EP 2 - Autovalores e Autovetores de Matrizes Reais Simétricas - O Algoritmo QR

Rhenan Silva Nehlsen - 11374871
Vinicius Maalouli Vinha - 11257421

21 de Julho de 2021

Sumário

1 Introdução

Nesse Exercício Programa foi adicionado ao algoritmo QR, o qual encontrava os autovalores e autovetores de uma matriz tridiagonal simétrica, as transformações de Householder. Elas têm como função transformar uma matriz simétrica qualquer em uma matriz tridiagonal simétrica, por meio de um processo iterativo, utilizando equações de formação, matrizes e elementos da álgebra linear. Implementou-se o programa feito no EP1 em Python com as transformações de Householder e observou-se a aplicação de todo esse algoritmo nas treliças planas, visando encontrar os modos de vibração e as frequências, estudando seus deslocamentos. Essas tarefas serão melhor apresentadas, discutidas e detalhadas na seção "Questões Propostas" bem como as conclusões finais na seção "Conclusões e Comentários Finais".

2 Questões Propostas

2.1 Cálculo de Autovalores e Autovetores

Para calcular os autovalores e autovetores de uma matriz simétrica A, a partir do algoritmo elaborado no EP1, precisa-se primeiro tridiagonalizar esta matriz. Para tanto, aplica-se as transformações de Householder.
As transformações de Householder ($H_w : \mathbb{R}^n \to \mathbb{R}^n$) são do tipo:

$$H_\omega = I - \frac{2 * \omega\omega^T}{\omega \cdot \omega} \tag{1}$$

Onde I é a matriz identidade e $\omega \in \mathbb{R}^n$

.

Estas transformações são úteis, pois seja A uma matriz simétrica de tamanho n e

$$H_\omega = H_{\omega_1} H_{\omega_2} \cdots H_{\omega_{n-2}} \tag{2}$$

Temos que $T = H_\omega \mathrm{A} H_\omega$ é tridiagonal simétrica.

Basta agora descobrir quem é o vetor ω. Para tridiagonalizar a matriz A, forma-se um vetor ω_i tal que quando aplicado H_{ω_i} sob A, não zera apenas os $(1+i)$ primeiros elementos da coluna i de A. Desta forma, define-se:

$$\omega_i = a_i + \delta \|a_i\| \hat{e}_i \tag{3}$$

Onde a_i é formado pelos $(n-i)$ últimos elementos da coluna i de A, $\hat{e}_i$ o versor da coluna i e δ o sinal do primeiro elemento de a_i.

Finalmente, para realizar as multiplicações $H_{\omega_i} A$, pode-se obter a partir de (1) que cada coluna nova de $x_j^{(k+1)}$ de A será:

$$x_j^{(k+1)} = x_j^{(k)} - \left(\frac{2 * \omega_i x_j^{(k)}}{\omega_i \cdot \omega_i} \right) \omega_i \tag{4}$$

Analogamente, na multiplicação a direita AH_{ω_i}, as novas linhas $y_j^{(k+1)}$ são dadas por:

$$y_j^{(k+1)} = y_j^{(k)} - \left(\frac{2 * \omega_i y_j^{(k)}}{\omega_i \cdot \omega_i} \right) \omega_i^T \tag{5}$$

Tendo em vista o embasamento teórico comentado até agora, passa-se para a implementação do algoritmo de triadigonalização de matrizes simétricas, a partir de transformações de Householder.

2.1.1 O algoritmo

Nesta subseção é explicado cada parte do algoritmo implementado. Começa-se com as funções auxiliares:

```
def produto_interno(x,y):
  #Retorna o produto interno entre os vetores x e y
  return sum([x[i]*y[i] for i in range(len(x))])

def norma(x):
  #Retorna a norma do vetor x
  return np.sqrt(produto_interno(x,x))

def monta_omega(a):
  #Monta o vetor w das transformações de Householder a partir
  # de um vetor a
  a0 = a[0] + np.sign(a[0])*norma(a)

  return  [a0] + a[1:] #Apenas o primeiro elemento é modificado

def househoulder_vetor(x,w,prod_int):
  #Aplica a transfromação de Householder a um vetor x
  escalar = 2*produto_interno(x,w)/prod_int
  vetor =[x[i] - escalar*w[i] for i in range(len(x))]
  return vetor
```

Figura 1: Funções auxiliares

Na Figura 1, as funções "produto_interno"e "norma"são bem intuitivas e calculam o produto interno entre dois vetores e a norma de um vetor, respectivamente. Já a função "monta_omega"forma o vetor ω_i a partir de a_i, conforme (3). Por fim, a função "householder_vetor"realiza as operações de (4) e (5).

Partindo para a função principal:

```python
def tridiagonalizacao_householder(A):
  #Tridagonaliza uma matriz simétrica A
  # por meio das Trasformações de Householder

  H_transposto = np.eye(len(A)) #Inicia H^t como a identidade
  for i in range(0,len(A)-2):
    w = monta_omega(A[i+1:,i].tolist()) #Monta o vetor omega a partir das colunas
    norma2w = produto_interno(w,w) # Calcula a norma^2 de w

    # Aplica a trasformação na coluna i da matriz, aproveitando que os outros elementos são nulos
    primeiro_elemento = A[i+1,i] - (2*(produto_interno(A[i+1:, i],w))/norma2w)*w[0]
    A[i+1:, i] = [primeiro_elemento] + [0]*(len(A)-(2+i))


    #Faz a multiplicação por Hw a esquerda
    for j in range(i+1, len(A)):
      A[i+1:,j] = househoulder_vetor(A[i+1:,j],w, norma2w)

    #Copia a coluna i para a linha i, aproveitando a simetria
    A[i, i+1:] = A[i+1:, i]

    #Faz a multiplicação por Hw a direita
    for j in range(i+1, len(A)):
      A[j,i+1:] = househoulder_vetor(A[j,i+1:],w, norma2w)

    #Faz a multiplicação de H^t por Hw
    for j in range(len(A)):
      H_transposto[j, i+1:] = househoulder_vetor( H_transposto[j, i+1:], w, norma2w)

  return A, H_transposto #Retorna a matriz tridiagonal e H^t
```

Figura 2: Tridiagonalização Householder

Nesta função, todos os passos da tridiagonalização são executados. Começa-se definindo H^T, que será utilizado para obtenção dos autovetores, como a identidade. A partir deste ponto, entra-se num loop para fazer as multiplicações de H_ω por A à esquerda e à direita $n-2$ vezes. Dentro do loop, monta-se o vetor ω_i e calcula-se $\omega_i \cdot \omega_i$. Como forma de otimização, calcula-se o termo $i+1$ e é zerado os $n-i$ últimos termos da coluna i , a refletindo para a linha i, devido a simetria. Por fim, faz-se as multiplicações $H_{\omega_i} A^i$, $H_{\omega_i} A^i H_{\omega_i}$ e $H^T H_{\omega_i}$.

Para finalizar, tendo a matriz T tridiagonal simétrica e a matriz H^T, pode-se aplicar o algoritmo QR do EP1 para encontrar os autovalores e autovetores de A:

```python
def separa_diagonais(A):
    #Separa a diagonal e a subdiagonal da matriz tridiagonal para
    # aplicação do algoritmo QR

    diagonal = []
    subdiagonal = []
    for i in range(len(A)):
        if i==len(A)-1:
            subdiagonal.append(0)
        else:
            subdiagonal.append(A[i+1][i])
        diagonal.append(A[i][i])

    return diagonal,subdiagonal


def calcula_valores(A):

    #Calcula os autovalores e autovetores de uma matriz simétrica

    tridiagonal, H_t = tridiagonalizacao_householder(A)


    diagonal, subdiagonal = separa_diagonais(tridiagonal)

    autovalores, autovetores, _ = QR(diagonal, subdiagonal, H_t)
    return autovalores, autovetores
```

Figura 3: Aplicação do algoritmo QR para encontrar λ e V

A função "separa_valores"passa a diagonal e a subdiagonal da matriz T para dois vetores, conforme a implementação do QR. Já a função "calcula_valores"junta todas as partes do algoritmo.

2.1.2 Teste a

Para testar as funções implementadas, começa-se utilizando a seguinte matriz:

$$A = \begin{pmatrix} 2 & 4 & 1 & 1 \\ 4 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \end{pmatrix}$$

cujos autovalores são conhecidos e valem (7,-2,2 e 1)

Os valores obtidos foram:

λ_1 analítico $= 7$

λ_1 encontrado pelo algoritmo $= 7.0$

Erro $\lambda_1 = 0.0$

Autovetor associado à λ_1

$$\begin{array}{c|c|c} \text{V} & \text{A x V} & \lambda_1 \text{ x V} \\ \begin{bmatrix} 0.632455537675667 \\ 0.632455537675666 \\ 0.316227754732857 \\ 0.316227754732857 \end{bmatrix} & \begin{bmatrix} 4.42718873551971 \\ 4.42718873551971 \\ 2.2135943395499 \\ 2.2135943395499 \end{bmatrix} & \begin{bmatrix} 4.42718876372967 \\ 4.42718876372966 \\ 2.21359428313 \\ 2.21359428313 \end{bmatrix} \end{array}$$

λ_2 analítico $= -2$

λ_2 encontrado pelo algoritmo $= -1.99999999999996$

Erro $\lambda_2 = 4.39648317751562 \cdot 10^{-14}$

Autovetor associado à λ_2

$$\begin{bmatrix} 0.707106748403919 \\ -0.707106813969169 \\ 6.55652520886729 \cdot 10^{-8} \\ 6.55652520886729 \cdot 10^{-8} \end{bmatrix} \quad \Bigg| \quad \begin{array}{c} \text{A x V} \\ \begin{bmatrix} -1.41421362793834 \\ 1.41421349680784 \\ 1.31130505481858 \cdot 10^{-7} \\ 1.31130505481858 \cdot 10^{-7} \end{bmatrix} \end{array} \quad \Bigg| \quad \begin{array}{c} \lambda_2 \text{ x V} \\ \begin{bmatrix} -1.41421349680781 \\ 1.41421362793831 \\ -1.31130504177343 \cdot 10^{-7} \\ -1.31130504177343 \cdot 10^{-7} \end{bmatrix} \end{array}$$

λ_3 analítico $= 2$
λ_3 encontrado pelo algoritmo $= 1.99999999999996$
Erro $\lambda_3 = 4.30766533554561 \cdot 10^{-14}$

Autovetor associado à λ_3

$$\begin{bmatrix} 0.316227828037035 \\ 0.316227681428675 \\ -0.632455537675663 \\ -0.632455537675663 \end{bmatrix} \quad \Bigg| \quad \begin{array}{c} \text{A x V} \\ \begin{bmatrix} 0.632455306437446 \\ 0.632455599654166 \\ -1.26491110356128 \\ -1.26491110356128 \end{bmatrix} \end{array} \quad \Bigg| \quad \begin{array}{c} \lambda_3 \text{ x V} \\ \begin{bmatrix} 0.632455656074057 \\ 0.632455362857337 \\ -1.2649110753513 \\ -1.2649110753513 \end{bmatrix} \end{array}$$

λ_4 analítico $= -1$
λ_4 encontrado pelo algoritmo $= -1.0$
Erro $\lambda_4 = 0.0$

Autovetor associado à λ_4

$$\begin{bmatrix} 2.37904933848248 \cdot 10^{-17} \\ 5.28677630773884 \cdot 10^{-18} \\ -0.707106781186548 \\ 0.707106781186547 \end{bmatrix} \quad \Bigg| \quad \begin{array}{c} \text{A x V} \\ \begin{bmatrix} -4.22942104619107 \cdot 10^{-17} \\ -5.28677630773881 \cdot 10^{-18} \\ 0.707106781186547 \\ -0.707106781186548 \end{bmatrix} \end{array} \quad \Bigg| \quad \begin{array}{c} \lambda_4 \text{ x V} \\ \begin{bmatrix} -2.37904933848248 \cdot 10^{-17} \\ -5.28677630773884 \cdot 10^{-18} \\ 0.707106781186548 \\ -0.707106781186547 \end{bmatrix} \end{array}$$

Por fim, para testar a ortogonalidade da matriz de autovetores, é utilizada a seguinte propriedade:

$$Q \; ortogonal \rightarrow Q^t = Q^{-1} \rightarrow QQ^t = I \tag{6}$$

Dessa forma, pode-se verificar se $V \times V^t$ é igual a Identidade:

$$V \times V^t = \begin{bmatrix} 1.0 & -3.3307 \cdot 10^{-16} & -1.014 \cdot 10^{-16} & -6.775 \cdot 10^{-17} \\ -3.3307 \cdot 10^{-16} & 1.0 & 3.4637 \cdot 10^{-18} & 1.094 \cdot 10^{-17} \\ -1.014 \cdot 10^{-16} & 3.4637 \cdot 10^{-18} & 1.0 & 1.1102 \cdot 10^{-16} \\ -6.775 \cdot 10^{-17} & 1.094 \cdot 10^{-17} & 1.1102 \cdot 10^{-16} & 1.0 \end{bmatrix}$$

2.1.3 Teste b

Para o proxímo teste do algoritmo implementado, utiliza-se a seguinte matriz:

$$A = \begin{pmatrix} n & n-1 & n-2 & \cdots & 2 & 1 \\ n-1 & n-1 & n-2 & \cdots & 2 & 1 \\ n-1 & n-2 & n-2 & \cdots & 2 & 1 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 2 & 2 & 2 & \cdots & 2 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

cujo autovalores são:

$$\lambda_i = \frac{1}{2}\left[1 - cos\left(\frac{(2i-1)\pi}{2n+1}\right)\right]^{-1}, i = 1, 2, \cdots, n.$$

com $n = 20$.

Para este problema, obteve-se os seguintes resultados:

$$\lambda_1 \text{ analítico} = 170.404267505427$$
$$\lambda_1 \text{ encontrado pelo algoritmo} = 170.404267505428$$
$$\text{Erro } \lambda_1 = 1.36424205265939 \cdot 10^{-12}$$

Autovetor associado à λ_1

V	A x V	λ_1 x V
0.312118317245532	53.1862932252516	53.1862932252517
0.310286682851543	52.8741749080061	52.8741749080061
0.306634162822505	52.251769907909	52.2517699079091
0.30118219159831	51.3227307449894	51.3227307449895
0.293962763513983	50.0925093904715	50.0925093904716
0.285018245044201	48.5683252724397	48.5683252724397
0.274401126179978	46.7591229093636	46.7591229093636
0.262173712396519	44.6755194201075	44.6755194201076
0.248407759019924	42.329742218455	42.329742218455
0.233184050138397	39.7355572577825	39.7355572577825
0.216591924529096	36.9081882469716	36.9081882469716
0.198728751382667	33.8642273116316	33.8642273116316
0.179699358902109	30.6215376249089	30.621537624909
0.159615419129206	27.1991485792842	27.1991485792842
0.138594792608571	23.6171441145302	23.6171441145302
0.116760836735112	19.8965448571677	19.8965448571677
0.0942416818437865	16.05918476307	16.05918476307
0.071169479289844	12.1275829871286	12.1275829871286
0.0476796259321293	8.1248117318973	8.12481173189731
0.0239099695704751	4.07436085073389	4.07436085073389

λ_2 analítico = 19.0080994910092

λ_2 encontrado pelo algoritmo = 19.0080994910092

Erro $\lambda_2 = 3.19744231092045 \cdot 10^{-14}$

Autovetor associado à λ_2

V	A x V	λ_2 x V
0.310286682851542	5.89796013837733	5.89796013837734
0.293962763513983	5.58767345552579	5.58767345552581
0.262173712396519	4.98342400916026	4.98342400916028
0.216591924529097	4.11700085039822	4.11700085039823
0.159615419129206	3.03398576710707	3.03398576710708
0.0942416818437865	1.79135526468673	1.79135526468673
0.0239099695704751	0.454483080422593	0.454483080422593
−0.0476796259321294	−0.906299073412016	−0.90629907341202
−0.116760836735112	−2.2194016013145	−2.2194016013145
−0.17969935890211	−3.41574329248186	−3.41574329248188
−0.233184050138397	−4.43238562474712	−4.43238562474713
−0.274401126179978	−5.21584390687398	−5.215843906874
−0.30118219159831	−5.72490106282086	−5.72490106282088
−0.312118317245532	−5.93277602716944	−5.93277602716945
−0.306634162822505	−5.82853267427248	−5.82853267427249
−0.285018245044202	−5.41765515855301	−5.41765515855303
−0.248407759019924	−4.72175939778935	−4.72175939778936
−0.198728751382667	−3.77745587800576	−3.77745587800577
−0.138594792608571	−2.6344236068395	−2.63442360683951
−0.071169479289844	−1.35279654306467	−1.35279654306468

λ_3 analítico $= 6.89678489274341$

λ_3 encontrado pelo algoritmo $= 6.89678489274342$

Erro $\lambda_3 = 1.77635683940025 \cdot 10^{-15}$

Autovetor associado à λ_3

V	A x V	λ_3 x V
-0.306634162822505	-2.11478986175327	-2.11478986175328
-0.262173712396519	-1.80815569893077	-1.80815569893077
-0.179699358902109	-1.23934782371174	-1.23934782371175
-0.0711694792898437	-0.49084058959061	-0.49084058959061
0.0476796259321297	0.328836123820368	0.328836123820369
0.159615419129206	1.10083321129922	1.10083321129922
0.248407759019924	1.71321487964886	1.71321487964886
0.30118219159831	2.07718878897858	2.07718878897858
0.310286682851542	2.13998050670998	2.13998050670998
0.274401126179978	1.89248554158985	1.89248554158985
0.198728751382667	1.37058945028973	1.37058945028974
0.0942416818437861	0.649964607606956	0.649964607606956
-0.0239099695704754	-0.16490191691961	-0.16490191691961
-0.138594792608571	-0.9558584718757	-0.9558584718757
-0.233184050138397	-1.60822023422322	-1.60822023422322
-0.293962763513983	-2.02739794643234	-2.02739794643234
-0.312118317245532	-2.15261289512748	-2.15261289512748
-0.285018245044201	-1.96570952657709	-1.96570952657709
-0.216591924529096	-1.49378791298249	-1.49378791298249
-0.116760836735112	-0.805274374858803	-0.805274374858804

λ_4 analítico $= 3.56048280769555$

λ_4 encontrado pelo algoritmo $= 3.56048280769555$

Erro $\lambda_4 = 3.99680288865056 \cdot 10^{-15}$

Autovetor associado à λ_4

V	A x V	λ_4 x V
0.301182191598311	1.07235401516984	1.07235401516985
0.216591924529096	0.771171823571532	0.771171823571538
0.0711694792898431	0.253397707444126	0.253397707444131
−0.0942416818437871	−0.335545887973125	−0.335545887973118
−0.233184050138397	−0.830247801546587	−0.83024780154658
−0.306634162822505	−1.09176566498165	−1.09176566498165
−0.293962763513982	−1.04664936559421	−1.04664936559421
−0.198728751382666	−0.707570302692792	−0.707570302692786
−0.0476796259321288	−0.169762488408705	−0.1697624884087
0.116760836735112	0.415724951807511	0.415724951807514
0.248407759019924	0.884451555288616	0.884451555288618
0.310286682851542	1.1047703997498	1.1047703997498
0.285018245044201	1.01480256135943	1.01480256135944
0.179699358902109	0.63981647792487	0.639816477924873
0.0239099695704751	0.0851310355881971	0.0851310355882004
−0.138594792608571	−0.493464376318951	−0.493464376318948
−0.262173712396519	−0.933464995617527	−0.933464995617524
−0.312118317245532	−1.11129190251959	−1.11129190251958
−0.274401126179978	−0.97700049217611	−0.977000492176108
−0.159615419129206	−0.568307955652658	−0.568307955652657

λ_5 analítico $= 2.18808019511022$
λ_5 encontrado pelo algoritmo $= 2.18808019511022$
Erro $\lambda_5 = 1.77635683940025 \cdot 10^{-15}$

Autovetor associado à λ_5

V	A x V	λ_5 x V
-0.293962763513985	-0.643214100944825	-0.643214100944821
-0.159615419129206	-0.34925133743084	-0.349251337430832
0.0476796259321315	0.10432684521235	0.104326845212361
0.233184050138398	0.510225401923409	0.510225401923418
0.312118317245532	0.682939908496071	0.682939908496079
0.248407759019923	0.543536097823199	0.543536097823208
0.0711694792898425	0.155724528130406	0.155724528130412
-0.138594792608573	-0.303256520852232	-0.303256520852227
-0.285018245044202	-0.623642777226296	-0.623642777226291
-0.301182191598309	-0.659010788556159	-0.659010788556153
-0.179699358902108	-0.393196608287712	-0.393196608287707
0.0239099695704762	0.0523169308828422	0.0523169308828471
0.216591924529097	0.473920500482921	0.473920500482925
0.310286682851542	0.678932145553903	0.678932145553906
0.262173712396518	0.573657107773343	0.573657107773345
0.0942416818437858	0.206208357596265	0.206208357596266
-0.116760836735112	-0.255482074424599	-0.255482074424598
-0.274401126179977	-0.600411669710351	-0.600411669710349
-0.306634162822504	-0.670940138816125	-0.670940138816124
-0.198728751382666	-0.434834445099396	-0.434834445099396

λ_6 analítico $= 1.49398982905874$

λ_6 encontrado pelo algoritmo $= 1.49398982905874$

Erro $\lambda_6 = 1.77635683940025 \cdot 10^{-15}$

Autovetor associado à λ_6

$$
\begin{array}{c|c|c}
\text{V} & \text{A x V} & \lambda_6 \text{ x V} \\
\begin{bmatrix}
0.285018245044202 \\
0.0942416818437857 \\
-0.159615419129209 \\
-0.306634162822505 \\
-0.248407759019922 \\
-0.023909969570472 \\
0.216591924529098 \\
0.312118317245532 \\
0.198728751382664 \\
-0.0476796259321315 \\
-0.26217371239652 \\
-0.301182191598309 \\
-0.138594792608569 \\
0.116760836735114 \\
0.293962763513983 \\
0.274401126179977 \\
0.0711694792898432 \\
-0.17969935890211 \\
-0.310286682851542 \\
-0.233184050138397
\end{bmatrix}
&
\begin{bmatrix}
0.425814359192209 \\
0.140796114148007 \\
-0.238463812739981 \\
-0.458108320498759 \\
-0.371118665435033 \\
-0.0357212513513852 \\
0.323586132302735 \\
0.466301591427757 \\
0.296898733307247 \\
-0.0712328761959262 \\
-0.391684859766968 \\
-0.449963130941491 \\
-0.207059210517704 \\
0.174439502514651 \\
0.439177378811893 \\
0.409952491595152 \\
0.106326478198434 \\
-0.268469014488127 \\
-0.463565148272579 \\
-0.348374599205488
\end{bmatrix}
&
\begin{bmatrix}
0.42581435919221 \\
0.140796114148006 \\
-0.238463812739987 \\
-0.458108320498764 \\
-0.371118665435038 \\
-0.0357212513513891 \\
0.323586132302731 \\
0.466301591427754 \\
0.296898733307243 \\
-0.0712328761959298 \\
-0.391684859766972 \\
-0.449963130941494 \\
-0.207059210517708 \\
0.174439502514648 \\
0.43917737881189 \\
0.40995249159515 \\
0.106326478198432 \\
-0.268469014488128 \\
-0.463565148272579 \\
-0.348374599205488
\end{bmatrix}
\end{array}
$$

λ_7 analítico $= 1.09545235007138$

λ_7 encontrado pelo algoritmo $= 1.09545235007138$

Erro $\lambda_7 = 4.44089209850063 \cdot 10^{-16}$

Autovetor associado à λ_7

V	A x V	λ_7 x V
-0.274401126179965	-0.300593358536076	-0.300593358536076
-0.0239099695704764	-0.0261922323561112	-0.0261922323561135
0.248407759019913	0.272118863394329	0.272118863394329
0.293962763513971	0.322022200124859	0.322022200124857
0.0711694792898464	0.077962773341417	0.0779627733414186
-0.216591924529084	-0.237266132731872	-0.237266132731868
-0.306634162822497	-0.335903114276077	-0.335903114276074
-0.116760836735119	-0.127905932997785	-0.127905932997786
0.179699358902096	0.196852085015625	0.196852085015622
0.312118317245529	0.34191074412694	0.341910744126939
0.159615419129217	0.174851085992726	0.174851085992729
-0.138594792608558	-0.151823991270705	-0.151823991270701
-0.310286682851545	-0.339904275925579	-0.339904275925578
-0.198728751382681	-0.217697877728907	-0.217697877728909
0.094241681843777	0.103237271850447	0.103237271850445
0.301182191598318	0.329930739586024	0.329930739586026
0.233184050138413	0.255442015723283	0.255442015723287
-0.0476796259321237	-0.052230758277872	-0.0522307582778692
-0.285018245044215	-0.312223906346903	-0.312223906346905
-0.262173712396535	-0.287198809371719	-0.287198809371723

λ_8 analítico $= 0.846121955021322$

λ_8 encontrado pelo algoritmo $= 0.846121955021324$

Erro $\lambda_8 = 2.33146835171283 \cdot 10^{-15}$

Autovetor associado à λ_8

V	A x V	λ_8 x V
0.262173712395292	0.221830934087301	0.221830934087103
−0.0476796259315735	−0.0403427783079922	−0.0403427783079084
−0.301182191596769	−0.254836864771711	−0.254836864771465
−0.19872875138238	−0.168148759638662	−0.168148759638606
0.138594792607164	0.117268096876769	0.117268096876549
0.312118317244463	0.264090160785034	0.264090160784851
0.116760836735959	0.0987939074488358	0.0987939074489552
−0.216591924527535	−0.183263182623322	−0.183263182623069
−0.293962763513988	−0.248728348167943	−0.248728348167927
−0.0239099695720819	−0.0202307501985768	−0.0202307501988304
0.274401126179102	0.232176817342871	0.232176817342715
0.248407759021097	0.210183258705216	0.210183258705397
−0.0711694792883119	−0.0602180589535355	−0.0602180589532761
−0.306634162822888	−0.259449897323975	−0.259449897324029
−0.179699358903878	−0.152047572871527	−0.152047572871828
0.159615419128673	0.1350541104848	0.135054110484701
0.310286682853066	0.262540374712453	0.262540374712718
0.0942416818451042	0.0797399560870407	0.0797399560872772
−0.233184050139271	−0.197302144383476	−0.197302144383631
−0.285018245045967	−0.241160194714721	−0.24116019471504

λ_9 analítico $= 0.680254988812241$
λ_9 encontrado pelo algoritmo $= 0.680254988812241$
Erro $\lambda_9 = 2.22044604925031 \cdot 10^{-16}$

Autovetor associado à λ_9

V	A x V	λ_9 x V
−0.248407758978082	−0.16898061730949	−0.168980617304509
0.116760836702923	0.0794271416685923	0.0794271416650547
0.310286682801566	0.211074063943748	0.211074063937766
0.0476796259529298	0.0324343034173395	0.032434303419182
−0.285018244987884	−0.193885083061996	−0.193885083055517
−0.198728751390115	−0.135186224553447	−0.135186224553554
0.179699358842098	0.122241385345216	0.122241385338698
0.293962763506423	0.19996963640178	0.199969636400277
−0.0239099695104315	−0.0162648760480781	−0.0162648760418196
−0.306634162799369	−0.208589418987505	−0.208589418984536
−0.138594792664423	−0.0942797991275652	−0.0942797991333717
0.233184050100723	0.158624613396801	0.15862461339246
0.262173712444275	0.178344975820443	0.178344975825644
−0.0942416817941234	−0.06410837420019	−0.0641083741945082
−0.312118317282425	−0.2123200424267	−0.212320042431051
−0.0711694793481071	−0.0484133933707844	−0.0484133933777196
0.274401126204678	0.186662735033238	0.186662735036429
0.216591924592576	0.147337737232582	0.147337737240544
−0.159615419141478	−0.108579185160649	−0.108579185162347
−0.301182191664155	−0.204880688412402	−0.204880688420946

λ_{10} analítico $= 0.564769726834725$

λ_{10} encontrado pelo algoritmo $= 0.564769726834726$

Erro $\lambda_{10} = 5.55111512312578 \cdot 10^{-16}$

Autovetor associado à λ_{10}

$$
\begin{array}{c|c|c}
\text{V} & \text{A x V} & \lambda_{10} \text{ x V} \\
\begin{bmatrix}
0.233184049475902 \\
-0.179699358173418 \\
-0.274401125568201 \\
0.116760835930551 \\
0.301182191020179 \\
-0.0476796250500205 \\
-0.312118316687663 \\
-0.0239099705240014 \\
0.306634162280147 \\
0.0942416828571175 \\
-0.285018244524591 \\
-0.159615420188671 \\
0.248407758543047 \\
0.216591925622046 \\
-0.198728750978813 \\
-0.262173713512803 \\
0.138594792312564 \\
0.293962764645468 \\
-0.0711694791326568 \\
-0.310286683990728
\end{bmatrix}
&
\begin{bmatrix}
0.131695291986656 \\
-0.101488757489246 \\
-0.154973448791729 \\
0.0659429854739866 \\
0.170098583809152 \\
-0.0269280088758617 \\
-0.176274976510859 \\
-0.0135036274581868 \\
0.173177692118484 \\
0.0532248494150096 \\
-0.160969676145584 \\
-0.090145957181587 \\
0.140293181971083 \\
0.122324562580703 \\
-0.112235982431724 \\
-0.148067776465335 \\
0.0782741430138554 \\
0.166021270180481 \\
-0.0401943672983589 \\
-0.175240525644544
\end{bmatrix}
&
\begin{bmatrix}
0.13169529192472 \\
-0.101488757427977 \\
-0.154973448730294 \\
0.0659429854134916 \\
0.17009858374995 \\
-0.0269280088150822 \\
-0.176274976455806 \\
-0.0135036275214666 \\
0.173177692069154 \\
0.0532248494836591 \\
-0.160969676103066 \\
-0.0901459572585655 \\
0.140293181935983 \\
0.12232456266817 \\
-0.112235982404511 \\
-0.148067776563871 \\
0.0782741429950821 \\
0.166021270288402 \\
-0.0401943672887203 \\
-0.175240525757896
\end{bmatrix}
\end{array}
$$

λ_{11} analítico $= 0.481555122390023$

λ_{11} encontrado pelo algoritmo $= 0.48155512239002$

Erro $\lambda_{11} = 2.44249065417534 \cdot 10^{-15}$

Autovetor associado à λ_{11}

V	A x V	λ_{11} x V
-0.216591918638184	-0.104300948347659	-0.1043009478885
0.233184041639473	0.112290970290526	0.112290969711096
0.19872874889599	0.0956988472892339	0.0956988469970243
-0.248407748986871	-0.119622024608043	-0.119622023966002
-0.179699360267547	-0.0865351475184504	-0.0865351474270467
0.262173702098051	0.126251089838685	0.126251089201272
0.159615424384654	0.0768636250977739	0.0768636252248868
-0.274401116977522	-0.132139264027793	-0.132139263470069
-0.138594801332677	-0.0667410361758405	-0.066741036518378
0.285018238263246	0.137251993008792	0.137251992610246
0.116760848017826	0.0562267839301773	0.0562267844575868
-0.293962760277345	-0.141559273166263	-0.141559273003465
-0.094241694299999	-0.0453825699853558	-0.0453825706328789
0.301182192618591	0.145035827495545	0.14503582762814
0.0711694911550918	0.0342720323578551	0.0342720330236257
-0.306634168173573	-0.147661253934922	-0.147661254383787
-0.047679635292872	-0.0229603720541305	-0.0229603726089705
0.310286691794273	0.149420145119535	0.149420145842986
0.0239099747485757	0.0115139704989276	0.0115139708163927
-0.31211832823944	-0.150302178870256	-0.150302179755512

λ_{12} analítico $= 0.4200300188321$

λ_{12} encontrado pelo algoritmo $= 0.420030018832097$

Erro $\lambda_{12} = 2.60902410786912 \cdot 10^{-15}$

Autovetor associado à λ_{12}

V	A x V	λ_{12} x V
0.198728718676355	0.0834720297020475	0.0834720274481082
−0.274401067465857	−0.115256688974307	−0.115256685535232
−0.0942416924177365	−0.0395843401848062	−0.0395843398409906
0.31028662648569	0.130329701022434	0.130329697566132
−0.0239099164744177	−0.0100428842560198	−0.0100428826670235
−0.301182165853279	−0.126505553060053	−0.126505550795245
0.138594716052819	0.0582139439891911	0.0582139411936947
0.248407780490023	0.104338724985618	0.104338724717264
−0.233183980648898	−0.0979442745079808	−0.0979442717833001
−0.159615484852933	−0.0670432933526754	−0.0670432951086716
0.293962730879013	0.123473172655556	0.123473171387047
0.0476797128021425	0.0200269077847803	0.0200269106661929
−0.31211833699042	−0.13109906988814	−0.131099070963929
0.0711694061953376	0.0298932894293573	0.0298932870244968
0.28501830923897	0.119716242551519	0.119716245797137
−0.179699330704479	−0.0754791135652901	−0.0754791132599176
−0.216592002759512	−0.0909751389776172	−0.0909751429979596
0.262173739807412	0.110120838369565	0.110120840868589
0.116760889634941	0.0490430759093354	0.0490430786722165
−0.306634228281004	−0.128795576185834	−0.128795580679436

λ_{13} analítico $= 0.373687363790749$

λ_{13} encontrado pelo algoritmo $= 0.373687363790708$

Erro $\lambda_{13} = 4.0467629247587 \cdot 10^{-14}$

Autovetor associado à λ_{13}

$$
\begin{array}{c|c|c}
\text{V} & \text{A x V} & \lambda_{13} \text{ x V} \\
\begin{bmatrix}
-0.179699241366643 \\
0.301181937313918 \\
-0.0239098122793701 \\
-0.285018124429572 \\
0.216591610435435 \\
0.138594970749127 \\
-0.310286492565474 \\
0.0711690985763657 \\
0.262173848003421 \\
-0.248407468947375 \\
-0.0942420765815297 \\
0.31211836094082 \\
-0.116760483222021 \\
-0.233184393566802 \\
0.274401090402644 \\
0.0476799691468048 \\
-0.306634436146477 \\
0.15961537041369 \\
0.198729037149866 \\
-0.293963011202426
\end{bmatrix}
&
\begin{bmatrix}
-0.0671513434892201 \\
0.112547897877422 \\
-0.00893479806985287 \\
-0.106507681737757 \\
0.0809375590239076 \\
0.0517911893501424 \\
-0.115950151072755 \\
0.0265950010698237 \\
0.0979710546360364 \\
-0.0928267398011711 \\
-0.0352170652910047 \\
0.116634685800691 \\
-0.0436319240484334 \\
-0.0871380506755364 \\
0.102540216264162 \\
0.0178173928012181 \\
-0.114585399808532 \\
0.059646243728195 \\
0.0742625168512322 \\
-0.109850247175597
\end{bmatrix}
&
\begin{bmatrix}
-0.0671513357814909 \\
0.112547884176216 \\
-0.00893479471940851 \\
-0.106507671550659 \\
0.0809375479228017 \\
0.0517911892538917 \\
-0.115950141426657 \\
0.0265949928303631 \\
0.0979710541152643 \\
-0.0928267322168666 \\
-0.0352170731559139 \\
0.116634687490652 \\
-0.0436319171701661 \\
-0.0871380613091133 \\
0.10254022009386 \\
0.0178174019760918 \\
-0.114585414091027 \\
0.0596462469903693 \\
0.0742625300011991 \\
-0.109850262708213
\end{bmatrix}
\end{array}
$$

λ_{14} analítico $= 0.338359135294937$

λ_{14} encontrado pelo algoritmo $= 0.338359135294667$

Erro $\lambda_{14} = 2.70061750740069 \cdot 10^{-13}$

Autovetor associado à λ_{14}

V	A x V	λ_{14} x V
-0.159615164260947	-0.0540072673490632	-0.0540072489592504
0.312117655210202	0.105607896911884	0.105607859927123
-0.13859408282613	-0.0468945940373695	-0.0468945740220066
-0.179699561679836	-0.0608030021604948	-0.06080298830282
0.310286050924566	0.104988151396216	0.104988119884833
-0.116759744845889	-0.0395067459716381	-0.0395067263032809
-0.198729355207374	-0.0672418984936036	-0.0672418927856338
0.306633610492069	0.103752304191805	0.103752283298378
-0.0942403564176891	-0.0318871036148551	-0.0318870855073505
-0.216592825172114	-0.0732861550038266	-0.0732861610362655
0.301181798261629	0.101907618779317	0.101907612826298
-0.0711681399348627	-0.024080405699169	-0.02408039028889
-0.233185103103593	-0.0789002902427922	-0.0789003098497296
0.293962624420322	0.099464928317178	0.0994649394078112
-0.0476785298382703	-0.0161324775431737	-0.0161324661281981
-0.24840879538965	-0.0840513535652558	-0.0840513852076318
0.285018438484216	0.0964385658023128	0.0964385923885553
-0.0239093504578425	-0.0080899533143346	-0.00808994714637273
-0.262174568320923	-0.0887091219731394	-0.0887091602333201
0.274401677351097	0.092846277688979	0.0928463142719235

λ_{15} analítico $= 0.311288812009951$

λ_{15} encontrado pelo algoritmo $= 0.311288812009458$

Erro $\lambda_{15} = 4.92828000631107 \cdot 10^{-13}$

Autovetor associado à λ_{15}

V	A x V	λ_{15} x V
0.13859460322567	0.0431429777456263	0.0431429493890412
−0.30663345138629	−0.0954516254800439	−0.095451562804398
0.233182725971488	0.0725872226805748	0.0725871737487914
0.023911384066264	0.00744334486970555	0.00744334633948921
−0.262174140963142	−0.081611917007427	−0.0816118768800168
0.293961466245146	0.0915069620785829	0.0915069156040101
−0.0942391848805049	−0.0293356250805537	−0.0293356039061921
−0.179701347309583	−0.0559390273591851	−0.0559390189204992
0.312118179853314	0.0971589176717661	0.0971588974130924
−0.198726388275092	−0.0618613171505968	−0.0618613013210836
−0.0711724101815732	−0.0221551636978665	−0.0221551750132718
0.285019604458431	0.0887233999364359	0.0887234140712706
−0.274400013267919	−0.0854176408876918	−0.0854176541455501
0.0476770313276963	0.0148413315560991	0.0148413264421363
0.216594059431699	0.0674232726721943	0.0674233074487995
−0.31028707179606	−0.0965888456434106	−0.096588893961289
0.159614253712092	0.0496861078370457	0.0496861314178133
0.116762423422735	0.0363468076054103	0.0363468360746085
−0.301183271250826	−0.0937549160489609	−0.0937549827047919
0.248408179143949	0.0773266315474941	0.0773266869791526

λ_{16} analítico $= 0.290609546465254$

λ_{16} encontrado pelo algoritmo $= 0.290609546465799$

Erro $\lambda_{16} = 5.45175016242183 \cdot 10^{-13}$

Autovetor associado à λ_{16}

V	A x V	λ_{16} x V
0.116761285546196	0.0339319668844449	0.0339319442373438
-0.285019080264018	-0.0828293186617515	-0.0828292656496254
0.293962906550184	0.0854284760560702	0.0854284269503169
-0.138593589801117	-0.0402766357762923	-0.0402766202751695
-0.0942437538879432	-0.027388157807537	-0.0273881345746095
0.274402662453829	0.0797440740491608	0.0797440332847149
-0.301181876550325	-0.0875263565479705	-0.087526328548008
0.159613100216923	0.0463850894052236	0.0463850906640402
0.0711725175004587	0.0206834351414937	0.0206834130316374
-0.262175551156911	-0.0761907366226942	-0.0761907180161306
0.306633582761693	0.0891106427700283	0.0891106464175584
-0.179696722818436	-0.0522215605989422	-0.052221583119656
-0.0476826141052274	-0.0138570411494767	-0.0138570228594238
0.248409255714769	0.0721900924052163	0.0721901011411757
-0.310285975402553	-0.0901720297548599	-0.0901720665864339
0.198726656131896	0.0577518234876173	0.0577518634091551
0.023911845945917	0.0069490205981978	0.00694901070550298
-0.233184571837042	-0.0677656282371387	-0.0677656626643841
0.312117602723145	0.0907042947645666	0.0907043549713655
-0.216591064678313	-0.0629433849568731	-0.062943431074709

λ_{17} analítico $= 0.275038189486704$

λ_{17} encontrado pelo algoritmo $= 0.27503818948696$

Erro $\lambda_{17} = 2.55351295663786 \cdot 10^{-13}$

Autovetor associado à λ_{17}

V	A x V	λ_{17} x V
0.0942421615090161	0.0259201999334122	0.0259201934747774
−0.248408942720729	−0.0683219615756041	−0.0683219458582793
0.312119578103385	0.085844819636109	0.0858448036649887
−0.262174393453223	−0.0721079772555639	−0.0721079705052163
0.116760609045676	0.0321136193059881	0.0321136265153174
0.0711704379373787	0.019574606821863	0.0195745883952907
−0.233185155299392	−0.0641348435996409	−0.0641348229287804
0.310287269717344	0.0853408612782487	0.0853408488839103
−0.274400824033601	−0.0754707035612067	−0.0754707058359315
0.138593739725789	0.0381185556329381	0.038118571248408
0.0476808499181907	0.0131140751012959	0.0131140546346986
−0.216592616533633	−0.0595712553485395	−0.0595712411076537
0.306633890933957	0.084336030735259	0.0843360301978174
−0.285017098753179	−0.0783905741149002	−0.0783905868139004
0.159613970258972	0.0438999197881204	0.043899937396853
0.023910979201185	0.00657644343216901	0.00657643242835429
−0.19872882311372	−0.0546580121249673	−0.0546580157080717
0.301181349865067	0.0828363554316161	0.0828363731741266
−0.293961540528931	−0.0808506268768674	−0.0808506498858746
0.179698489564027	0.0494239313435796	0.0494239472232312

λ_{18} analítico $= 0.263690054997803$
λ_{18} encontrado pelo algoritmo $= 0.263690054997603$
Erro $\lambda_{18} = 1.99396055222678 \cdot 10^{-13}$

Autovetor associado à λ_{18}

V	A x V	λ_{18} x V
-0.0711692782448202	-0.0187666323167006	-0.0187666308945164
0.198728169022119	0.0524026459281197	0.0524026418190155
-0.28501734170763	-0.0751562448491789	-0.0751562385101558
0.312117187071237	0.0823022060811525	0.0823021982245116
-0.274399888097561	-0.0723565300597526	-0.0723565215837821
0.179698145228746	0.0473846218969047	0.0473846137983354
-0.0476785696429895	-0.0125723713751853	-0.012572364651367
-0.0942424599751442	-0.0248507950042864	-0.0248507994539552
0.216592327710831	0.0571132413417585	0.0571132428061278
-0.293962728526333	-0.0775150500230279	-0.0775150480523543
0.310286186540251	0.0818193871385184	0.0818193815937954
-0.262172774450026	-0.0691323622401867	-0.0691323533136017
0.159614100866279	0.0420886628311354	0.0420886510358222
-0.0239083688366873	-0.00630441296382189	-0.00630439909344906
-0.116762594100705	-0.0307891199220913	-0.0307891348600777
0.233185821484755	0.061488767220343	0.0614887820919765
-0.301183830262764	-0.0794191671219777	-0.0794191807663772
0.30663553136851	0.0808567287984666	0.0808567401307817
-0.248408741688497	-0.0655029066495998	-0.0655029147577251
0.138595305831261	0.0365461995908308	0.036546203817055

λ_{19} analítico $= 0.251473581905183$

λ_{19} encontrado pelo algoritmo $= 0.25147358190581$

Erro $\lambda_{19} = 6.26332319342282 \cdot 10^{-13}$

Autovetor associado à λ_{19}

V	A x V	λ_{19} x V
0.0239093340092529	0.00601256245188964	0.00601256586428922
−0.0711676401699048	−0.0178967715573641	−0.0178967813893097
0.11675798892571	0.0293615346032876	0.0293615496912672
−0.159611862324354	−0.0401381481617715	−0.0401381667333622
0.198724855787588	0.0499740313975252	0.0499740512986202
−0.233180214619591	−0.0586386448307668	−0.0586386637999541
0.262170320296935	0.0659288935605318	0.0659289095144637
−0.285015623082641	−0.0716738883451029	−0.071673899635708
0.301180576768853	0.0757389528318999	0.075738958440521
−0.310286201075963	−0.0780287827599455	−0.0780287824005187
0.31211897586141	0.0784896827241681	0.0784896768406418
−0.306635857040146	−0.0771108276531258	−0.0771108173106433
0.293965295028941	0.0739245190097257	0.0739245056969259
−0.27440423049153	−0.0690054293563618	−0.0690054147318125
0.248411136970072	0.0624688527690767	0.0624688383991586
−0.216595273362654	−0.0544680020755557	−0.0544679892163746
0.179702398839914	0.0451904164424684	0.0451904059133396
−0.1385972864044	−0.0348535638794232	−0.0348535560545398
0.0942434467118264	0.023699742203086	0.0236997371157723
−0.04768053905555	−0.011990398426232	−0.011990395943499

λ_{20} analítico $= 0.255964433042702$

λ_{20} encontrado pelo algoritmo $= 0.255964433042276$

Erro $\lambda_{20} = 4.25715018792516 \cdot 10^{-13}$

Autovetor associado à λ_{20}

V	A x V	λ_{20} x V
−0.0476802107958094	−0.0122044388970703	−0.0122044381236856
0.138596481853102	0.0354757718987395	0.0354757698991834
−0.216594533006516	−0.0554404991585535	−0.0554404968410694
0.274404379793089	0.0702377627906703	0.0702377614980555
−0.306637744149665	−0.0784883550531967	−0.0784883563306316
0.310290283788778	0.0794232712526028	0.0794232765685215
−0.285021617347147	−0.072955386230376	−0.0729553966890551
0.233187044825632	0.0596875736337914	0.0596875897215968
−0.159618007262796	−0.0408565113276711	−0.0408565327323594
0.0711717495085674	0.0182174109736608	0.0182174365115874
0.0239078375680409	0.00611958376642585	0.00611955608837043
−0.116758619246151	−0.0298860810088524	−0.02988605377814
0.198726240532413	0.0508668734620226	0.0508668494885021
−0.262170777446563	−0.0671064125995164	−0.0671063944093624
0.301178828384067	0.0770910787855074	0.0770910680516647
−0.312114672754947	−0.0798902582135345	−0.0798902552558957
0.293959126223064	0.0752430775423703	0.0752430810812895
−0.248404512704061	−0.0635827129247889	−0.0635827202594379
0.179696907248155	0.0459960093121134	0.0459960169832246
−0.0942403607103932	−0.0241221756991399	−0.0241221804989354

Novamente, para testar a ortogonalidade da matriz de autovetores:

$$V \times V^t = \begin{bmatrix} 1.0 & 3.0 \cdot 10^{-16} & 4.2 \cdot 10^{-16} & 1.1 \cdot 10^{-16} & -8.4 \cdot 10^{-17} & \cdots \\ 3.0 \cdot 10^{-16} & 1.0 & 4.0 \cdot 10^{-16} & 1.5 \cdot 10^{-16} & 6.2 \cdot 10^{-16} & \cdots \\ 4.2 \cdot 10^{-16} & 4.0 \cdot 10^{-16} & 1.0 & 6.7 \cdot 10^{-16} & 5.6 \cdot 10^{-16} & \cdots \\ 1.1 \cdot 10^{-16} & 1.5 \cdot 10^{-16} & 6.7 \cdot 10^{-16} & 1.0 & -2.8 \cdot 10^{-17} & \cdots \\ -8.4 \cdot 10^{-17} & 6.2 \cdot 10^{-16} & 5.6 \cdot 10^{-16} & -2.8 \cdot 10^{-17} & 1.0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

OBS : A matriz 20x20 não cabe na folha.

2.1.4 Análise dos Testes

Nessa seção serão feitas algumas análises sobre a primeira tarefa do EP2, comentando a eficiência do método, tendo como base a comparação do erro entre os valores, utilizando autovalores obtidos do algoritmo e autovalores obtidos analiticamente.

Teste a

A partir da visualização dos autovetores encontrados da matriz A de dimensão 4x4, foi possível comparar o resultado de sua multiplicação pelos autovalores encontrados pelo algoritmo QR com a multiplicação dos autovetores encontrados pelos autovalores analíticos. Percebeu-se que o algoritmo cumpriu bem o seu papel, já que o erro foi igual a 0 para λ_1 e

λ_4 e na ordem de 10^{-14} para λ_2 e λ_3.

Teste b

A partir da visualização dos autovetores encontrados da matriz A, considerando n = 20, foi possível comparar o resultado de sua multiplicação pelos autovalores encontrados pelo algoritmo QR com a multiplicação dos autovetores encontrados pelos autovalores analíticos. Percebeu-se que o algoritmo também cumpriu bem o seu papel, já que o erros foram na odem de 10^{-13} ou menos.

Conclui-se que tanto a implementação das transformações de Householder quanto o uso do algorítmo QR para a determinação de autovalores e autovetores foi eficiente para ambos os testes. O programa conseguiu chegar nos resultados esperados com uma margem de erro aceitável.

2.2 Aplicação em Treliças Planas

Nessa tarefa, utiliza-se o programa para o cálculo das frequências e dos modos de vibração de treliças planas. As frequências são as raízes dos autovalores da matriz $\tilde{K}$ e os modos de vibração são os autovetores da matriz $\tilde{K}$. A matriz $\tilde{K}$ é obtida a partir dos passos que estão descritos abaixo.

Para a obtenção dos valores de frequências e modos de vibração , parte-se de uma equação diferencial para o movimento da treliça, vista a seguir:

$$M\ddot{x} + Kx = 0 \tag{7}$$

Os valores da matriz K representam a rigidez total do sistema de barras, visto a seguir:

$$K^{\{i,j\}} = \frac{AE}{L_{\{i,j\}}} \cdot \begin{pmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{pmatrix} \tag{8}$$

L representa o comprimento da barra, A representa as áreas de seções transversais e E representa o módulo de elasticidade. Os elementos C e S que aparecem na montagem dos valores da matriz K são cossenos e senos do ângulo que a barra sem deformação forma com o eixo horizontal, como são vistos a seguir:

$$C = cos\theta_{\{i,j\}}, S = sen\theta_{\{i,j\}} \tag{9}$$

Para a montagem da matriz K, determina-se sua posição através do seguinte padrão, sendo i e j o número dos nós da treliça:

$$\begin{matrix} (2i-1, 2i-1) & (2i-1, 2i) & (2i-1, 2j-1) & (2i-1, 2j) \\ (2i, 2i-1) & (2i, 2i) & (2i, 2j-1) & (2i, 2j) \\ (2j-1, 2i-1) & (2j-1, 2i) & (2j-1, 2j-1) & (2j-1, 2j) \\ (2j, 2i-1) & (2j, 2i) & (2j, 2j-1) & (2j, 2j) \end{matrix}$$

Já a matriz M diagonal representa a massa de cada nó da treliça expressa em cada elemento da diagonal principal, lembrando que foi considerada uma barra ideal, em que metade da massa vai para cada extremo. No programa, essa matriz foi representada por um único vetor contendo os valores da diagonal principal, a fim de economizar no espaço e número de contas, otimizando o algoritmo. Suas entradas, portanto, são:

$$M_{2i-1,2i-1} = m_i \tag{10}$$

$$M_{2i,2i} = m_i \tag{11}$$

Por fim, o termo x presente na equação diferencial representa o deslocamento dos nós da treliça.

Como solução geral dessa equação, temos a expressão a seguir, que são os deslocamentos feitos pelos nós da treliça:

$$x(t) = ze^{i\omega t} \tag{12}$$

O termo z expressa os modos de vibração e ele se relaciona com o termo y, que são os autovetores procurados da seguinte maneira:

$$z = M^{-\frac{1}{2}}y \tag{13}$$

A partir do uso do termo y e do uso de $\tilde{K}$ sendo uma matriz simétrica positiva, é possível chegar em uma equação, cujo modelo é conhecido e fornece os autovalores representados pelo w e autovetores representados pelo y, vista ambas a seguir:

$$\tilde{K} = M^{-\frac{1}{2}}KM^{-\frac{1}{2}} \tag{14}$$

$$\tilde{K}y = \omega^2 y \tag{15}$$

Obtida essa equação, é aplicado o algoritmo utilizado no EP1 para resolver e encontrar os autovalores e autovetores da matriz desejada.

2.2.1 O algoritmo

Novamente, iniciando pelas funções auxiliares utilizadas.

```python
def contribuicaoK(K,i,j, angulo, tamanho, AE):
    # Calula as contribuições de cada barra na matriz de ridigez global

    C = np.cos(angulo*np.pi/180)
    S = np.sin(angulo*np.pi/180)
    C2 = C*C
    S2 = S*S
    CS = C*S

    valores = [C2*AE/tamanho,CS*AE/tamanho, -C2*AE/tamanho, -CS*AE/tamanho,S2*AE/tamanho,  -S2*AE/tamanho]
    posicoes = {
      0  : [(2*(i-1), 2*(i-1)), (2*(j-1), 2*(j-1))],
      1  : [(2*(i-1), 2*i-1), (2*i -1, 2*(i-1)), (2*j -1, 2*(j-1)), (2*(j-1), 2*j -1)],
      2 : [(2*(j-1), 2*(i-1)), (2*(i-1), 2*(j-1))],
      3 : [(2*j -1, 2*(i-1)), (2*(j-1), 2*i -1), (2*i -1, 2*(j-1)), (2*(i-1), 2*j -1)],
      4 : [(2*i -1, 2*i -1),(2*j -1, 2*j -1)],
      5 : [(2*j -1, 2*i -1), (2*i -1, 2*j-1)]
    }

    for valor in posicoes:
        for indice in posicoes[valor]:
          try:
            K[indice[0], indice[1]] += valores[valor]
          except IndexError:
            #As colunas e linhas referentes aos nos 13 e 14 não entram na matriz K
            pass

    return K
```

Figura 4: Cálculo das contribuições na matriz K de cada barra

A função "contribuicaoK" aplica a equação (8) para formar a matriz de rigidez global K. Como os nós fixos podem ser desconsiderados em K, os erros de indexação gerados por esses nós foram apenas ignorados.

```python
def ler_arquivoT2(filename):
    # Le os arquivos de input do segundo teste
    #Retorna a matriz K e a diagonal da matriz de Massas

    with open(filename, 'r') as arquivo:
        n_nos, n_nos_livres, n_barras = map(int, arquivo.readline().strip().split(' '))
        densidade, area, elasticidade = map(float, arquivo.readline().strip().split(' '))
        AE = area*elasticidade*(10**9)
        K = np.zeros((2*n_nos_livres, 2*n_nos_livres))
        M = [0]*(n_nos_livres)
        for linha in arquivo:
            i, j , angulo, tamanho = map(float, linha.strip().split(' '))
            i, j = int(i), int(j)
            K = contribuicaoK(K, i, j, angulo, tamanho, AE)
            massa = tamanho*area*densidade/2
            if i<= n_nos_livres:
                M[i-1] += massa
            if j<= n_nos_livres:
                M[j-1] += massa
    return K, M
```

Figura 5: Rotina para ler o arquivo de input e montar as matrizes K e M

A função "ler_ArquivoT2" pega os parâmetros do input e monta matriz de rigidez global K e a matriz de massas M, representada somente por sua diagonal em um vetor.

```python
def raiz_M(M):
    #Calcula M^(-1/2)

    M_raiz = []
    for elem in M:
        M_raiz += [1/np.sqrt(elem)]*2
    return M_raiz

def monta_K_til(K,M_raiz):
    #Calcula a matriz K~

    #Multiplica a direita por M^(-1/2)
    for i in range(len(K)):
        for j in range(len(K)):
            K[i,j] = K[i,j]*M_raiz[j]

    #Multiplica a esquera por M^(-1/2)
    for i in range(len(K)):
        for j in range(len(K)):
            K[i,j] = M_raiz[i]*K[i,j]

    return K
```

Figura 6: Calcula $M^{-\frac{1}{2}}$ e $\tilde{K}$

A função "raiz_M" calcula o valor de $M^{-\frac{1}{2}}$ e com este valor calcula-se $\tilde{K}$ na função "monta_K_til" conforme a equação (14).

```python
def trelicas(filename, printa=True):
  #Calcula as 5 menores frequências de vibração da treliça e seus respectivos modos de vibração

  K, M = ler_arquivoT2(filename)
  K_original = K.copy()
  M_raiz = raiz_M(M)
  K = monta_K_til(K, M_raiz)
  autovalores,  modos_de_vibracao = calcula_valores(K)

  # w = raiz(autovalor)
  frequencias = [np.sqrt(autovalor) for autovalor in autovalores]

  # Z = M^(-1/2)*y
  for i in range(len( modos_de_vibracao)):
    for j in range(len( modos_de_vibracao)):
      modos_de_vibracao[i,j] = M_raiz[i]* modos_de_vibracao[i,j]

  #Ordena os menores w mantendo seus indices
  indices = {i : frequencias[i] for i in range(len(frequencias))}
  indices = sorted(indices.items(), key = operator.itemgetter(1))


  menores_frequencias = []
  menores_modos = np.zeros((24,5))
  for i in range(5):
    menores_frequencias.append(indices[i][1])
    menores_modos[:,i] = modos_de_vibracao[:, indices[i][0]]

    if printa:
      print(f"w{i+1} = {indices[i][1]} (rad/s)")
      print(f'Modos de vibração: {modos_de_vibracao[:, indices[i][0]]}')
      print('---------------------------------------------------------')

  return menores_frequencias, menores_modos
```

Figura 7: Calculo das menores frequências

Para finalizar, a função "trelicas" organiza o fluxo do algoritmo. Ela começa montando as matrizes $\tilde{K}$ e $M^{-\frac{1}{2}}$ a partir do arquivo de input. Posteriormente, é calculado os autovalores e autovetores de $\tilde{K}$ de acordo com (15). Com os autovalores, obtem-se as frequências pela relação $\omega_i = \sqrt{\lambda_i}$. Os modos de vibração são obtidos por (13). Ao final, imprime-se as 5 menores frequências angulares e seus respectivos modos de vibração.

2.2.2 Resultados

As 5 menores frequências angulares e seus modos de vibração são apresentados a seguir:

$\omega\,[rad/s]$	24.59255	92.01244	94.70337
Modo de vibração	$\begin{bmatrix} 0.003495723 \\ -0.0006120484 \\ 0.003495723 \\ 0.0006120484 \\ 0.002269078 \\ -0.001813025 \\ 0.002248355 \\ -0.0005955082 \\ 0.002248355 \\ 0.0005955082 \\ 0.002269078 \\ 0.001813025 \\ 0.0009983716 \\ -0.001817312 \\ 0.0009960167 \\ -0.0005071916 \\ 0.0009960167 \\ 0.0005071916 \\ 0.0009983716 \\ 0.001817312 \\ 4.181565 \cdot 10^{-5} \\ -0.0002960973 \\ 4.181565 \cdot 10^{-5} \\ 0.0002960973 \end{bmatrix}$	$\begin{bmatrix} -6.261856 \cdot 10^{-6} \\ -0.002186166 \\ 6.261856 \cdot 10^{-6} \\ -0.002186166 \\ 0.0004729349 \\ -0.002985179 \\ 0.0001760825 \\ -0.002050769 \\ -0.0001760825 \\ -0.002050769 \\ -0.0004729349 \\ -0.002985179 \\ -4.166106 \cdot 10^{-6} \\ -0.003087111 \\ -4.028547 \cdot 10^{-6} \\ -0.001743989 \\ 4.028547 \cdot 10^{-6} \\ -0.001743989 \\ 4.166106 \cdot 10^{-6} \\ -0.003087111 \\ 4.211796 \cdot 10^{-5} \\ -0.001097249 \\ -4.211796 \cdot 10^{-5} \\ -0.001097249 \end{bmatrix}$	$\begin{bmatrix} -0.0007788999 \\ 0.0008083593 \\ -0.0007788999 \\ -0.0008083593 \\ 0.000640509 \\ 0.002837993 \\ 0.000874589 \\ 0.0007135821 \\ 0.000874589 \\ -0.0007135821 \\ 0.000640509 \\ -0.002837993 \\ 0.002484207 \\ 0.002940858 \\ 0.002397314 \\ 0.0003109346 \\ 0.002397314 \\ -0.0003109346 \\ 0.002484207 \\ -0.002940858 \\ 0.001156625 \\ -5.634362 \cdot 10^{-5} \\ 0.001156625 \\ 5.634362 \cdot 10^{-5} \end{bmatrix}$

$\omega\,[rad/s]$	142.8097	150.8221
Modo de vibração	$\begin{bmatrix} 0.003868795 \\ -0.001930401 \\ 0.003868795 \\ 0.001930401 \\ -0.001488139 \\ 0.002680953 \\ -0.0006903867 \\ -0.001142977 \\ -0.0006903867 \\ 0.001142977 \\ -0.001488139 \\ -0.002680953 \\ -0.0005565432 \\ 0.00291262 \\ -0.0005122764 \\ 2.02225 \cdot 10^{-5} \\ -0.0005122764 \\ -2.02225 \cdot 10^{-5} \\ -0.0005565432 \\ -0.00291262 \\ -0.000231133 \\ 4.474872 \cdot 10^{-5} \\ -0.000231133 \\ -4.474872 \cdot 10^{-5} \end{bmatrix}$	$\begin{bmatrix} -8.982146 \cdot 10^{-5} \\ 0.001922543 \\ 8.982146 \cdot 10^{-5} \\ 0.001922543 \\ 0.001505122 \\ -0.003387696 \\ 0.0004343333 \\ 0.001797423 \\ -0.0004343333 \\ 0.001797423 \\ -0.001505122 \\ -0.003387696 \\ -0.0006892737 \\ -0.003717491 \\ -0.0006281251 \\ 0.001129479 \\ 0.0006281251 \\ 0.001129479 \\ 0.0006892737 \\ -0.003717491 \\ 5.00037 \cdot 10^{-5} \\ 0.0006787439 \\ -5.00037 \cdot 10^{-5} \\ 0.0006787439 \end{bmatrix}$

2.2.3 Tarefa Bônus

A tarefa bônus pedia a montagem de um GIF animado da treliça vibrando nas menores frequências. A fim de realizar essa tarefa, precisa-se inicialmente calcular os deslocamentos dos nós das barras. A partir da equação (12) e da fórmula de Euler ($e^{ix} = cos(x) + isin(x)$), chega-se que a parte real, que nos interessa, dos deslocamentos vale:

$$[\mathbf{X}(t)] = cos(\omega t)[\mathbf{Z}] \tag{16}$$

Tendo calculado os deslocamentos, precisa-se definir as coordenadas iniciais dos nós da treliça. Para centralizá-la na imagem do GIF utiliza-se os seguintes pontos:

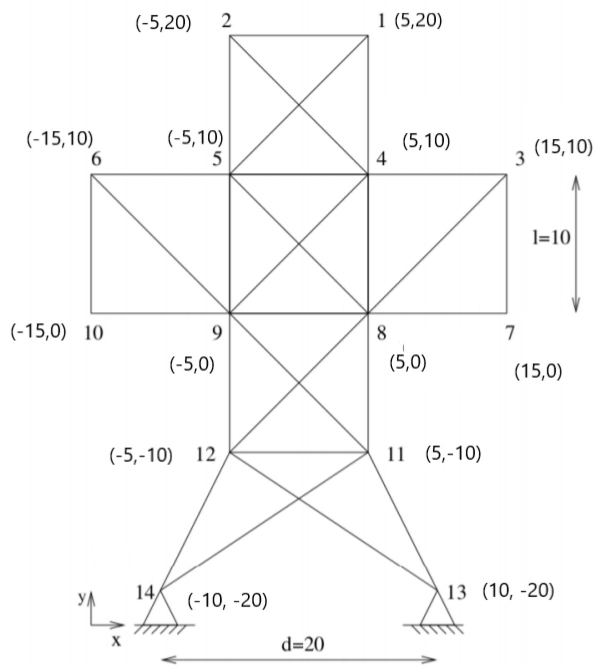

Figura 8: Nós da treliça mapeados no eixo cartesiano

Agora a tarefa ficou simples, para cada valor de t no intervalo desejado, calcula-se $zcos(\omega t)$ e adiciona-se este valor à coordenada do nó.

Para finalizar, vale ressaltar que utiliza-se nesta tarefa a biblioteca OpenGL, que dada a coordenada de dois pontos ela desenha uma reta entre eles. Deste modo, não foi necessário calcular a deformação das barras.

3 Conclusões e Comentários Finais

Nesse segundo Exercício Programa foi possível aprimorar uma série de habilidades como trabalho em equipe, criatividade, raciocínio lógico, entre outras. Foi possível desenvolver um algoritmo iterativo através de uma linguagem de programação e vivenciar na prática a aplicação desse método na rodagem do programa, saindo da teoria e, com uso de métodos numéricos, chegar nos resultados esperados. Conseguiu-se relacionar a mecânica das treliças planas com o algoritmo QR, utilizando conceitos aprendidos em álgebra linear e aplicando-os em Python, descobrindo autovalores e autovetores de matrizes. Além disso, foi possível construir um gif (novo aprendizado em python), representando a treliça vibrando, resultado dos deslocamentos horizontais e verticais de cada nó móvel.

4 Referências Bibliográficas

- Enunciado do Exercício Programa 2