CHED National Center of Excellence in Information Technology Education
INSTITUTE OF COMPUTER SCIENCE
College of Arts and Sciences
University of the Philippines Los Baños
College 4031, Laguna, PHILIPPINES
☎ Phone (63-49) 536-2313 ✆ Fax (63-49) 536-2302 ✆ Campus VoIP 6123
✉ E-mail jppabico@up.edu.ph 💻 Web www.ics.uplb.edu.ph/jppabico

CMSC 180
Introduction to Parallel Computing
Second Semester AY 2023-2024

Laboratory Research Problem 01
Computing the Pearson Correlation Coefficient of a Matrix and a Vector

Introduction

Given a $m \times n$ matrix $\mathbf{X}$ with m rows and n columns and a $m \times 1$ vector $\boldsymbol{y}$, a $1 \times n$ vector $\boldsymbol{r}$ holds the Pearson Correlation Coefficient of the columns in $\mathbf{X}$ and $\mathbf{y}$, such that

$$\boldsymbol{r}(j) = \{m\,[\mathbf{X}_j\mathbf{y}] - [\mathbf{X}]_j\,[\mathbf{y}]\} \;/\; \{[m\,[\mathbf{X}^2]_j - ([\mathbf{X}])^2][m\,[\mathbf{y}^2] - ([\mathbf{y}])^2]\}^{1/2} \qquad \text{(Equation 1)}$$

where,

$$\begin{aligned}
[\mathbf{X}]_j &= \textstyle\sum_{i=1\ldots m} \mathbf{X}(i,j) = \mathbf{X}(1,j) + \mathbf{X}(2,j) + \ldots + \mathbf{X}(m,j),\\
[\mathbf{X}^2]_j &= \textstyle\sum_{i=1\ldots m} \mathbf{X}(i,j)^2 = \mathbf{X}(1,j)^2 + \mathbf{X}(2,j)^2 + \ldots + \mathbf{X}(m,j)^2,\\
[\mathbf{y}] &= \textstyle\sum_{i=1\ldots m} \mathbf{y}(i) = \mathbf{y}(1) + \mathbf{y}(1) + \ldots + \mathbf{y}(m),\\
[\mathbf{y}^2] &= \textstyle\sum_{i=1\ldots m} \mathbf{y}(i)^2 = \mathbf{y}(1)^2 + \mathbf{y}(1)^2 + \ldots + \mathbf{y}(m)^2,\\
[\mathbf{X}_j\mathbf{y}] &= \textstyle\sum_{i=1\ldots m} \mathbf{X}(i,j)\mathbf{y}(i) = \mathbf{X}(1,j)\mathbf{y}(1) + \mathbf{X}(2,j)\mathbf{y}(2) + \ldots + \mathbf{X}(m,j)\mathbf{y}(m),
\end{aligned}$$

$\boldsymbol{r}(j)$ is the jth element of $\boldsymbol{r}$, and $\mathbf{X}(i, j)$ is the element in the ith row and jth column of $\mathbf{X}$. The vector $\boldsymbol{r}$ is said to be the Pearson Correlation Coefficient vector of a matrix $\mathbf{X}$ and a vector $\mathbf{y}$. Specifically, $\boldsymbol{r}(j)$ is the Pearson Correlation Coefficient of the jth column of $\mathbf{X}$ and $\mathbf{y}$, for all $j = 1, 2, \ldots, n$.

Research Question 1: What do you think is the complexity of solving the Pearson Correlation Coefficient vector of an $n \times n$ square matrix $\mathbf{X}$ with a $n \times 1$ vector $\mathbf{y}$? (*hint*: CMSC 142)

Research Activity 1: Write a computer program using the programming language of your choice for computing the Pearson Correlation Coefficient vector of an $n \times n$ square matrix $\mathbf{X}$ with a $n \times 1$ vector $\mathbf{y}$. In other words, transform Equation 1 above into a computer program given $\mathbf{X}$ and $\mathbf{y}$.

How to do it?

1. Write a function `pearson_cor` that accepts as parameters the matrix $\mathbf{X}$ and the vector **y,** and outputs the vector $\boldsymbol{v}$. For example, in pseudocode:

```
func pearson_cor(X as matrix, y as vector, m as integer, n as integer) as vector
begin
   define v(n) as vector;
   for i:=1 to n do
   begin
      v(i):=0;
      for j:=1 to m do
```

```
        begin
            v(i):= {see equation 1 above};
        end;
    end;
    pearson_cor:=v;
end;
```

2. Write the main program `lab01` that includes the following:
   (1) Read n as a user input (maybe from a command line or as a data stream);
   (2) Create a non-zero $n \times n$ square matrix **X** whose elements are assigned with random integers (make sure that any integer $i \neq 0$);
   (3) Create a non-zero $n \times 1$ vector **y** whose elements are assigned with random integers;
   (4) Create a $1 \times n$ vector ***v***;
   (5) Take note of the system time `time_before`;
   (6) call `pearson_cor(X, y, n, n);`
   (7) Take note of the system time `time_after`;
   (8) Obtain the elapsed time `time_elapsed:=time_after - time_before`;
   (9) output `time_elapsed`;

   For example, for computing the column sum of a 100×100 square matrix **M**:
   ```
   $ lab01 < 100
   $ time elapsed: 10.2345 seconds
   ```

3. Fill in the following table with your time readings:

n	**Time Elapsed (seconds)**			**Average Runtime (seconds)**	**Complexity***
	Run 1	**Run 2**	**Run 3**		
100					
200					
300					
400					
500					
600					
700					
800					
900					
1,000					
2,000					
4,000					
8,000					
16,000					
20,000					

*What does your answer to **Research Question 1** say, but converted into a time?

Research Question 2: Were you able to run up to $n > 10{,}000{,}000$? If so, can you make it higher to 50,000,000 or even 100,000,000? If not, why do you think so and what do you need to do to make it so?

4. Using a graphing software (such as LibreOffice Calc), create a line graph of *n* versus **Average** obtained from the Table above. On the same graph, plot *n* versus **Complexity** as well (at least up to the *n* where your program worked).

Research Question 3: Do the two lines agree, at least in the form? If not, provide an explanation why so?

Research Question 4: Discuss ways on how we can make it better (lower average runtime) without using any extra processors or cores (notice that the word "ways" is in plural form).