



Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря  
Сікорського» Інститут прикладного системного аналізу

**Лабораторна робота № 1**  
з курсу «Спеціальні розділи обчислювальної математики»  
з теми «Методи розв'язування нелінійних алгебраїчних рівнянь»

Варіант № 14

Виконав студент 3 курсу групи КА-91  
Панченко Єгор Станіславович  
перевірила старший викладач  
Хоменко Ольга Володимирівна

Київ-2021

**Завдання:** знайти найбільше власне число та відповідний власний вектор степеневим методом або методом скалярних добутків з точністю  $\varepsilon = 0,001$ . Також знайти всі власні числа матриці методом Якобі,  $LU$ ,  $QR$ .

## 1. Теоретичні відомості

Використаємо метод скалярних добутків у першому завданні та  $QR$  алгоритм у другому.

## 2. Текст обох програм

FileEditViewInsertCellKernelHelp

TrustedPython 3

RunCode

```
Panchenko Yehor, variant 14
```

```
In [25]: import numpy as np
np.set_printoptions(suppress=True)
```

The input matrix is A

```
In [26]: n = 5
A = np.array([[ 894, 207, -248, -269, -281],
               [ 207, 646, -42, -42, 464],
               [-248, -42, 970, 225, -15],
               [-269, -42, 225, 174, -5],
               [-281, 464, -15, -5, 917]])
A
```

```
Out[26]: array([[ 894, 207, -248, -269, -281],
                 [ 207, 646, -42, -42, 464],
                 [-248, -42, 970, 225, -15],
                 [-269, -42, 225, 174, -5],
                 [-281, 464, -15, -5, 917]])
```

### Task 1

Use dot product method to find the biggest eigenvalue of the matrix A

```
In [27]: y = list([np.ones(5)])
eps = 1e-5
s = list([np.dot(y[0], y[0])])
x = list([y / np.sqrt(s)])
t = list([0])
eigenvalues = list([0])

step = 1000
eps = 1e-4

for k in range(1, step):
    #print(A.shape, x[k-1].T.shape)
    y.append((np.matmul(A, x[k-1].T).T).flatten())
    #print('y = ', y)
    #print(y[k].shape)
    s.append(np.dot(y[k], y[k]))
    t.append(np.dot(y[k], x[k-1].T))
    x.append(y[k] / np.sqrt(s[k]))
    eigenvalues.append(s[k] / t[k])
    if abs(eigenvalues[-2] - eigenvalues[-1]) <= eps:
        break;

print('eigenvalue = ', eigenvalues[-1])
print('eigenvector = ', x[-1])

print('bias = ', np.matmul(A, x[-1].T) - eigenvalues[-1] * x[-1])

eigenvalue = 1366.46306365909
eigenvector = [-0.66678069  0.05519263  0.53424171  0.24734458  0.45361188]
bias = [-0.03845347 -0.13856489  0.08476505  0.03296986 -0.15738077]
```

Print intermediate results

```
In [28]: for yk, lk in zip(y[:7], eigenvalues[:7]):
          print('y, lambda = ', yk, lk)

y, lambda = [1. 1. 1. 1. 1.] 0
y, lambda = [135.50571944 551.41436325 398.02009999 37.11872843 482.99068314]
[996.79214266]
y, lambda = [-10.79292299 697.61861151 390.62884231 40.18082465 773.78923991]
1147.437592531269
y, lambda = [-170.96708761 708.95006252 314.09076945 58.03407276 925.2716
0936] 1240.4623724503488
y, lambda = [-294.59604888 685.11675499 259.26380054 75.66147116 999.8494
9263] 1285.1755677795443
y, lambda = [-381.71039878 651.49270825 233.33749017 91.65262967 1029.0074
3462] 1304.6005974552088
y, lambda = [-444.42121201 619.39989259 229.74022453 106.57316296 1037.2749
8685] 1313.9407765467097
```

## Task 2

Use QR-algorithm to find all eigenvalues of the matrix A

```
In [29]: List_A = [A]
          List_Q = []
          List_R = []

          step = 100
          for k in range(step):
              Q, R = np.linalg.qr(List_A[-1])
              List_Q.append(Q)
              List_R.append(R)
              List_A.append(np.matmul(R, Q))

          print('eigenvalues = ', np.diagonal(List_A[-1]))

eigenvalues = [1366.46364021 1266.16093029 759.32940591 177.14309036 31.90
293322]
```

Print intermediate results

```
In [30]: for a in List_A[:7]:
          print(a)

[[ 894  207 -248 -269 -281]
 [ 207  646 -42  -42  464]
 [-248  -42  970  225  -15]
 [-269  -42  225  174  -5]
 [-281  464 -15  -5  917]]
[[1257.15416656 -76.10134608 -213.3302318 41.1768058 29.63350668]
 [-76.10134608 1215.18886253 -97.11937374 -155.33920023 -70.55751836]
 [-213.3302318 -97.11937374 887.90325613 44.95636941 15.90073699]
 [41.1768058 -155.33920023 44.95636941 189.42645395 54.66347638]
 [29.63350668 -70.55751836 15.90073699 54.66347638 51.32726084]]
[[1326.18760647 -47.37863709 -136.09667485 6.76293005 -0.77702635]
 [-47.37863709 1253.25760281 -87.38102684 -24.0332383 1.8425771]
 [-136.09667485 -87.38102684 811.8776396 7.41391008 -0.52984172]
 [6.76293005 -24.0332383 7.41391008 177.2894755 -8.39241434]
 [-0.77702635 1.8425771 -0.52984172 -8.39241434 32.38767562]]
[[1351.10161239 -29.09219469 -80.01498181 0.90261547 0.01861246]
 [-29.09219469 1262.52705829 -60.07532721 -3.34356977 -0.04569765]
 [-80.01498181 -60.07532721 778.31222597 1.27344808 0.01776458]
 [0.90261547 -3.34356977 1.27344808 177.14053015 1.50714482]
 [0.01861246 -0.04569765 0.01776458 1.50714482 31.9185732]]
[[1359.59550059 -21.21361059 -45.40539959 0.11812678 -0.00043864]
 [-21.21361059 1266.42881417 -38.4308133 -0.46400355 0.00113591]
 [-45.40539959 -38.4308133 765.92935552 0.25217422 -0.00066042]
 [0.11812678 -0.46400355 0.25217422 177.14288925 -0.27142628]
 [-0.00043864 0.00113591 -0.00066042 -0.27142628 31.90344047]]
[[1362.56135572 -17.64258212 -25.44405644 0.01538049 0.00001029]
 [-17.64258212 1267.79140665 -23.9300491 -0.06441437 -0.00002829]
 [-25.44405644 -23.9300491 761.60120514 0.05467952 0.00002624]
 [0.01538049 -0.06441437 0.05467952 177.14308281 0.04888307]
 [0.00001029 -0.00002829 0.00002624 0.04888307 31.90294968]]
[[1363.73087754 -15.6819675 -14.19556003 0.00199906 -0.00000024]
```

## 3. Висновок

У ході виконання лабораторної роботи було реалізовано метод скалярних добутків та  $QR$  алгоритм. Я навчився застосовувати ці методи, зрозумів теорію, яка за цим стоїть.