CENG 371 SCIENTIFIC COMPUTING

HW3

Name: Andaç Berkay Seval

ID: 2235521

Q1)

a) Function is implemented in power_method.m file.
b) Function is implemented in inverse_power.m file.
c) For the largest eigenvalue, we can use power method. Thus, when we apply power method, we find that the largest eigenvalue of A is $\lambda_{largest} = 3.7321$, and corresponding eigenvector is $v_l = [0.2887 \ -0.5 \ 0.5774 \ -0.5 \ 0.2887]^T$. For the smallest eigenvalue, we can use shifted inverse power method with $\alpha = 0$ (since we want to find the smallest eigenvalue in magnitude). Thus, when we apply shifted inverse power method, we find that the smallest eigenvalue of A is $\lambda_{smallest} = 0.2679$, and the corresponding eigenvector is $v_s = [0.2887 \ 0.5 \ 0.5774 \ 0.5 \ 0.2887]^T$.

```
Largest eigenvalue of A:
3.7321
Corresponding eigenvector:
   0.2887
  -0.5000
   0.5774
  -0.5000
   0.2887
Smallest eigenvalue of A:
0.2679
Corresponding eigenvector:
   0.2887
   0.5000
   0.5774
   0.5000
   0.2887
```

Figure 1: Largest and smallest eigenvalues and corresponding eigenvectors of A.

d) Let's try to find the largest eigenvalue of B with pen and paper starting $x = [1 \ 1 \ 1]^T$:

for i = 1:k
  x = Bx
  x = x / ||x||
end
eigenvector = x
largest eigenvalue = $(x^TBx) / (x^Tx)$

where starting $x = [1 \ 1 \ 1]^T$ and B = [0.2 0.3 -0.5; 0.6 -0.8 0.2; -1.0 0.1 0.9]. Thus, in the first iteration, $Bx = [0 \ 0 \ 0]^T$ and x becomes $x = [0 \ 0 \ 0]^T$. Therefore, this algorithm does not work in theory for matrix B. However, when we applied power method, we can see that it works with largest eigenvalue of B $\lambda_{largest} = 1.3427$, and corresponding eigenvector is $v_l = [-0.407031 \ -0.028761 \ 0.912961]^T$. The reason behind that the power method is working is rounding errors. Thus, due to rounding errors, this algorithm works for matrix B.

```
Largest eigenvalue of B:
1.3427
Corresponding eigenvector:
  -0.407031
  -0.028761
   0.912961
```

Figure 2: Largest eigenvalue of B and corresponding eigenvector

Q2)

a) This idea works since we can find the next largest eigenvalue by deleting the largest eigenvalue. Let's look at the eigendecomposition of a matrix:
Let matrix A is a n x n matrix. Then, A can be factorized as $A = Q\Lambda Q^{-1}$ where Q is the n x n matrix with its ith column is the ith eigenvector v_i of A. Λ is the diagonal matrix with its diagonal elements are the eigenvalues $\Lambda_{ii} = \lambda_i$. This decomposition can be derived from:
   - $Av = \lambda v$
   - $AQ = Q\Lambda$
   - $A = Q\Lambda Q^{-1}$

   Therefore, this factorization is equal to

$$A = \sum_{i=1}^{n} \lambda_i \frac{v_i v_i^T}{v_i^T v_i}.$$

   Thus, if we subtract

$$\lambda_i \frac{v_i v_i^T}{v_i^T v_i}$$

   from the current matrix, we obtain

$$\sum_{i=2}^{n} \lambda_i \frac{v_i v_i^T}{v_i^T v_i}$$

   this matrix. Therefore, we make the magnitude of largest eigenvalue 0. If we apply power method to this new matrix, we will get the second largest eigenvalue of A. Moreover, if we subtract it successively from the current matrix at each step while applying power method, we get k largest eigenvalues of A.

b) Function is implemented in power_k.m file.
c) Function is implemented in subspace_iteration.m file.
d) Let's compare the performances of power_k method and subspace_iteration method on can_{229} matrix. I used the same iteration number 10^3 for both methods. Therefore, I compare them in terms of their total runtime and eigenvalues. Firstly, in terms of their runtime, for small k values (number of largest eigenvalues), power_k method performs better than the subspace_iteration method. I compare them for k values k = 10, 20, 50, 100, 200, 229. For k = 10, 20, 50 and 100, power_k method has a lesser total runtime than subspace_iteration method. However, for k = 200 and 229, subspace iteration method performs better. Furthermore, regardless of the k values, total runtime of the subspace_iteration method is almost the same for every k value. Thus, in terms of runtime, we can conclude that for small k values like up to 100, power_k method performs better. However, for large k values like after 100, subspace_iteration method performs better. Secondly, in terms of their eigenvalues, after some nth largest eigenvalue, there were some very little differences between the computed eigenvalues of methods.

```
Total runtime of power_k method when k is 10 : 0.679753
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 10 : 7.386512
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.997679
10'th largest eigenvalue is 6.989658
Total runtime of power_k method when k is 20 : 1.355952
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 20 : 7.407177
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.997068
10'th largest eigenvalue is 6.990268
```

Figure 3: Comparison between the methods in terms of their total runtime and 10 largest eigenvalues when k = 10, 20

```
Total runtime of power_k method when k is 50 : 3.391848
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 50 : 7.399687
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.996843
10'th largest eigenvalue is 6.990493
Total runtime of power_k method when k is 100 : 6.774190
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 100 : 7.393976
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.998526
10'th largest eigenvalue is 6.988811
```

Figure 4: Comparison between the methods when k = 50, 100

```
Total runtime of power_k method when k is 200 : 13.563591
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 200 : 7.400067
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.998532
10'th largest eigenvalue is 6.988806
Total runtime of power_k method when k is 229 : 15.518432
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727337
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.995875
10'th largest eigenvalue is 6.991464
Total runtime of subspace_iteration method when k is 229 : 7.415263
1'th largest eigenvalue is 8.696485
2'th largest eigenvalue is 8.474378
3'th largest eigenvalue is 8.474378
4'th largest eigenvalue is 7.901888
5'th largest eigenvalue is 7.901888
6'th largest eigenvalue is 7.727336
7'th largest eigenvalue is 7.442476
8'th largest eigenvalue is 7.442476
9'th largest eigenvalue is 6.994780
10'th largest eigenvalue is 6.992557
```

Figure 5: Comparison between the methods when k = 200, 229