

Algebra Linear Computacional COC473 - Lista 2

Bruno Dantas de Paiva
DRE: 118048097

September 24, 2020

1 Questão 1

1.1 Método de Potência

```
def power_method(matrix):
    number_of_rows = len(matrix)
    eigenvector = [1.0 for _ in range(number_of_rows)]

    y_vector = Matrix_Utils.multiply_matrix_vector(matrix, eigenvector)
    first_element = 1
    eigenvalue = y_vector[0]
    steps = 1

    for i in range(number_of_rows):
        y_vector[i] = y_vector[i]/eigenvalue
        eigenvector = y_vector

    residue = math.fabs(eigenvalue - first_element)/eigenvalue
    tol = 10**-3

    while (residue>=tol):
        first_element = eigenvalue
        y_vector = Matrix_Utils.multiply_matrix_vector(matrix, eigenvector)
        eigenvalue = y_vector[0]

        for i in range(number_of_rows):
            y_vector[i] = y_vector[i]/eigenvalue
            eigenvector = y_vector

        residue = math.fabs(eigenvalue-first_element)/eigenvalue
        steps+=1

    print("Eigenvalue:" + str(eigenvalue))
    print("Eigenvector:" + str(eigenvector))
    print("Steps:" + str(steps))
```

Nota-se que o power method utiliza uma função que faz o produto de um vetor com uma matriz, onde o código está logo abaixo.

```
def multiply_matrix_vector(matrix_a, vector):
    number_of_columns_of_a      = len(matrix_a)
    number_of_columns_of_vector = range(len(vector))
    result                      = [0.0 for _ in range(number_of_columns_of_a)]
    
    for j in range(number_of_columns_of_a):
        summation = 0

        for i in number_of_columns_of_vector:
            summation += matrix_a[j][i]*vector[i]

        result[j] = summation

    return result
```

2 Questão 2

```
def jacobi_method(matrix):
    if(not Matrix_Utils.check_simetry(matrix)):
        return -1

    number_of_rows = len(matrix)
    identity_matrix = [[float(i==j) for j in range(number_of_rows)] for i in range(number_of_rows)
    tol = 10**(-3)
    biggest_element = Matrix_Utils.get_biggest_element(matrix)
    
    while (math.fabs(matrix[biggest_element[0]][biggest_element[1]]) > tol):
        p_matrix = Matrix_Utils.calculate_p_matrix(matrix, biggest_element)
        p_matrix_transposed = Matrix_Utils.get_transposed_matrix(p_matrix)
        matrix = Matrix_Utils.multiply_matrixes(p_matrix_transposed, Matrix_Utils.get_transposed_matrix(identity_matrix))
        identity_matrix = Matrix_Utils.multiply_matrixes(identity_matrix, p_matrix)
        biggest_element = Matrix_Utils.get_biggest_element(matrix)
        
        result = []
        for i in range(number_of_rows):
            result.append((i+1, matrix[i][i], identity_matrix[i]))
            print(str(i+1) + "Autovalor:" + str(matrix[i][i]) + ", Autovetor:" + str(identity_matrix[i]))
    
    return (result)
```

Nota-se que o método de Jacobi não iterativo utiliza algumas funções externas, onde o código está logo abaixo. Note que a função check simetry é a mesma utilizada na lista anterior, logo seu código não estará incluso.

```

def get_biggest_element(matrix_a):
    number_of_rows = len(matrix_a)
    biggest_value = -math.inf
    for i in range(number_of_rows):
        for j in range(number_of_rows):
            if (i != j and math.fabs(matrix_a[i][j]) > biggest_value):
                biggest_value = math.fabs(matrix_a[i][j])
                index = (i, j)
    return index

def multiply_matrixes(matrix_a, matrix_b):
    number_of_rows = len(matrix_a)
    result = [[0.0 for _ in range(number_of_rows)] for _ in range(number_of_rows)]
    for i in range(len(matrix_a)):
        for j in range(len(matrix_b[0])):
            for k in range(len(matrix_b)):
                result[i][j] += matrix_a[i][k] * matrix_b[k][j]
    return result

def get_transposed_matrix(matrix):
    answer = [[0.0]*len(matrix) for i in range(len(matrix))]

    for i in range(len(matrix)):
        for j in range(len(matrix)):
            answer[j][i] = matrix[i][j]

    return answer

def calculate_p_matrix(matrix, indexes):
    number_of_rows = len(matrix)
    phi_value = 0

    p_matrix = [[0.0 for _ in range(number_of_rows)] for _ in range(number_of_rows)]
    for i in range(number_of_rows):
        p_matrix[i][i] = 1.0

    phi_value = phi(matrix, indexes)

    p_matrix[indexes[0]][indexes[0]] = math.cos(phi_value)
    p_matrix[indexes[1]][indexes[1]] = math.cos(phi_value)
    p_matrix[indexes[0]][indexes[1]] = -math.sin(phi_value)
    p_matrix[indexes[1]][indexes[0]] = math.sin(phi_value)
    return p_matrix

def phi(matrix, indexes):
    denominator = (matrix[indexes[0]][indexes[0]] - matrix[indexes[1]][indexes[1]])
    if (matrix[indexes[0]][indexes[0]] == matrix[indexes[1]][indexes[1]]):
        return math.pi/4
    else:
        return math.atan(2*matrix[indexes[0]][indexes[1]]/denominator)/2

```

3 Questão 3

- 3.1 Usando o polinômio determinístico, calcule os autovalores e autovetores exatos de A

$$\begin{aligned}
 A &= \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} & D(\lambda) = 0 = A - \lambda I \Rightarrow \begin{bmatrix} 3-\lambda & 2 & 0 \\ 2 & 3-\lambda & -1 \\ 0 & -1 & 3-\lambda \end{bmatrix} = 0. \\
 D(\lambda) &= (3-\lambda)((3-\lambda)^2 - 9) = 0 & \lambda = [3, 3+\sqrt{5}, 3-\sqrt{5}] \\
 \text{Autovetores:} \\
 \lambda = 3: & \begin{bmatrix} 0 & 2 & 0 & | & 0 \\ 2 & 0 & -1 & | & 0 \\ 0 & -1 & 0 & | & 0 \end{bmatrix} \xrightarrow[L_2 \leftrightarrow L_3]{L_1 \leftrightarrow L_2} \begin{bmatrix} 2 & 0 & -1 & | & 0 \\ 0 & 2 & 0 & | & 0 \\ 0 & 0 & 0 & | & 0 \end{bmatrix} \therefore x_2 = 0, x_3 = x_1 \\
 X &= x_1 \begin{bmatrix} 0.5 & 0 & 1 \end{bmatrix} \\
 \lambda = 3 + \sqrt{5}: & \begin{bmatrix} -\sqrt{5} & 2 & 0 & | & 0 \\ 2 & -\sqrt{5} & 1 & | & 0 \\ 0 & -1 & \sqrt{5} & | & 0 \end{bmatrix} \xrightarrow[L_2 + L_2 + 0.8142L_1]{L_1 \leftrightarrow L_2} \begin{bmatrix} -\sqrt{5} & 2 & 0 & | & 0 \\ 0 & -0.447 & -1 & | & 0 \\ 0 & -1 & -2.24 & | & 0 \end{bmatrix} \xrightarrow[L_3 - 2.24L_2]{\frac{1}{\sqrt{5}}} \begin{bmatrix} -2.24 & x_1 & 0 & | & 0 \\ 0 & x_2 & -2.24x_3 & | & 0 \\ 0 & 0 & x_3 & | & 0 \end{bmatrix} \\
 &\begin{cases} -2.24x_1 + 2x_2 = 0 \Rightarrow x_2 = -2.24x_3 \\ -0.447x_2 - x_3 = 0 \Rightarrow x_1 = -2x_3 \\ x_3 = x_3 \end{cases} \\
 &x = x_3 \begin{bmatrix} -2 & -\sqrt{5} & 1 \end{bmatrix} \\
 \lambda = 3 - \sqrt{5}: & \text{Como a única coisa que muda é o sinal da soma, pode se afirmar, com base em } \lambda_2, \text{ que:} \\
 X &= x_3 \begin{bmatrix} -2 & \sqrt{5} & 1 \end{bmatrix}
 \end{aligned}$$

Figure 1: Imagem contendo o cálculo dos autovalores e autovetores da matriz do enunciado

- 3.2 A matriz A é positiva definida?

$$\begin{aligned}
 \text{B} \rightarrow \text{criterio de Sylvester:} \\
 \text{Para todo submatriz da Matriz A, se } \det(\text{submatriz}) > 0, \\
 \text{A m\~ot\~a \'e definida positiva.}
 \end{aligned}$$

$$\begin{aligned}
 A_{1,1} &= [3], \det(A_{1,1}) > 0 \\
 A_{2,2} &= \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}, \det(A_{2,2}) > 0 & \text{Portanto, A \'e definida positiva} \\
 A_{3,3} &= \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix}, \det(A_{3,3}) > 0
 \end{aligned}$$

Figure 2: Imagem contendo os cálculos da verificação se a matriz é definida positiva ou não

3.3 Calcule pelo “Power Method” o maior autovalor e o correspondente autovetor de A

Passo 1:
 $x_0 = [1, 1, 1]^T$ $x_{100} = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix}$ $R = \frac{|5-1|}{5} = 0.8$

$$x^1 = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 2 \end{bmatrix} \rightarrow \lambda = 5, x^1 = [1, 0.8, 0.4]^T$$

Passo 2:
 $x_1 = [1, 0.8, 0.4]^T$ $x_{100} = \begin{bmatrix} 1 \\ 4.6 \\ 4.6 \end{bmatrix}$ $R = \frac{|4.6-0.4|}{4.6} = 0.9$.

$$x^2 = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4.6 \\ 4.6 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 0.4 \end{bmatrix} \rightarrow \lambda = 4.6, x^2 = [1, 0.87, 0.09]^T$$

Passo 3:
 $x_2 = [1, 0.87, 0.09]^T$ $x_{100} = \begin{bmatrix} 1 \\ 4.74 \\ 4.74 \end{bmatrix}$ $R = \frac{|4.74-0.09|}{4.74} = 0.03$.

$$x^3 = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4.74 \\ 4.74 \end{bmatrix} = \begin{bmatrix} 4.92 \\ 4.92 \\ -0.6 \end{bmatrix} \rightarrow \lambda = 4.74, x^3 = [1, 0.95, 0.13]^T$$

Passo 4:
 $x_3 = [1, 0.95, 0.13]^T$ $x_{100} = \begin{bmatrix} 1 \\ 4.98 \\ 4.98 \end{bmatrix}$ $R = \frac{|4.98-0.13|}{4.98} = 0.03$.

$$x^4 = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4.98 \\ 4.98 \end{bmatrix} = \begin{bmatrix} 5.02 \\ 5.02 \\ -0.27 \end{bmatrix} \rightarrow \lambda = 5.02, x^4 = [1, 1.02, -0.27]^T$$

Passo 5:
 $x_4 = [1, 1.02, -0.27]^T$ $x_{100} = \begin{bmatrix} 1 \\ 5.04 \\ 5.04 \end{bmatrix}$ $R = \frac{|5.04-0.27|}{5.04} = 0.03$.

$$x^5 = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5.04 \\ 5.04 \end{bmatrix} = \begin{bmatrix} 5.33 \\ 5.33 \\ -1.83 \end{bmatrix} \rightarrow \lambda = 5.04, x^5 = [1, 1.06, -0.36]^T$$

(kajom)

Figure 3: Imagem contendo o cálculo do power method até o passo 5

3.4 Usando o método de Jacobi, obtenha todos autovalores e autovetores de A; use uma tolerância $|a_{i,j}| <= 10^{-3}$ para os elementos fora da diagonal

D S T Q Q S S Tolv.: 10^{-3}

$$A = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Término 0 - vetor gerado (0,1)

$$P = \begin{bmatrix} 0,707 & -0,707 & 0,0 \\ 0,707 & 0,707 & 0,0 \\ 0,0 & 0,0 & 1,0 \end{bmatrix} \quad A_2 = P^T A_1 P = \begin{bmatrix} 3,0 & 0 & -0,707 \\ 0 & 1,0 & -0,707 \\ -0,707 & -0,707 & 3,0 \end{bmatrix} \quad X_2 = X_1 P = \begin{bmatrix} 0,707 & -0,707 & 0 \\ 0,707 & 0,707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Término 0 - vetor gerado (1,2)

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0,95 & -0,30 \\ 0,30 & 0,95 \end{bmatrix} \quad A_3 = P^T A_2 P = \begin{bmatrix} 5 & -0,21 & -0,67 \\ -0,21 & 0,77 & 0 \\ -0,67 & 0,77 & 3,22 \end{bmatrix} \quad X_3 = X_2 P = \begin{bmatrix} 0,707 & -0,67 & 0,21 \\ 0,707 & 0,17 & -0,21 \\ 0 & 0,30 & 0,95 \end{bmatrix}$$

Término 0 - vetor gerado (0,2)

$$P = \begin{bmatrix} 0,84 & 0 & 0,31 \\ 0 & 1 & 0 \\ 0,31 & 0 & 0,94 \end{bmatrix} \quad A_4 = P^T A_3 P = \begin{bmatrix} 5,23 & -0,20 & 0 \\ -0,20 & 0,77 & -0,77 \\ 0 & -0,77 & 2,99 \end{bmatrix} \quad X_4 = X_3 P = \begin{bmatrix} 0,60 & -0,67 & 0,43 \\ 0,74 & 0,17 & 0,02 \\ 0,90 & 0,30 & 0,99 \end{bmatrix}$$

Término 0 - vetor gerado (0,1)

$$P = \begin{bmatrix} 0,99 & 0,04 & 0 \\ 0,04 & 0,99 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A_5 = P^T A_4 P = \begin{bmatrix} 5,23 & 0 & 0,003 \\ 0 & 0,77 & -0,91 \\ 0,003 & 0 & 3 \end{bmatrix} \quad X_5 = X_4 P = \begin{bmatrix} 0,63 & -0,64 & 0,43 \\ 0,71 & 0,71 & 0,0009 \\ -0,32 & 0,32 & 0,89 \end{bmatrix}$$

Figure 4: Imagem contendo o cálculo da obtenção dos autovalores e autovetores utilizando o método de Jacobi. Parte 1

Término 0 - vetor gerado (2,1)

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0,99 & -0,03 \\ 0,03 & 0,99 \end{bmatrix} \quad A_6 = P^T A_5 P = \begin{bmatrix} 5,24 & 0 & 0,003 \\ 0 & 0,76 & 0 \\ 0,003 & 0 & 3 \end{bmatrix} \quad X_6 = X_5 P = \begin{bmatrix} 0,63 & -0,63 & 0,45 \\ 0,71 & 0,71 & 0,0009 \\ -0,32 & 0,32 & 0,89 \end{bmatrix}$$

Término 0 - vetor gerado (2,0)

$$P = \begin{bmatrix} 1 & 0 & -0,001 \\ 0 & 1 & 0 \\ 0,001 & 0 & 1 \end{bmatrix} \quad A_7 = P^T A_6 P = \begin{bmatrix} 5,24 & 0 & 0 \\ 0,001 & 0,76 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad X_7 = X_6 P = \begin{bmatrix} 0,63 & -0,63 & 0,45 \\ 0,71 & 0,71 & 0 \\ -0,32 & 0,32 & 0,89 \end{bmatrix}$$

Figure 5: Imagem contendo o cálculo da obtenção dos autovalores e autovetores utilizando o método de Jacobi. Parte 2

3.5 Obtenha o vetor solução X usando os métodos de Cholesky (direto), Jacobi e Gauss-Seidel (iterativos) (use $\text{tol} = 10^{-5}$) e também pela técnica que utiliza os autovalores e autovetores de A.

3.5.1 Obtenha o vetor solução X usando o método de Cholesky (direto)

$$\begin{array}{l}
 \left[\begin{array}{ccc} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{array} \right] \left[\begin{array}{ccc} L_{1,1}^2 & L_{1,1}L_{2,1} & L_{1,1}L_{3,1} \\ L_{1,1}L_{2,1} & L_{2,2}^2 + L_{2,1}^2 & L_{2,2}L_{3,1} + L_{2,1}L_{3,2} \\ L_{1,1}L_{3,1} & L_{2,2}L_{3,1} + L_{2,1}L_{3,2} & L_{3,3}^2 + L_{2,2}^2 + L_{3,1}^2 \end{array} \right] \\
 L_{1,1}^2 = 3 \Rightarrow \sqrt{3} = 1,73 = L_{1,1} \\
 L_{1,1}L_{2,1} = 2 \Rightarrow 2/1,73 = 1,15 = L_{2,1} \\
 L_{1,1}L_{3,1} = 0 \Rightarrow 0 = L_{3,1} \\
 L_{2,2}^2 + L_{2,1}^2 = 3 \Rightarrow L_{2,2} = 1,29 \\
 L_{2,2}L_{3,1} + L_{2,1}L_{3,2} = -1 \Rightarrow L_{3,2} = -0,775 \\
 L_{2,2}L_{3,1} + L_{2,1}L_{3,2} = -1 \Rightarrow L_{3,2} = -0,775 \\
 L_{3,3}^2 + L_{2,2}^2 + L_{3,1}^2 = 3 \Rightarrow L_{3,3} = \sqrt{3,79} = 1,95
 \end{array}$$

$$L = \left[\begin{array}{cc} 1,73 & 0 \\ 1,15 & 1,29 \\ 0 & -0,775 \end{array} \right], \quad U = \left[\begin{array}{cc} 1,73 & 1,15 \\ 0 & 1,29 \\ 0 & 0 \end{array} \right]$$

Resolução por eliminação de Gauß:

$$\left[\begin{array}{ccc|c} 1,73 & 0 & 0 & 1 \\ 1,15 & 1,29 & 0 & -1 \\ 0 & 0 & 1,95 & 1 \end{array} \right] \xrightarrow{\text{L}_2 \leftarrow \text{L}_2 - \frac{1,15}{1,73} \text{L}_1} \left[\begin{array}{ccc|c} 1,73 & 0 & 0 & 1 \\ 0 & 1,29 & 0 & -1,666 \\ 0 & 0 & 1,95 & 1 \end{array} \right] \xrightarrow{\text{L}_3 \leftarrow \text{L}_3 + 1,666 \text{L}_1} \left[\begin{array}{ccc|c} 1,73 & 0 & 0 & 1 \\ 0 & 1,29 & 0 & -0,775 \\ 0 & 0 & 1,551 & 1 \end{array} \right]$$

$$\left\{ \begin{array}{l} 1,73X_1 = 1 \\ 1,29X_2 = -0,775 \\ 1,551X_3 = -0,007 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} X_1 = 0,578 \\ X_2 = -0,602 \\ X_3 = -0,007 \end{array} \right.$$

$$\boxed{\text{kajoma} \quad X = [1, -1, 0]}$$

Figure 6: Imagem contendo o cálculo da solução $AX=B$ utilizando o método de Cholesky

3.5.2 Obtenha o vetor solução X usando o método de Jacobi iterativo

A		B		Tentativa	X_1	X_2	X_3	R.
9 2 0	2 9 -1	0 -1 3	1					
$x_1^1 = b_1 - (A_{1,2} \cdot x_2 + A_{1,3} \cdot x_3) / A_{1,1}$				1	0,33	-0,33	0,33	1,0
$x_2^1 = b_2 - (A_{2,1} \cdot x_1 + A_{2,3} \cdot x_3) / A_{2,2}$				2	0,55	-0,44	0,22	0,365
$x_3^1 = b_3 - (A_{3,1} \cdot x_1 + A_{3,2} \cdot x_2) / A_{3,3}$				3	0,63	-0,63	0,18	0,22
				4	0,75	-0,69	0,12	0,196
				5	0,77	-0,79	0,10	0,099
				6	0,86	-0,828	0,068	0,070
				7	0,83	-0,83	0,053	0,049
				8	0,92	-0,905	0,038	0,036
				9	0,94	-0,94	0,032	0,026
				10	0,96	-0,95	0,021	0,019
				11	0,969	-0,96	0,017	0,014
				12	0,976	-0,97	0,012	0,010
				13	0,980	-0,98	0,0098	0,0077
				14	0,986	-0,983	0,0065	0,0057
				15	0,989	-0,987	0,0054	0,0042
				16	0,993	-0,990	0,0036	0,0031
				17	0,994	-0,994	0,0030	0,0024
				18	0,996	-0,997	0,0020	0,0017
				19	0,997	-0,9992	0,0016	0,0013
				20	0,998	-0,9972	0,0011	0,00097 $\rightarrow 2^4$
$x = [0,998, -0,9972, 0,00097]$								

Figure 7: Imagem contendo o cálculo da solução $AX=B$ utilizando o método de jacobi

3.5.3 Obtenha o vetor solução X usando o método de Gauss-Seidel iterativo

A		B		Tentativa	X_1	X_2	X_3	R.
9 2 0	2 9 -1	0 -1 3	1					
$x_1^1 = b_1 - (A_{1,2} \cdot x_2 + A_{1,3} \cdot x_3) / A_{1,1}$				1	-0,33	0,22	0,407	2,89
$x_2^1 = b_2 - (A_{2,1} \cdot x_1 + A_{2,3} \cdot x_3) / A_{2,2}$				2	0,18	-0,32	0,23	1,77
$x_3^1 = b_3 - (A_{3,1} \cdot x_1 + A_{3,2} \cdot x_2) / A_{3,3}$				3	0,55	-0,62	0,13	0,57
				4	0,75	-0,79	0,07	0,24
				5	0,86	-0,88	0,04	0,12
				6	0,92	-0,94	0,02	0,067
				7	0,96	-0,96	0,012	0,033
				8	0,98	-0,98	0,007	0,018
				9	0,99	-0,989	0,004	0,010
				10	0,993	-0,994	0,002	0,006
				11	0,996	-0,997	0,001	0,003
				12	0,998	-0,998	0,0006	0,002
				13	0,999	-0,999	0,0003	0,0009
$\hookrightarrow R \approx 10^{-3}$								
$x = [0,999, -0,999, 0,0003]$								

Figure 8: Imagem contendo o cálculo da solução $AX=B$ utilizando o método de gauss-seidel

3.5.4 Obtenha o vetor solução X usando o método de Jacobi que usa os autovalores e autovetores de A

Handwritten notes showing the steps to solve the system $AX=B$ using eigenvalues and eigenvectors of matrix A .

Given:

$$A = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Calculated values:

$$\lambda = \begin{bmatrix} 5,24 & 0 & 0 \\ 0 & 0,71 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad Q = \begin{bmatrix} 0,63 & -0,63 & 0,45 \\ 0,71 & 0,71 & 0 \\ 0,32 & 0,32 & 0,89 \end{bmatrix} \quad Q^T = \begin{bmatrix} 0,63 & 0,71 & -0,32 \\ -0,63 & 0,71 & 0,32 \\ 0,45 & 0 & 0,89 \end{bmatrix}$$

Inverse of λ :

$$\lambda^{-1} = \begin{bmatrix} 0,19 & 0 & 0 \\ 0 & 1,31 & 0 \\ 0 & 0 & 0,33 \end{bmatrix}$$

Solution:

$$X = Q \lambda^{-1} Q^T B = \begin{bmatrix} 0,99 \\ -1 \\ 0 \end{bmatrix}$$

Figure 9: Imagem contendo o resultado de $AX=B$ utilizando os autovetores e autovalores de A

3.6 Calcule o determinante de A utilizando os seus autovalores

Handwritten notes showing the calculation of the determinant of matrix A using its eigenvalues.

Dado que os autovalores são:

$\lambda = [3, 3+\sqrt{5}, 3-\sqrt{5}]$:

$|\det(A)| = \prod_{i=1}^n |\lambda_i| \therefore 3 \cdot (3+\sqrt{5}) \cdot (3-\sqrt{5}) = |\det(A)|$ (1)

$3 \cdot (3^2 - (\sqrt{5})^2) = 3 \cdot (9-5) = 3 \cdot 4 = 12 \therefore |\det(A)| = 12.$

Figure 10: Imagem contendo o cálculo da determinante da matriz do enunciado, por meio dos autovalores

4 Questão 4

4.1 A matriz A é positiva definida?

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_2$ python3 main.py
True
```

Figure 11: Imagem contendo os cálculos da verificação se a matriz é definida positiva ou não, no pc

4.2 Calcule pelo “Power Method” o maior autovalor e o correspondente autovetor de A

Como é possível observar abaixo, o resultado é um pouco diferente do resultado obtido manualmente, visto que o número de iterações foram diferentes.

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_2$ python3 main.py
Eigenvalue: 5.2315495656305195
Eigenvector: [1.0, 1.1167384607358495, -0.49710311111638605]
Number of iterations: 12
```

Figure 12: Imagem contendo o cálculo do power method pelo computador

4.3 Usando o método de Jacobi, obtenha todos autovalores e autovetores de A; use uma tolerância $|a_{i,j}| \leq 10^{-3}$ para os elementos fora da diagonal

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_2$ python3 main.py
1º Eigenvalue: 5.236067975487744, Eigenvector: [0.6324421432111451, 0.44721355824702347]
2º Eigenvalue: 0.7639320245122635, Eigenvector: [0.7070917825774846, 0.7071217794774768, 4.1648279142060193e-08]
3º Eigenvalue: 2.9999999999999925, Eigenvector: [-0.31623447345421635, 0.3162210057524644, 0.8944272096263812]
Steps: 7
```

Figure 13: Imagem contendo o resultado de Jacobi para obtenção dos autovalores e autovetores de A

4.4 Obtenha o vetor solução X usando os métodos de Cholesky (direto), Jacobi e Gauss-Seidel (iterativos) (use tol = 10^{-5}) e também pela técnica que utiliza os autovalores e autovetores de A

4.4.1 Obtenha o vetor solução X usando o método de Cholesky (direto)

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_1$ python3 main.py
[1.0000000000000004, -1.0000000000000007, -1.850371707708594e-16]
```

Figure 14: Imagem contendo o cálculo da solução da matriz usando o método da decomposição de Cholesky

4.4.2 Obtenha o vetor solução X usando o método de Jacobi iterativo

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_1$ python3 main.py
x1: [0.9999796677755572, -0.9999745847194466, 1.016611222133997e-05]
residue: 8.804312828820505e-06
iteration_number: 36
```

Figure 15: Imagem contendo o cálculo da solução da matriz utilizando o método de jacobi iterativo

4.4.3 Obtenha o vetor solução X usando o método de Gauss-Seidel iterativo

```
bdantas@Oracle:~/Área de Trabalho/ALC_Lists/List_1$ python3 main.py
x1: [0.9999884951404902, -0.9999904126170751, 3.195794308297195e-06]
residue: 8.662521591198997e-06
iteration_number: 21
```

Figure 16: Imagem contendo o cálculo da solução da matriz utilizando o método de gauss-seidel iterativo