Topic: Linear Algebra

Concepts: Eigenvalue problem

Note-1: *Feel free to use resources from the internet to help you write the python script, find commands, or understand concepts.*

Note-2: *This assignment uses random numbers, so no two student submissions show have exactly the same solutions.*

(1) Provide the python script to create a **vector** of size N (i.e. N number of elements) with all elements randomly generated within the interval [0,1]. Execute the code for N=10 and show the result.

(2) Provide the python script to create a **full matrix** of size NxN with all elements randomly generated within the interval [0,1]. Execute the code for N=4 and show the result.

```
import numpy as np

n = 3  # Example size
matrix_uniform = np.random.rand(n, n)
```

(3) Provide the python script to create a **sparse matrix** of size NxN with all elements randomly generated within the interval [0,1]. Execute the code for N=4 and show the result. Change the “density” value so that you have at least 3 non-zero elements in the matrix.

```
from scipy import sparse
import numpy as np

# Define the dimensions of the square matrix
n = 100

# Define the density of non-zero elements (e.g., 0.01 for 1% non-zero)
density = 0.01

# Generate a random sparse matrix in COO format (default)
# The 'format' parameter can be set to 'csc', 'csr', 'lil', etc.
# The 'random_state' parameter ensures reproducibility
sparse_matrix = sparse.rand(n, n, density=density, format="csr", random_state=42)

# You can convert to a dense NumPy array for inspection (for small matrices)
# dense_matrix = sparse_matrix.toarray()
# print(dense_matrix)

print(f"Shape of the sparse matrix: {sparse_matrix.shape}")
print(f"Number of non-zero elements: {sparse_matrix.nnz}")
print(f"Sparsity of the matrix: {1 - (sparse_matrix.nnz / (n * n)):.4f}")
```

(4) Find eigenvalues of a **full matrix** of random numbers (size NxN) using the "eig" function of *numpy* for N={10,100,500,1000,5000,10000,50000,10^5, 5*10^5}. For each case, calculate the time taken for solving the problem and make a log-log plot of N vs. time taken. Check whether the scaling is O(N^3), especially for larger values of N. Report any issues that you may have faced. You may use the code below if you need it. Also, try running for each N value separately instead of using a "for" loop.

```
import time

# "tic" - record the start time
start_time = time.perf_counter()

# Code to be timed
for i in range(1000000):
    _ = i * 2

# "toc" - calculate elapsed time
end_time = time.perf_counter()
elapsed_time = end_time - start_time

print(f"Elapsed time: {elapsed_time:.6f} seconds")
```

(5) Find 5 largest (magnitude) eigenvalues of a **sparse matrix** (refer problem 3) of random numbers (size NxN) using the "eigs" function of *scipy* for N={10,100,500,1000,5000,10000,50000,10^5, 5*10^5}. For each case, calculate the time taken for solving the problem and make a log-log plot of N vs. time taken. Check whether the scaling is O(N). Report any issues that you may have faced. You may use the code below if you need it. Also, try running for each N value separately instead of using a "for" loop.

```
import numpy as np
from scipy.sparse.linalg import eigs
```

```
# Find the 3 largest magnitude eigenvalues
n_eigenvalues = 3
eigenvalues, eigenvectors = eigs(A, k=n_eigenvalues, which='LM')

print("First", n_eigenvalues, "eigenvalues:", eigenvalues)
```

(6) Generate a random sparse matrix (with a fixed random seed or random state) of size N=1000. Using "eigs", for the same matrix, find "k" largest magnitude eigenvalues, where k={5,10,20,50,100}. Print the first 5 eigenvalues obtained in each case to 15 decimals and compare the accuracy as more and more eigenvalues (k) are requested to be solved. You will find that as more eigenmodes are solved for, you get more accurate eigenvalues. Also, report the time taken in each calculation (as a table or plot). As k increases, the time taken also increases (indicating that more iterations are being carried out). Use a symmetric matrix:

```
density=2/100 # define 2% zeros in the ranom sparse matrix
sparse_matrix=sparse.rand(n,n,density=density,format="csr",random_state=50)
A_sparse = sparse_matrix
A_sparse = (A_sparse + A_sparse.T) / 2   # make it symmetric
```

Note that you should not generate a new random matrix for each time you are using a different value of "k", instead use a fixed sparse matrix generated using a fixed random seed.

See:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.eigsh.html

```
# compute k eigenvalues
eigvals, eigvecs = eigsh(A_sparse, k=k,which='LM')
```

Notes: In structural dynamics, it is equivalent to finding the first few natural modes and frequencies accurately. For a structure with stiffness matrix size going into 1000x1000 or more, it is required to calculate several of the eigenmodes (say 100 or 200) to ensure the first 30-50 modes that are used in mode superposition are calculated accurately. "eigs" uses an iterative method whereas "eig" uses a direct method. Hence, "eigs" gives approximate results and accuracy improves only if more iterations are carried out (which is equivalent to requesting calculation of more number of eigenvalues).

(7) Generate two random vectors (N=4). Find a 3rd vector which is linearly dependent on the two of them.

(8) Generate four random vectors (N=3). Find whether they are linearly independent or dependent.