

3. EIGENVALUES & EIGENVECTORS

b) Properties of eigenvalues

November 27, 2021

1 AIM

We will input one matrix, generate one random matrix, and one random skew-matrix. For these matrices, we will present various properties regarding eigenvalues.

Matrix input functions...

```
[11]: from numpy import linalg, matrix, identity, tril, random, round, zeros, trace, \
      ↪ product
def inputPositiveInteger(prompt):
    while True:
        try:
            i = input(prompt)
            if i == "x": return 0
            i = int(i)
            if i <= 0: i = 1/0
            return i
        except:
            print("Invalid integer, please re-enter.")
def floatInput(prompt):
    while True:
        try:
            i = float(input(prompt))
            return i
        except:
            print("Invalid number, please re-enter.")

def matrixInput(nRow, nCol):
    print("\nEnter row by row, each element in the row separated by comma...")
    A, i = zeros((nRow, nCol)), 0
    while i < nRow:
        row = input("R{0}: ".format(i + 1)).split(",")
        if "x" in row: break # To stop inputting anymore
        if len(row) != nCol:
            print("ERROR: You must only enter", nCol, "per row")
            continue
        for j in range(0, nCol):
            try:
```

```

        A[i][j] = float(row[j])
    except:
        print("ERROR: Non-numeric inputs.")
        j = -1
        break
    if j != -1: i = i + 1
return A

```

Matrix generation functions...

```

[12]: from numpy import linalg, matrix, identity, tril, random, round
# tril returns the lower triangular matrix for an array or matrix.
# The 1st argument is the array or matrix.
# The 2nd argument specifies whether zeros should be at (k = -1) or above (k = 0) the diagonal.
# Some functions...

def randomMatrix(n, m):
    A = zeros((n, m))
    for i in range(0, n):
        for j in range(0, m):
            A[i][j] = random.randint(1, 25)
    return A

def randomSkewMatrix(n, m):
    A = zeros((n, m))
    # Creating upper triangle...
    for i in range(0, n):
        A[i][i] = 0
        for j in range(i + 1, m):
            A[i][j] = random.randint(1, 25)
    for i in range(0, n):
        for j in range(0, i):
            A[i][j] = -A[j][i]
    return A

```

Main...

```

[13]: # Matrix input
n = inputPositiveInteger("n (rows or columns in the square matrix): ")
M = matrixInput(n, n)
print("Matrix:\n{0}".format(M))
# Proving properties of eigenvalues and eigenvectors...
# 1. For a nxn matrix, the number of eigen values is n.
print("-----")
print("PROPERTY 1:")
print("Eigen values:")
eigenValues = linalg.eigvals(M)
for e in eigenValues: print(e)

```

```

print("\nNumber of eigen values:", len(eigenValues))
# 2. The sum of eigen values is equal to the sum of the diagonal elements of
    ↪ matrix.
print("-----")
print("PROPERTY 2:")
print("Sum of eigen values:", round(sum(eigenValues), 3))
print("Sum of diagonal values:", trace(M))
# 3. The product of eigenvalues is equal to the determinant of the matrix.
print("-----")
print("PROPERTY 3:")
print("Product of eigen values:", round(product(eigenValues), 3))
print("Determinant of matrix of:", (linalg.det(M)))
# 4. The eigen value for an identity matrix is 1.
print("-----")
print("PROPERTY 4:")
I = identity(3)
print("Identity matrix:\n{}".format(I))
print("Eigen values:")
tmp = linalg.eigvals(I)
for e in tmp: print(e)
# 5. The eigen value of a triangular matrix is same as the diagonal elements of
    ↪ a matrix.
print("-----")
print("PROPERTY 5:")
# Random matrix...
A = matrix(randomMatrix(5, 5))
# Lower triangular matrix...
T = tril(A, 0)
print("Lower triangular matrix:\n{}".format(T))
print("Eigen values:")
tmp = linalg.eigvals(T)
for e in tmp: print(e)
# 6. For a skew symmetric matrix, the eigenvalues are imaginary.
print("-----")
print("PROPERTY 6:")
A = matrix(randomSkewMatrix(3, 3))
print("Matrix:\n{}".format(A))
print("Eigen values:")
tmp = linalg.eigvals(A)
for e in tmp: print(e)
# 7. For orthogonal matrix the values of eigenvalues are 1 or -1.
print("-----")
print("PROPERTY 7:")
A = matrix([[1, 2, 2], [2, 1, -2], [-2, 2, -1]]) / 3
print("Orthogonal matrix:\n{}".format(A))
print("Eigen values:")
tmp = linalg.eigvals(A)

```

```

for e in tmp: print(round(e, 3))
# 8. For idempotent matrix the eigenvalues are 0 and 1.
print("-----")
print("PROPERTY 8:")
A = matrix([[2, -2, -4], [-1, 3, 4], [1, -2, -3]])
print("Idempotent matrix:\n{0}".format(A))
print("Eigen values:")
tmp = linalg.eigvals(A)
for e in tmp: print(round(e, 3))

```

n (rows or columns in the square matrix): 3

Enter row by row, each element in the row separated by comma...

R1: 2,3,4

R2: 2,6,3

R3: 0,7,4

Matrix:

```

[[2. 3. 4.]
 [2. 6. 3.]
 [0. 7. 4.]]

```

PROPERTY 1:

Eigen values:

(10.747189497992093+0j)

(0.6264052510039564+1.7729705556948898j)

(0.6264052510039564-1.7729705556948898j)

Number of eigen values: 3

PROPERTY 2:

Sum of eigen values: (12+0j)

Sum of diagonal values: 12.0

PROPERTY 3:

Product of eigen values: (38+0j)

Determinant of matrix of: 37.99999999999999

PROPERTY 4:

Identity matrix:

```

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

```

Eigen values:

1.0

1.0

1.0

PROPERTY 5:

Lower triangular matrix:

```
[[11.  0.  0.  0.  0.]
 [11.  4.  0.  0.  0.]
 [11. 21. 18.  0.  0.]
 [13. 18. 21. 17.  0.]
 [12.  9. 23.  8. 18.]]
```

Eigen values:

18.0

17.0

18.0

4.0

11.0

PROPERTY 6:

Matrix:

```
[[ 0.  2.  8.]
 [-2.  0. 13.]
 [-8. -13.  0.]]
```

Eigen values:

(3.8556595020785216e-17+0j)

(4.440892098500626e-16+15.394804318340654j)

(4.440892098500626e-16-15.394804318340654j)

PROPERTY 7:

Orthogonal matrix:

```
[[ 0.33333333  0.66666667  0.66666667]
 [ 0.66666667  0.33333333 -0.66666667]
 [-0.66666667  0.66666667 -0.33333333]]
```

Eigen values:

(1+0j)

(-0.333+0.943j)

(-0.333-0.943j)

PROPERTY 8:

Idempotent matrix:

```
[[ 2 -2 -4]
 [-1  3  4]
 [ 1 -2 -3]]
```

Eigen values:

0.0

1.0

1.0