In [11]:
```
import numpy as np

from numpy.linalg import inv, eig, svd

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

In [12]:
```
# Inverse of A

A_inv = inv(A)
print("Inverse of A:", A_inv)
```

```
Inverse of A: [[ 5.00000000e+00 -3.00000000e+01  1.00000000e+00  1.3
0000000e+01]
 [ 4.00000000e+00 -2.50000000e+01  1.00000000e+00  1.10000000e+01]
 [-1.00000000e+00  7.00000000e+00  5.82867088e-16 -3.00000000e+00]
 [-9.00000000e+00  5.60000000e+01 -2.00000000e+00 -2.40000000e+01]]
```

In [13]:
```
# Eigenvalues and Eigenvectors

eigenvalues, eigenvectors = eig(A)
print("\nEigenvalues:\n", eigenvalues)
print("\nEigenvectors (columns of P):\n", eigenvectors)
```

```
Eigenvalues:
 [-0.02231724+0.j          3.6104532 +1.72034086j  3.6104532 -1.7203
4086j
  2.80141085+0.j        ]

Eigenvectors (columns of P):
 [[-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+0.2105
225j
  -0.34893941+0.j        ]
 [ 0.10239683+0.j         -0.00821729+0.55142262j -0.00821729-0.5514
2262j
  -0.06120796+0.j        ]
 [ 0.81399778+0.j         -0.18869091-0.53300366j -0.18869091+0.5330
0366j
  -0.89607183+0.j        ]]
```

In [14]:
```
# Diagonalization

P = eigenvectors
D = np.diag(eigenvalues)
P_inv = inv(P)

# Verify A = P D P⁻¹
A_diagonalized = P @ D @ P_inv
print("\nDiagonalized A (P D P⁻¹):\n", A_diagonalized)
```

```
Diagonalized A (P D P⁻¹):
 [[ 4.00000000e+00-1.27549228e-16j -2.00000000e+00+1.48378393e-16j
   2.00000000e+00+1.93217153e-16j  1.00000000e+00-1.17034130e-16j]
 [ 1.00000000e+00+9.11516538e-18j  1.00000000e+00-2.39847431e-17j
  -4.14357081e-15+1.91320470e-17j  1.00000000e+00-3.81278032e-17j]
 [-2.00000000e+00-4.39044044e-16j  1.00000000e+00+2.43494279e-17j
   3.00000000e+00-4.18421915e-17j -1.00000000e+00-7.35341107e-17j]
 [ 1.00000000e+00-2.67485770e-16j  3.00000000e+00-1.08632006e-16j
  -1.00000000e+00-9.16637678e-17j  2.00000000e+00-3.98164530e-17j]]
```

In [15]:
```
# Check if A_diagonalized ≈ A
is_identical = np.allclose(A, A_diagonalized)
print("\nIs A_diagonalized identical to A?", is_identical)
```

```
Is A_diagonalized identical to A? True
```

Explanation

Diagonalization is important because it simplifies matrix operations. When a matrix A can be written as $A = P D P^{-1}$, where D is diagonal:

- Powers of A become easy: $A^n = P D^n P^{-1}$
- It reveals the matrix's structure via eigenvalues
- It enables efficient computation in systems of differential equations, quantum mechanics, and PCA

In [16]:
```
# SVD Reconstruction

U, S, Vt = svd(A)
```

In [19]:
```
# Choose a threshold to keep significant singular values

sigma_threshold = 1.0
S_filtered = np.array([s if s > sigma_threshold else 0 for s in S])
A_reconstructed = U @ np.diag(S_filtered) @ Vt

print("Reconstructed A using filtered SVD:\n", A_reconstructed)
print("Is reconstructed A ≈ original A? ", np.allclose(A, A_reconstruc
```

```
Reconstructed A using filtered SVD:
 [[ 3.99915045e+00 -2.00070915e+00  2.00019707e+00  1.00158201e+00]
 [ 1.00525801e+00  1.00438901e+00 -1.21971810e-03  9.90208714e-01]
 [-2.00018645e+00  9.99844369e-01  3.00004325e+00 -9.99652809e-01]
 [ 9.97733776e-01  2.99810832e+00 -9.99474297e-01  2.00422008e+00]]
Is reconstructed A ≈ original A?  False
```

In [ ]:
```

```