2. MATRIX RELATED

c) SymPy matrices

November 27, 2021

1 AIM

Matrices are available in both NumPy and SymPy. In SymPy, we have functionalities that are not offered in NumPy matrices. SymPy matrices can be considered as extensions of NumPy matrices, since they add on to the already wide range of features available for matrix computation, such as row echelon forms, rank, nullspaces, etc.

2 Row echelon form

Getting the row echelon form of matrices…

[1]:
```
from sympy import Matrix, pprint
B = Matrix([[1, 1, 1], [3, 1, -2], [2, 4, 7]])
C = Matrix([[7, 5, 3], [9, 8, -1], [8, 4, 3]])
print("------------\nMatrix B is...")
pprint(B)
print("Row echelon form of B...")
pprint(B.rref()[0])
# Matrix.rref(M) returns a tuple.
# The first element is the row reduced matrix.
# The second element is a tuple of non-zero i.e. pivot column numbers.
print("------------\nMatrix C is...")
pprint(C)
print("Row echelon form of C...")
pprint(C.rref()[0])
```

```
------------
Matrix B is…
⎡1  1  1 ⎤
⎢        ⎥
⎢3  1  -2⎥
⎢        ⎥
⎣2  4  7 ⎦
Row echelon form of B…
⎡1  0  -3/2⎤
⎢          ⎥
⎢0  1  5/2 ⎥
⎢          ⎥
⎣0  0  0   ⎦
```

```
 0  0   0
------------
Matrix C is…
 7  5  3

 9  8  -1

 8  4  3
Row echelon form of C…
 1  0  0

 0  1  0

 0  0  1
```

3 Matrix as sum of symmetric and skew-symmetric matrices

Expressing a matrix as a sum of symmetric and skew-symmetric matrices…

A square matrix M can be expressed as

$M = \frac{1}{2}(M + M^T) + \frac{1}{2}(M - M^T)$

Now, a transpose of any matrix M is such that row i of M becomes column i of M's transpose. Hence, row i in M's transpose is column i in M. Hence, by adding M and its transpose, we are essentially adding row i of M to column i of M, and adding every column i of M to row i. Hence, every row becomes row i + column i, and every column becomes column i + row i, meaning the rows and columns become equal i.e. interchangeable. Hence, by definition, $M + M^T$ is a symmetric matrix, meaning $\frac{1}{2}(M + M^T)$ is also symmetric.

Now, if we subtract M's transpose from M, we get that every row i of M becomes row i - column i, and the left-most element becomes zero. However, every column i of M becomes column i - row i by the same operation, and the top element becomes zero. Hence, we have the 1st row and 1st column as zero, and every other row and column such that row i = negative of column i i.e. every row becomes interchangeable with the corresponding column's negative. Hence, by definition, $M - M^T$ is a skew-symmetric matrix, meaning $\frac{1}{2}(M + M^T)$ is also skew-symmetric.

This, way, M can be expressed as a sum of symmetric and skew-symmetric matrices.

[20]:
```
from sympy import Matrix, pprint
M = Matrix([[1, 8, 0], [3, 5, 2], [-8, 9, -2]])
M_t = M.transpose()
M1 = (M + M_t)/2
M2 = (M - M_t)/2
# Transposes of M1 and M2...
M1_t = M1.transpose()
M2_t = M2.transpose()
print("We have that...")
pprint(M)
```

```
print("can be expressed as the sum of the following...")
print("------------")
pprint(M1)
print("Is symmetric? {0}".format(M1 == M1_t))
print("------------")
pprint(M2)
print("Is skew-symmetric? {0}".format(M2 == -M2_t))
print("------------")
print("To confirm...")
pprint(M1 + M2)
```

```
We have that…
1   8  0

3   5  2

-8  9  -2
can be expressed as the sum of the following…
------------
 1    11/2   -4

11/2   5    11/2

 -4   11/2   -2
Is symmetric? True
------------
 0    5/2   4

-5/2   0   -7/2

 -4   7/2   0
Is skew-symmetric? True
------------
To confirm…
1   8  0

3   5  2

-8  9  -2
```

4 Measures available for SymPy matrices

Various measures of a user-defined sympy matrix…

[9]:
```
# 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
```

[34]:
```
# Main program
singular = False
print("MATRIX 1")
nRow = inputPositiveInteger("Rows\t: ")
nCol = inputPositiveInteger("Columns\t: ")
A = matrixInput(nRow, nCol)
print("Your matrix:")
pprint(A)
print("------------\nResults of certain functions on this matrix...")
# Determinant
try:
```

```
    det = A.det()
    print("Determinant:", det)
    if det == 0: singular = True
except:
    print("Determinant: Unobtainable for non-square matrix!")
# Row echelon form
print("Row echelon form:")
pprint(A.rref()[0])
# Is singular?
print("Is singular:", singular)
# Trace
try:
    print("Trace:", A.trace())
except:
    print("Trace: Unobtainable for non-square matrix!")
# Rank
print("Rank:", A.rank())
# Nullity
nullspace = A.nullspace()
nullity = len(nullspace)
print("Nullity:", nullity)
# Nullspace
if nullity == 0:
    print("Nullspace: Empty")
else:
    print("Nullspace:")
    for M in nullspace:
        pprint(M)
```

```
MATRIX 1
Rows    : 3
Columns : 3

Enter row by row, each element in the row separated by comma…
R1: 32,-2, 0
R2: 21, 0, 231
R3: 2, 3, -3
Your matrix:
 32.0  -2.0   0.0

 21.0  0.0   231.0

 2.0   3.0   -3.0
------------
Results of certain functions on this matrix…
Determinant: -23226.0000000000
Row echelon form:
```

```
1  0  0

0  1  0

0  0  1
Is singular: False
Trace: 29.0000000000000
Rank: 3
Nullity: 0
Nullspace: Empty
```

4.1 NOTES

4.1.1 Nullspace

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.

5 CONCLUSION

As we can see, SymPy matrices offer feaures and functions such as rank, nullspace and row echelon forms, that are widely used and very important in linear algebra. On top of this, pretty printing function as well a the default rendering of SymPy matrices makes them much more appealing to present and study.