# Numerical Computing :: Project Four


## Julia Troni


In [1]:
```
%matplotlib notebook
from matplotlib import pyplot
import matplotlib.pyplot as plt
import numpy as np
import math
```

# Numerical Experiements, Method Implementation, and Data Visualization

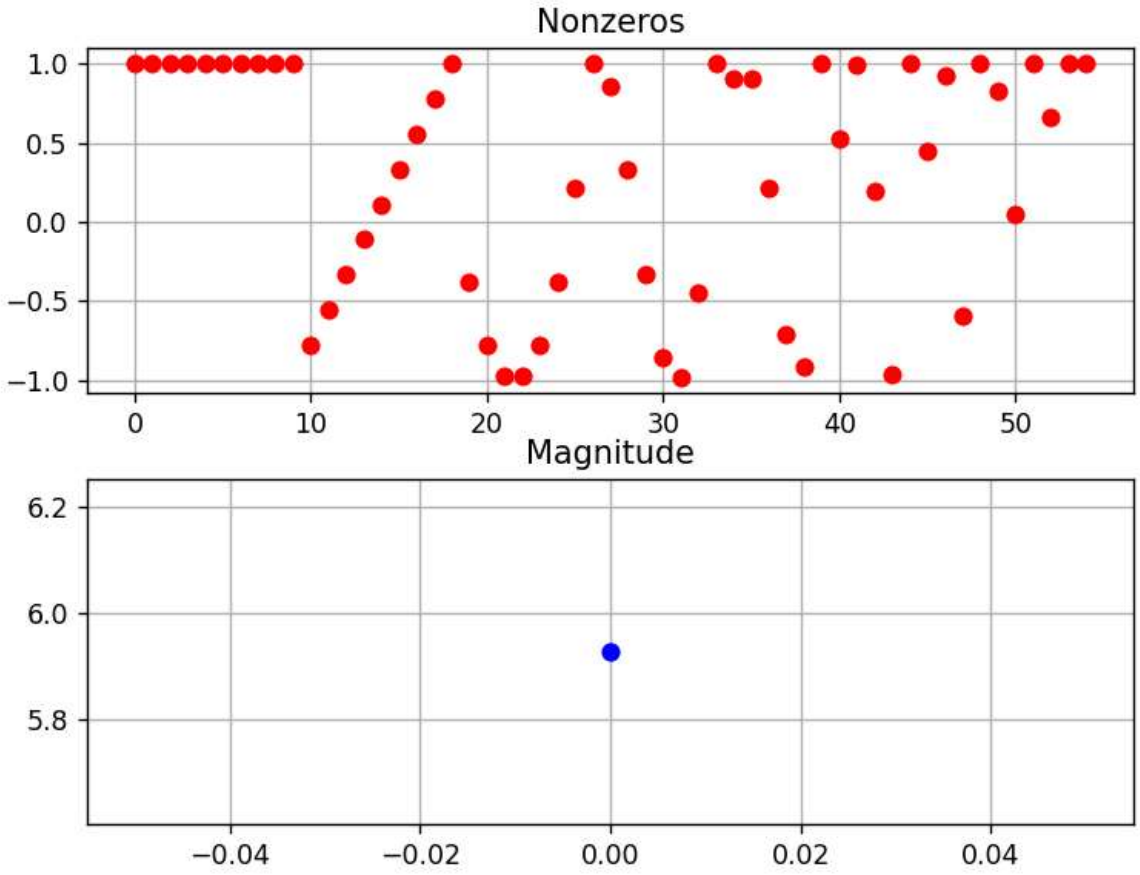
In [2]:
```
def MatrixEvaluation(mat):
    #1. What are the matrix dimensions?
    dim=mat.shape
    print("Dimensions: ", dim)

    #2. How many nonzeros are there?
    nonzeroMatrix= mat[mat.nonzero()]
    nonzeroCount= len(nonzeroMatrix)
    print("Nonzero elements :", nonzeroCount )

    #3. Is it symmetric?
    ##first transpose
    trans= mat.transpose()
    # now compare matrices using array_equal() method
    if np.array_equal(trans, mat):
        print("Symmetric")
    else:
        print("Not Symmetric")

    #4. Is it diagonal?
    checkDiagonal= np.all(mat == np.diag(np.diagonal(mat)))
    print("Is it diagonal? ", checkDiagonal)

    #5. Is it orthogonal?

    #6. What is the rank?
    rank= np.linalg.matrix_rank(mat)
    print("The rank is: ", rank)

    #7. What is the smallest singular value?
    small=mat.min()
    print("The smallest singular value is: ", small)

    # 8. What is the largest singular value?
    big=mat.max()
    print("The largest singular value is: ", big)

    # 9. What is the condition number?
    print("The condition number is:  ", np.linalg.cond(mat))
```

```
    #find magnitude of the elements of the matrix
    magnitude = np.linalg.norm(mat)
    print("The Magnitude is: ", magnitude)

    # Generate five random right-hand-sides.
    #For each right-hand-side b, try to solve Ax = b with the appropriate solver (like
    #Did the solver have any issues solving the systems?
    bs = []
    flag = True
    for i in range(5):
        bs.append(np.random.rand(mat.shape[0],1))
        try:
            solve = np.linalg.solve(mat,bs[i])
        except:
            flag= False
            print("Solver had problems")
            break;
    if (flag):
        print("Solving success")

    plt.figure()
     #Plot the nonzero elements of the matrix.
    plt.subplot(2,1,1)
    plt.plot(nonzeroMatrix, 'ro')
    plt.title('Nonzeros')
    plt.grid(True)

    # Plot the magnitude of the elements of the matrix.
    plt.subplot(2,1,2)
    plt.plot(magnitude, 'bo')
    plt.title('Magnitude')
    plt.grid(True)
    plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25,
                        wspace=0.35)

    plt.show()
    return;
```

In [3]:

```
mat1= np.loadtxt('mat1.txt',dtype=float, encoding=None, delimiter=",")
MatrixEvaluation(mat1)
```

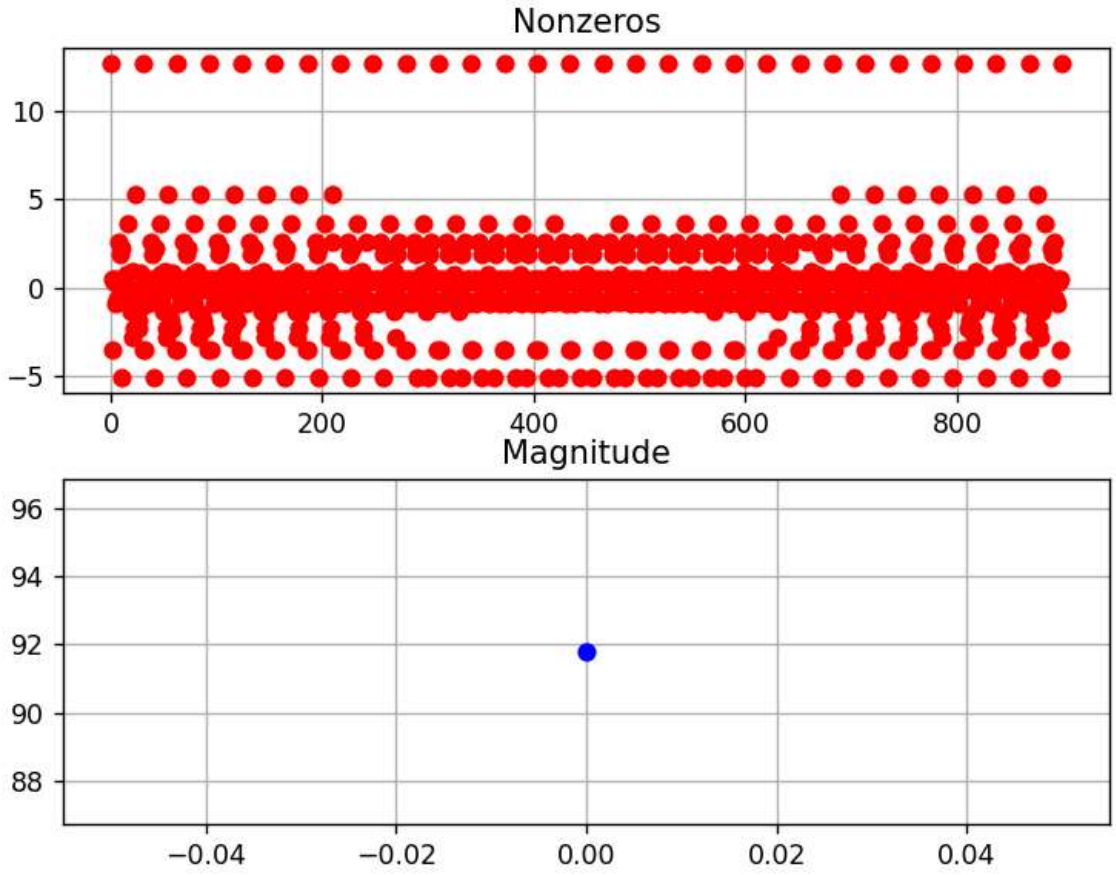
```
Dimensions:  (10, 10)
Nonzero elements : 55
Not Symmetric
Is it diagonal?  False
The rank is:  10
The smallest singular value is:  -0.98079561042524
The largest singular value is:  1.0
The condition number is:   124.39975871662227
The Magnitude is:  5.927702902597504
Solving success
```




In [4]:

```
mat2= np.loadtxt('mat2.txt',dtype=float, encoding=None, delimiter=",")
MatrixEvaluation(mat2)
```

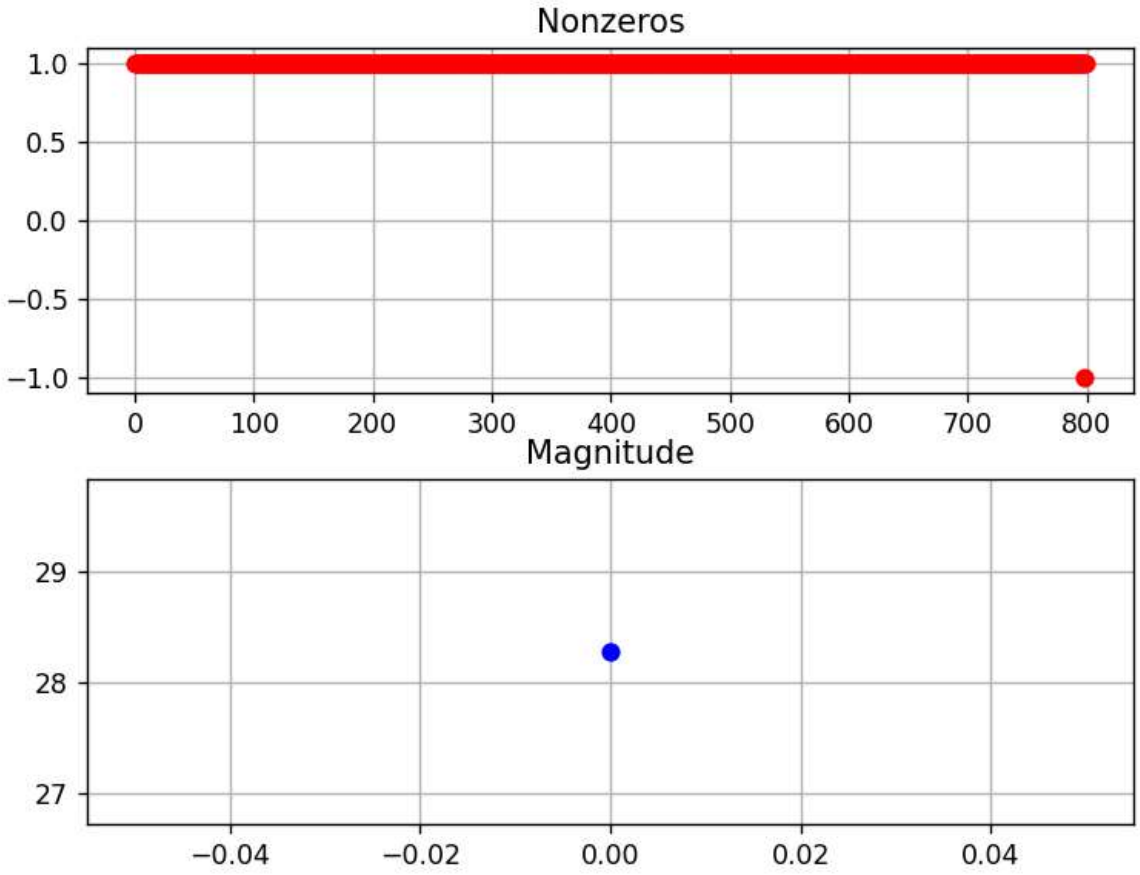
```
Dimensions:  (30, 30)
Nonzero elements : 900
Symmetric
Is it diagonal?  False
The rank is:  30
The smallest singular value is:  -5.043681919480303
The largest singular value is:  12.633181597604256
The condition number is:   206.6726633597007
The Magnitude is:  91.78606079905931
Solving success
```




In [5]:

```
mat3= np.loadtxt('mat3.txt',dtype=float, encoding=None, delimiter=",")
MatrixEvaluation(mat3)
```

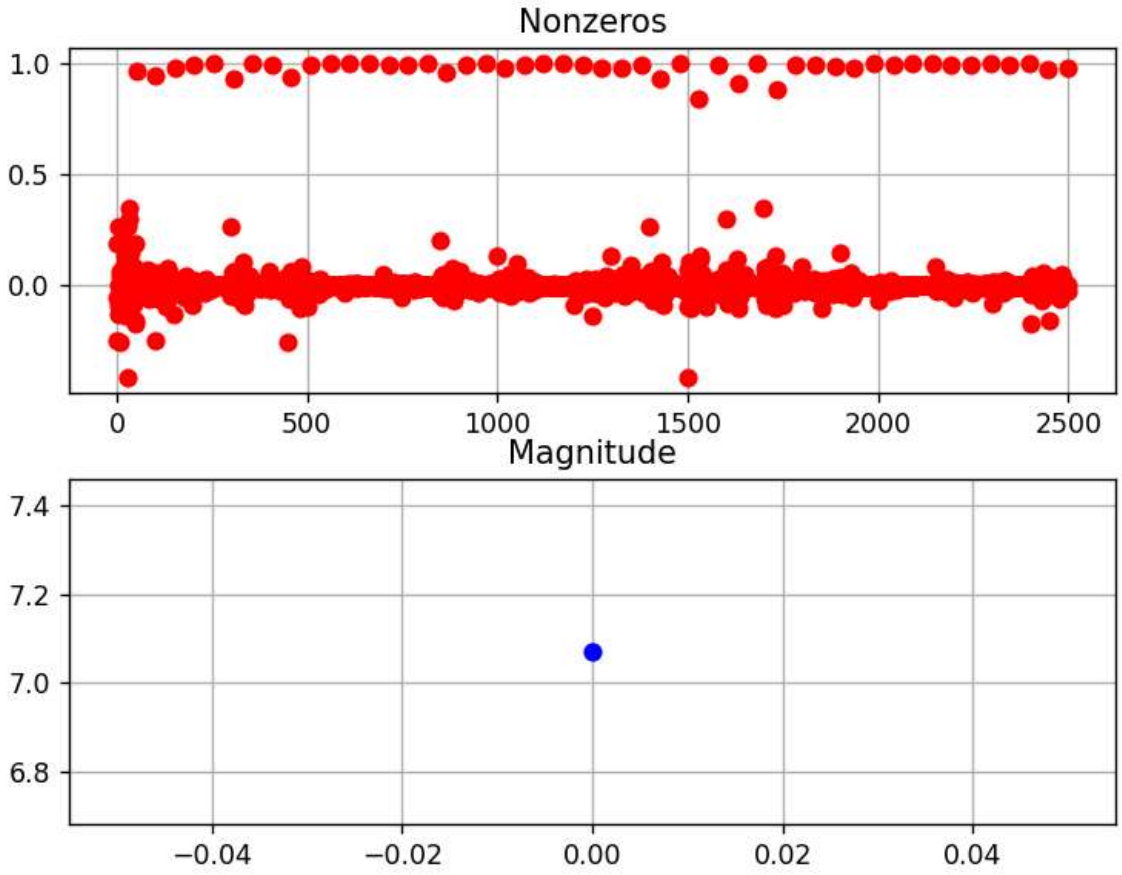
```
Dimensions:  (400, 400)
Nonzero elements : 800
Not Symmetric
Is it diagonal?  False
The rank is:  399
The smallest singular value is:  -1.0
The largest singular value is:  1.0
The condition number is:   3.1340633742956184e+16
The Magnitude is:  28.284271247461902
Solver had problems
```




In [6]:

```
mat4= np.loadtxt('mat4.txt',dtype=float, encoding=None, delimiter=",")
MatrixEvaluation(mat4)
```

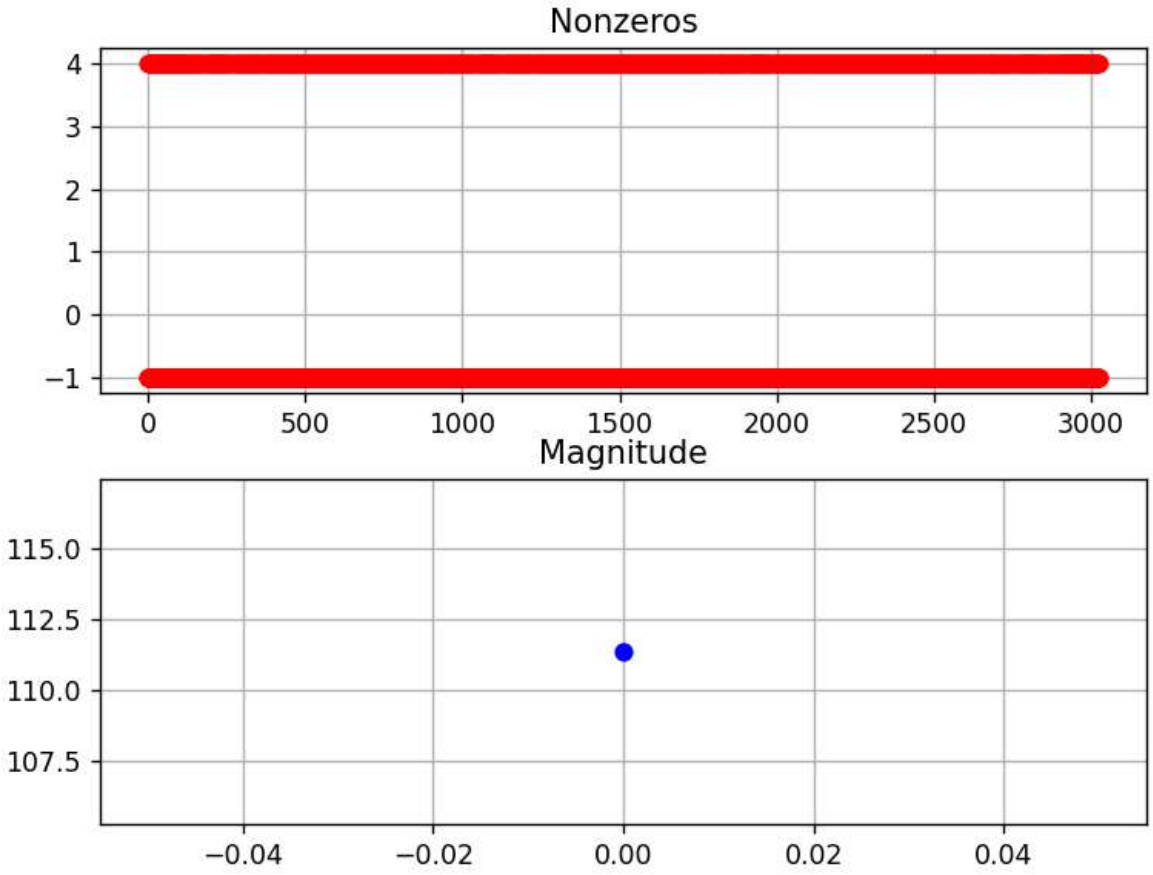
```
Dimensions:  (50, 50)
Nonzero elements : 2500
Not Symmetric
Is it diagonal?  False
The rank is:  50
The smallest singular value is:  -0.41156824931968294
The largest singular value is:  0.9999995198007274
The condition number is:   1.0000000000000016
The Magnitude is:  7.0710678118654755
Solving success
```




In [7]:

```
mat5= np.loadtxt('mat5.txt',dtype=float, encoding=None, delimiter=",")
MatrixEvaluation(mat5)
```

```
Dimensions:  (625, 625)
Nonzero elements : 3025
Symmetric
Is it diagonal?  False
The rank is:  625
The smallest singular value is:  -1.0
The largest singular value is:  4.0
The condition number is:   273.3060573767079
The Magnitude is:  111.35528725660043
Solving success
```




## Observations

- It was interesting that I noticed that depending on the randomly generated b's, the solver was sometimes successful and sometimes failed, although I only observed this for matrix 3. I am curious if this is due to the linear distribution of nonzero elements as that appears to be the only significant distinguishing factor between matrix 3 and the others

## References


- this was incredibly useful https://pythonnumericalmethods.berkeley.edu/notebooks/chapter14.01-Basics-of-Linear-Algebra.html
- https://www.geeksforgeeks.org/numpy-nonzero-in-python/#:~:text=nonzero()%20function%20is%20used,arr%5Bnonzero(arr)%5D%20.
- https://www.delftstack.com/howto/numpy/python-numpy-magnitude/
- https://matplotlib.org/stable/tutorials/introductory/pyplot.html


In [ ]: