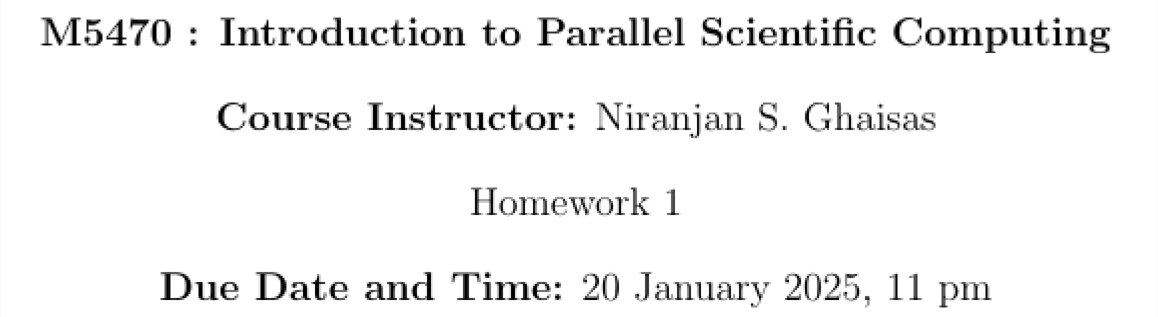


M5470 : Introduction to Parallel Scientific Computing

Course Instructor: Niranjan S. Ghaisas

Homework 1

Due Date and Time: 20 January 2025, 11 pm

Name: **Abhishek Kumar**
Roll: **CO22BTECH11001**

Q1):

- Designed to read a matrix to read a matrix size **n** from an input file, generate a 2D array of dimensions **n x n** filled with predefined data, and write the array to output files in both ASCII and binary formats.

Below Steps

- **Input Handling:** Reads the size of the array **n** from a file named input.in. Ensures the input file exists before proceeding.
- **Dynamic Memory Allocation:** Allocated memory for a 2D array (double**) dynamically using malloc of size n with each element 1D array (double *) of size n.
- **Array Initialization:** Filled the array with data, where each element is the sum of its row and column indices (i + j).
- **File Writing:** Calls print_to_file twice to save the array in **ASCII** and **binary formats**.
- **Memory Deallocation:** Frees all dynamically allocated memory to prevent memory leaks.

File name with its format and size

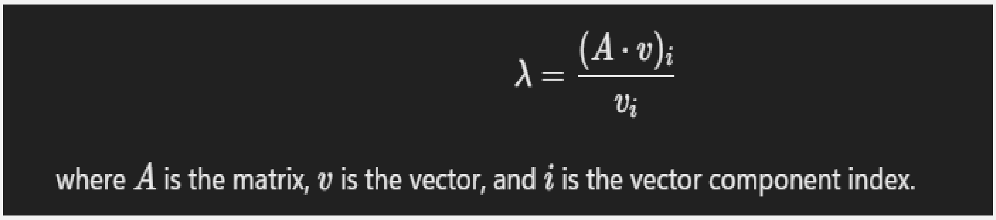
File Type	Name	Size
ASCII	array_004000_asc.out	319MB
Binary	array_004000_bin.out	122MB

Q2):

- I have read matrices and vectors from input files to determine if each vector is an eigenvector of the corresponding matrix.
- If a vector is identified as an eigenvector, its associated eigenvalue is appended to the vector's file. The program demonstrates matrix-vector operations, eigenvalue computation, and file handling in C.

Major steps:

1. **Matrix and Vector Reading:**
    - The `readMatrix` and `readVectors` functions read matrices and vectors from files into dynamically allocated memory.
    - Input files are named systematically using matrix size (n) and vector indices.
2. **Eigenvalue Computation:** The findEigenval function computes the eigenvalue for a given matrix and vector using
    - If the eigenvalues across components are within a specified tolerance, the vector is confirmed as an eigenvector.

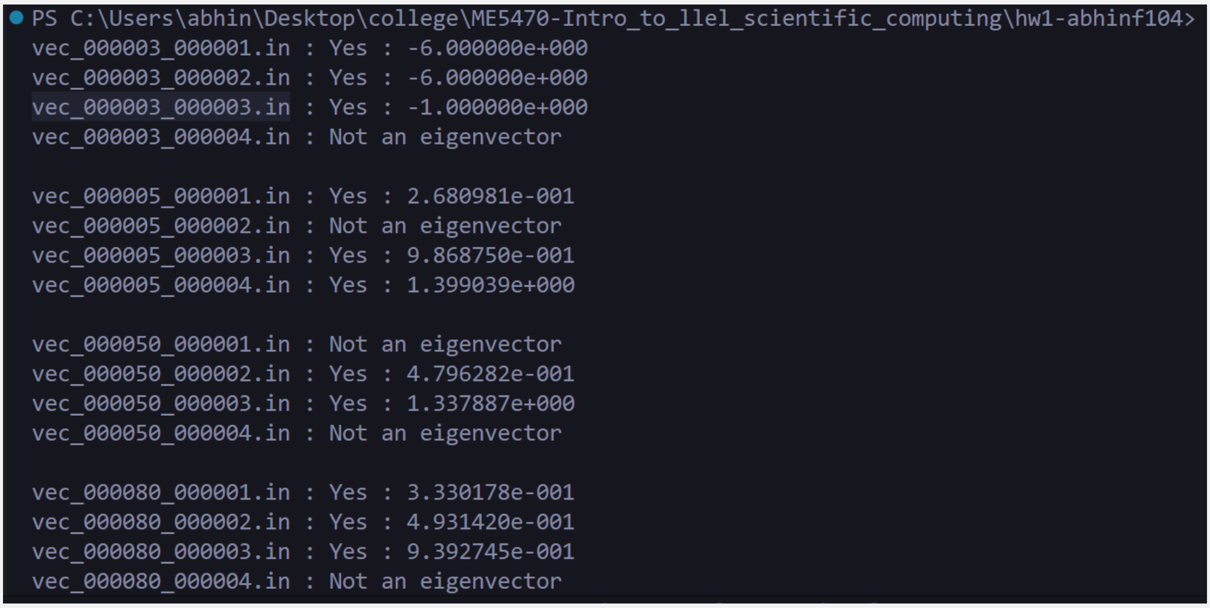


$$\lambda = \frac{(A \cdot v)_i}{v_i}$$

where A is the matrix, v is the vector, and i is the vector component index.

3. **File Handling:**
    - Eigenvalues are appended to vector files using the *appendEigenvalue* function.
    - Input files and output operations are handled systematically, ensuring clear organization.
4. **Memory Management:**
    - The program uses dynamic memory allocation (malloc) for matrices and vectors.
    - The *freeMemory* function ensures proper deallocation of the allocated memory.

5. **Input Structure:**
    - The input.in file specifies the matrix sizes to be processed.
    - Matrix and vector input files are stored in an **inputfiles/** directory with consistent naming conventions.
6. **Output:**
    - Prints results for each vector:
        - If it is an eigenvector, the eigenvalue is displayed.
        - Otherwise, it notes that the vector is not an eigenvector.



```
PS C:\Users\abhin\Desktop\college\ME5470-Intro_to_llel_scientific_computing\hw1-abhinf104>
 vec_000003_000001.in : Yes : -6.000000e+000
 vec_000003_000002.in : Yes : -6.000000e+000
 vec_000003_000003.in : Yes : -1.000000e+000
 vec_000003_000004.in : Not an eigenvector

 vec_000005_000001.in : Yes : 2.680981e-001
 vec_000005_000002.in : Not an eigenvector
 vec_000005_000003.in : Yes : 9.868750e-001
 vec_000005_000004.in : Yes : 1.399039e+000

 vec_000050_000001.in : Not an eigenvector
 vec_000050_000002.in : Yes : 4.796282e-001
 vec_000050_000003.in : Yes : 1.337887e+000
 vec_000050_000004.in : Not an eigenvector

 vec_000080_000001.in : Yes : 3.330178e-001
 vec_000080_000002.in : Yes : 4.931420e-001
 vec_000080_000003.in : Yes : 9.392745e-001
 vec_000080_000004.in : Not an eigenvector
```