ME5470 : Introduction to Parallel Scientific Computing

HOMEWORK 1 Report

Abhinav kalala

Co21btech11007

1.

Explanation:

1. Input Reading: The program reads the matrix size n from the file input.in.
2. Dynamic Allocation: A 2D array of size n×n is dynamically allocated using malloc.
3. Array Filling: Each element of the array is filled using the formula A[i][j]= i+j
4. File Writing: The function print_to_file writes the array to either an ASCII or binary file based on the format_flag.
    - For ASCII, the data is written using fprintf with 15-decimal precision.
    - For binary, the data is written using fwrite.
5. Filename Generation: The filename includes a zero-padded representation of n.
6. Memory Deallocation: Allocated memory is freed after use.

After compiling we use commands

du -sh array_004000_asc.out

du -sh array_004000_bin.out

output is

123M array_004000_bin.out

320M array_004000_asc.out

1. Binary File (array_004000_bin.out):
    - Size: 123 MB
    - Reason: Each element in the 4000 × 4000 array is a double (8 bytes). The size calculation is: 4000×4000 × 8=128,000,000 bytes (approx. 123 MB).
    - This matches the observed size of the binary file.
2. ASCII File (array_004000_asc.out):
    - Size: 320 MB

- Reason: Each double is stored as a human-readable number with 15 decimal places. On average, a number like 12345.678901234567 takes approximately 20 bytes (including spaces or newline characters). The size calculation is: $4000 \times 4000 \times 20 = 320{,}000{,}000$ bytes (approx. 320 MB).
- This aligns with the observed size of the ASCII file.

Memory Size of the Array:

- In Memory:
  - The array is stored as raw double values in memory, so the size is: $4000 \times 4000 \times 8 = 128{,}000{,}000$ bytes (128 MB).

Comments on File Sizes:

1. Binary Format:
   - Much smaller on disk since it directly stores raw data without extra characters for formatting.
   - Faster to write and read, but not human-readable.
2. ASCII Format:
   - Larger due to extra characters (spaces, newline) and conversion of binary data to human-readable form.
   - Slower to write and read but easier for debugging and manual inspection.

2.

The program implements an algorithm to verify eigenvectors of a n×n matrix and find their corresponding eigenvalues. Here's the analysis:

Program Structure and Implementation:

- The program reads a n×n matrix and multiple test vectors from separate input files

- Uses a numerical approach with epsilon tolerance (1e-6) for floating-point comparisons

- Implements efficient matrix-vector multiplication and eigenvector verification

Test Results for n = 3:

1. Vector [1, 1, 1]:

  - Result: Not an eigenvector

- Verification: Av = [3, 4, 3] which is not a scalar multiple of v

2. Vector [1, 0, -1]:

- Result: Is an eigenvector
- Eigenvalue: 2.000000
- Verification: Av = [2, 0, -2] = 2[1, 0, -1]

3. Vector [1, 2, 1]:

- Result: Not an eigenvector
- Verification: Av = [4, 6, 4] is not a scalar multiple of v

Key Mathematical Findings:

- The matrix has at least one eigenvector [1, 0, -1] with eigenvalue 2
- The symmetric nature of the matrix guarantees all eigenvalues are real
- The program successfully distinguishes between true eigenvectors and non-eigenvectors

Test Results for n = 5:

1. Vector [0, 0, 0, 0, 5]:
    - Result: Is an eigenvector
    - Eigenvalue: 2.000000
    - Verification: A × v=[0,0,0,0,10] which is 2×[0,0,0,0,5]
2. Vector [0, 0, 0, 0, 10]:
    - Result: Is an eigenvector
    - Eigenvalue: 3.000000
    - Verification: A × v= [0,0,0,0,30] which is 3×[0,0,0,0,10]
3. Vector [0, 0, 0, 5, 0]:
    - Result: Not an eigenvector
    - Verification: A × v = [0,0,0,15,0] is not a scalar multiple of v
4. Vector [0, 0, 5, 0, 0]:
    - Result: Not an eigenvector

- Verification: A × v = [0,0,15,0,0] is not a scalar multiple of v

Key Mathematical Findings:

- The matrix has eigenvectors with eigenvalues 2 and 3.
- The vectors [0, 0, 0, 0, 5] and [0, 0, 0, 0, 10] are confirmed eigenvectors with corresponding eigenvalues 2 and 3, respectively.
- The vectors [0, 0, 0, 5, 0] and [0, 0, 5, 0, 0] do not satisfy the eigenvector conditions, as their matrix products are not scalar multiples of the original vectors.

File Handling:

- Successfully processes sequential vector files

- Appends eigenvalues to vector files when eigenvectors are identified

- Handles file I/O errors appropriately

The program demonstrates robust numerical computation and accurate eigenvector verification for the given n×n matrix case.