Experiment - 3

* AIM - Exercise to find the eigen value and eigen vectors in scilab.

* materials - A Computer with scilab installed.

* Procedure -

(i) Basic matrix operations -

- familiarize yourself with basic matrix creation and manipulation in scilab, create a simple matrix by enclosing elements in square brackets separated with commas for ex.
  type A = [[1,2], [3,4]] to make a 2x2 matrix A.

- Access individual elements using rows and col. indices for ex ~~type~~ A(1,2) to see the element at row 1 and col 2

- Practice basic operations like matrix addition, subtraction, multiplication and transportation.

--> A = [[1,2]] ; [3,4]]

A = 1. 2.
    3. 4.

--> disp (A (1,2))

2.

--> ~~B =~~ [[5,6]]; [7, 8]]

B = 5. 6.
    7. 8.

--> C. add = A+B

C. add = 6. 8.
         10. 12.

--> disp (C. add)

6. 8.
10. 12.

--> C_sub = A-B

C_sub = -4. -4.

-4. -4.

--> disp(C_sub)

-4. -4.

-4. -4.

--> C_transpose = A'

C_transpose = 1. 3.

2. 4.

--> disp(C_transpose)

~~1. 3.~~

2. 4.

(ii) Finding Eigen Values -

- Scilab provide the builtin function (spec) to find eigen value of the matrix for example type eig to get a vector containing the eigen of matrix
- Explore the obtained eigen values and understand their relationship to the matrix.
- Experiment with different types of matrixes including symmetric, triangular and diagonal and observe how their eigen values are affected.

--> ~~A~~ [4, -2; 1, 1]

A = 4. -2.

1. 1.

--> eigenvalue = spec(A)

eigenvalue =

3. + 0.i

2. + 0.i

--> Symmetric_Matrix = [2, 1; 1, 3

Symmetric_Matrix =

2. 1.

1. 3.

--> eigen_symmetric = spec(symmetric_matrix)

eigen_symmetric = 1.3819660
3.6180340

--> triangular_matrix = [1, 2 ; 0, 3]

triangular_matrix = 1. 2.
0. 3.

--> eigen_triangular = spec(triangular_matrix)

eigen_triangular = 1. + 0.i
3. + 0.i

--> disp(eigen_triangular)

1. + 0.i
3. + 0.i

--> diagonal_matrix = [4, 0 ; 0, -1]

diagonal_matrix = 4. 0.
0. -1.

--> eigen_diagonal = spec(diagonal_matrix)

eigen_diagonal = -1.
4.

--> disp(eigen_diagonal)

-1.
4.

(iii) Finding Eigenvectors -

- we the eigs function to find the eigenvectors corresponding to the eigen value. For example type eigs(A) to get the matrix containing the eigen vectors as columns.
- Each column in the obtained matrix correspond to the eigenvector associated with a corresponding eigen value.

- verify the relation between eigenvectors and eigen-values using the equation A * v = lambda * v where v is an eigenvector and lambda is corresponding eigenvalue

--> A = [[4, -2]; [1, 1]]

A =   4.   -2.
      1.   1.

--> [v, D] = spec (A)

v =   0.8944272 + 0.i   0.7071068 + 0.
      ~~0.4472136~~ + 0.i   0.7071068 + 0.

D =   3. + 0.i   0. + 0.i
      0. + 0.i   2. + 0.i

--> for i = 1 : size (D, 1)
> lambda = D (1, i)
> t = v (:, i) ;
> result = A * t - lambda * t
> disp (result)
> end

0.
0.
0.
0.

(iv) Application of Eigenvalues and eigenvectors -
Discuss and research some real world engineering applications of eigenvalues and eigenvectors such as modal analysis of vibration, principal component analysis (PCA) for the data compression and data image processing.
Consider how these concepts can be applied in your specific engineering discipline.

>> import numpy as np

Example 1. Model Analysis of vibrations.

Eigenvalues and eigenvectors are commonly used in model analysis to study vibration in structure.

Define a symmetric matrix corresponding the structure stiffness or mass matrix.

>> structure_matrix = np.array([[[4,-1],[-1,3]])

calculate eigen values and eigenvectors

>> eigenvalues, eignvectors = np.linalg.eig (structure_matrix)

>> print ("Example ~~1 Model Analysis~~ of vibrations")

>> print (" Eigenvectors:")

>> print (eigenvectors)

=> Example 1 ⟶ Model Analysis and vibrations

Eigenvalue - [4.6180 3399 2.38196608]

Eigenvector - [[0.85065081 0.5257 3111]
[-0.5857 3111 0.85065081]]

Example - 2 Principal components Analysis (PCA) for data compression.

Create a sample datasheet

>> data = np.array ~~([[1~~,2,3],[4,5,6],[7,8,9]])

Calculate the ~~covariance~~ matrix

>> covariance_matrix : np.cov(data, rowvar= false)

calculate eigenvalues and eigenvectors

>> eigenvalues_pca, eigenvectors_pca =
np.linalg.eig (covariance_matrix)

>> print ("\n Example 2- PCA for data compression")

>> print (" Eigenvalues (PCA): ", eigenvalues_pca)

>> print (" Eigenvalues (PCA): ")

>> print (eigenvectors_pca)

→ Example 2 → PCA for data compression
Eigenvalue (PCA) : [0. 27. 0.]
Eigenvectors (PCA) : [[-0.81649658 0.57735 0.
[0.40824829 0.57735 -0.
[0.408248 0.57735 0.70

Example 3: Image Processing -
consider a grayscale image as 3×3 matrix
>> image_matrix = np.array([[10, 20, 30],
[40, 50, 60], [70, 80,

Calculate eigenvalue and eigenvectors
>> eigenvalues_image, eigenvectors_image = np.linalg.eig(image_matrix)
>> print("\n Example 3 - Image Processing:")
>> print("Eigenvalues (Image):", eigenvalues_image
>> print("Eigenvectors (Image):")
>> print(eigenvectors_image).

→ Example 3 → Image Processing.
Eigenvalues (Image) : [1.61168 -1.116843 2.0680
Eigenvectors (Images) : [[0.23197 -0.7858 0.408
[-0.5253 2209 -0.0867513 -0.81
[-0.8186 735 0.612327 0.40824869]

* Conclusion - The experiment provides a fundamental understanding of finding eigenvalue and eigenvectors in Scilab. By continuing to explore these concepts and their applications, you unlock powerful tools for solving complex engineering problems involving matrix analysis and data interpretation