21960197 向平 Assignment 7

Ex 12.1.1

Using the supplied code (or code of your own) for the Power Iteration method, find the dominant eigenvector of *A*, and estimate the dominant eigenvalue by calculating a Rayleigh quotient. Compare your conclusions with the corresponding part of Exercise 5.

(a) $\begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix}$

```
A = [10 -12 -6; 5 -5 -4; -1 0 3];
initial = ones(3,1);
steps = 50;
[lam, u] = power_iteration(A, initial, steps)
```

```
lam =
   4.000000503397112
u = 3×1
  -0.577350487166988
  -0.577350269189598
   0.577350051212209
```

```
% correct results
[V,D] = eig(A)
```

```
V = 3×3
   0.816496580927726  -0.577350269189623   0.000000000000003
   0.408248290463862  -0.577350269189626   0.447213595499959
   0.408248290463864   0.577350269189628  -0.894427190999915
D = 3×3
   1.000000000000003                   0                   0
                   0   3.999999999999990                   0
                   0                   0   3.000000000000003
```

(b) $\begin{bmatrix} -14 & 20 & 10 \\ -19 & 27 & 12 \\ 23 & -32 & -13 \end{bmatrix}$

```
A = [-14 20 10; -19 27 12; 23 -32 -13];
[lam, u] = power_iteration(A, initial, steps)
```

```
lam =
  -3.999996476221397
u = 3×1
  -0.577350487166988
  -0.577350269189598
   0.577350051212209
```

```
% correct results
[V, D] = eig(A)
```

```
V = 3×3
  -0.577350269189626  -0.816496580927729  -0.000000000000001
  -0.577350269189626  -0.408248290463872   0.447213595499957
```

```
  0.577350269189625  -0.408248290463847  -0.894427190999917
D = 3×3
 -3.999999999999968                   0                   0
                  0   0.999999999999980                   0
                  0                   0   2.999999999999989
```

(c) $\begin{bmatrix} 8 & -8 & -4 \\ 12 & -15 & -7 \\ -18 & 26 & 12 \end{bmatrix}$

```
A = [8 -8 -4; 12 -15 -7; -18 26 12];
[lam, u] = power_iteration(A, initial, steps)
```

```
lam =
  4.000000000000007
u = 3×1
 -0.577350269189625
 -0.577350269189626
  0.577350269189627
```

```
% correct results
[V, D] = eig(A)
```

```
V = 3×3
  0.577350269189626   0.816496580927726   0.000000000000000
  0.577350269189626   0.408248290463863   0.447213595499958
 -0.577350269189626   0.408248290463863  -0.894427190999916
D = 3×3
  3.999999999999998                   0                   0
                  0   1.999999999999999                   0
                  0                   0  -0.999999999999999
```

(d) $\begin{bmatrix} 12 & -4 & -2 \\ 19 & -19 & -10 \\ -35 & 52 & 27 \end{bmatrix}$

```
A = [12 -4 -2; 19 -19 -10; -35 52 27];
[lam, u] = power_iteration(A, initial, steps)
```

```
lam =
 10.002528723735843
u = 3×1
 -0.574693977338165
 -0.576682202205000
  0.580658651938669
```

```
% correct results
[V, D] = eig(A)
```

```
V = 3×3
 -0.000000000000000  -0.577350269189631   0.816496580927733
 -0.447213595499958  -0.577350269189627   0.408248290463879
  0.894427190999916   0.577350269189619   0.408248290463832
D = 3×3
  1.000000000000007                   0                   0
                  0   9.999999999999975                   0
                  0                   0   9.000000000000020
```

From the above results, we can observe that the Power Iteration will converge to the eigenvalue with the largest absolute value, in this exercise, are $4, -4, 4,$ and 10 successively.

Ex 12.2.1

1. Apply the shifted QR algorithm (preliminary version shiftedqr0) with tolerance 10^{-14} directly to the following matrices:

(a) $\begin{bmatrix} -3 & 3 & 5 \\ 1 & -5 & -5 \\ 6 & 6 & 4 \end{bmatrix}$

```
A = [-3 3 5; 1 -5 -5; 6 6 4];
lam = shiftedqr0(A)
```

```
lam = 3×1
  -6.000000000000006
  -2.000000000000000
   4.000000000000000
```

```
% correct results
eig(A)
```

```
ans = 3×1
   4.000000000000000
  -6.000000000000003
  -2.000000000000001
```

(b) $\begin{bmatrix} 3 & 1 & 2 \\ 1 & 3 & -2 \\ 2 & 2 & 6 \end{bmatrix}$

```
A = [3 1 2; 1 3 -2; 2 2 6];
lam = shiftedqr0(A)
```

```
lam = 3×1
     2
     4
     6
```

```
% correct results
eig(A)
```

```
ans = 3×1
   2.000000000000001
   6.000000000000000
   3.999999999999997
```

(c) $\begin{bmatrix} 17 & 1 & 2 \\ 1 & 17 & -2 \\ 2 & 2 & 20 \end{bmatrix}$

```
A = [17 1 2; 1 17 -2; 2 2 20];
```

```
lam = shiftedqr0(A)
```

```
lam = 3×1
    16
    18
    20
```

```
% correct results
eig(A)
```

```
ans = 3×1
  15.999999999999982
  19.999999999999982
  17.999999999999996
```

(d) $\begin{bmatrix} -7 & -8 & 1 \\ 17 & 18 & -1 \\ -8 & -8 & 2 \end{bmatrix}$

```
A = [-7 -8 1; 17 18 -1; -8 -8 2];
lam = shiftedqr0(A)
```

```
lam = 3×1
   9.999999999999996
   1.000000000000002
   2.000000000000001
```

```
% correct results
eig(A)
```

```
ans = 3×1
  10.000000000000018
   0.999999999999997
   1.999999999999994
```

From the above results, we can observe that the shifted QR algorithm works well with very high precision, and even better than the Matlab built-in function in some cases.