ME5470: Introduction to Parallel Scientific Computing Homework 1 Report

Om Dave (CO22BTECH11006)

January 20, 2025

Question 1

(a) File Sizes

For $n = 4000$ and `format.flag` set to 0 (binary) and then 1 (ASCII), we generated two files:

- `array_004000_bin.out`
- `array_004000_ascii.out`

Using the command `du -sh`, we obtained:

```
array_004000_bin.out    -> 123M
array_004000_ascii.out  -> 320M
```

(b) Memory Size Estimation

The array in memory has size

$$n \times n \quad \text{with each entry a double (8 bytes).}$$

Hence, for $n = 4000$,

$$\text{size in bytes} = n^2 \times 8 = 4000 \times 4000 \times 8 = 128 \times 10^6 \text{ (bytes)} = 128\,\text{MB}.$$

Comparing this to the file sizes on disk:

- **Binary file:** 123 MB
- **ASCII file:** 320 MB

Note that the binary file on disk (123 MB) is close to the raw memory footprint (128 MB), while the ASCII file is significantly larger (320 MB). Thus, for large data sets, binary output is preferable because it is more space-efficient on disk.

Question 2

We wrote a code that checks whether a given vector $\mathbf{x}$ is an eigenvector of a matrix $\mathbf{A}$, and if so, computes the corresponding eigenvalue. The key idea is to see if

$$\mathbf{Ax} \;=\; \lambda\mathbf{x}$$

for some scalar λ.

Algorithm Outline

1. **Read matrix and vectors:** We first read n from `input.in`, then read the $n \times n$ matrix $\mathbf{A}$ from `mat_{nnnnnn}.in`, and subsequently read one or more candidate vectors $\mathbf{x}$ from `vec_{nnnnnn}-{vecnum}.in`.

2. **Compute $\mathbf{p} = \mathbf{Ax}$**: We allocate a vector $\mathbf{p}$ of size n and perform the standard matrix-vector multiplication:

$$p_i \;=\; \sum_{j=1}^{n} A_{ij}\, x_j, \quad i = 1, \ldots, n.$$

3. **Check the eigenvector property:** If $\mathbf{x}$ is truly an eigenvector, then there exists a constant λ such that

$$\mathbf{p} = \lambda\mathbf{x}.$$

   To check numerically, for each consecutive pair of indices $(i, i+1)$, we see if the ratios $\frac{p_i}{x_i}$ and $\frac{p_{i+1}}{x_{i+1}}$ match within a small tolerance ε. In code, we often rearrange this ratio check to avoid dividing by zero:

$$p_i\, x_{i+1} \;\approx\; p_{i+1}\, x_i.$$

   If any pair violates this within the tolerance ($\texttt{fabs}(p_i \cdot x_{i+1} - p_{i+1} \cdot x_i) > \varepsilon$), we conclude $\mathbf{x}$ is *not* an eigenvector and return `NULL`.

4. **Compute the eigenvalue λ** (if valid): If the vector passes the above checks, we pick the first nonzero entry x_k of $\mathbf{x}$ and set

$$\lambda \;=\; \frac{p_k}{x_k}.$$

   Then we know $\mathbf{p} = \lambda\mathbf{x}$. We print this λ and also append it to the corresponding vector output file, if required.

5. **Role of ε:** Real-number computations in finite precision often incur small floating-point errors. We introduce a small threshold, ε (e.g., $\varepsilon = 10^{-9}$), to decide whether two real numbers are "close enough" to be considered equal.

Testing: The above routines were tested using provided input files for $n = 3$, $n = 5$, $n = 50$, and $n = 80$. In each case, the code outputs whether the given vector is an eigenvector or not, and if it is, prints the corresponding eigenvalue.

Outputs:

```
Reading matrix from file ./inputfiles/mat_000003.in
vec_000003_000001.in : Yes : -6.000000
vec_000003_000002.in : Yes : -6.000000
vec_000003_000003.in : Yes : -1.000000
vec_000003_000004.in : Not an eigenvector
```

Figure 1: Results for n = 3

```
Reading matrix from file ./inputfiles/mat_000005.in
vec_000005_000001.in : Yes : 0.268098
vec_000005_000002.in : Not an eigenvector
vec_000005_000003.in : Yes : 0.986875
vec_000005_000004.in : Yes : 1.399039
```

Figure 2: Results for n = 5

```
Reading matrix from file ./inputfiles/mat_000050.in
vec_000050_000001.in : Not an eigenvector
vec_000050_000002.in : Yes : 0.479628
vec_000050_000003.in : Yes : 1.337887
vec_000050_000004.in : Not an eigenvector
```

Figure 3: Results for n = 50

```
Reading matrix from file ./inputfiles/mat_000080.in
vec_000080_000001.in : Yes : 0.333018
vec_000080_000002.in : Yes : 0.493142
vec_000080_000003.in : Yes : 0.939275
vec_000080_000004.in : Not an eigenvector
```

Figure 4: Results for n = 80