静岡理工科大学　情報学部　幸谷担当講義

レポート用紙

講義名　：数値解析１	年月日：　2024 年　　6 月　　17 日(月)
学籍番号：99999999	氏名：幸谷　智紀

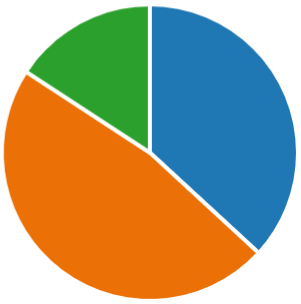
本日の課題 問題 5.1, 5.2

感想

2. 課題の難易度はどうでしたか？

詳細

難しかった	7
少し難しかった	9
ちょうど良かった	3
簡単だった	0
とても簡単だった	0



問題 5.1

プログラム例：

```
# clinsolve.py: 複素連立一次方程式を高速に解く
import numpy as np # NumPy
import scipy.linalg as sclinalg # Scipy.linalg
import time # time 関数

# 乱数 seed の指定
rng = np.random.default_rng(seed=20240521)

# 正方行列の次数
input_str = input('Input size of square matrix > ')
n = int(input_str)

# 乱数行列生成
mat_a = rng.random((n, n)) + rng.random((n, n)) * 1j # n x n 行列
print('||mat_a||_2 = ', np.linalg.norm(mat_a))
# 解を生成
true_x = rng.random((n, 1)) + rng.random((n, 1)) * 1j # n 次元ベクトル
print('||x||_2 = ', np.linalg.norm(true_x))
```

```
# 定数ベクトルｂを生成
b = mat_a @ true_x

# (1)逆行列を求めて連立一次方程式を解く
start_time = time.time()
inv_x = sclinalg.inv(mat_a) @ b
end_time_inv = time.time() - start_time
relerr_inv = np.linalg.norm(inv_x - true_x) / np.linalg.norm(true_x)

# (2) solve 関数を用いて連立一次方程式を解く
start_time = time.time()
solve_x = sclinalg.solve(mat_a, b)
end_time_solve = time.time() - start_time
relerr_solve = np.linalg.norm(solve_x - true_x) / np.linalg.norm(true_x)

# ノルム相対誤差と計算時間の表示
print( '                        inv         solve')
print(f'Computational        time        (s):        {end_time_inv:10.2g},
{end_time_solve:10.2g}')
print(f'Norm2 Relative error  : {relerr_inv:10.2e}, {relerr_solve:10.2e}')
```

実行結果：

```
Input size of square matrix > 2
||mat_a||_2 =  2.057714638198031
||x||_2 =  1.1823661968140318
                          inv           solve
Computational time (s):       0.005,      0.0045
Norm2 Relative error  :    3.39e-16,    2.48e-16
PS C:\Users\tkouy\OneDrive - 静岡理工科大学\講義
Input size of square matrix > 1000
||mat_a||_2 =  816.1805789223445
||x||_2 =  25.340629320855 48
                          inv           solve
Computational time (s):       0.14,       0.087
Norm2 Relative error  :    3.37e-12,    1.61e-12
```

問題 5.2

プログラム例：

```
# ceig.py: 複素正方行列の固有値・固有ベクトル
import numpy as np # NumPy
import scipy.linalg as sclinalg # SciPy.linalg
```

```
# 乱数 seed の指定
rng = np.random.default_rng(seed=20240521)

# 正方行列の次数
input_str = input('Input size of square matrix > ')
n = int(input_str)

# 乱数行列生成
mat_a = rng.random((n, n)) + rng.random((n, n)) * 1j # 複素ベクトル
print('mat_a = ¥n', mat_a)

# 全ての固有値
eigen_values = sclinalg.eig(mat_a)

# 固有値，(右)固有ベクトル
eigen_values, Vr = sclinalg.eig(mat_a, right=True)
print('Eigenvalues = ', eigen_values)
print('Right eigenvectors = ', Vr)

# 固有値，左固有ベクトル，(右)固有ベクトル
eigen_values, Vl, Vr = sclinalg.eig(mat_a, left=True, right=True)
print('Eigenvalues = ', eigen_values)
print('Left eigenvectors = ¥n', Vl)
print('Right eigenvectors = ¥n', Vr)

print('index ||(A - eig * I) Vr || / ||Vr||  ||(A - eig * I) Vl|| /
||Vl||')
for index in range(n):
    # A * Vr == Labmda * Vr ?
    print(f'{index:5d} {np.linalg.norm((mat_a - eigen_values[index]
* np.eye(n)) @ Vr[:, index]) / np.linalg.norm(Vr[:, index]):30.17e}
{np.linalg.norm((mat_a.conj().T - eigen_values[index].conj() *
np.eye(n)) @ Vl[:, index]) / np.linalg.norm(Vl[:, index]):30.17e}')
```

実行結果：

レポート用紙

```
Input size of square matrix > 2
mat_a =
 [[0.7763316 +0.97887112j 0.54878817+0.22280498j]
 [0.97934203+0.35788684j 0.65359049+0.89895787j]]
Eigenvalues =  [ 1.45061101+1.22404376j -0.02068892+0.65378523j]
Right eigenvectors =  [[ 0.63619237+0.02349786j -0.56686221+0.00091524j]
 [ 0.77117256+0.j          0.82381211+0.j        ]]
Eigenvalues =  [ 1.45061101+1.22404376j -0.02068892+0.65378523j]
Left eigenvectors =
 [[ 0.82381211+0.j          0.77117256+0.j        ]
 [ 0.56686221+0.00091524j -0.63619237+0.02349786j]]
Right eigenvectors =
 [[ 0.63619237+0.02349786j -0.56686221+0.00091524j]
 [ 0.77117256+0.j          0.82381211+0.j        ]]
index ||(A - eig * I) Vr || / ||Vr||  ||(A - eig * I) Vl|| / ||Vl||
    0        7.69685683180812282e-16        8.955206467987465 23e-16
    1        2.54384052431380011e-16        3.18887285829407209e-16
```

以上