1. General Remarks

In this assignment you are asked to write pthread and openmp codes which

- find inverse A^{-1} of a matrix A,

via Gaussian elimination with partial pivoting. (Partial pivoting is needed for numerical stability).

2. A pseudo code

Say we want to solve for x the system of linear equations

$$Ax = b$$

where $A = (a_{i,j})$ is an $n \times n$ matrix and $b = (b_i)$ is an $n \times 1$ vector, both real. We will use Matlab notation

- $a_{:,j}$ to denote column j of A,
- $a_{i:j,k}$ to denote elements from row i to row j in column k of A, etc.

The Gaussian elimination method

- first transforms a matrix to an upper triangulat form, and
- next solves the upper triangular system of linear equations by backsubstitution.

The following is an outline of a pseudocode written for a sequential implementation of Gaussian elimination with partial pivoting.

```
/* Step 1: triangularization                     */
/* Matrix A and vector b are updated as follows */
/* step 1a, find a pivot                         */
for i = 0 to n - 1 {
   find the largest in absolute value element in a(i:n-1,i), say it is a(m,i)
   swap rows i and m of the current A
   swap b(i) with b(m)

/* step 1b, update a(i+1:n-1,i+1:n-1) and b(i:n-1)    */
   for j = i+1 to n-1 {
     p = a(j,i)/a(i,i)
```

```
      a(j,i:n-1) = a(j,i:n-1) - p*a(i,i:n-1)    /* now new a(j,i) = 0 */
      b(j) = b(j) - p*b(i)
   }
}
/* end of the "i" for loop                          */
/* now the new matrix A is upper trianugular        */

/* Step 2: backsubstitution                        */
for i = n-1 to 0 {
   x(i) = b(i)/a(i,i)
   b(0:i-1) = b(0:i-1) - a(0:i-1,i)*x(i)
}
```

(This is not the only way of implementing Gaussian elimination or backsubstitution. You can rearrange the computations in the above pseudocde as long as they produce correct results).

Remark 1: It is not always beneficial to distribute work among multiple thread when the workload is insignificant. For example, in the triangularization step, when `i` becomes close to `n-1`, there is not much work left. It that case it may be beneficial to let a single thread finish the work as there is an overhead for running multiple threads. Please try to exploit this observation in your code.

Now we want to adapt Step 1 and Step 2 to find the inverse A^{-1} of a matrix A. We can get the inverse in the following way.

- Execute the triangularization process in Step 1 abovea.
- Execute Step 2 for n rhs vector e_i, $i = 0, 1, ..., n-1$, where e_i is a vector of all 0s except for the position $i+1$ where it is 1.

  For example, when $n = 4$ we have

$$e_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad e_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad e_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

  Note that

$$(e_0\, e_1\, e_2\, e_3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  is the identity matrix of dimension 4.

  Let x_i be the solution to $Ax_i = e_i$. Then

$$(x_0\, x_1\, \cdots\, x_{n-1}) = A^{-1}.$$

3. Requirements

Templates for the `pthreads` and `openmp` codes will be placed in

`/classes/ece5720/assignments/hw2`

Please populate the $n \times n$ matrix A as done in the template.

(While developing your code you may want to set A to an upper triangular matrix of all integers, and set $b = e_3$. For example,

$$A = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This way you will be able to see whether your code performs correctly on this simple data.)

For finding the inverse of A

- triangularize A,
- create n rhs vectors as columns of an $n \times n$ identity matrix,
- repeat backsubstitution n time, once for each vector e_i, $i = 0, 1, ..., n-1$.

4. Checking for corretness.

For a general A and b, to check the numerical correctness of your algorithm, compute the following

$$r = Ax - b, \quad ||r||_2 = \sqrt{\sum_{i=1}^{n} r_i^2}$$

where r is called the residual vector and $||r||_2$ is its Euclidean norm (norms measure the length of a vector). Next compute norms of A and x

$$||A||_F = \left(\sum_{i,j=1}^{n} a_{i,j}^2 \right), \quad ||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Finaly, compute a normalized residual error

$$\rho = \frac{||r||_2}{||A||_F \cdot ||x||_2}.$$

ρ should be of the order of the machine relative precision, 10^{-5} for the single precision, 10^{-14} for the double precision (it can be larger for very large n).

For the case of inverting A, compute the normalized residual error for a single column of A^{-1}.

5. Benchmarking

Your algorithm should be benchmarked for a range of parameters n and p where

- n is the size of a matrix, followed by
- p, the number of threads.

n should start from MIN_DIM, then doubled until $n =$ MAX_DIM. Try to make MAX_DIM as large as the ecelinux cluster will allow you.

p should start from 1, then doubled until it is larger than the number of cores.

Measure the execution time for all pairs (n, p). (Do not include the time for numerical verification of your results).

It is good to include comments in your codes so it is clear what particular sections of codes do.

6. Write-up

Write a document that describes how your programs work. Sketch the key elements of your parallelization strategy. Explain how your program partitions the data and work among threads and how they are synchronized.

Explain whether the workload is distributed evenly among threads. Justify your implementation choices. Did you exploit the observation in Remark 1?

Compare results of the `pthreads` implementation with those of the `openmp` implemantation (ease of programming, timing results, etc.).

Your findings, a discussion of results, graphs and tables (if any) should be saved in a file

`your_net_id_hw2_writeup.pdf`.

Please present your tables and graphs in a way so they are easily readable.

Your codes (with instructions how to compile and execute it) should be saved in files

`your_net_id_hw2_pthread.c` and `your_net_id_hw2_openmp.c`.

All files should be archived and named `your_net_id_hw2.x` where `x` stands for `zip` or `tar`.