1. General Remarks

In this assignment you are asked to write a pthread code which

- solves a system of linear equtions
- inverts a matrix

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 */
for i = 1 to n - 1 {
   find the largest in magnitude element in a(i:n,i), say it is a(m,i)
   swap rows i and m of the current A
   swap b(i) with b(m)

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

```
      a(j,i:n) = a(j,i:n) - p*a(i,i:n)
      b(j) = b(j) - p*b(i)
   }
}

/* Step 2: backsubbstitution                         */
for i = n to 1 {
   x(i) = b(i)/a(i,i)
   for j = i - 1 to 1 {
     b(1:i-1) = b(1:i-1) - a(1: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).

If there are $K > 1$ right hand side vectors, Step 1 is performed only one time but Step 2 is repeated for each of the K right hand side vectors.

3. Requirements

A template for the code can be found in

`/classes/ece5720/assignments/hw2`

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

For solving a system with a single rhs vector b proceed as follows,

- set the true solution x_{true} to all 1s,
- calculate $b = A \cdot x$
- then, you can easily compare the calculated solution $x_{calculated}$ with x_{true}.

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.

To check the numerical correctness of your algorithm compute the following the following

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

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

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

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^{-6} for the single precision, 10^{-15} for the double precision.

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

4. 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 is 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 twice the number of cores is reached.

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

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.

Your findings, a discussion of results, graphs and tables 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. For example, if certain combinations of (number o threads, matrix dimension) do not bring any new information, you may omit them from your graphs. But then explain why you are omitting them.

Your code (with instructions how to compile and execute it) should be saved in a file

`your_net_id_hw2_code.c`.

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