main

July 21, 2024

1 Question 6

Part A: Generate a 10x10 random matrix

[ ]:
```
import numpy as np
from scipy.linalg import svd, eig

np.random.seed(0)  # For reproducibility
A = 20 * (np.random.rand(10, 10))
print(A)
```

```
[[10.97627008 14.30378733 12.05526752 10.89766366  8.47309599 12.91788226
   8.75174423 17.83546002 19.27325521  7.66883038]
 [15.83450076 10.5778984  11.36089122 18.51193277  1.42072116  1.74258599
   0.40436795 16.65239691 15.56313502 17.40024296]
 [19.57236684 15.98317128  9.22958725 15.61058353  2.36548852 12.79842043
   2.86706575 18.89337834 10.43696644  8.2932388 ]
 [ 5.29111224 15.48467379  9.12300664 11.36867898  0.37579601 12.35270994
  12.24191445 12.33867994 18.87496157 13.63640598]
 [ 7.19015801  8.74063908 13.95262392  1.20450943 13.33533431 13.41275739
   4.20765122  2.57852595  6.30856702  7.27421542]
 [11.40393541  8.77203027 19.76747676  2.04089621  4.17753512  3.22619036
  13.06216651  5.06583205  9.32621546  4.88851184]
 [ 3.17939167  2.20750282 13.12659179  2.76365903  3.93164723  7.37450341
  16.4198646   1.94202552 16.75889815  1.92196816]
 [19.5291893   9.37302403 19.53522176 12.09691039 14.78527159  0.78375585
   5.65613925  2.40393122  5.92280395  2.37455438]
 [ 6.35966359  8.28525989  1.28294993 13.84944239 11.33202908  5.30778982
  10.46496107  1.87881022 11.51892991 18.58592395]
 [ 6.37137905 13.3482076   2.63595725 14.32654408  5.78812186  3.66382724
  11.7302587   0.40215092 16.57880058  0.09390952]]
```

part B: Generate Eigenvalues

[ ]:
```
eigenvalues, _ = eig(A)
print("Eigenvalues of A:")
print(eigenvalues)
```

```
Eigenvalues of A:
[ 96.51219018+0.j          19.21023184+0.j
```

```
  16.44808113+0.j         -15.96781383+1.49688563j
 -15.96781383-1.49688563j  -8.92601657+0.65850417j
  -8.92601657-0.65850417j   1.47572628+7.06282219j
   1.47572628-7.06282219j   3.8162997 +0.j        ]
```

Part C: Eigenvalue real check

```
[ ]: real_eigenvalues = [val for val in eigenvalues if np.isclose(val.imag, 0)]
     print("Real Eigenvalues of A:")
     print(real_eigenvalues)
```

```
Real Eigenvalues of A:
[(96.51219017786045+0j), (19.210231844527975+0j), (16.44808113372758+0j),
(3.81629970407358+0j)]
```

Part D: find Singular values of A

```
[ ]: singular_values = svd(A, compute_uv=False)
     print("Singular values:", singular_values)
```

```
Singular values: [98.2983481  29.28287938 26.15214231 20.52939995 17.099897
15.07640661
  8.07955393  5.85349896  4.3683713   1.51667655]
```

Part E: SVD real check

```
[ ]: are_singular_values_real = np.all(np.isreal(singular_values))
     print("Are all singular values real?", are_singular_values_real)
```

```
Are all singular values real? True
```

Step F: Compare singular values to eigenvalues

```
[ ]: eigenvalues_sorted = np.sort(np.abs(eigenvalues))[::-1]
     singular_values_sorted = np.sort(singular_values)[::-1]
     print("Are singular values the same as the eigenvalues? ", np.
      ↪allclose(eigenvalues_sorted, singular_values_sorted))
```

```
Are singular values the same as the eigenvalues?  False
```

Part G: symmetric matrix conversion

```
[ ]: As = (A + A.T) / 2
     print(As)
```

```
[[10.97627008 15.06914404 15.81381718  8.09438795  7.831627   12.16090883
   5.96556795 18.68232466 12.8164594   7.02010471]
 [15.06914404 10.5778984  13.67203125 16.99830328  5.08068012  5.25730813
   1.30593539 13.01271047 11.92419745 15.37422528]
 [15.81381718 13.67203125  9.22958725 12.36679509  8.15905622 16.28294859
   7.99682877 19.21430005  5.85995818  5.46459802]
 [ 8.09438795 16.99830328 12.36679509 11.36867898  0.79015272  7.19680308
   7.50278674 12.21779517 16.36220198 13.98147503]
```

```
 [ 7.831627    5.08068012  8.15905622  0.79015272 13.33533431  8.79514626
   4.06964923  8.68189877  8.82029805  6.53116864]
 [12.16090883  5.25730813 16.28294859  7.19680308  8.79514626  3.22619036
  10.21833496  2.92479395  7.31700264  4.27616954]
 [ 5.96556795  1.30593539  7.99682877  7.50278674  4.06964923 10.21833496
  16.4198646   3.79908238 13.61192961  6.82611343]
 [18.68232466 13.01271047 19.21430005 12.21779517  8.68189877  2.92479395
   3.79908238  2.40393122  3.90080708  1.38835265]
 [12.8164594  11.92419745  5.85995818 16.36220198  8.82029805  7.31700264
  13.61192961  3.90080708 11.51892991 17.58236227]
 [ 7.02010471 15.37422528  5.46459802 13.98147503  6.53116864  4.27616954
   6.82611343  1.38835265 17.58236227  0.09390952]]
```

Part H: Repeat with symmetric matrix

[ ]:
```python
eigenvalues, _ = eig(As)
print("Eigenvalues of As:")
print(eigenvalues)

real_eigenvalues = [val for val in eigenvalues if np.isclose(val.imag, 0)]
print("Real Eigenvalues of A:")
print(real_eigenvalues)

singular_values = svd(As, compute_uv=False)
print("Singular values:", singular_values)

are_singular_values_real = np.all(np.isreal(singular_values))
print("Are all singular values real?", are_singular_values_real)

eigenvalues_sorted = np.sort(np.abs(eigenvalues))[::-1]
singular_values_sorted = np.sort(singular_values)[::-1]
print("Are singular values the same as the eigenvalues? ", np.
 ↪allclose(eigenvalues_sorted, singular_values_sorted))
```

```
Eigenvalues of As:
[ 97.52207344+0.j  22.62768354+0.j  18.52487549+0.j -21.54583336+0.j
  10.38597708+0.j -15.73403841+0.j -12.80116478+0.j  -0.72156405+0.j
  -2.27128689+0.j  -6.83612742+0.j]
Real Eigenvalues of A:
[(97.52207343632773+0j), (22.627683535121125+0j), (18.524875487590425+0j),
(-21.545833361488512+0j), (10.385977078262073+0j), (-15.734038411787171+0j),
(-12.801164778700912+0j), (-0.7215640530867955+0j), (-2.271286892963698+0j),
(-6.836127420427997+0j)]
Singular values: [97.52207344 22.62768354 21.54583336 18.52487549 15.73403841
12.80116478
 10.38597708  6.83612742  2.27128689  0.72156405]
Are all singular values real? True
Are singular values the same as the eigenvalues?  True
```

Step i: Sort absolute values of eigenvalues of As in descending order

[ ]:
```
eigenvalues_sorted_desc = np.sort(np.abs(eigenvalues))[::-1]
print("Sorted absolute eigenvalues of As in descending order:",␣
↪eigenvalues_sorted_desc)
```

```
Sorted absolute eigenvalues of As in descending order: [97.52207344 22.62768354
21.54583336 18.52487549 15.73403841 12.80116478
 10.38597708  6.83612742  2.27128689  0.72156405]
```

step j: difference etween vectors of eigenvalues and vector of singular values

[ ]:
```
r = eigenvalues_sorted_desc - singular_values_sorted
print("Difference r:", r)
# This vector is small
```

```
Difference r: [ 1.56319402e-13  5.32907052e-14  3.55271368e-14  1.06581410e-14
  1.77635684e-15  1.77635684e-15  1.42108547e-14  3.55271368e-15
  0.00000000e+00 -1.55431223e-15]
```

Part k: take norm of r

[ ]:
```
norm_r = np.linalg.norm(r)
print("Norm of r:", norm_r)
```

```
Norm of r: 1.6992564555596449e-13
```

Compare norm of r to machine epsilon

[ ]:
```
machine_epsilon = 1e-16
comparison_value = norm_r / (0.5 * (np.linalg.norm(eigenvalues_sorted_desc) +␣
↪np.linalg.norm(singular_values_sorted)))
print("Comparison value:", comparison_value)
```

```
Comparison value: 1.5913828481644784e-15
```

[ ]:
```
print("Is the comparison value close to the machine epsilon?", comparison_value␣
↪< machine_epsilon)
```

```
Is the comparison value close to the machine epsilon? False
```