2. MATRIX RELATED

e) Exercise on row echelon form, rank & nullity

November 27, 2021

1 Finding rank of 2 given matrices

[26]:
```
from sympy import Matrix, zeros, pprint
G = Matrix([[1, 2, 1], [3, 1, -2], [2, 5, 7]])
S = Matrix([[7, 5, 1], [9, 8, -1], [8, 4, 3]])
print("G =")
pprint(G)
print("S =")
pprint(S)
```

```
G =
⎡1  2  1 ⎤
⎢         ⎥
⎢3  1  -2⎥
⎢         ⎥
⎣2  5  7 ⎦
S =
⎡7  5  1 ⎤
⎢         ⎥
⎢9  8  -1⎥
⎢         ⎥
⎣8  4  3 ⎦
```

[28]:
```
print("Ranks of G and S")
print("G rank =", G.rank())
print("S rank =", S.rank())
```

```
Ranks of G and S
G rank = 3
S rank = 3
```

2 Working on 2 user-inputted matrices

[1]:
```
# Support functions...
from sympy import Matrix, zeros, pprint
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
```

[3]:
```
print("MATRIX 1")
A = matrixInput(3, 3)
print("\nMATRIX 2")
B = matrixInput(3, 3)
```

```
MATRIX 1

Enter row by row, each element in the row separated by comma…
R1: 1, 4, 5
R2: 2, 7, 4
R3: 9, 9, 0
```

```
MATRIX 2

Enter row by row, each element in the row separated by comma…
R1: 0, 3, 1
R2: 9, 6, 4
R3: 0, 8, 2
```

[9]:
```
print("A =")
pprint(A)
print("B =")
pprint(B)
```

```
A =
 1.0  4.0  5.0

 2.0  7.0  4.0

 9.0  9.0  0.0
B =
 0.0  3.0  1.0

 9.0  6.0  4.0

 0.0  8.0  2.0
```

[12]:
```
print("Row echelon forms...")
print("For A:")
pprint(A.rref()[0])
print("For B:")
pprint(B.rref()[0])
```

```
Row echelon forms…
For A:
 1  0  0

 0  1  0

 0  0  1
For B:
 1  0  0

 0  1  0

 0  0  1
```

[17]:
```
print("Checking if singular or not...")
def isSingular(M, name):
    try:
        M.det()
        print(name + " is non-singular!")
    except: print(name + " is singular!")

isSingular(A, 'A')
isSingular(B, 'B')
```

```
Checking if singular or not…
A is non-singular!
B is non-singular!
```

[19]:
```
print("Rank and nullity...")
def rankAndNullity(M, name):
    print("------------\nFor " + name + ":")
    # Rank
    print("Rank:", M.rank())
    # Nullity
    nullspace = M.nullspace()
    nullity = len(nullspace)
    print("Nullity:", nullity)

rankAndNullity(A, 'A')
rankAndNullity(B, 'B')
```

```
Rank and nullity…
------------
For A:
Rank: 3
Nullity: 0
------------
For B:
Rank: 3
Nullity: 0
```

NULLITY: The null space of any matrix A consists of all the vectors B such that AB = 0 and B is not zero. It can also be thought as the solution obtained from AB = 0 where A is known matrix of size m x n and B is matrix to be found of size n x 1. The nullity of A is the number of vectors in its nullspace. The nullspace function for sympy matrices returns a list of vectors that are in the nullspace of A, as defined above. By getting its length, we can figure out the nullity of A.

[21]:
```
print("Ranks of A+B, A-B and A•B...")
print("Rank:", (A + B).rank())
print("Rank:", (A - B).rank())
print("Rank:", (A * B).rank())
```

```
Ranks of A+B, A-B and A•B…
Rank: 3
Rank: 3
Rank: 3
```

As we can see, the ranks of all the above matrices are 3.