University of Oslo
Department of Mathematics

Author: Jonas Semprini Næss

Fall, 2023
Mandatory assingment: 1

MAT4110: Introduction to Numerical Analysis

Problem 1.

a.)

```
1
2  import numpy as np
3  import pandas as pd
4  import random
5
6  matrix = np.array(
7      [
8          [5, 1 / np.sqrt(2), -1 / np.sqrt(2)],
9          [1 / np.sqrt(2), 5 / 2, 7 / 2],
10         [-1 / np.sqrt(2), 7 / 2, 5 / 2],
11     ]
12 )
13
14 result = pd.DataFrame(columns=["Approx Eigenvalue", "Actual Eigenvalue", "Error"])
15
16 num_iterations = 10
17 actual_eigenvalues, _ = np.linalg.eig(matrix)
18 print(actual_eigenvalues)
19 actual_largest_eigenvalue = max(actual_eigenvalues)
20
21 n = matrix.shape[0]
22 b = np.random.rand(n)
23
24 for i in range(num_iterations):
25     # Power iteration
26     b = np.dot(matrix, b)
27     # Normalize the vector
28     eigenvalue = np.linalg.norm(b)
29     error = abs(eigenvalue - actual_largest_eigenvalue)
30     b /= eigenvalue
31     result.loc[i] = [eigenvalue, actual_largest_eigenvalue, error]
32
33
34 with pd.option_context(
35     "display.max_rows",
36     None,
37     "display.max_columns",
38     None,
39     "display.precision",
40     7,
41 ):
42     print(result)
```

```
43
44
45 # def inverse_power_method(A, mu, iter, tol=1e-15):
46 #     Ashift = A - mu * np.identity(A.shape[0])
47 #     b = np.zeros((len(A), iter + 1))
48 #     b[:, 0] = np.random.rand(A.shape[0])
49 #     print(b, b[0])
50 #     rn = np.ones((iter + 1,))
51 #     for k in range(num_iterations):
52 #         b[:, k] = b[:, k] / np.linalg.norm(b[:, k])
53 #         b[:, k + 1] = np.linalg.solve(Ashift, b[:, k])
54 #         rn[k + 1] = np.sum(b[:, k + 1]) / np.sum(b[:, k])
55 #         if abs(rn[k + 1] - rn[k]) < tol:
56 #             break
57 #     if k < iter:
58 #         rn[k + 2 :] = rn[k + 1]
59 #     return (
60 #         1.0 / rn[k + 1] + mu,
61 #         1.0 / rn + mu,
62 #         b[:, k + 1] / np.linalg.norm(b[:, k + 1]),
63 #     )
64
65
66 # lamda, v = np.linalg.eig(matrix)
67 # order = np.abs(lamda).argsort()
68 # lamda = lamda[order]
69 # mu = 2
70 # lamda_shift, lamda_seq, vpm = inverse_power_method(matrix, mu, iter=num_iterations
    )
71
72 # print(
73 #     "The eigenvalue closest to {} from the shifted power method is {} (exact is
    {}, error is {})".format(
74 #         mu, lamda_shift, lamda[1], abs(lamda_shift - lamda[1])
75 #     )
76 # )
```

Listing 1: Python example

Problem 2.

a.)

b.)

c.)

Problem 3.

a.)

b.)

c.)

d.)