. Em tais casos, pode-se verificar a pertinência da aplicação dos modelos lineares para o preenchimento de tais falhas nas medições.

A pertinência do modelo pode ser verificada a partir do coeficiente de determinação R^2, que mede a qualidade do ajuste do modelo aos dados experimentais.

Exercício resolvido

Seja uma bacia hidrográfica com três pluviômetros, similar ao mostrado na Figura, localizados em diferentes locais de uma bacia. Em um dos locais, o pluviômetro apresentou defeito, e não foi possível medir a precipitação em um dado mês. Verifique a possibilidade de se preencher tal falha a partir de registros realizados nos dois outros locais. Em caso afirmativo, preencha tal falha.

Figura 4 – Imagem de um pluviômetro automático (fonte: www.cemaden.gov.br).

Mês	Pluviômetro 1 (mm)	Pluviômetro 2 (mm)	Pluviômetro 3 (mm)
Janeiro	40	43	49
Fevereiro	30	34	37
Março	22	Falha	31
Abril	12	15	19
Maio	53	57	63
Junho	58	63	68

Solução: Como se deseja estimar as leituras em Pluviômetro 2 a partir das medições em Pluviômetro 1 e Pluviômetro 3, então as leituras em Pluviômetro 2 serão dependentes das leituras em Pluviômetro 1 e Pluviômetro 3. Matematicamente, as leituras em Pluviômetro 2 serão denominadas y, e as de Pluviômetro 1 e Pluviômetro 3 serão denominadas de x1 e x2, respectivamente (nota: A solução não muda se Pluviômetro 1 fosse chamado de x2 e Pluviômetro 3 de x1).

Para a construção do modelo linear múltiplo serão considerados apenas os meses que ocorrem registros em todos os pluviômetros. Ou seja, serão considerados apenas os meses de janeiro, fevereiro, abril, maio e junho, totalizando 5 leituras (n=5).

Solicita-se aos alunos que preencham os elementos do sistema matricial:

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum {x_{1i}}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum {x_{2i}}^2 \end{bmatrix} . \begin{Bmatrix} a \\ b_1 \\ b_2 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_{1i}\, y_i \\ \sum x_{2i}\, y_i \end{Bmatrix}$$

Este sistema pode ser expresso, de forma compactada, como:

$[A].\{Z\} = [B]$

Donde

$\{Z\} = [A]^{-1}[B]$

A partir do cálculo da matriz inversa. Recomenda-se aos leitores consultar sobre inversão de matrizes. A solução está apresentada abaixo, a partir de uma planilha Excel.

	x1	x2	y	x1^2	x2^2	x1.x2	x1.y	x2.y
1	40	49	43	1600	2401	1960	1720	2107
2	30	37	34	900	1369	1110	1020	1258
3	12	19	15	144	361	228	180	285
4	53	63	57	2809	3969	3339	3021	3591
5	58	68	63	3364	4624	3944	3654	4284
Σ	193	236	212	8817	12724	10581	9595	11525
		ȳ	42.4					

|A|		x	{Z}	=	|B|
5	193	236		a		212
193	8817	10581	x	b1	=	9595
236	10581	12724		b2		11525

{Z}	=	|A|^1			x	|B|
a		26.7287	4.8104	-4.4960		212
b1	=	4.8104	0.9211	-0.8552	x	9595
b2		-4.4960	-0.8552	0.7946		11525

{Z}	=	|A|^-1 x |B|
a		6.1338
b1	=	1.7160
b2		-0.6350

$$y = 1.7160 \times b1 - 0.6350 \times b2 + 6.1338$$

Notem que o R^2 do modelo linear múltiplo foi elevado (alto). Portanto, o modelo pode ser utilizado para estimar a precipitação em Pluv2 no mês de março. Segundo a planilha, o valor estimado para a falha é de 24,20.

Encoraja-se os leitores a inverterem a Matriz de Coeficientes com base nas regras de Álgebra Linear. Maior detalhamento teórico pode ser encontrado em Hoffman & Kunze (1979).

No caso, a unidade de medida é mm. Ou seja, a falha é estimada em 24,20 mm.

A seguir é apresentado algoritmo em linguagem Python para resolução de regressão múltipla. Optou-se pelo uso da biblioteca Numpy para realizar operações com matrizes (inversão e multiplicação de matrizes):

```
import numpy as np

x1, x2, y = [], [], []
temp = 0

soma_x1, soma_x2, soma_x12, soma_x22, soma_x1x2, soma_y, soma_x1y, soma_x2y = 0, 0, 0, 0, 0, 0, 0, 0
sxx, sxy = 0, 0

ve, vt = 0, 0

print('Insira os valores para a regressao. Para parar de inserir, digite o valor -9999')

while temp != -9999:
    temp = float(input('Insira um valor para x1: '))
    if temp == -9999:
        break
    x1.append(temp)

    temp = float(input('Insira um valor para x2: '))
    if temp == -9999:
        break
    x2.append(temp)

    temp = float(input('Insira um valor para y: '))
    if temp == -9999:
        x1.pop()
        x2.pop()
        break
    y.append(temp)
    print(f'Valores x1: {x1}')
    print(f'Valores x2: {x2}')
    print(f'Valores y: {y}')

n = len(x1)

soma_x1 = sum(x1)
soma_x2 = sum(x2)
soma_y = sum(y)

for i in range(len(x1)):
    soma_x12 += x1[i] ** 2
    soma_x22 += x2[i] ** 2
    soma_x1x2 += x1[i] * x2[i]
    soma_x1y += x1[i] * y[i]
    soma_x2y += x2[i] * y[i]

matrizA = np.array([[n, soma_x1, soma_x2],
                    [soma_x1, soma_x12, soma_x1x2],
                    [soma_x2, soma_x1x2, soma_x22]])

matrizB = np.array([[soma_y],
                    [soma_x1y],
                    [soma_x2y]])

matrizA_inversa = np.linalg.inv(matrizA)

coeficientes = np.dot(matrizA_inversa, matrizB)

b1 = float(coeficientes[1][0])
b2 = float(coeficientes[2][0])
a = float(coeficientes[0][0])

ym = soma_y / n

for i in range(len(x1)):
    ye = a + x1[i]*b1 + x2[i]*b2
    ve += (ye - ym) ** 2
    vt += (y[i] - ym) ** 2

r2 = ve / vt

print(f'A funcao de regressao e: y = {b1:.4f}x1 + {b2:.4f}x2 + {a:.4f} (R2 = {R2:.4f})')
```

Exercícios numéricos

1. Determine a função linear que modela o seguinte conjunto de dados e determine o Coeficiente de Determinação (R2) para a função ajustada:

x1	x2	Y
2	3	4
3	7	8
4	9	15
7	16	26

2. Considere os dados abaixo, referentes a chuvas em uma bacia hidrográfica. Utilizando uma regressão linear múltipla, preencha as falhas ocorridas. Verifique a qualidade do ajuste do modelo aos dados obtidos.

I	P1	P2	P3
1	30	39	43
2	38	43	48
3	2	3	5
4	12	Falha	18
5	25	31	36
6	28	34	Falha
7	92	110	121
8	15	17	21
9	45	49	52

Exercícios computacionais

1. Desenvolva um algoritmo em linguagem de programação Python para realizar a modelagem dos dados inseridos pelo usuário e determinar o Coeficiente de Determinação (R^2).

Resolução de sistemas de equações lineares

Nas páginas anteriores, foram verificadas aplicações matemáticas que exigiam a solução de sistemas de equações lineares e simultâneas.

De uma forma geral, pode-se escrever um sistema de equações da seguinte forma:

$[A].\{Z\} = [B]$

em que [A] é uma matriz de coeficientes, {Z} o vetor com as incógnitas e [B] o vetor de termos independentes.

Caso a matriz [A] seja inversível, então se pode escrever:

$\{Z\} = [A]^{-1}[B]$

Assim, as estimativas em {Z} podem ser obtidas a partir do cálculo da matriz inversa. Recomenda-se aos leitores consultar sobre inversão de matrizes. Recomenda-se ao leitor pesquisar a metodologia de cálculo da matriz Adjunta e da Inversa, quando existir.

Outras técnicas podem ser utilizadas para obtenção do vetor {Z}, podendo ser classificadas em técnicas diretas ou em técnicas iterativas. Veremos a seguir o detalhamento de tais técnicas, para solução de sistemas lineares.

Técnica direta - Eliminação de Gauss

Esta técnica se aplica à matriz expandida do sistema anterior, composta por {A:B}. Para um sistema nxn, a matriz expandida terá dimensões nx(n+1). O Método da Eliminação de Gauss é baseado em operações elementares entre linhas, tais que não alterem a solução do sistema original. Dentre essas técnicas elementares, podem ser citadas:

- Multiplicação de uma equação por uma constante real diferente de zero;
- Soma entre equações;
- Mudança da ordem das equações.

A combinação de tais técnicas deve ser adotada de modo a transformar um sistema de equações independentes em um sistema tipo triangular.

De forma geral, podemos representar um sistema com n equações e incógnitas a serem resolvidas através da triangularização da seguinte forma, a ser desenvolvida na matriz expandida:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{in} & b_i \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & b_n \end{bmatrix} \quad \begin{matrix} l_1^1 \\ l_2^1 \\ \vdots \\ l_i^1 \\ \vdots \\ l_n^1 \end{matrix}$$

Na qual cada linha $l_i^1, i = 1{:}n$ representa uma equação do sistema linear a ser triangularizado.

Para eliminação da primeira coluna, A PARTIR da linha 2, assumindo-se que $a_{11} \neq 0$ devemos multiplicar a primeira linha por um fator e somar com os elementos da linha considerada.

Assim, para a linha i, devemos multiplicar a primeira linha pelo fator constante

$m_{i1} = (-1) * a_{i1}/a_{11}$

e em seguida realizar a soma da linha 1 transformada pela linha i.

Com isso, podemos escrever para $l_i, i = 2 : n$ os seguintes termos

$$a_{ij}^{(2)} = a_{ij} + m_{i1}a_{1j}\,, j = 2{:}\,n$$

$$b_i^{(2)} = b_i + m_{i1}b_1$$

O índice 2 em $a_{ij}^{(2)}$ e $b_i^{(2)}$ indica que será utilizado um segundo valor para a_{ij} e b_i.

Com isso temos a seguinte configuração para a matriz aumentada ou expandida:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} & b_n^{(2)} \end{bmatrix}$$

Para eliminação da segunda coluna, procedimento semelhante deve ser adotado a partir da segunda linha.

Para zerar o segundo elemento (da segunda coluna) na terceira equação (ou linha) até a última repetimos o procedimento para eliminação da primeira coluna, desta vez tomando como base a segunda equação, assumindo-se que $a_{22} \neq 0$.

Para a linha i o fator será, então:

$$m_{i2} = (-1) * a_{i2}/a_{22} \quad l_i^{(3)} = l_i^{(2)} + m_{i2}l_2^{(2)}, i = 3 : n$$

Os novos elementos obtidos a partir da eliminação da segunda coluna podem ser expressos como:

$$a_{ij}^{(3)} = a_{ij}^{(2)} + m_{i2}a_{2j}^{(2)}, j = 3{:}\,n$$

$$b_i^{(3)} = b_i^{(2)} + m_{i2}b_2^{(2)}$$

As colunas que estão abaixo da diagonal principal da matriz de coeficientes [A] serão eliminadas utilizando o mesmo procedimento até que a matriz modificada se torne triangular.

Os elementos $a_{11}, a_{22}^{(2)}, \dots, a_{jj}^{(j)}$ são denominados pivôs. Após o procedimento de eliminação, chega-se a uma matriz triangular com a seguinte configuração:

$$\begin{aligned} a_{11}x_1 \quad a_{12}x_2 + \dots \quad + a_{1n}x_n &= b_1 \\ a_{22}^{(2)}x_2 + \dots \quad + a_{2n}^{(2)}x_n &= b_2^{(2)} \\ \vdots \qquad\qquad\quad \vdots &\quad\ \vdots \\ a_{nn}^{(n)}x_n &= b_n^{(n)} \end{aligned}$$

Caso algum dos pivôs se anule durante o procedimento de eliminação das colunas, devemos trocar linhas, escolhendo a linha imediatamente abaixo para evitar a perda da eliminação anterior.

Exercício resolvido

Utilizar o Método da Eliminação de Gauss para triangularizar o sistema:

$$2x_1 + 3x_2 - x_3 = 5$$

$$4x_1 + 4x_2 - 3x_3 = 3$$

$$2x_1 - 3x_2 + x_3 = -1$$

A matriz expandida por ser escrita como:

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 1 & 1 \end{bmatrix}$$

O primeiro pivô é a_{11}= 2.

Para eliminação da primeira coluna a partir da linha 2 devemos considerar os sucessivos fatores:

$$m_{i1} = (-1) * a_{i1}/a_{11}$$

para i=2 e i=3.

Desse modo, temos para i=2:

$$m_{21} = \frac{(-1) * a_{21}}{a_{11}} = (-1) * \frac{4}{2} = -2$$

$$a_{2j}^{(2)} = a_{2j} + m_{21}a_{1j}\,, j = 2{:}n$$

$$b_2^{(2)} = b_2 + m_{21}b_1$$

Substituindo,

$$a_{2j}^{(2)} = a_{2j} + (-2) * a_{1j}\,, j = 2{:}n$$

$$b_2^{(2)} = b_2 + (-2) * b_1$$

Para i=3, temos:

$$m_{31} = \frac{(-1) * a_{31}}{a_{11}} = (-1) * \frac{2}{2} = -1$$

$$a_{3j}^{(2)} = a_{3j} + m_{31}a_{1j}\,, j = 2{:}n$$

$$b_3^{(2)} = b_3 + m_{31}b_1$$

$$a_{3j}^{(2)} = a_{3j} + (-1) * a_{1j}\,, j = 2{:}n$$

$$b_3^{(2)} = b_3 + (-1) * b_1$$

Desse modo, podemos escrever que a matriz expandida

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 1 & 1 \end{bmatrix}$$

Transforma-se em:

$$\begin{matrix} 2 & 3 & -1 & 5 \\ 0 & 4+(-2)*3 & -3+(-2)*(-1) & 3+(-2)*5 \\ 0 & -3+(-1)*3 & 1+(-1)*(-1) & -1+(-1)*5 \end{matrix}$$

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & -6 & 2 & -6 \end{bmatrix}$$

Para eliminar a segunda coluna na terceira linha, devemos considerar o pivô $a_{22}^{(2)} = -2$.

O fator será $m_{32} = (-1) * a_{32}/a_{22}$=(-1)*(-6)/(-2)= -3

e a nova linha 3 poderá ser expressa a partir de:

$$a_{3j}^{(3)} = a_{3j}^{(2)} + m_{32}a_{2j}^{(2)} = a_{3j}^{(2)} + (-3) * a_{2j}^{(2)}$$

$$b_3^{(3)} = b_3^{(2)} + m_{32}b_2^{(2)} = b_3^{(2)} + (-3) * b_2^{(2)}$$

Então, a matriz expandida

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & -6 & 2 & -6 \end{bmatrix}$$

Pode ser reescrita como:

$$\begin{matrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & 2+(-3)*(-1) & -6+(-3)*(-7) \end{matrix}$$

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & 5 & 15 \end{bmatrix}$$

A partir deste sistema triangular, obtém-se:

$$x_3 = \frac{15}{5} = 3$$

$$-2x_2 - x_3 = -7;\ -2x_2 = -7 + x_3 = -7 + 3;\ x_2 = 2$$

$$2x_1 + 3x_2 - x_3 = 5;\ \ 2x_1 = 5 - 3x_2 + x_3 = 5 - 3 * 2 + 3;\ \ 2x_1 = 2;\ x_1 = 1$$

Técnica indireta - Iterativa: Método de Jacobi

Considerem o sistema linear

$$\begin{array}{lllll} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \dots & a_{1n}x_n = \quad b_1 \\ a_{21}x_1 + & a_{22}x_2 + & a_{23}x_3 + & \dots & a_{2n}x_n = \quad b_2 \\ \vdots & \vdots & \vdots & & \vdots \quad \vdots \\ a_{i1}x_1 + & a_{i2}x_2 + & a_{i3}x_3 + & \dots & a_{in}x_n = \quad b_i \\ \vdots & \vdots & \vdots & & \vdots \quad \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & a_{n3}x_3 + & \dots & a_{nn}x_n = \quad b_n \end{array}$$

Pode-se escrever

$$x_i = (b_i - \sum_{\substack{j=1 \\ i \neq j}}^{n} a_i * x_j)/a_{ii}$$

∀ i, i = 1...n

Desse modo,

$$\left\{ \begin{array}{l} x_1 = \dfrac{b_1 - a_{12} * x_2 - a_{13} * x_3 - \cdots - a_{1n} * x_n}{a_{11}} \\ x_2 = \dfrac{b_2 - a_{21} * x_1 - a_{23} * x_3 - \cdots a_{2n} * x_n.}{a_{22}} \\ \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots. \\ x_n = \dfrac{b_n - a_{n1} * x_1 - a_{n2} * x_2 - \cdots . - a_{nn-1} * x_{n-1}}{a_{nn}} \end{array} \right\}$$

O equacionamento acima pode ser utilizado como base para um processo iterativo, ou "passo-a-passo", desde que se forneça uma inicialização para o processo. Ao final de cada passo, deve-se calcular o erro para cada estimativa, por exemplo, calculando-se o valor absoluto da diferença entre as iterações consecutivas. O máximo erro ao final de cada iteração deve ser inferior ao erro admissível, para que o processo convirja.

Para detalharmos este processo, vamos considerar a aplicação abaixo.

Exercício resolvido

Resolver o sistema de equações

$$4x_1 + 0{,}24x_2 - 0{,}08x_3 = 8$$

$$0{,}09x_1 + 3x_2 - 0{,}15x_3 = 9$$

$$0{,}04x_1 - 0{,}08x_2 + 4x_3 = 20$$

Com a seguinte inicialização:

$$\boldsymbol{I}_{\mathrm{x}}^{(\mathbf{0})} =< 2{,}0; 3{,}0; 5{,}0 >$$

Trabalhe com 4 casas decimais, e adote **erro admissível de 0,001.**

Solução:

Pode-se atualizar cada valor de x_i da seguinte forma:

Iteração 1:

$$x_1 = \frac{8 - 0{,}24 * 3 + 0{,}08 * 5}{4} = 1{,}9200$$

$$x_2 = \frac{9 - 0{,}09 * 2 + 0{,}15 * 5}{3} = 3{,}1900$$

$$x_3 = \frac{20 - 0{,}04 * 2 + 0{,}08 * 3}{4} = 5{,}0400$$

Estimativa do erro na iteração 1:

E_1:

$$E_{x_1} = |1{,}92 - 2{,}0| = 0{,}08$$
$$E_{x_2} = |3{,}19 - 3{,}0| = 0{,}19$$
$$E_{x_3} = |5{,}04 - 5{,}0| = 0{,}04$$

Como o erro máximo é maior que o erro admissível, então se deve continuar o processo.

Iteração 2:

$$\boldsymbol{I_x^{(2)} =< 1{,}9200; 3{,}1900; 5{,}0400 >}$$

$$x_1 = \frac{8 - 0{,}24 * 3{,}1900 + 0{,}08 * 5{,}0400}{4} = 1{,}9094$$

$$x_2 = \frac{9 - 0{,}09 * 1{,}9200 + 0{,}15 * 5{,}0400}{3} = 3{,}1944$$

$$x_3 = \frac{20 - 0{,}04 * 1{,}9200 + 0{,}08 * 3{,}1900}{4} = 5{,}0446$$

Estimativa do erro na iteração 2:

E_2:

$$E_{x_1} = |1{,}9094 - 1{,}9200| = 0{,}0106$$
$$E_{x_2} = |3{,}1944 - 3{,}1900| = 0{,}0044$$
$$E_{x_3} = |5{,}0446 - 5{,}0400| = 0{,}0046$$

Como o erro máximo é maior que o erro admissível, então se deve continuar o processo.

Iteração 3:

$$\boldsymbol{I_x^{(3)} =< 1,9094; 3,1944; 5,0446 >}$$

$$x_1 = \frac{8 - 0{,}24 * 3{,}1944 + 0{,}08 * 5{,}0446}{4} = 1{,}9092$$

$$x_2 = \frac{9 - 0{,}09 * 1{,}9094 + 0{,}15 * 5{,}0446}{3} = 3{,}1949$$

$$x_3 = \frac{20 - 0{,}04 * 1{,}9094 + 0{,}08 * 3{,}1944}{4} = 5{,}0448$$

E_3:

$$E_{x_1} = |1{,}9092 - 1{,}9094| = 0{,}0002$$
$$E_{x_2} = |3{,}1949 - 3{,}1944| = 0{,}0005$$
$$E_{x_3} = |5{,}0448 - 5{,}0446| = 0{,}0002$$

$$\boldsymbol{E_{adm} = 0{,}001 > E_{máx} = 0{,}0005}$$

CONVERGIU!

Plotando-se o erro máximo em função da iteração, obtemos:

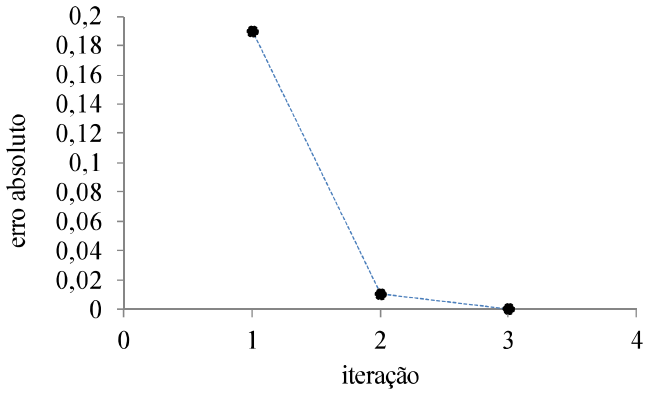


Figura 5 - Gráfico do processo de convergência do método iterativo de Jacobi.

Algoritmo do Método de Jacobi

De posse no que foi apresentado no exercício anterior, o Método de Jacobi se desenvolve a partir da seguinte equação básica, na iteração k+1:

$$x_i^{(k+1)} = (b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} * x_j^{(k)})/a_{ii}$$

i=1, ..., n

que se repetirá até que Erro(k+1)<= Eadm

Pensando computacionalmente, o algoritmo para a aplicação do método iterativo de Jacobi segue o seguinte fluxo:

1. Recebe-se a entrada de dados, contendo as equações de recorrência, os valores estimados iniciais e um erro admissível adotado.
2. A cada iteração, calcula-se um novo valor aproximado para as variáveis de interesse e verifica-se o erro.
3. SE o maior erro for inferior ao erro admissível, parar a execução do programa, SENÃO, continuar para a próxima iteração
4. ENQUANTO o maior erro for superior ao erro permitido adotado pelo problema, voltar ao passo 2, realizando novas aproximações.

O algoritmo está representado visualmente a seguir:

Método de Gauss-Seidel

Este método é baseado no Método de Jacobi, apresentando como modificação a atualização das estimativas já obtidas na mesma iteração para estimativa das variáveis seguintes. Assim, o valor recém calculado de $x_1^{(k+1)}$ será utilizado para calcular $x_2^{(k+1)}$, e assim por diante, na expectativa de acelerar o processo iterativo.

A equação básica do Método de Gauss-Seidel pode então ser expressa como:

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} * x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} * x_j^{(k)}))/a_{ii}$$

Para um sistema 3x3, podemos escrever:

$$x_1^{(k+1)} = (b_1 - (\sum_{j=2}^{3} a_{1j} * x_j^{(k)}))/a_{11}$$

$$x_2^{(k+1)} = (b_2 - (a_{21} * x_1^{(k+1)} + a_{23} * x_3^{(k)})/a_{22}$$

$$x_3^{(k+1)} = (b_3 - (\sum_{j=1}^{2} a_{3j} * x_j^{(k+1)}))/a_{33}$$

Exercício resolvido

Resolver o sistema abaixo pelo Método de Gauss-Seidel, com erro admissível de 0,001.

$$4x + 0.2y - 0.1z = 12$$

$$0.06x + 3y - 0.18z = 9$$

$$0.05x + 0.07y + 4z = 24$$

A matriz expandida pode ser escrita como:

4	0.2	-0.1	12
0.06	3	-0.18	9
0.05	0.07	4	24

Erro admissível:
0.001

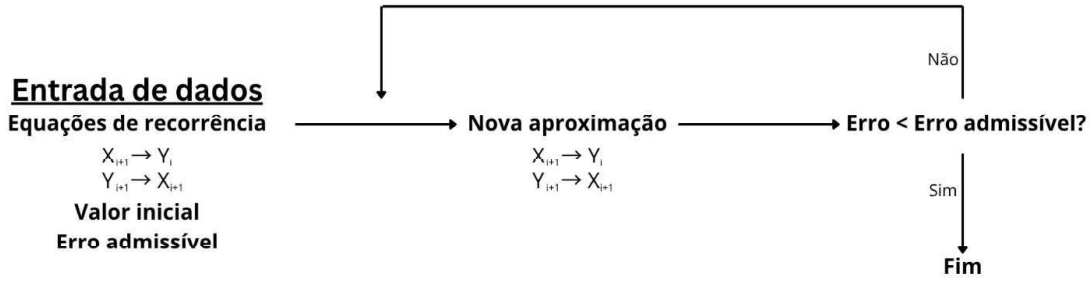
Que pode ser simplificada para:

1	0.05	-0.025	3
0.02	1	-0.06	3
0.0125	0.0175	1	6

Iteração	0	1	2	3	4	5
x1	0	3.0000	3.0007	2.9828	2.9828	2.9828
x2	0	2.9400	3.2946	3.2946	3.2946	3.2946
x3	0	5.9110	5.9048	5.9050	5.9050	5.9050
Erro1	-	3.0000	0.0007	0.0178	0.0000	0.0000
Erro2	-	2.9400	0.3546	0.0000	0.0000	0.0000
Erro3	-	5.9110	0.0062	0.0002	0.0000	0.0000
Máximo		5.9110	0.3546	0.0178	0.0000	0.0000
Se		Não convergiu	Não convergiu	Não convergiu	Convergiu	Convergiu

Algoritmo do Método de Gauss-Seidel

O algoritmo do Método Iterativo de Gauss-Seidel é semelhante ao apresentado para o Método Iterativo de Jacobi, observando-se a modificação referente à utilização da aproximação mais recente no cálculo das aproximações das variáveis, conforme representação:



Observa-se que para o cálculo de uma nova aproximação do valor de Y, por exemplo, pode ser utilizada a aproximação mais recente do X.

Programação aplicada

Aplicando os entendimentos de programação na linguagem Python para o desenvolvimento de um programa simples que resolva sistemas de equações 2x2, podemos obter diversos resultados diferentes.

Abaixo estão apresentados códigos desenvolvidos utilizando a linguagem Python:

1. Método Iterativo de Jacobi:

```
print('Entrada:')
print('#' * 100)
recorrencia = []

print(f'Escreva a equacao de recorrencia para a variavel x: ')
recorrencia.append(input('>>> ').lower())

print(f'Escreva a equacao de recorrencia para a variavel y: ')
recorrencia.append(input('>>> ').lower())

print(f'Escreva a equacao de recorrencia para a variavel z: ')
recorrencia.append(input('>>> ').lower())

print(f'Indique a inicializacao para a variavel x: ')
x = float(input('>>> '))

print(f'Indique a inicializacao para a variavel y: ')
y = float(input('>>> '))

print(f'Indique a inicializacao para a variavel z: ')
z = float(input('>>> '))

print('Indique o erro admissivel: ')
erroadm = float(input('>>> '))

xtemp = (eval(recorrencia[0]))
ytemp = (eval(recorrencia[1]))
ztemp = (eval(recorrencia[2]))

erroatual1 = abs(x - xtemp)
erroatual2 = abs(y - ytemp)
erroatual3 = abs(z - ztemp)

while erroatual1 > erroadm or erroatual2 > erroadm or erroatual3 > erroadm:
    x = xtemp
    y = ytemp
    z = ztemp

    print(f'x = {x:.2f} ({erroatual1:.4f})| y = {y:.2f} ({erroatual2:.4f}) | z = {z:.2f} ({erroatual3:.4f})')

    xtemp = (eval(recorrencia[0]))
    ytemp = (eval(recorrencia[1]))
    ztemp = (eval(recorrencia[2]))

    erroatual1 = abs(x - xtemp)
    erroatual2 = abs(y - ytemp)
    erroatual3 = abs(z - ztemp)
```

2. Método Iterativo de Gauss-Seidel:

```
print('Entrada:')
print('#' * 100)
recorrencia = []

print(f'Escreva a equacao de recorrencia para a variavel x: ')
recorrencia.append(input('>>> ').lower())

print(f'Escreva a equacao de recorrencia para a variavel y: ')
recorrencia.append(input('>>> ').lower())

print(f'Escreva a equacao de recorrencia para a variavel z: ')
recorrencia.append(input('>>> ').lower())

print(f'Indique o chute inicial para a variavel x: ')
x = float(input('>>> '))

print(f'Indique o chute inicial para a variavel y: ')
y = float(input('>>> '))

print(f'Indique o chute inicial para a variavel z: ')
z = float(input('>>> '))

print('Indique o erro admissivel: ')
erroadm = float(input('>>> '))

xtemp = (eval(recorrencia[0]))
temp = x
x=xtemp
xtemp = temp

ytemp = (eval(recorrencia[1]))

ztemp = (eval(recorrencia[2]))

erroatual1 = abs(x - xtemp)
erroatual2 = abs(y - ytemp)
erroatual3 = abs(z - ztemp)
print(f'x = {xtemp:.2f} | y = {y:.2f} | z = {z:.2f}')

while erroatual1 > erroadm or erroatual2 > erroadm:
    y = ytemp
    z = ztemp
    print(f'x = {x:.2f} ({erroatual1:.4f}) | y = {y:.2f} ({erroatual2:.4f} | z = {z:.2f} ({erroatual3:.4f})')
    xtemp = (eval(recorrencia[0]))
    temp = x
    x = xtemp
    xtemp = temp
    ytemp = (eval(recorrencia[1]))
    ztemp = (eval(recorrencia[2]))
    erroatual1 = abs(x - xtemp)
    erroatual2 = abs(y - ytemp)
    erroatual3 = abs(z - ztemp)
```

Exercícios numéricos

1. Resolva o seguinte sistema de equações lineares por método direto:

$$8x + 2y - 2z = -2$$
$$10x + 2y + 4z = 4$$
$$12x + 2y + 2z = 6$$

2. Triangularizar o sistema de equações a seguir utilizando o método de Gauss

$$2x1 + 3x2 - x3 = 5$$
$$4x1 + 4x2 - 3x3 = 3$$
$$2x1 + 3x2 + x3 = -1$$

3. No sistema da questão 2, a partir do sistema já triangularizado, estime os valores de x1, x2, e x3 por substituição sucessiva, a partir da última equação.
4. Resolva os sistemas de equações lineares das questões anteriores utilizando método iterativo (indireto)

REFERÊNCIAS

Cunha, M.C.C. Métodos Numéricos, Editora da UNICAMP, 276p., 2003

Hoffman, K.; Kunze, R. Algebra linear, Livros Técnicos e Científicos Editora S.A., 1979, 514p.

Lopes, D.C; Melo, E.C. Desenvolvimento de Algoritmos, Notas de Aula, Departamento de Engenharia Agrícola, Viçosa, 2002.
ftp://ftp.ufv.br/dea/Disciplinas/Evandro/Eng691/Material%20Didatico/ApostilaAlgoritmos.pdf

Martins, G. Estatística Geral e Aplicada, Editora Atlas, 204p., 2010

Raízes de funções implícitas

As funções implícitas são aquelas para as quais não é possível isolar ou explicitar a variável independente ou incógnita em função das demais variáveis. Assim, requerem usualmente métodos alternativos, sejam gráficos ou numéricos, para se estimar a incógnita.
Nesta seção, iremos apresentar e discutir os métodos numéricos iterativos para estimar a raiz de uma função implícita f(x).

Método de Newton

Este método se desenvolve a partir de uma inicialização (estimativa inicial) para a incógnita, que é sucessivamente atualizada com base na derivada de primeira ordem da função implícita em análise.

Da definição do cálculo diferencial,

$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x} \cong \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$, para Δx pequeno.

Aplicando-se a equação acima como base de um processo iterativo em busca da raiz, e assumindo que $f(x_0 + \Delta x)$ está próximo de zero (o que equivale dizer que $x_0 + \Delta x$ está próximo da raiz, então:

$$f'(x_0) * \Delta x \cong -f(x_0)$$

Considerando-se que $\Delta x = x_1 - x_0$, então:

$$\Delta x = x_1 - x_0 = \frac{-f(x_0)}{f'(x_0)}$$

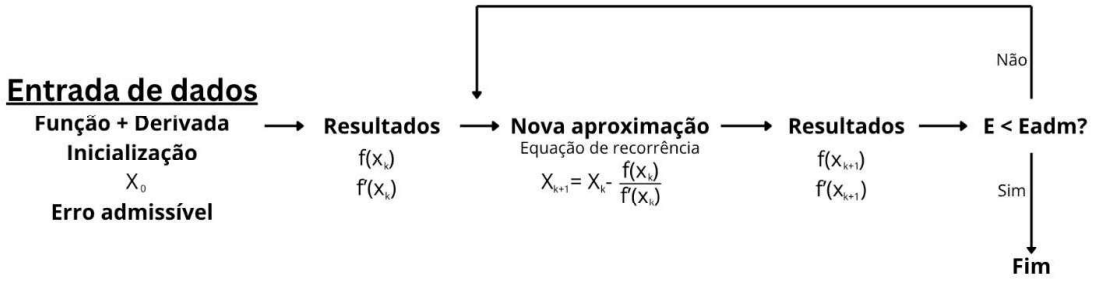
e daí,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

A equação pode ser generalizada para um processo iterativo, da seguinte forma:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Podemos então esquematizar o algoritmo do método de Newton, conforme fluxograma abaixo:



Construído o algoritmo, podemos guiar a construção do software utilizando a linguagem Python:

Exercício resolvido

O problema abaixo sugere utilizarmos o método de Newton para encontrar uma aproximação para uma raiz da função $\mathbf{x^3 + 10x^2 + 2x - 15}$. A questão propõe uma inicialização igual a **1,0** e, além disso, o erro admissível na questão é de **0,10%**.

Resolução:
1º Passo - Montar a tabela "x0; f(x); f'(x); Erro";
2º Passo - Começar o preenchimento da tabela conforme a inicialização escolhida;
3º Passo - Para obter o próximo *x0,* aplicar a Equação de Newton: $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
4º Passo - Repetir o esquema até que o mesmo obedeça à tolerância estabelecida no problema.

Utilizar o método de Newton para encontrar uma aproximação para a raiz da seguinte função: $x^3 + 10x^2 + 2x - 15$

Função:	$x^3 + 10x^2 + 2x - 15$
Erro admissível:	0.10%
Inicialização:	1

x	f(x)	f'(x)	Erro	
1.0000	-2.0000	25.0000		
1.0800	0.0837	27.0992	2.0992	
1.0769	0.0001	27.0174	0.0818	
1.0769	0.0000	27.0173	0.0001	Convergiu!!! (Eadm < 0.001)

Método das Secantes

Este método se baseia no Método de Newton, considerando uma aproximação para a derivada $f'(x_k)$

Pode-se escrever:

$$f'(x_k) \cong \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Substituindo-se na equação de Newton,

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}}$$

Realizando os algebrismos necessários, chega-se a:

$$x_{k+1} = \frac{f(x_k) * x_{k-1} - f(x_{k-1}) * x_k}{f(x_k) - f(x_{k-1})}$$

Exercício resolvido

O problema abaixo sugere utilizarmos o método das Secantes para encontrar uma aproximação para a raiz da função **$x^3 + 10x^2 + 2x - 15$**. A questão propõe DUAS inicializações iguais a **1,0 e 1,5** e, além disso, o erro admissível na questão é de **0,10%**.

Resolução:

1º Passo - Montar a tabela "x0; f(x); Erro";

2º Passo - Começar o preenchimento da tabela conforme a inicialização escolhida;

3º Passo - Para obter o próximo *x0,* aplicar a Equação das secantes: $x_{k+1} = \frac{f(x_k)*x_{k-1} - f(x_{k-1})*x_k}{f(x_k) - f(x_{k-1})}$

4° Passo - Repetir o esquema até que o mesmo obedeça a tolerância estabelecida no problema.

Obs: A vantagem deste método é a não utilização da derivada da função-problema para a descoberta da raiz da função. Entretanto, requer DUAS inicializações.

Utilizar o método das Secantes para encontrar uma aproximação para a raiz da seguinte função: $x^3 + 10x^2 + 2x - 15$

Função:	$x^3 + 10x^2 + 2x - 15$
Erro admissível:	0.10%
Inicializações:	1.00 1.50

x	f(x)	Erro	
1.0000	-2.0000		
1.5000	13.8750		
1.0630	-0.3734	0.3069	
1.0744	-0.0665	0.0669	
1.0769	0.0005	0.0005	Convergiu!!! (Eadm < 0.001)

Aplicação de Engenharia do Método das Secantes

Uma importante aplicação do Método das Secantes é a estimativa da lâmina d´água que escoa em um canal, para uma certa vazão. Abaixo são apresentadas algumas características físicas e geométricas, para compreensão da equação matemática a ser aplicada. Este processo físico pode ser descrito pela Equação de Manning.

Equação de Manning

Essa equação pode ser aplicada em canais com escoamentos permanentes (invariáveis no tempo) e uniformes (invariáveis no espaço), para cálculo da vazão Q:

$$Q = \frac{\sqrt{I}}{n} \mathrm{A}\, R_h^{\frac{2}{3}}$$

em que:

$$R_h = \frac{A}{P}$$

sendo:

n – Rugosidade da parede ou revestimento do canal;

R_h – Raio hidráulico, definido como igual à relação entre a área transversal da seção de escoamento A, e o perímetro molhado P, este último definido a partir da linha de contato entre a água e a parede e fundo do canal;

I- Declividade de fundo (m/m).

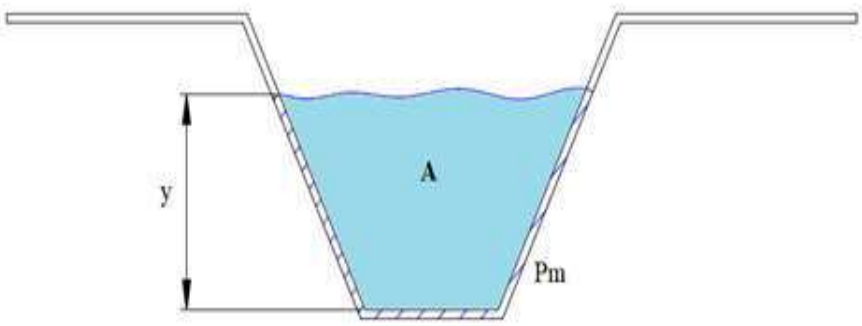


Figura 6 - Seção transversal de um canal trapezoidal com lâmina d´água y.

Para aplicação do Método das Secantes, vamos considerar uma seção retangular de base b. Então, a área e o perímetro molhado podem ser dados por:

A= by e P= b+2y

Substituindo-se na Equação de Manning, obtém-se:

$$Q = \frac{\sqrt{I}}{n} * (b * y) * (\frac{b * y}{b + 2y})^{2/3}$$

que é implícita em relação à lâmina y. Desse modo, encontrar y que satisfaça à equação de Manning equivale a encontrar a raiz da seguinte função:

$$F(y) = \frac{\sqrt{I}}{n} * (b * y) * (\frac{b * y}{b + 2y})^{2/3} - Q$$

Exercício resolvido

Estimar a lâmina d´água y em um canal retangular de base b= 2 m, rugosidade n= 0,01, e declividade de fundo I= 0,001, transportando uma vazão Q= 2 m^3/s. Aplicar o Método das Secantes, adotando erro admissível de 0,001 e inicializações y0= 0,5 m e y1= 0,6 m.

Rugosidade:	0.01	
Vazão:	2	m3/s
Declividade de fundo:	0.001	m/m
Base:	2	m
Lâminas estimadas iniciais:		
y0:	0.5	m
y1:	0.6	m
Erro admissível:	0.001	

i	Lâmina	f(y)	Erro
0	0.5	-0.48	0.4797
1	0.6	-0.027	0.0266
2	0.6058	0.0008	0.000797
3	0.6057	0	0.000001

Para o caso de canais trapezoidais, então as expressões de área e perímetro molhado passam a ser, respectivamente:

$$A = (b + zy)y$$

$$P = b + 2y\sqrt{1 + z^2}$$

onde "z" representa o talude lateral do canal. A função implícita resultante oriunda da Equação de Manning pode ser expressa por:

$$F(y) = \frac{\sqrt{I}}{n} * ((b + zy) * y) * (\frac{(b + zy) * y}{b + 2y\sqrt{1 + z^2}})^{2/3} - Q$$

que pode ser resolvida pelos Métodos de Newton e Secantes:
Algoritmos aplicados aos Métodos Iterativos de Newton e Secantes:

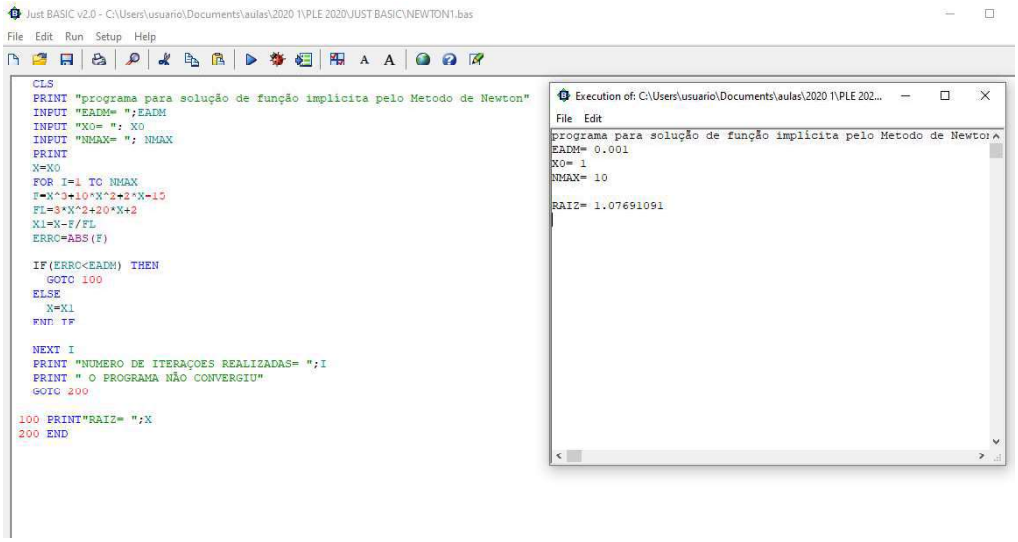
O **Método de Newton** pode ser executado seguindo o seguinte algoritmo:

a) Sejam dados f(x), f´(x), Eadm;
b) Entre com x0, e com um "número máximo de iterações" nmax;
c) Faça k=1, 2, ..., nmax;
d) Calcule $x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$;
e) Calcule ABS $(f(x_k))$;
f) SE ABS $(f(x_k))$< Eadm, ENTÃO: x*=x_k, e encerra-se o processo.
g) SENÃO incrementa-se o valor de k e retorna-se ao passo "d", desde que o número máximo de iterações nmax não tenha sido atingido.

Para o **Método das Secantes**, o algoritmo acima pode ser levemente modificado, ficando:

a) Sejam dados f(x), f´(x), Eadm;
b) Entre com x0, x1 e com um "número máximo de iterações" nmax;
c) Faça k= 1,2, ..., nmax;
d) Calcule $x_{k+1} = \frac{f(x_k)*x_{k-1} - f(x_{k-1})*x_k}{f(x_k) - f(x_{k-1})}$
e) Calcule ABS $(f(x_{k+1}))$;
f) SE ABS $(f(x_{k+1}))$< Eadm, ENTÃO: x*=x_{k+1}, e encerra-se o processo.
g) SENÃO incrementa-se o valor de k e retorna-se ao passo "d", desde que o número máximo de iterações nmax não tenha sido atingido.

Usando o algoritmo de Newton, e programando em Basic, temos abaixo a solução para o polinômio y= **$x^3 + 10x^2 + 2x - 15$.**



A implementação acima foi realizada em JustBasic, que é um ambiente de programação em Basic gratuito e muito útil. Recomenda-se sua instalação e aplicação. Duas estruturas de programação são muito úteis, tendo sido aplicadas acima:

A estrutura FOR-NEXT, que estabelece um laço, onde um conjunto de comandos é repetido um certo número de vezes;

A estrutura IF-THEN-ELSE-ENDIF, para testes lógicos e tomadas de decisão.

Outro comando que merece destaque é o de desvio, o GOTO, que desvia para um rótulo especificado. Notem que no programa acima duas linhas de programação tinham rótulos: 100 e 200.

Programação em BASIC - Armazenando Dados

Primeiramente, vamos considerar a estrutura de armazenamento por meio de matrizes ou vetores.

$DIM\ X\ (10)$

O comando **DIM** dimensiona um vetor, no caso um vetor X(I), com 10 posições.

Exemplo:

X	1	2	3	4	5	X(I)
	↑	↑	↑	↑	↑	
	1ª	2ª	3ª	4ª	5ª	
Y	1,8	4,3	5,9	9,4	10,1	Y(I)

$$Y(4) = 9{,}4$$

$$Y(2) = 4{,}3$$

Laço de programação

No Basic, o laço de programação pode ser representado pelos comandos **FOR-NEXT**, com uma sintaxe que deve envolver um contador, e um valor limite para que o laço seja executado.

Seja o somatório abaixo:

$$\sum_{I=1}^{5} X_I = X_1 + X_2 + X_3 + X_4 + X_5 = X(1) + X(2) + X(3) + X(4) + X(5)$$

↓

$SomaX$

Supondo que os valores de Xi foram previamente armazenados em um vetor de pelo menos 5 posições (através de, por exemplo, DIM X(7)), então o somatório pode ser programado como:

```
SomaX = 0
For I = 1 To 5
SomaX = SomaX + X(I)
Next I
```

A variável *SomaX* irá armazenar o valor do somatório procurado.

Suponha agora que se deseje somar o *quadrado* de N números. Então, matematicamente, pode-se escrever:

$$\sum_{I=1}^{N} X_I^2$$

↓

$SomaX2$

As linhas de programação serão:

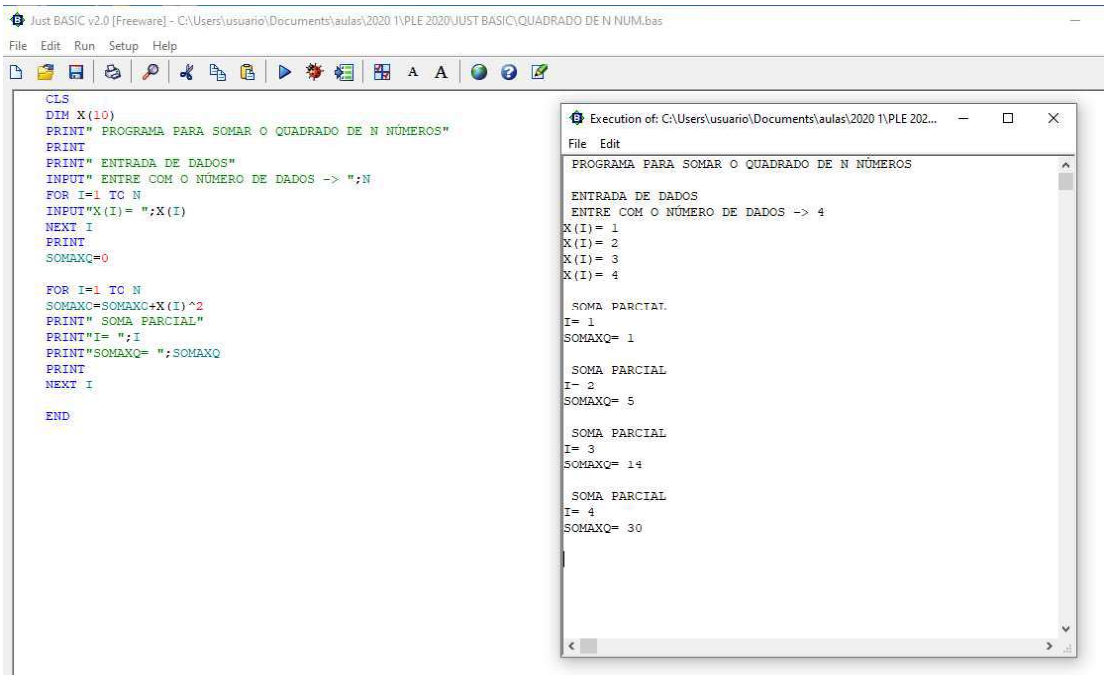
$SomaX2 = 0$

$For\ I = 1\ To\ N$

$SomaX2 = SomaX2 + X(I)^2$

$Next\ I$

Na Plataforma JustBasic, a programação do algoritmo acima pode ficar:



Em Python, possibilidades de resolução dos problemas pelos métodos de Newton e Secantes são apresentadas a seguir:

Método de Newton:

```
#Entrada
funcao = [] # 0 = funcao, 1 = derivada

print(f'Escreva a função implícita: ')
funcao.append(input('>>> ').lower())

print(f'Escreva a derivada da função implícita: ')
funcao.append(input('>>> ').lower())

print(f'Escreva a inicialização da variavel x: ')
x = float(input('>>> '))

print(f'Indique o erro admissivel: ')
eadm = float(input('>>> '))

fx = eval(funcao[0])
dx = eval(funcao[1])

print(f"xi-1 = {x:10.2f} | f(xi-1) = {fx:10.2f} | f'(x-1) = {dx:10.2f}")

xtemp = x
fxtemp = fx
dxtemp = dx

x = xtemp - (fx/dx)

fx = eval(funcao[0])
dx = eval(funcao[1])

while abs(x - xtemp) > eadm:
    print(f"xi-1 = {x:10.4f} | f(xi-1) = {fx:10.4f} | f'(x-1) = {dx:10.4f} | Erro = {abs(x - xtemp):10.4f}")

    xtemp = x
    fxtemp = fx
    dxtemp = dx

    x = xtemp - (fx / dx)
    fx = eval(funcao[0])
    dx = eval(funcao[1])

print(f"xi-1 = {x:10.4f} | f(xi-1) = {fx:10.4f} | f'(x-1) = {dx:10.4f} | Erro = {abs(x - xtemp):10.4f}\n")

print(f"Raiz aproximada: {x:10.5f} | f(x) = {fx:10.5f}")
```

Método das secantes:

```
# Entrada
funcao = []  # 0 = funcao
varx = []
varfx = []

print(f'Escreva a função implícita: ')
funcao.append(input('>>> ').lower())

print(f'Escreva a primeira inicialização da variável x: ')
varx.append(float(input('>>> ')))

print(f'Escreva a segunda inicialização da variável x: ')
varx.append(float(input('>>> ')))

print(f'Indique o erro admissível: ')
eadm = float(input('>>> '))

print(varx)

for i in range(2):
    x = varx[i]
    fx = eval(funcao[0], {"x": x})
    varfx.append(fx)
    print(f"{x:10.4f} | f({x:2.4f}) = {varfx[-1]:10.4f}")

while abs(varx[-1] - varx[-2]) > eadm:
    x = (varfx[-1] * varx[-2] - varfx[-2] * varx[-1]) / (varfx[-1] - varfx[-2])
    # x = varx[-1] - (varfx[-1] * (varx[-1] - varx[-2])) / (varfx[-1] - varfx[-2])
    fx = eval(funcao[0])

    varx.append(x)
    varfx.append(fx)

    print(f"{x:10.4f} | f({x:2.4f}) = {varfx[-1]:10.4f} | Erro = {abs(varx[-1] - varx[-2]):10.4f}")

print(f"Raiz aproximada: {varx[-1]:10.6f} | f(x) = {varfx[-1]:10.6f}")
```

Exercícios numéricos

1. Determine, utilizando método numérico, uma raiz para as funções. Inicialize o método com o valor 1.0:
    a. f(x) = $x^3 + 3x^2 + 10x - 10$
    b. f(h) = $h^3 - 0.5h^2 - 15h + 3$
    c. f(x) = $1.4x^4 - 4x$

Interpolação

Polinômio interpolador de Lagrange

Lagrange propôs um polinômio interpolador que passa por (n+1) pontos conhecidos (x_0,y_0), (x_1,y_1), ..., (x_n,y_n), de modo que $x_0 \neq x_1 \neq .... \neq x_n$.

Desse modo, conhece-se $y_0=f(x_0)$, $y_1=f(x_1)$, ..., $y_n=f(x_n)$, e deseja-se encontrar p(x) tal que $p(x_0)=f(x_0)$, $p(x_1)=f(x_1)$, ..., $p(x_n)=f(x_n)$.

Este polinômio pode ser expresso por:

$$p(x) = \sum_{i=0}^{n} Y_i L_i$$

onde os coeficientes L_i devem ser tais que $L_i(x_k)=1$ quando i=k, e $L_i(x_k)=0$, se i≠k.

$$L_i(x) = \frac{\prod_{\substack{j=0 \\ j\neq i}}^{n}(x - x_j)}{\prod_{\substack{j=0 \\ j\neq i}}^{n}(x_i - x_j)}$$

Por exemplo, para 4 pontos (x_0,y_0), (x_1,y_1), (x_2,y_2), (x_3,y_3), pode-se escrever:

$$p(x) = L_0 y_0 + L_1 y_1 + L_2 y_2 + L_3 y_3$$

$$L_0(x) = \frac{(x - x_1).(x - x_2).(x - x_3)}{(x_0 - x_1).(x_0 - x_2).(x_0 - x_3)}$$

$$L_1(x) = \frac{(x - x_0).(x - x_2).(x - x_3)}{(x_1 - x_0).(x_1 - x_2).(x_1 - x_3)}$$

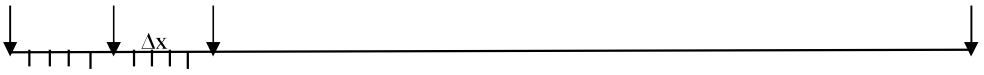
$$L_2(x) = \frac{(x - x_0).(x - x_1).(x - x_3)}{(x_2 - x_0).(x_2 - x_1).(x_2 - x_3)}$$

$$L_3(x) = \frac{(x - x_0).(x - x_1).(x - x_2)}{(x_3 - x_0).(x_3 - x_1).(x_3 - x_2)}$$

Exercício resolvido

Suponha que foi realizada uma batimetria para avaliar a profundidade de um rio. A primeira medida foi realizada na extremidade esquerda, e a última na extremidade direita (profundidade zero) (as margens direita e esquerda são convencionadas com base no sentido do fluxo, com o observador olhando para jusante). A partir das N+1 medições, deseja-se aplicar um polinômio interpolador de grau N para refinar a avaliação do fundo do rio, tomando-se intervalos regulares Δx (discretização). Este problema está resolvido a seguir, utilizando-se plataforma Excel.

No exercício, as medições foram efetuadas a cada 50 cm (indicadas pelas setas), e suponha que se deseja refinar para intervalos de 10 cm (traços menores).



	Pontos	nível
A0	0,00	0,00
A1	0,50	-0,40
A2	1,00	-1,40
A3	1,50	-2,10
A4	2,00	-2,60
A5	2,50	-2,20
A6	3,00	-1,80
A7	3,50	-1,20
A8	4,00	0,00

Construir um polinômio que represente o leito de fundo de um rio, a partir de medições de profundidades , tomadas em intervalos regulares. Iniciar em uma das margens, adotar intervalos de 0,1 m, até atingir a margem oposta.

X	L0	L1	L2	L3	L4	L5	L6	L7	L8	P(x)
0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
0,10	0,56	1,12	-1,75	2,24	-2,07	1,31	-0,54	0,13	-0,01	-0,08
0,20	0,29	1,54	-2,02	2,48	-2,24	1,40	-0,58	0,14	-0,02	-0,14
0,30	0,13	1,53	1,53	1,79	1,58	0,97	0,40	0,10	0,01	0,20
0,40	0,04	1,31	-0,76	0,83	-0,71	0,44	-0,18	0,04	0,00	-0,29
0,50	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,40
0,60	-0,01	0,69	0,60	-0,54	0,43	-0,25	0,10	-0,02	0,00	-0,55
0,70	-0,02	0,42	0,99	-0,74	0,57	-0,33	0,13	-0,03	0,00	-0,73
0,80	-0,01	0,22	1,16	-0,66	0,48	-0,27	0,11	-0,02	0,00	-0,94
0,90	0,00	0,08	1,14	-0,38	0,26	-0,14	0,05	-0,01	0,00	-1,16
1,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	-1,40
1,10	0,00	-0,04	0,78	0,39	-0,22	0,11	-0,04	0,01	0,00	-1,64
1,20	0,00	-0,04	0,54	0,72	-0,34	0,17	-0,06	0,01	0,00	-1,86
1,30	0,00	-0,03	0,31	0,94	-0,34	0,16	-0,06	0,01	0,00	-2,07
1,40	0,00	-0,02	0,13	1,03	-0,22	0,09	-0,03	0,01	0,00	-2,25
1,50	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	-2,40
1,60	0,00	0,01	-0,07	0,86	0,27	-0,10	0,03	-0,01	0,00	-2,51
1,70	0,00	0,02	-0,09	0,66	0,55	-0,16	0,05	-0,01	0,00	-2,59
1,80	0,00	0,01	-0,08	0,42	0,78	-0,18	0,05	-0,01	0,00	-2,63
1,90	0,00	0,01	-0,04	0,19	0,94	-0,13	0,03	-0,01	0,00	-2,63
2,00	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	-2,60
2,10	0,00	-0,01	0,03	-0,13	0,94	0,19	-0,04	0,01	0,00	-2,55
2,20	0,00	-0,01	0,05	-0,18	0,78	0,42	-0,08	0,01	0,00	-2,47
2,30	0,00	-0,01	0,05	-0,16	0,55	0,66	-0,09	0,02	0,00	-2,39
2,40	0,00	-0,01	0,03	-0,10	0,27	0,86	-0,07	0,01	0,00	-2,29
2,50	0,00	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00	-2,20
2,60	0,00	0,01	-0,03	0,09	-0,22	1,03	0,13	-0,02	0,00	-2,11
2,70	0,00	0,01	-0,06	0,16	-0,34	0,94	0,31	-0,03	0,00	-2,02
2,80	0,00	0,01	-0,06	0,17	-0,34	0,72	0,54	-0,04	0,00	-1,95
2,90	0,00	0,01	-0,04	0,11	-0,22	0,39	0,78	-0,04	0,00	-1,87
3,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	0,00	-1,80
3,10	0,00	-0,01	0,05	-0,14	0,26	-0,38	1,14	0,08	0,00	-1,72
3,20	0,00	-0,02	0,11	-0,27	0,48	-0,66	1,16	0,22	-0,01	-1,63
3,30	0,00	-0,03	0,13	-0,33	0,57	-0,74	0,99	0,42	-0,02	-1,52
3,40	0,00	-0,02	0,10	-0,25	0,43	-0,54	0,60	0,69	-0,01	-1,38
3,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	-1,20
3,60	0,00	0,04	-0,10	0,44	-0,71	0,03	-0,76	1,31	0,04	-0,98
3,70	-0,01	0,10	-0,40	0,97	-1,58	1,79	-1,53	1,53	0,13	-0,73
3,80	-0,02	0,14	-0,58	1,40	-2,24	2,48	-2,02	1,54	0,29	-0,46
3,90	-0,01	0,13	-0,54	1,31	-2,07	2,24	-1,75	1,12	0,56	-0,20
4,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00

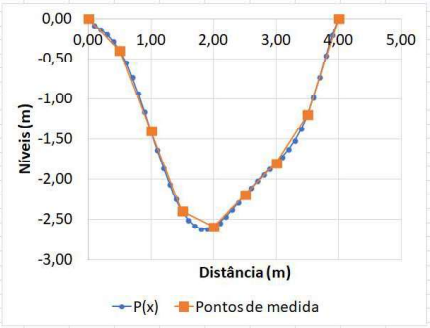


Figura 7- Interpolação aplicada à batimetria de um riacho.

A estimativa de x* a partir de um polinômio de Lagrange pode-se tornar repetitivo e enfadonho, a depender da dimensão do problema. Assim, sua implementação computacional é indicada.

Exercício resolvido

Construir um polinômio de Lagrange passando pelos pontos (1, 1); (2, 4); (3, 9)

Resolução:

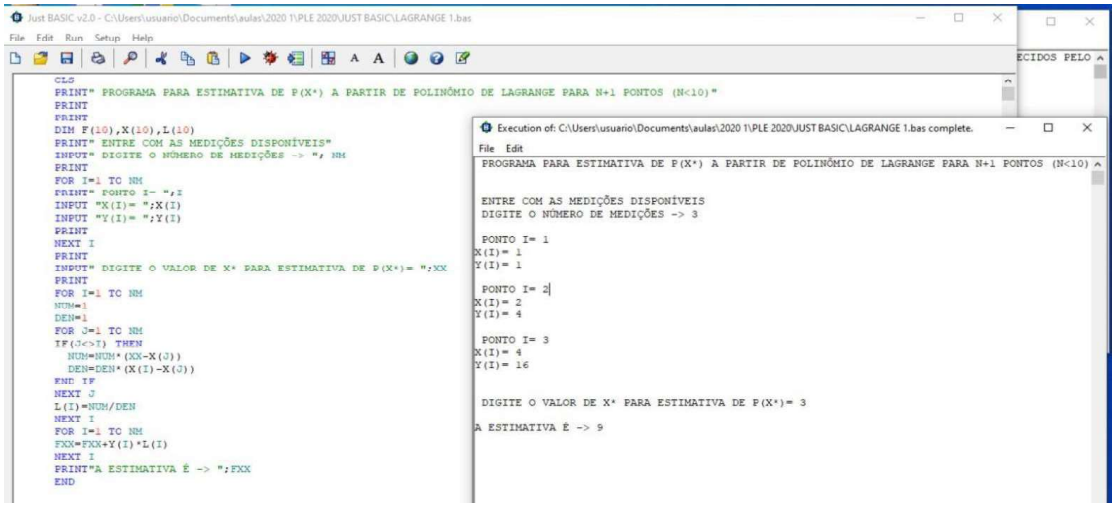
Aplicando a equação do polinômio de Lagrange, temos:

$$P_2(x) = 1 * \frac{(x-2).(x-3)}{(1-2).(1-3)} + 4 * \frac{(x-1).(x-3)}{(2-1).(2-3)} + 9 * \frac{(x-1).(x-2)}{(3-1).(3-2)}$$

Realizando-se os procedimentos algébricos necessários à determinação do polinômio, temos que:

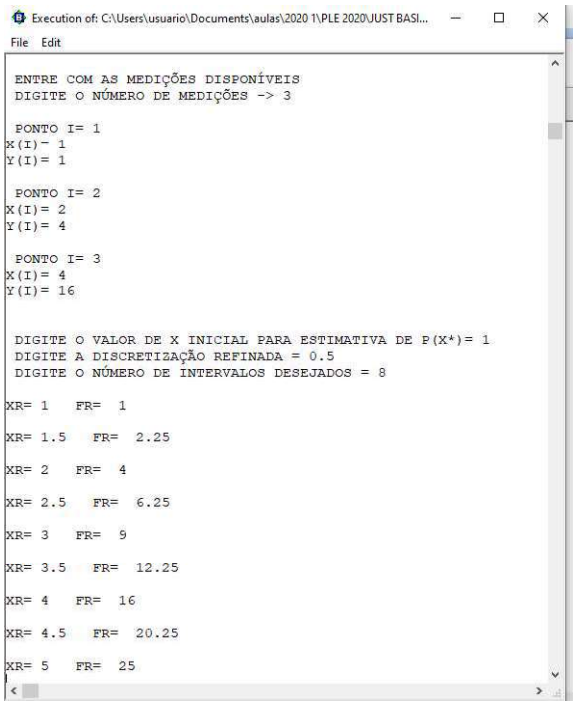
$$P_2(x) = x^2$$

O código em Basic para estimativa de p(x*), sendo p(x) um polinômio de Lagrange, está apresentado a seguir, juntamente com sua execução.



Para codificar o cálculo efetuado na planilha Excel, pode-se facilmente inserir um laço externo, e também se deve dimensionar as matrizes (vetores) correspondentes. O código abaixo apresenta a solução.

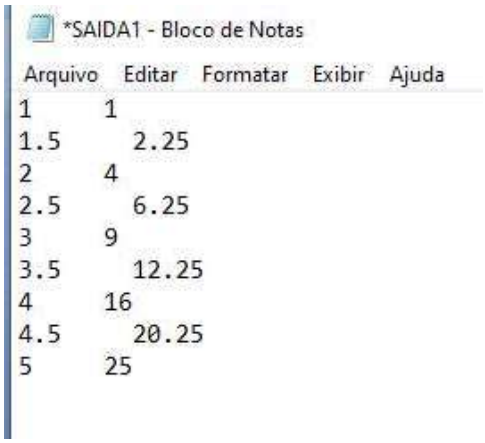
```
CLS
PRINT" PROGRAMA PARA ESTIMATIVA DE P(X*) A PARTIR DE POLINÔMIO DE LAGRANGE PARA N+1 PONTOS (N<10)"
PRINT
PRINT" FORNECE VALORES DE P(X*) PARA UMA DISCRETIZAÇÃO MAIS REFINADA DX"
PRINT
DIM Y(10),X(10),L(10),XR(100),FR(100)
PRINT
PRINT" ENTRE COM AS MEDIÇÕES DISPONÍVEIS"
INPUT" DIGITE O NÚMERO DE MEDIÇÕES -> "; NM
PRINT
FOR I=1 TO NM
PRINT" PONTO I= ";I
INPUT "X(I)= ";X(I)
INPUT "Y(I)= ";Y(I)
PRINT
NEXT I
PRINT
INPUT" DIGITE O VALOR DE X INICIAL PARA ESTIMATIVA DE P(X*)= ";XX
INPUT" DIGITE A DISCRETIZAÇÃO REFINADA = ";DX
INPUT" DIGITE O NÚMERO DE INTERVALOS DESEJADOS = ";NINTERV
FOR K=1 TO NINTERV+1
XR(K)=XX+(K-1)*DX
PRINT
FOR I=1 TO NM
NUM=1
DEN=1
FOR J=1 TO NM
IF(J<>I) THEN
 NUM=NUM*(XR(K)-X(J))
 DEN=DEN*(X(I)-X(J))
END IF
NEXT J
L(I)=NUM/DEN
NEXT I
FOR I=1 TO NM
FR(K)=FR(K)+Y(I)*L(I)
```



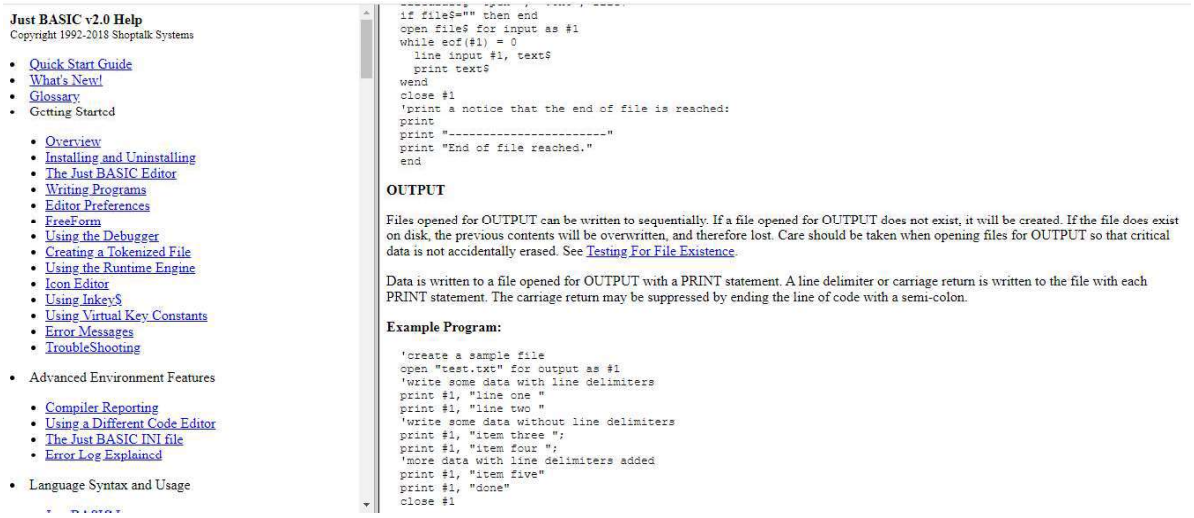
```
NEXT I
PRINT"XR= ";XR(K);
PRINT"   FR=  ";FR(K)
NEXT K
OPEN "SAIDA1.OUT" FOR OUTPUT AS #1
FOR I=1 TO NINTERV+1
PRINT #1,XR(I);
PRINT #1,"     ";
PRINT #1,FR(I)
NEXT I
CLOSE #1
END
```

Encoraja-se os leitores a implementarem o código acima, e atentarem para a estrutura de I/O (entrada/saída) adotada para salvar os resultados em um arquivo, com nome fornecido pelo usuário. A execução do programa está apresentada na janela ao lado. Notem a discretização refinada adotada, com incrementos de 0,5 a partir de 1, gerando assim valores de x de 1; 1,5; 2; 2,5; 3; 3,5; 4; 4,5; 5. Temos um total de nove números, uma vez que foram adotados na simulação oito intervalos.

O arquivo de saída "saida1.out" tem o seguinte formato:



Mais detalhes podem ser encontrados no HELP do JustBasic, disponível em: file:///C:/Program%20Files%20(x86)/Just%20BASIC%20v2.0/jb2help/jb20help.html.



Exercício complementar

Utilizando a formulação de Lagrange, apresente a solução analítica para um polinômio linear passando pelos pontos (x_1, y_1) e (x_2,y_2).

A seguir é apresentada uma versão do algoritmo solucionador de interpolação pelo método de Lagrange utilizando a linguagem Python:

```
x = []
y = []
l = []
n = int(input("Qual o número de pontos desejado? "))
xint = float(input("Para qual valor deseja interpolar? "))

i = 0
print('Insira os valores conhecidos para x e y.')
while i < n:
    x.append(float(input("x: ")))
    y.append(float(input("y: ")))
    i += 1

k = 0
for k in range(0, n):
    num = 1
    den = 1
    i = 0
    for i in range(0, n):
        if i != k:
            num *= (xint - x[i])
            den *= (x[k] - x[i])
    l.append(num/den)

p = 0
for i in range(len(l)):
    p += l[i]*y[i]

print(f'O valor interpolado para x = {xint} é y = {p:.2f}')
```

Exercícios numéricos

1. Estabelecer o polinômio de Lagrange que passe pelos pontos (0,2), (1,4), (2,3).
2. Estabelecer o polinômio de Lagrange que passe pelos pontos (0,2), (1,5), (3,8), (5, 2).
3. Estabelecer o polinômio de Lagrange que passe pelos pontos (0,3), (2,6), (3,2), (4,8). Plote este polinômio no Excel, no intervalo [-1, 5], com discretização de 0,5.

Exercícios computacionais

Motobombas Centrífugas Monoestágio
BC-22 - Rotor fechado

Aplicações Gerais:

Irrigação, abastecimento predial, sistemas de refrigeração, sistemas de prevenção e combate a incêndio, indústrias.


BC-22 R


BC-22 R Mancal


BC-22 F

MODELO	Potência (cv)	Monofásico	Trifásico	Ø Sucção (pol)	Ø Recalque (pol)	Pressão máxima sem vazão (m.c.a.)	Altura máxima de sucção (m.c.a.)	Ø Rotor (mm)	CARACTERÍSTICAS HIDRÁULICAS																				
									Altura Manométrica Total (m.c.a.)																				
									20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	56			
									Vazão em m^3/h válida para sucção de 0 m.c.a.																				
BC-22 R 1 A	3	x	x	1 1/4	1	39	8	155	10,2	9,7	9,2	8,7	8,1	7,5	6,8	6,1	5,2												
	4	x	x	1 1/4	1	48	8	169	*	*	10,7	10,3	9,9	9,4	8,9	8,4	7,9	7,3	6,7	6,0	5,2	4,3							
	5	x	x	1 1/4	1	59	8	186	*	*	*	11,1	10,8	10,4	10,1	9,7	9,4	9,0	8,6	8,2	7,7	7,2	6,7	6,1	5,5				

1. Estabeleça um polinômio para representar a curva da bomba de 3 CV, adotando três pontos da Tabela. A curva da bomba deve apresentar como variável independente a vazão e como variável dependente a "altura manométrica".

Integração Numérica

A integral de uma função contínua f(x), em um intervalo fechado [a,b], corresponde à área compreendida entre a função e o eixo dos x na figura 8.

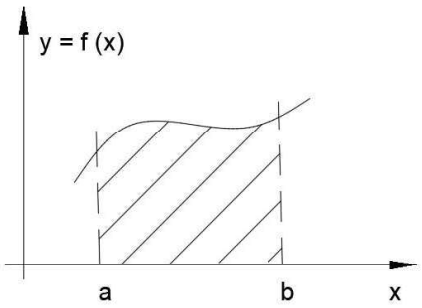


Figura 8- Representação gráfica da integração de f(x).

Suponha agora que se adotou uma discretização uniforme Δx para o intervalo [a,b]. então, pode-se escrever a seguinte aproximação:

$$I = \int_a^b f(x)dx \cong \sum f(x)\Delta x$$

desde que o intervalo Δx seja pequeno.

A aproximação acima é particularmente útil quando não se conhece a expressão analítica de f(x), porém se dispõe de estimativas de f(x) em pontos x especificados.

Método dos Trapézios

O Método dos Trapézios pode ser desenvolvido a partir da equação anterior, considerando os trapézios formados a partir da interpolação linear entre dois pontos $f(x_i)$ e $f(x_i + \Delta x)$, conforme a Figura 9.

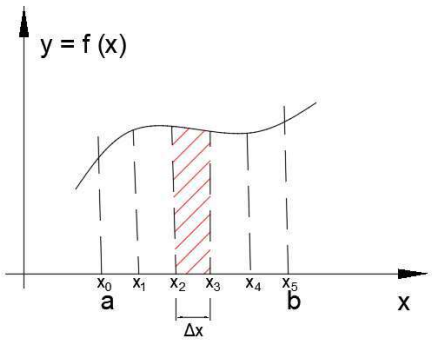
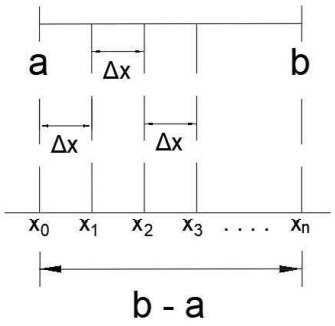


Figura 9- Discretização do intervalo [a,b].

Discretização

$$I \cong \frac{\Delta x}{2}\big(f(x_0) + f(x_1)\big) + \frac{\Delta x}{2}\big(f(x_1) + f(x_2)\big) + \frac{\Delta x}{2}\big(f(x_2) + f(x_3)\big) + \frac{\Delta x}{2}\big(f(x_3) + f(x_4)\big) + \frac{\Delta x}{2}\big(f(x_4) + f(x_5)\big)$$

$$I \cong \frac{\Delta x}{2}\big(f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + 2f(x_4) + f(x_5)\big)$$



$$I \cong \frac{\Delta x}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n) \right)$$

$$\Delta x = \frac{b-a}{n} \rightarrow x_0, x_1, \ldots, x_{n-1}, x_n$$

$$n \rightarrow intervalos$$

Exercício resolvido

Aplicar o Método dos Trapézios para estimar a integral de $f(x) = x^2$, no intervalo [0;4]. Adotar primeiramente uma discretização Δx igual a 1. Em seguida, repita o processo para Δx igual a 0,5. Compare as soluções aproximadas, e calcule também a solução analítica. Quantos pontos foram considerados para estimativa da integral, em cada discretização? E quantos intervalos?

Função:	$f(x) = x^2$
Intervalo:	[0;4]
Discretizações:	[1;0.5]

Solução analítica:
I = x^3/3
21.333

Solução numérica:			
Δx=1		Δx=0.5	
x	f(x)	x	f(x)
0.00	0.00	0.00	0.00
1.00	1.00	0.50	0.25
2.00	4.00	1.00	1.00
3.00	9.00	1.50	2.25
4	16.00	2.00	4.00
		2.50	6.25
		3.00	9.00
		3.50	12.25
		4.00	16.00
I = 1/2 * (0 + 2*(1+4+9) + 16))		I = 0.5/2 * (0 + 2*(0.25+1+2.25+4+6.25+9+12.25) + 16))	
I = 22		I = 21.5	

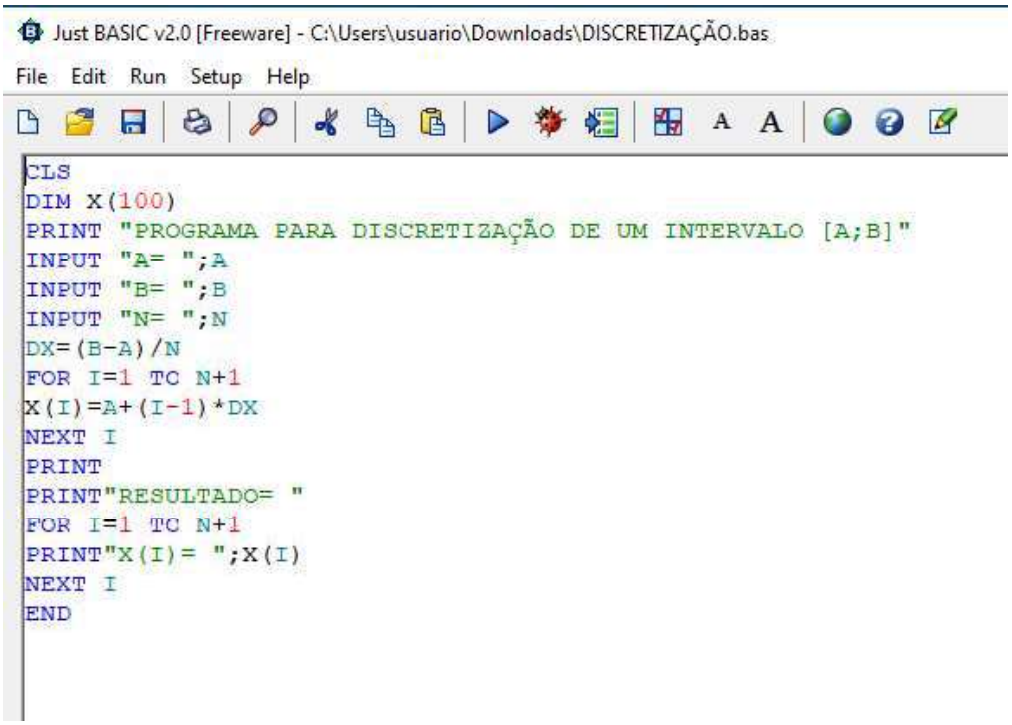
Discussão a respeito da discretização do intervalo de integração

Os procedimentos de integração discutidos até o momento requerem a discretização do intervalo de interesse [a,b] em subintervalos de comprimento $\Delta x = \frac{b-a}{n}$, onde n é o número de subintervalos.

Vamos fixar este procedimento, focando apenas na construção do intervalo discretizado.

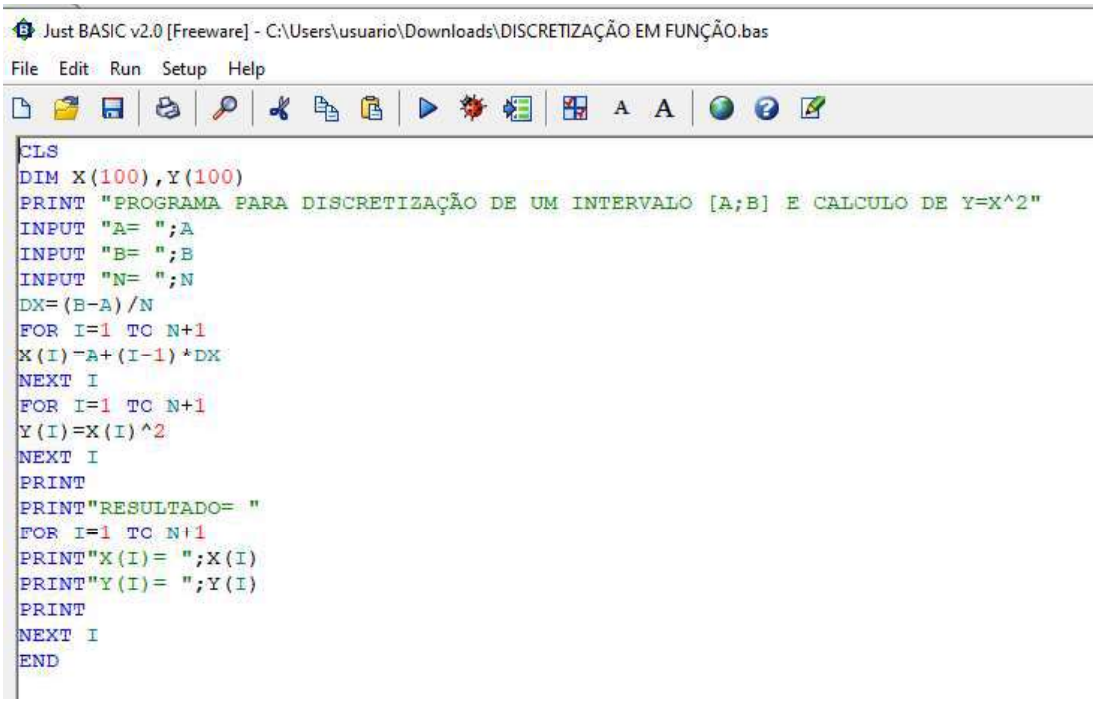
Aplicação: Estimar os valores de x_i no intervalo [a,b], obtidos a partir da discretização do intervalo em n subintervalos. Elaborar um programa em Basic para armazenar esses (n+1) valores de x_i. Utilizar o comando INPUT para entrada de a, b e n.

A programação para a aplicação acima está transcrita abaixo (Rotina 1):



Aplicação: Com base na discretização obtida no exercício anterior, calcule o valor de y(x)=x^2 em cada um dos pontos da discretização. Elabore um programa em Basic para realizar tal cálculo e exibir na tela.

A rotina computacional para solução está apresentada a seguir (Rotina 2).



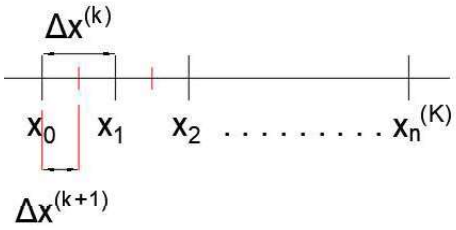
Implementação complementar em Basic

Implementar a rotina computacional (Rotina 3) abaixo, para realizar a integral de A*X^2 +B*X+C no intervalo [L1;L2].

```
CLS
DIM F(100),X(100)
PRINT "VAMOS CALCULAR A INTEGRAL DE
A*X^2+B*X+C NO INTERVALO [L1; L2]"
INPUT " DIGITE O VALOR DE L1= "; L1
INPUT " DIGITE O VALOR DE L2= "; L2
INPUT " DIGITE O VALOR DE A= "; A
INPUT " DIGITE O VALOR DE B= "; B
INPUT " DIGITE O VALOR DE C= "; C
INPUT " DIGITE O VALOR DE N= "; N
VX=(L2-L1)/N
SOMAF=0
X(0)=L1
FOR I=0 TO N
X(I+1)=X(I)+VX
NEXT I
FOR J=0 TO N
F(I)=A*X(J)^2+B*X(J)+C
NEXT J
FOR I=1 TO N-1
SOMAF=F(I)+SOMAF
NEXT I
K=(VX/2)*(F(0)+2*SOMAF+F(N))
PRINT ""; K
END
```

Considere um processo iterativo tal que:

$$\Delta x^{(k+1)} = \frac{\Delta x^{(k)}}{2}$$



$$DX^{(0)} = \frac{DXI}{2^0} = \frac{DXI}{1} = DXI$$

$$DX^{(1)} = \frac{DXI}{2^1} = \frac{DXI}{2}$$

$$DX^{(2)} = \frac{DXI}{2^2} = \frac{DXI}{4}$$

A rotina computacional pode ser escrita como:

```
CLS
DIM F(100),X(100)
PRINT "VAMOS CALCULAR A*X^2+B*X+C"
INPUT " DIGITE O VALOR DE L1= "; L1
INPUT " DIGITE O VALOR DE L2= "; L2
INPUT " DIGITE O VALOR DE A= "; A
INPUT " DIGITE O VALOR DE B= "; B
INPUT " DIGITE O VALOR DE C= "; C
INPUT " DIGITE O VALOR DE N= "; N
DXI=(B-A)/N
NI=N
FOR K=0 TO NINTER
DX=DXI/(2^K)
N=(2^K)*NI
FOR I=1 TO N
X(I+1)=X(I)+DX
NEXT I
FOR I=0 TO N
F(I)=A*X(I)^2+B*X(I)+C
NEXT I
SOMA=0
FOR I=1 TO N-1
SOMA=SOMA+F(I)
NEXT I
INTEG=(DX/2)*(F(0)+2*SOMA+F(N))
END
```

Inserir no programa anterior um laço externo para o processo iterativo. $NITER = 50$. Incluir um alerta ao usuário quando o programa não conseguir em $NITER$ iterações.

Teste de convergência

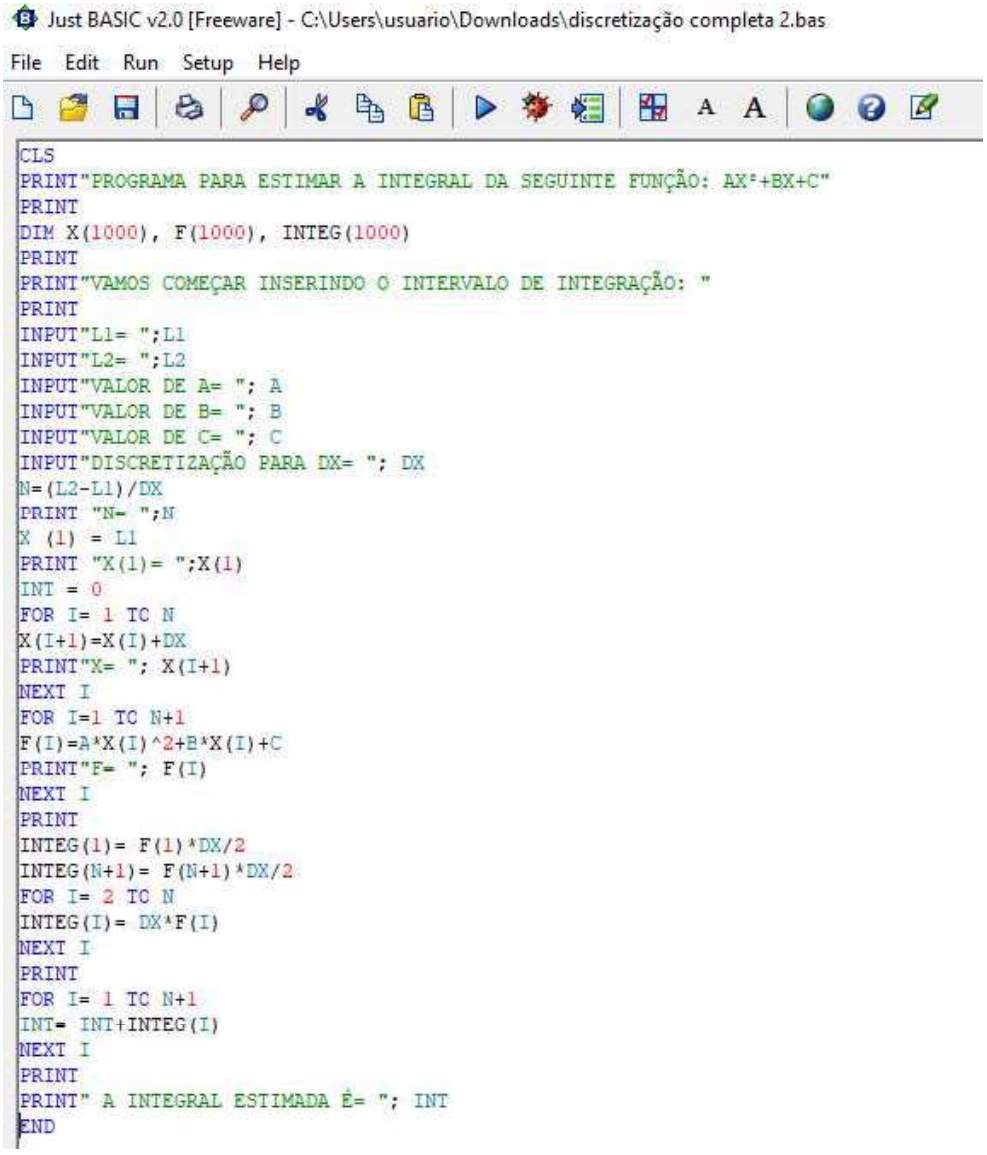
Tomar por base E_{ADM}

$$ERRO = ABS\big(INTEG(K+1) - INTEG(K)\big)$$

Exercícios complementares:

a) Resolver as integrações numéricas de X^2 e A*X^2 +B*X+C de forma manuscrita e em Excel. Adote o intervalo de [0,5] e discretização de 0,5.
b) Implementar as rotinas computacionais 1, 2 e 3 em plataforma JustBasic;
c) Rodar as rotinas computacionais 1,2 e 3 implementadas.

A rotina abaixo foi elaborada para cálculo da integral numérica de um polinômio de grau 2.




```
CLS
PRINT"PROGRAMA PARA ESTIMAR A INTEGRAL DA SEGUINTE FUNÇÃO: AX²+BX+C"
PRINT
DIM X(1000), F(1000), INTEG(1000)
PRINT
PRINT"VAMOS COMEÇAR INSERINDO O INTERVALO DE INTEGRAÇÃO: "
PRINT
INPUT"L1= ";L1
INPUT"L2= ";L2
INPUT"VALOR DE A= "; A
INPUT"VALOR DE B= "; B
INPUT"VALOR DE C= "; C
INPUT"DISCRETIZAÇÃO PARA DX= "; DX
N=(L2-L1)/DX
PRINT "N= ";N
X (1) = L1
PRINT "X(1)= ";X(1)
INT = 0
FOR I= 1 TO N
X(I+1)=X(I)+DX
PRINT"X= "; X(I+1)
NEXT I
FOR I=1 TO N+1
F(I)=A*X(I)^2+B*X(I)+C
PRINT"F= "; F(I)
NEXT I
PRINT
INTEG(1)= F(1)*DX/2
INTEG(N+1)= F(N+1)*DX/2
FOR I= 2 TO N
INTEG(I)= DX*F(I)
NEXT I
PRINT
FOR I= 1 TO N+1
INT= INT+INTEG(I)
NEXT I
PRINT
PRINT" A INTEGRAL ESTIMADA É= "; INT
END
```

A execução manual de parte deste programa, até o cálculo de F(I) está apresentada a seguir:

			1	2	3	4	.....	1000
		X	X(1)	X(2)	X(3)	X(4)	.....	X(1000)
		F						
		INTEG						
	.....							
	L1=	0						
	L2=	2						
	A=	0						
	B=	1						
	C=	0						
	DX=	0.5						
	N=(2-0)/0.5=	4						
	X(1)=	0						
	"X(1)=0"							
	INT=0							
	I=1							
	X(2)=	X(1)+DX=	0.5					
	I=2							
	X(3)=	X(2)+DX=	1.0					
	I=3							
	X(4)=	X(3)+DX=	1.5					
	I=4							
	X(5)=	X(4)+DX=	2.0					
	I=1	F(1)=	0					
	I=2	F(2)=	0.5					
	I=3	F(3)=	1.0					
	I=4	F(4)=	1.5					
	I=5	F(5)=	2.0					
		X	0	0.5	1.0	1.5	2.0	
		F	0	0.5	1.0	1.5	2.0	
			1	2	3	4	5	
	.....	(CONTINUAR...)						

É importante observar a construção da discretização, sendo armazenada no vetor X(I), onde I é a posição do registro numérico no vetor. Os laços FOR-NEXT possibilitam o preenchimento gradual do vetor X com os elementos da discretização, além do cálculo e armazenamento de F(I). Para o exemplo em questão, temos:

I	*1*	*2*	*3*	*4*	*5*
X(I)	**0**	**0,5**	**1**	**1,5**	**2**

I	*1*	*2*	*3*	*4*	*5*
F(I)	**0**	**0,5**	**1**	**1,5**	**2**

que são iguais, uma vez que a função adotada foi F(x) = x.

A escolha da discretização Δx é decisiva para a precisão da estimativa. Assim, à medida que se refina a discretização, dividindo-se o intervalo em um maior número de subintervalos, obtém-se estimativas cada vez mais precisas. Para otimização da escolha da discretização, é necessário assumir um erro admissível para o processo, que pode ser estimado a partir do valor absoluto da diferença entre estimativas sucessivas, para diferentes discretizações, como visto anteriormente.

Um possível critério é adotar uma progressão geométrica de tamanho de subintervalos, com fator 1/2 (por exemplo), e um processo iterativo para estimativa sucessiva das integrais, como já mostrado acima, e repetido aqui.

$$\Delta x^{(k+1)} = \frac{\Delta x^{(k)}}{2}$$

Elaborou-se a rotina abaixo, construída a partir da rotina anterior, adicionando-se um laço externo a todo o processo (FOR K=1 TO 50; .....;NEXT K) para controlar o cálculo da discretização desejada. Além desta modificação, foi necessário armazenar as diversas estimativas de integral, no vetor INTEGR(50), com 50 posições de armazenamento, correspondendo ao número de iterações.

Tomando por base agora um erro admissível, especificado pelo usuário (EADM), inseriu-se também uma sequência de dois testes lógicos IF-THEN_ENDIF, para testar a convergência do processo iterativo, quando o erro calculado se tornar menor ou igual a EADM. Percebam que este teste só deve ser realizado a partir da segunda iteração (K=2), para que se disponha de estimativas de integrais sucessivas INTEGR(K) e INTEGR(K-1), onde K é a variável inteira que controla o processo iterativo. A rotina está apresentada abaixo, incluindo um desvio através de GOTO, para interromper o processo iterativo quando a convergência for obtida:

```
DIM X(1000), F(1000), INTEG(1000),INTEGR(50)

PRINT
PRINT"VAMOS COMEÇAR INSERINDO O INTERVALO DE INTEGRAÇÃO: "
PRINT
INPUT"L1= ";L1
INPUT"L2= ";L2
INPUT"VALOR DE A= "; A
INPUT"VALOR DE B= "; B
INPUT"VALOR DE C= "; C
INPUT"DISCRETIZAÇÃO PARA DX INICIAL= "; DX
EADM=0.001
FOR K=1 TO 50
DX=DX/K
N=(L2-L1)/DX
PRINT "N= ";N
X (1) = L1
PRINT "X(1)= ";X(1)
INTEGR (K) = 0
FOR I= 1 TO N
X(I+1)=X(I)+DX
PRINT"X= "; X(I+1)
NEXT I
FOR I=1 TO N+1
F(I)=A*X(I)^2+B*X(I)+C
PRINT"F= "; F(I)
NEXT I
PRINT
INTEG(1)= F(1)*DX/2
INTEG(N+1)= F(N+1)*DX/2
FOR I= 2 TO N
INTEG(I)= DX*F(I)
NEXT I
PRINT
FOR I= 1 TO N+1
INTEGR(K)= INTEGR(K)+INTEG(I)
NEXT I
PRINT
PRINT "ITERAÇÃO K= "; K
PRINT" A INTEGRAL ESTIMADA É= "; INTEGR(K)
IF K>1 THEN
   ERRO=ABS(INTEGR(K)-INTEGR(K-1))
   PRINT "ERRO= ";ERRO
   IF ERRO<=EADM THEN
   PRINT "O PROCESSO CONVERGIU EM K= ";K;"ITERAÇÕES"
   PRINT "A INTEGRAL É:";INTEGR(K)
   GOTO 100
   END IF
END IF
NEXT K
100 END
```

A seguir é apresentada uma versão de algoritmo em linguagem Python:

```
from time import sleep

valores_x = []
valores_fx = []
valores_integral = []
erro = 999

intervalo = float(input("Insira o delta X inicial: "))
limite_inferior = float(input("Indique o limite inferior da integral: "))
limite_superior = float(input("Indique o limite superior da integral: "))
eadm = float(input('Informe o erro admissivel: '))

funcao = input("Escreva a funcao: ")

i=limite_inferior
# intervalo = (limite_superior-limite_inferior)/dx
while i <= limite_superior:
    x=i
    valores_x.append(x)
    fx = eval(funcao)
    valores_fx.append(fx)
    i+=intervalo

print('\nValores da funcao:')
for j in range(len(valores_fx)):
    print(f'f({valores_x[j]:8.4f}) = {valores_fx[j]:8.4f}')
    # sleep(0.5)

somatorio = 0
for k in range(len(valores_fx)-1):
    if k == 0:
        continue
    somatorio += valores_fx[k]

integral = intervalo * ((valores_fx[0]/2) + somatorio + (valores_fx[-1]/2))
print(f'Integral: {integral}')
valores_integral.append(integral)

while erro > eadm:
    valores_x = []
    valores_fx = []
    intervalo = intervalo/2
    i = limite_inferior
    # intervalo = (limite_superior - limite_inferior) / dx
    while i <= limite_superior:
        x = i
        valores_x.append(x)
        fx = eval(funcao)
        valores_fx.append(fx)
        i += intervalo

    print('\nValores da funcao:')
    for j in range(len(valores_fx)):
        print(f'f({valores_x[j]:8.4f}) = {valores_fx[j]:8.4f}')
        # sleep(0.1)

    somatorio = 0
    for k in range(len(valores_fx) - 1):
        if k == 0:
            continue
        somatorio += valores_fx[k]

    integral = intervalo * ((valores_fx[0] / 2) + somatorio + (valores_fx[-1] / 2))
    print(f'Integral: {integral}')

    valores_integral.append(integral)

    erro = abs(valores_integral[-1] - valores_integral[-2])

print(f'\nValor da integral calculada numericamente\nI = {valores_integral[-1]:5.4f}\ndx = {intervalo}\nErro =
{erro:.4f} < {eadm}')
```

Exercícios numéricos

1. Determinar numericamente a integral para as seguintes equações:
   a. $\int_0^4 x^2\,dx$
   b. $\int_0^{10} 2x^2 - 4x + 12\,dx$
   c. $\int_0^2 (x^2 + 2x + 1)dx$

Exercícios computacionais

1. Implementar as rotinas para determinação de integração numérica e realizar a integração das funções abaixo, nos intervalos especificados. Construir o gráfico do processo de convergência, relacionando a iteração K com o ERRO associado.
   a. $I = \int_0^2 (x^2 + 2x + 1)dx$
   b. $I = \int_0^4 (3x^2 - 3x + 3)dx$
2. Atualizar a rotina anterior para integrar polinômios de terceira ordem. Em seguida, realizar a integração da função abaixo, no intervalo especificado. Construir o gráfico do processo de convergência, relacionando a iteração K com o ERRO associado.
   a. $I = \int_0^3 (x^3 + 2x + 1)dx$
3. Atualizar a rotina denominada "discretização completa 2.bas" para realizar a integração numérica a partir de valores de F(I) fornecidos pelo usuário, a partir de medições experimentais obtidas com base em uma discretização Δx de um intervalo [a;b] especificado. Em seguida, estimar a integral numérica oriunda das seguintes medições:

I	*1*	*2*	*3*	*4*	*5*
X(I)	0.0	2.0	4.0	6.0	8.0
F(I)	0.0	3.2	7.6	5.0	0.0

4. Resolver as integrações numéricas de X^2 e A*X^2 +B*X+C em Excel. Adote o intervalo de [0,5] e discretização de 0,5.

Derivação numérica

A aproximação numérica para uma derivada de primeira ordem pode ser expressa com base na expressão estabelecida para o cálculo diferencial. Seja f(x) uma função contínua. Pode-se escrever que:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

Considerando que

$x_1 = x_0 + \Delta x$

Então

$$f'(x_0) \cong \frac{f(x_1) - f(x_0)}{\Delta x}$$

Seja agora uma discretização Δx de x no intervalo [a;b], de modo que

$x_{i+1} = x_i + \Delta x$

Então,

$$f'(x_i) \cong \frac{f(x_{i+1}) - f(x_i)}{\Delta x}$$

que pode ser chamada de aproximação "progressiva" para a derivada primeira de $f(x_i)$. No caso de uma aproximação "regressiva", teremos então:

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{\Delta x}$$

Para uma derivada de **segunda ordem**, lembrando que esta representa a variação da derivada primeira, podemos combinar as expressões anteriores, estimando a variação de estimativas progressiva e regressiva em torno no ponto x_i.

Teremos, então:

$$f''(x_i) \cong \frac{\frac{f(x_{i+1}) - f(x_i)}{\Delta x} - \frac{f(x_i) - f(x_{i-1})}{\Delta x}}{\Delta x}$$

E, daí:

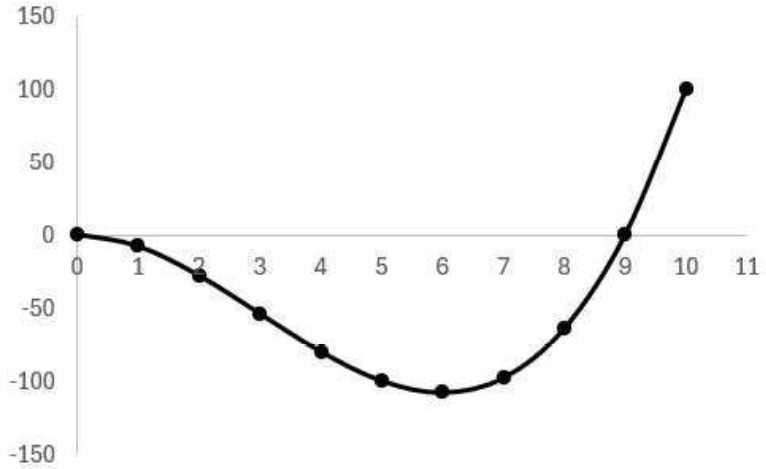
$$f''(x_i) \cong \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\Delta x^2}$$

Diversas podem ser as condições de contorno a serem utilizadas nas derivações numéricas, e na solução de equações diferenciais. Neste documento, focaremos apenas nas condições de contorno de "valores conhecidos" para a função desejada.

Exercício resolvido

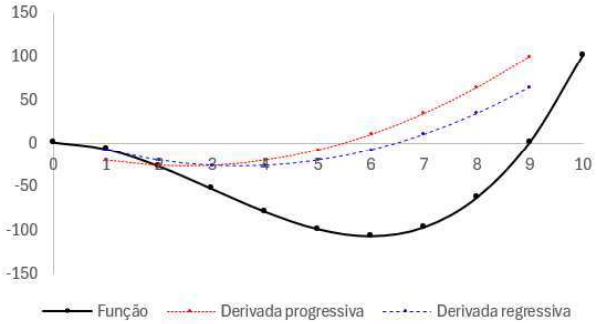
Com base em um esquema progressivo, estime as derivadas nos pontos internos de x no intervalo [0;10], com discretização unitária. Os valores da função f(x) estão apresentados na tabela abaixo.

x	f(x)
0	1
1	-7
2	-27
3	-53
4	-79
5	-99
6	-107
7	-97
8	-63
9	1
10	101



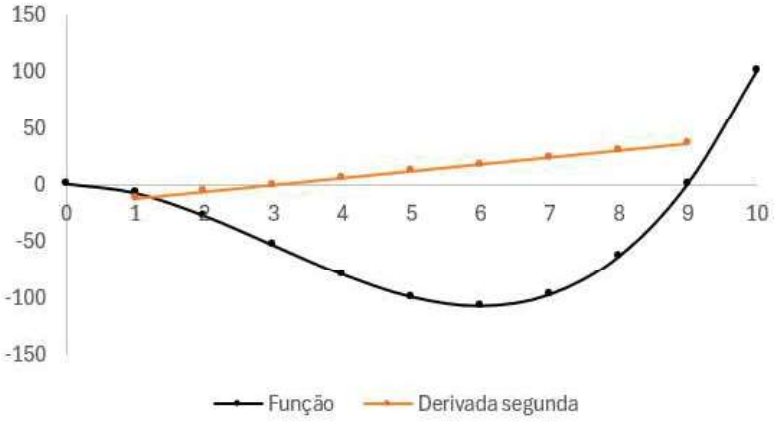
Aplicando-se os conceitos de derivada primeira progressiva e regressiva temos que:

x	f(x)	f'(x) - Progressivo	f'(x) - Regressivo
0	1		
1	-7	-20	-8
2	-27	-26	-20
3	-53	-26	-26
4	-79	-20	-26
5	-99	-8	-20
6	-107	10	-8
7	-97	34	10
8	-63	64	34
9	1	100	64
10	101		

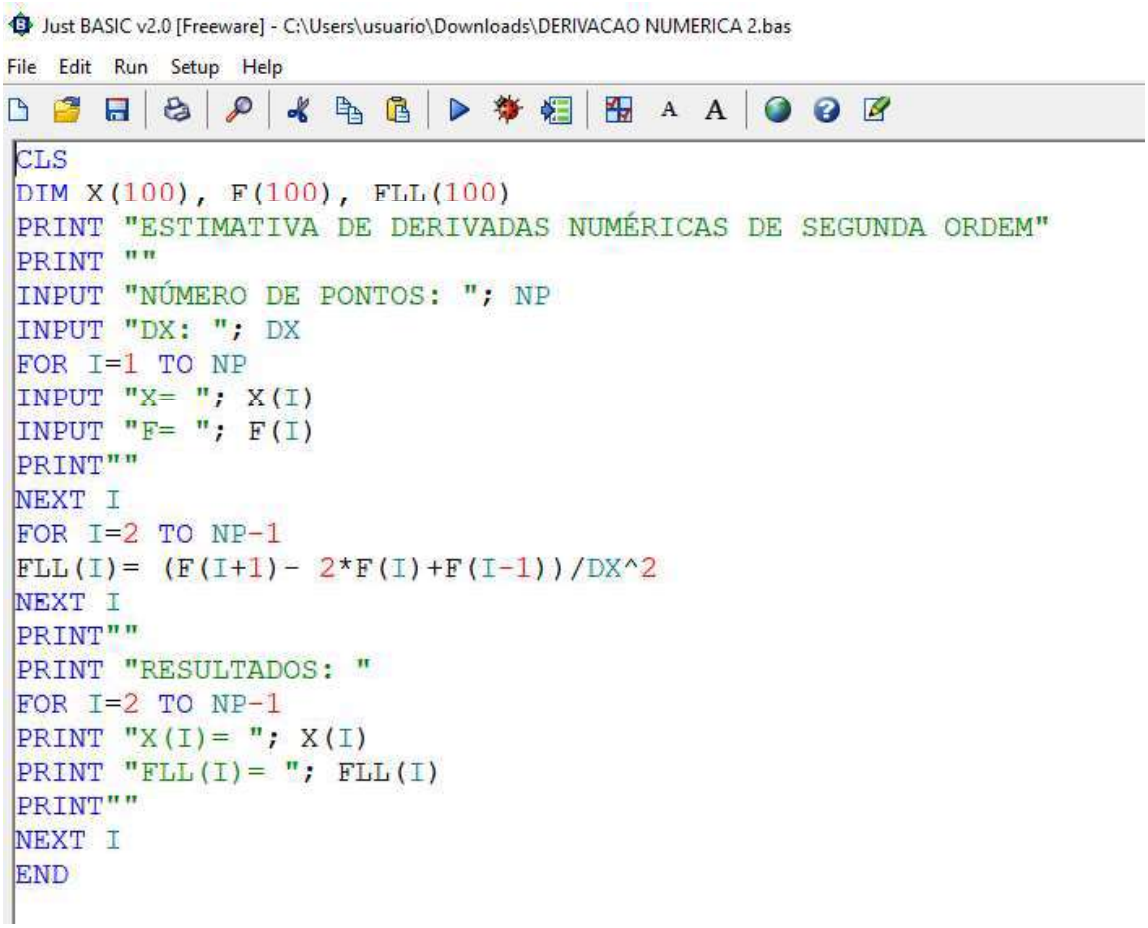


Com a determinação da derivada segunda para a função, temos:

x	f(x)	f''(x)
0	1	
1	-7	-12
2	-27	-6
3	-53	0
4	-79	6
5	-99	12
6	-107	18
7	-97	24
8	-63	30
9	1	36
10	101	



O Programa "Derivacao numerica 2.bas" está listado abaixo, para o cálculo da derivada numérica de segunda ordem:




```
CLS
DIM X(100), F(100), FLL(100)
PRINT "ESTIMATIVA DE DERIVADAS NUMÉRICAS DE SEGUNDA ORDEM"
PRINT ""
INPUT "NÚMERO DE PONTOS: "; NP
INPUT "DX: "; DX
FOR I=1 TO NP
INPUT "X= "; X(I)
INPUT "F= "; F(I)
PRINT""
NEXT I
FOR I=2 TO NP-1
FLL(I)= (F(I+1)- 2*F(I)+F(I-1))/DX^2
NEXT I
PRINT""
PRINT "RESULTADOS: "
FOR I=2 TO NP-1
PRINT "X(I)= "; X(I)
PRINT "FLL(I)= "; FLL(I)
PRINT""
NEXT I
END
```

Problema de fluxo e distribuição de cargas hidráulicas em aquiferos

Dentre as principais aplicações das derivadas numéricas está a solução de equações diferenciais, totais ou parciais.

Tais equações têm larga aplicação em Engenharia Agrícola e Ambiental, tais como a simulação de escoamentos em meios porosos, simulações de níveis freáticos em sistemas de drenagem subterrânea, transporte de poluentes, fluxo de calor, dentre outras.

Um exemplo de Equação Diferencial para escoamento em meios porosos é a Equação de Poisson, expressa em uma dimensão como:

$$\frac{d^2h}{dx^2} = -\frac{R}{T}$$

sendo h a carga hidráulica em um meio poroso (igual à aproximadamente à altura de pressão da água), R é taxa de recarga ou extração (vazão por unidade de área), e

T - Transmissividade do aquífero = kb (k - Condutividade hidráulica do meio poroso; b - espessura do aquífero)

As equações diferenciais requerem a adoção de condições de contorno para sua solução definida. No caso da Equação de Poisson, podem ser de carga h conhecida ou de fluxo especificado.

A solução numérica da Equação de Poisson em uma dimensão pode ser escrita como:

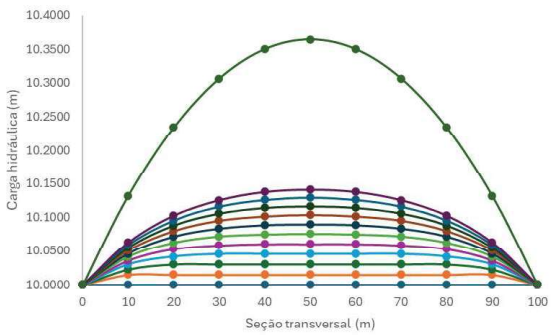
$$h_i = \frac{1}{2}\left(h_{i+1} + h_{i-1} + \frac{R\Delta x^2}{T}\right)$$

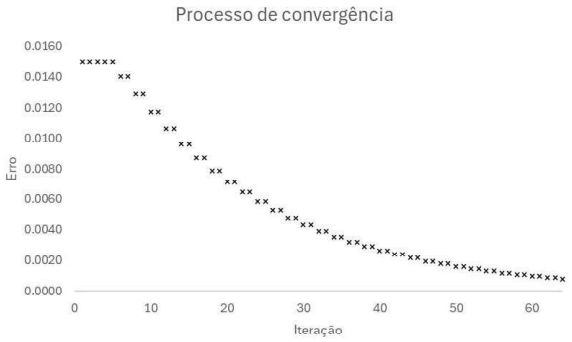
Aplicação: Estimar as cargas hidráulicas h ao longo de uma seção transversal de um aquífero poroso, através da solução numérica da Equação de Poisson. Considere que as cargas hidráulicas são conhecidas nas extremidades da seção de estudo, e que o aquífero sofre recarga de 1 mm/dia, proveniente de excessos de irrigação na superfície do solo. Assuma que T= 3 m²/dia, e que as cargas nas extremidades são iguais a 10 m. A seção em estudo tem 100 m de extensão, e considere uma discretização de 10 m.

A tela abaixo mostra a solução em Excel, que requer um processo iterativo. Foi adotado erro admissível de 0,001. A solução aproximada aceitável para a distribuição de cargas h foi obtida após 64 iterações. A recarga provocou uma ascensão das cargas hidráulicas, que atinge 10,4 m no meio da seção.

Inicialização	
R	1 mm/dia
T	3 m²/s
R/T	0.0003 m/dia
Δx	10 m
h	10 m

k=		0	1		2		3		4		5		6		7		8	
	Pos	h	h	erro	h	erro	h	erro	h	erro	h	erro	h	erro	h	erro	h	erro
1	0	10.0000	10.0000		10.0000		10.0000		10.0000		10.0000		10.0000		10.0000		10.0000	
2	10	10.0000	10.0150	0.0150	10.0225	0.0075	10.0300	0.0075	10.0356	0.0056	10.0413	0.0056	10.0453	0.0047	10.0506	0.0047	10.0547	0.0041
3	20	10.0000	10.0150	0.0150	10.0300	0.0150	10.0413	0.0113	10.0525	0.0113	10.0619	0.0094	10.0713	0.0094	10.0795	0.0082	10.0877	0.0082
4	30	10.0000	10.0150	0.0150	10.0300	0.0150	10.0450	0.0150	10.0581	0.0131	10.0713	0.0131	10.0830	0.0117	10.0947	0.0117	10.1052	0.0105
5	40	10.0000	10.0150	0.0150	10.0300	0.0150	10.0450	0.0150	10.0600	0.0150	10.0741	0.0141	10.0881	0.0141	10.1010	0.0129	10.1139	0.0129
6	50	10.0000	10.0150	0.0150	10.0300	0.0150	10.0450	0.0150	10.0600	0.0150	10.0750	0.0150	10.0891	0.0141	10.1031	0.0141	10.1160	0.0129
7	60	10.0000	10.0150	0.0150	10.0300	0.0150	10.0450	0.0150	10.0600	0.0150	10.0741	0.0141	10.0881	0.0141	10.1010	0.0129	10.1139	0.0129
8	70	10.0000	10.0150	0.0150	10.0300	0.0150	10.0450	0.0150	10.0581	0.0131	10.0713	0.0131	10.0830	0.0117	10.0947	0.0117	10.1052	0.0105
9	80	10.0000	10.0150	0.0150	10.0300	0.0150	10.0413	0.0113	10.0525	0.0113	10.0619	0.0094	10.0713	0.0094	10.0795	0.0082	10.0877	0.0082
10	90	10.0000	10.0150	0.0150	10.0225	0.0075	10.0300	0.0075	10.0356	0.0056	10.0413	0.0056	10.0459	0.0047	10.0506	0.0047	10.0547	0.0041
11	100	10.0000	10.0000		10.0000		10.0000		10.0000		10.0000		10.0000		10.0000		10.0000	
			Emáx	0.0150		0.0150		0.0150		0.0150		0.0150		0.0141		0.0141		0.0129





Em duas dimensões, a Equação de Poisson pode ser escrita como:

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} = -\frac{R}{T}$$

R - Taxa de recarga ou extração

T - Transmissividade do aquífero = kb (k - Condutividade hidráulica; b - espessura do aquífero)

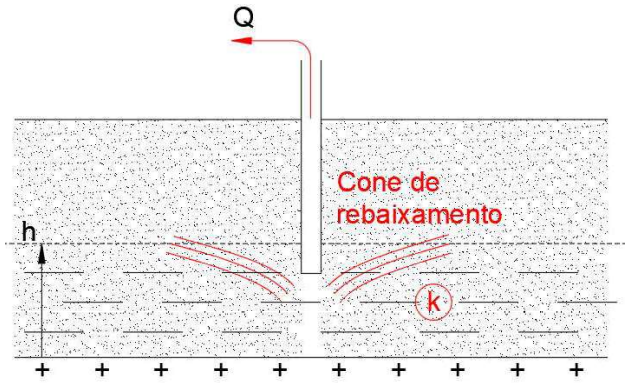
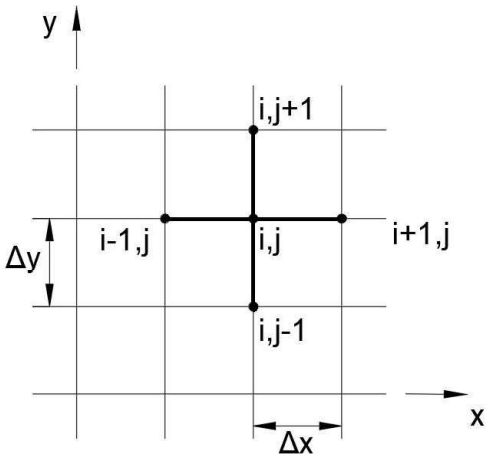


Figura 10: Discretização bidimensional.



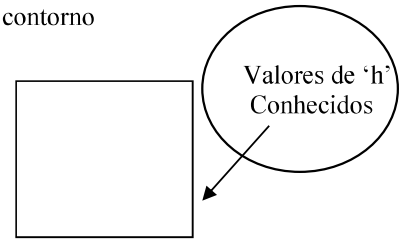
$$\frac{(h_{i+1,j}-2h_{i,j}+h_{i-1,j})}{\Delta x^2}+\frac{(h_{i,j+1}-2h_{i,j}+h_{i,j-1})}{\Delta y^2}=-\frac{R}{T}$$

$$\Delta x=\Delta y$$

$$h_{i+1,j}+h_{i-1,j}+h_{i,j+1}+h_{i,j-1}-4h_{i,j}=-\frac{R\Delta x^2}{T}$$

$$h_{i,j}=\frac{1}{4}\left(h_{i+1,j}+h_{i-1,j}+h_{i,j+1}+h_{i,j-1}+\frac{R\Delta x^2}{T}\right)$$

Condição de contorno



A seguir é apresentado código Python para cálculo de uma derivada de segunda ordem para dados armazenados em arquivo com extensão .txt:

```
linhas, x, fx, dx2 = [], [], [], []

with open('derivacao.txt', 'r') as dados:
    for linha in dados:
        x.append(float(linha[0:2]))
        fx.append(float(linha[3:-1]))

dx = x[0] - x[1]
i=1
while i < 10:
    derivada = (fx[i+1] - 2*fx[i] + fx[i-1]) / (dx ** 2)
    dx2.append(derivada)
    i+=1

print(dx2)
```

Exercício resolvido

Equação do Escoamento em meios porosos com Hipótese de Dupuit

Neste caso a espessura saturada varia com a altura do freático.
Assim, deve-se considerar a condutividade hidráulica K, ao invés da transmissividade. A Equação diferencial resulta em:

$$\frac{K}{2}\frac{d^2h^2}{dx^2} = -R$$

Fazendo-se uma substituição de variáveis $v=h^2$, teremos:

$$\frac{K}{2}\frac{d^2v}{dx^2} = -R$$

A aproximação por Diferenças Finitas fica:

$$v_i = \frac{1}{2}(v_{i+1} + v_{i-1} + \frac{2R\Delta x^2}{K})$$

onde

$v_i=h_i$ [2]

Aplicação: Utilizando a Hipótese de Dupuit, avaliar a elevação de um lençol freático devido a uma recarga de 2 mm/ dia, considerando a condutividade hidráulica igual a 2 m/dia. Considere espaçamento entre os drenos de 10 m, e uma discretização de 1 m.

	A	B	C
2	K=	2	m/dia
3	R=	2	mm/dia
4	Dx=	1	
5	F=	0,002	

	k	v0	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
17												
18	0	0	0	0	0	0	0	0	0	0	0	0
19	1	0	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0
20	2	0	0,0015	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,0015	0
21	3	0	0,002	0,00275	0,003	0,003	0,003	0,003	0,003	0,00275	0,002	0
22	4	0	0,00238	0,0035	0,00388	0,004	0,004	0,004	0,00388	0,0035	0,00238	0
23	5	0	0,00275	0,00413	0,00475	0,00494	0,005	0,00494	0,00475	0,00413	0,00275	0
24	6	0	0,00306	0,00475	0,00553	0,00588	0,00594	0,00588	0,00553	0,00475	0,00306	0
25	7	0	0,00338	0,0053	0,00631	0,00673	0,00688	0,00673	0,00631	0,0053	0,00338	0
26	8	0	0,00365	0,00584	0,00702	0,00759	0,00773	0,00759	0,00702	0,00584	0,00365	0
27	9	0	0,00392	0,00633	0,00772	0,00838	0,00859	0,00838	0,00772	0,00633	0,00392	0
28	10	0	0,00417	0,00682	0,00835	0,00916	0,00938	0,00916	0,00835	0,00682	0,00417	0
29	11	0	0,00441	0,00726	0,00899	0,00986	0,01016	0,00986	0,00899	0,00726	0,00441	0
30	12	0	0,00463	0,0077	0,00956	0,01057	0,01086	0,01057	0,00956	0,0077	0,00463	0

51	33	0	0,00748	0,0131	0,01702	0,01931	0,02008	0,01931	0,01702	0,0131	0,00748	0
52	34	0	0,00755	0,01325	0,01721	0,01955	0,02031	0,01955	0,01721	0,01325	0,00755	0
53	35	0	0,00762	0,01338	0,0174	0,01976	0,02055	0,01976	0,0174	0,01338	0,00762	0
54	36	0	0,00769	0,01351	0,01757	0,01997	0,02076	0,01997	0,01757	0,01351	0,00769	0
55												

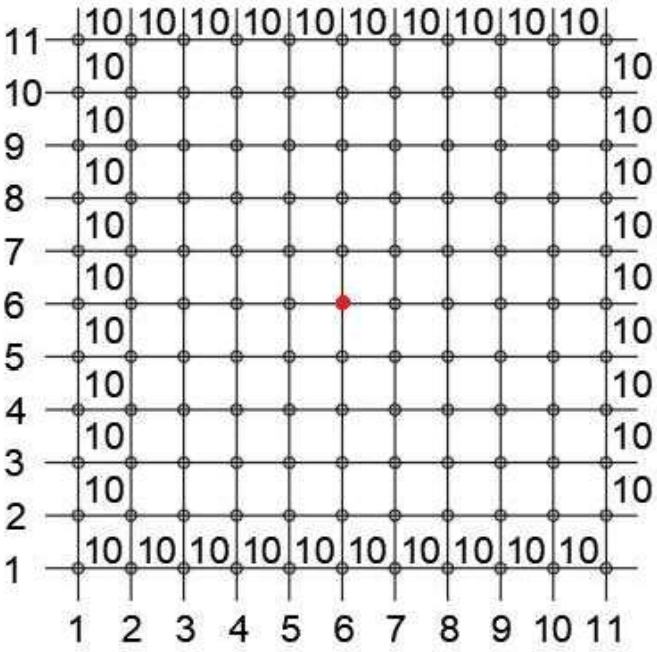
Para controle do processo iterativo, calculou-se o erro máximo em cada iteração a partir do valor absoluto da variação em v_i.

Exercícios numéricos

1. Considere um aquífero com área retangular de 100 m x 100 m. Avaliar o impacto da instalação de um poço no centro do aquífero para bombear 3 m³/dia. Considere uma discretização de 10 m (Δx) e adote h= 10 m no contorno.

   Dados adicionais: $T = 2\frac{m^2}{dia}$.

   Considere uma inicialização de 10 m em todos os nós. Adotar $\varepsilon_{adm} = 10^{-3}$.



Exercícios computacionais

1. Atualizar o código Python aplicado à derivação numérica para torná-lo iterativo. A cada iteração, verificar o erro e utilizar como critério de parada, um erro admissível informado pelo usuário.
2. Considere um aquífero com área retangular de 100 m x 100 m. Avaliar o impacto da recarga R distribuída no aquífero. Considere uma discretização de 10 m (Δx) e adote h= 10 m no contorno.

   Dados adicionais: $T = 2\frac{m^2}{dia}$.

   Considere uma inicialização de 10 m em todos os nós. Adotar $\varepsilon_{adm} = 10^{-3}$. Adotar recarga R de 2,5 mm dia^{-1}

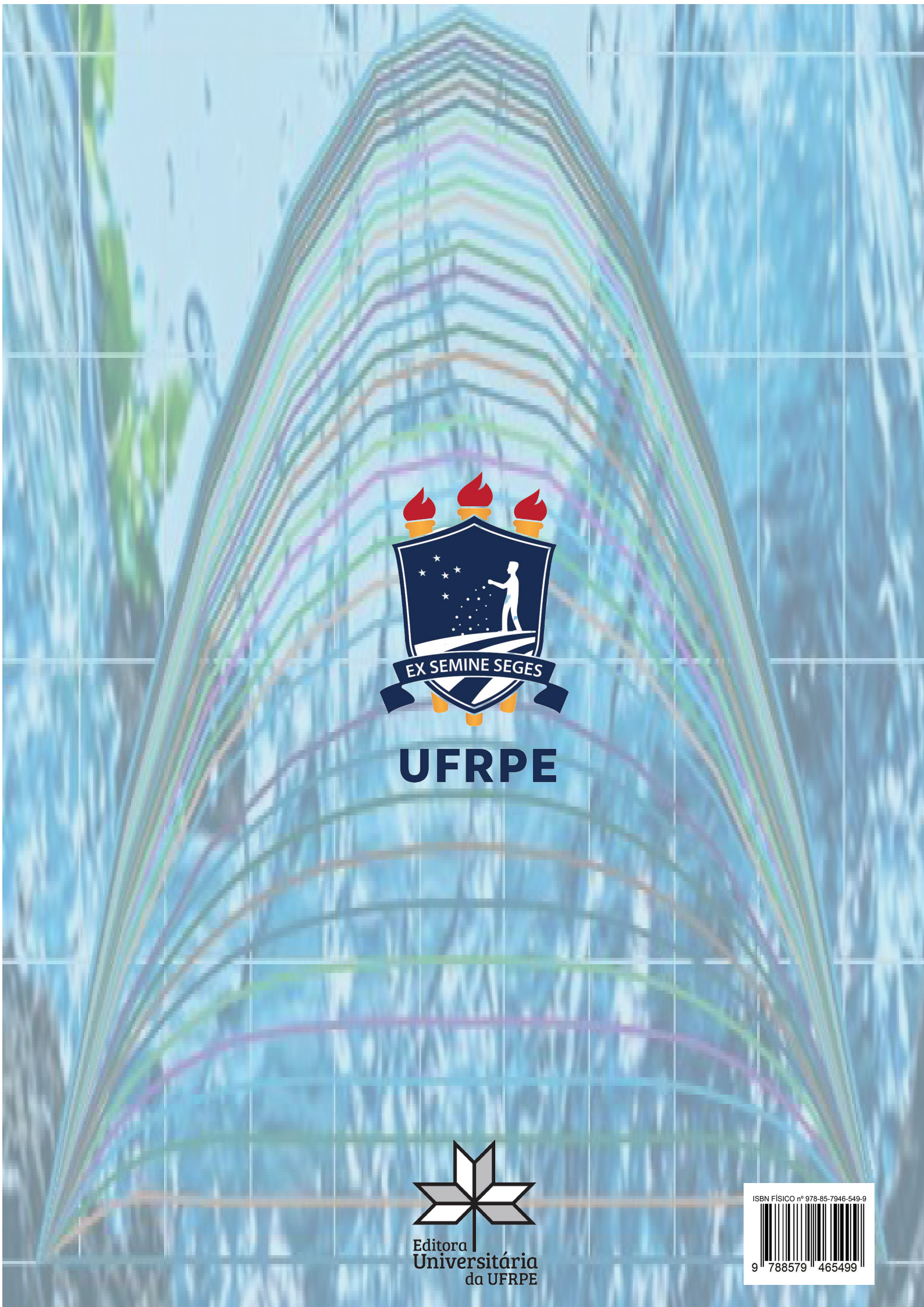
Leitura Complementar

Montenegro. S.M.G.L.; Montenegro, A.A.A. Aproveitamento Sustentável de Aquíferos Aluviais no Semiárido, In: Água Subterrânea: Aquíferos Costeiros e Aluviões, Vulnerabilidade e Aproveitamento, Cabral et al. (org.), 2004, 448p.

Wang, H.F.; Anderson, M.P. Introduction to Groundwater Modeling: Finite Difference and Finite Element Methods, W.H. Freeman and Company, São Francisco, 1982, 237p.

Todos os códigos Python apresentados estão disponíveis na plataforma GitHub

https://github.com/Fernandez-E/MatComp2024



ISBN FÍSICO nº 978-85-7946-549-9
9 788579 465499