ME5470-Introduction to Parallel Scientific Computing

Homework – 1 (Report)

Parthib Ghosh CO22BTECH11007

Answer 1:

a) After executing the code, we got two files – one is **ASCII** file, where the data is stored in ASCII format and another one is **binary** file, where the data is stored in binary format.

size of the ASCII file : 351 MB
size of the binary file : 122 MB

b) Size of double : 8 bytes, n = 4000
size of the file in memory : 8 x 4000 x 4000 = 128000000 bytes ≈ 128 MB
size in disk for ASCII format : 351 MB
size in disk for binary format : 122 MB

- Bin format takes less storage in disk than ASCII format because it's a direct dump of memory representation. The ASCII format takes more memory due to the text representation of numbers, decimal places, line endings, etc.
- A binary format is better for storing large data due to the following reasons
  - It's more space efficient.
  - Faster to read and write.
  - Maintains full numerical precision.
  - No risk of floating-point representation errors.
  - Can be easily compressed for further storage and transmission.

Code Explanation

- Declare the variables `double n` and `double** A`.
- Read the value of `n` from `input.in`, make sure to close the file after reading the value from it.

- Allocate memory for the 2D array of size `n x n` in heap using `malloc()`.
- Fill the array with mentioned values.
- Write into a binary file and ASCII file using the function `print_to_file()` with appropriate format flag.
    - Create the mentioned formatted file name.
    - Open the file in write mode, `w` for ASCII file and `wb` for binary file.
    - Write into the file using `fprintf()` and `fwrite()`.
- Free all the memory created in the heap.

Answer 2:

```
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
```

The above figures shows the results, showing whether the given vectors are the eigen vectors of the given matrix.

Here we have used the following algorithm to check whether a given vector is an eigen vector or not.

- We are given a matrix $A_{n\times n}$ and a vector $\vec{x}_{n\times 1}$.
- Multiply the matrix with the vector i.e., $\vec{y} = A\vec{x}$
- To be an eigen vector, the given vector must satisfy the following eigen vector equation

$$A\vec{x} = \lambda\vec{x}$$

- Now the get the value of λ using the following way.

$$\vec{y} = A\vec{x} = \lambda\vec{x}$$
$$\Rightarrow \vec{y}\cdot\vec{x} = \lambda\vec{x}\cdot\vec{x}$$

$$\Rightarrow \lambda = \frac{\vec{y} \cdot \vec{x}}{\vec{x} \cdot \vec{x}}$$

- Now using the obtained value of λ check the difference between each value of $\vec{y}$ and $\lambda\vec{x}$.
- If any one difference the not close to zero (some tolerance), the it is not an eigen vector.

Code Explanation

- Declare the variables `double n`, `double** A`, `double* x`, etc.
- Read the value of `n` from `input.in`, make sure to close the file after reading the value from it.
- Allocate memories for the respective 2D and 1D arrays.
- Read the matrix with the given formatted name.
- Perform the following task for each vector.
    - Read the vector with the given formatted name scheme.
    - Check whether it is an eigen vector using the above explained algorithm.
    - Print whether it is a eigenvector or not.
- Free the allocated memory.