MannanNaeem_HW7

October 30, 2025

[2]:
```
import numpy as np
```

[3]:
```
A = np.array([
    [4, -2, 2, 1],
    [1, 1, 0, 1],
    [-2, 1, 3, -1],
    [1, 3, -1, 2]
])
```

[5]:
```
inverse_a = np.linalg.inv(A)
print(inverse_a)
```

```
[[ 5.00000000e+00 -3.00000000e+01  1.00000000e+00  1.30000000e+01]
 [ 4.00000000e+00 -2.50000000e+01  1.00000000e+00  1.10000000e+01]
 [-1.00000000e+00  7.00000000e+00 -7.77156117e-16 -3.00000000e+00]
 [-9.00000000e+00  5.60000000e+01 -2.00000000e+00 -2.40000000e+01]]
```

[6]:
```
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:\n", eigenvalues)
print("\nEigenvectors (columns):\n", eigenvectors)
```

```
Eigenvalues:
 [-0.02231724+0.j          3.6104532 +1.72034086j  3.6104532 -1.72034086j
  2.80141085+0.j        ]

Eigenvectors (columns):
 [[-0.43676159+0.j          0.57598271+0.j          0.57598271-0.j
   0.26748859+0.j        ]
 [-0.36900111+0.j          0.00962337-0.2105225j   0.00962337+0.2105225j
  -0.34893941+0.j        ]
 [ 0.10239683+0.j         -0.00821729+0.55142262j -0.00821729-0.55142262j
  -0.06120796+0.j        ]
 [ 0.81399778+0.j         -0.18869091-0.53300366j -0.18869091+0.53300366j
  -0.89607183+0.j        ]]
```

[7]:
```
# Construct eigen matrix P (columns are eigenvectors)
P = eigenvectors
```

```python
# Construct diagonal matrix D from eigenvalues
D = np.diag(eigenvalues)

# Compute the inverse of P
P_inv = np.linalg.inv(P)

# Verify A = P * D * P ¹
A_diagonalized = P @ D @ P_inv

print("Diagonal matrix D:\n", D)
print("\nP ¹:\n", P_inv)
print("\nReconstructed A (P * D * P ¹):\n", A_diagonalized)

# Check numerical equality
print("\nIs A identical to P * D * P ¹? ->", np.allclose(A, A_diagonalized))
```

```
Diagonal matrix D:
 [[-0.02231724+0.j          0.        +0.j          0.        +0.j
   0.        +0.j        ]
 [ 0.        +0.j          3.6104532 +1.72034086j  0.        +0.j
   0.        +0.j        ]
 [ 0.        +0.j          0.        +0.j          3.6104532 -1.72034086j
   0.        +0.j        ]
 [ 0.        +0.j          0.        +0.j          0.        +0.j
   2.80141085+0.j        ]]

P ¹:
 [[ 0.24293955+2.72782814e-16j -1.52243125-6.81957034e-17j
   0.05807612+9.37690922e-17j  0.66140356+3.40978517e-17j]
 [ 1.00207027+1.76334426e-02j -0.26464982-6.27009101e-02j
   0.2698955 -8.46128808e-01j  0.38375201+8.74766701e-02j]
 [ 1.00207027-1.76334426e-02j -0.26464982+6.27009101e-02j
   0.2698955 +8.46128808e-01j  0.38375201-8.74766701e-02j]
 [-0.18035771+6.73859095e-18j -1.34612117+5.02942747e-17j
  -1.06750294+1.14609174e-16j -0.57270992+2.13977987e-17j]]

Reconstructed A (P * D * P ¹):
 [[ 4.00000000e+00+4.59988659e-16j -2.00000000e+00-1.25679570e-16j
   2.00000000e+00-5.55060892e-18j  1.00000000e+00-5.87274714e-17j]
 [ 1.00000000e+00-1.45688883e-16j  1.00000000e+00+2.06157796e-17j
   2.22044605e-16-4.41427991e-17j  1.00000000e+00-4.48739398e-17j]
 [-2.00000000e+00+2.70950622e-16j  1.00000000e+00-2.04476175e-16j
   3.00000000e+00-1.22420428e-16j -1.00000000e+00+6.26632910e-17j]
 [ 1.00000000e+00-4.16733564e-16j  3.00000000e+00+1.63580536e-16j
  -1.00000000e+00-2.23091523e-16j  2.00000000e+00-5.52727360e-17j]]

Is A identical to P * D * P ¹? -> True
```

```
[8]: # Perform Singular Value Decomposition
     U, sigma, VT = np.linalg.svd(A)

     print("U (Left singular vectors):\n", U)
     print("\nSigma (Singular values):\n", sigma)
     print("\nV^T (Right singular vectors):\n", VT)

     # Construct Sigma matrix (choose desired range for sigma)
     Sigma = np.zeros_like(A, dtype=float)
     np.fill_diagonal(Sigma, sigma)

     # Reconstruct A using SVD components
     A_reconstructed = U @ Sigma @ VT

     print("\nReconstructed A from SVD:\n", A_reconstructed)
     print("\nIs reconstructed A identical to original A? ->", np.allclose(A,␣
      ↪A_reconstructed))
```

```
U (Left singular vectors):
 [[-0.90088393  0.29808588 -0.27934466  0.14669534]
 [-0.15664896 -0.27968393 -0.27004158 -0.90791825]
 [ 0.40053155  0.44427396 -0.80072384  0.03219408]
 [-0.05868327 -0.79720911 -0.45594456  0.39131633]]

Sigma (Singular values):
 [5.25493707 4.48128809 3.20990942 0.0132293 ]

V^T (Right singular vectors):
 [[-0.87916011  0.35577983 -0.10304403 -0.29980016]
 [-0.17251679 -0.62999945  0.60835248 -0.45082802]
 [-0.07536571 -0.5856582  -0.78036977 -0.2057851 ]
 [ 0.43776188  0.36541191 -0.10154905 -0.81518492]]

Reconstructed A from SVD:
 [[ 4.00000000e+00 -2.00000000e+00  2.00000000e+00  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00 -4.18216367e-16  1.00000000e+00]
 [-2.00000000e+00  1.00000000e+00  3.00000000e+00 -1.00000000e+00]
 [ 1.00000000e+00  3.00000000e+00 -1.00000000e+00  2.00000000e+00]]

Is reconstructed A identical to original A? -> True
```