CO21BTECH11002
Aayush Kumar
ME5470 HW1 Report

Q1

a)

```
co21btech11002@edison9:~/ME5470/hw1-Random-Bee/codes$du -sh array_004000_asc.out
320M    array_004000_asc.out
co21btech11002@edison9:~/ME5470/hw1-Random-Bee/codes$du -sh array_004000_bin.out
123M    array_004000_bin.out
```

Type	Size
ASCII	320 MB
Binary	123 MB

b)

The total size of the array in memory is given by:

$$Size\ in\ memory\ (bytes)\ = Size\ of\ one\ double\ (bytes) \times n^2$$

For a given value of n, this can be expressed in megabytes (MB) as:

$$Size\ in\ MB\ = \frac{Size\ in\ memory\ (bytes)}{1024^2}$$

For n = 4000

$$Size\ in\ MB\ = \frac{8 \times 4000 \times 4000}{1024 \times 1024}$$

$$Size\ in\ MB\ = 122.07$$

Comparison:

- The size of the binary file (123 MB) matches the size of the array in memory because the binary format directly writes the raw memory representation of the data.

- The ASCII file (320 MB) is larger because it uses human-readable decimal representations, which require additional space for characters (digits, decimal point, and delimiters).

Binary format is best suited for saving large data as it is compact and preserves full precision while consuming less disk space. ASCII format should be used only if the data needs to be human-readable or needs to be used with tools that do not support binary formats.

Q2

n = 3

```
vec_000003_000001.out : Yes : -6.000000
vec_000003_000002.out : Yes : -6.000000
vec_000003_000003.out : Yes : -1.000000
vec_000003_000004.out : Not an eigenvector
```

n = 5

```
vec_000005_000001.out : Yes : 0.268098
vec_000005_000002.out : Not an eigenvector
vec_000005_000003.out : Yes : 0.986875
vec_000005_000004.out : Yes : 1.399039
```

n = 50

```
vec_000050_000001.out : Not an eigenvector
vec_000050_000002.out : Yes : 0.479628
vec_000050_000003.out : Yes : 1.337887
vec_000050_000004.out : Not an eigenvector
```

n = 80

```
vec_000080_000001.out : Yes : 0.333018
vec_000080_000002.out : Yes : 0.493142
vec_000080_000003.out : Yes : 0.939275
vec_000080_000004.out : Not an eigenvector
```