MATH 440A/540A Parallel Scientific Computing
Homework Assignment 3
Last Submission Date: April 5 (8:00am)

This assignment must be entirely your own work. If any part of your assignment has been copied, then your mark may be reduced to zero. Any consultation (given/taken) must be acknowledged. Late assignments will be penalized at the rate of 5 marks per day. In fact, you should aim to submit the assignment several days before the due date because extensions will not be granted for reasons such as busy Lab or computer breakdowns.

Submit all Assignment 4 computer program files (with a readme.txt file containing details required to run your code and reproduce your reported output and of the mathematical procedure you used for developing your code, following details below) and PDF/WORD documents of your reports (both of the form `xxxx_*.*` : replace `xxxx` with your last name) to `mganesh@mines.edu` and to the grader `breyes@mymail.mines.edu`.
Submit written/printed copies (of your reports, tables, figures, codes etc.) to your lecturer.

Let $P = mN$ be the total number of processing cores in `MPI_COMM_WORLD` induced by `mpiexec`, with N and m respectively being the number of nodes and cores in each node. (For Sayers Lab machines, choose $m = 4$ and for MIO nodes, choose m appropriately.)
Let `master` $= 0$ denote the first processing core in `MPI_COMM_WORLD`.

Let $J \geq 2$ and $L = JP$. Let C be a random $L \times L$ weighted covariance matrix (described below) with each processing core generating only J distinct random rows of C. Let

$$C_{\max} = \max\{C_{i,j} : 1 \leq i, j \leq L\}, \qquad C_{\min} = \min\{C_{i,j} : 1 \leq i, j \leq L\}.$$

Let $\text{core}_{\max}$ be the number of the processing core (with minimum index) that first generated the number $C_{\max}$ in `MPI_COMM_WORLD`. Let the location of $C_{\max}$ in C be $(i\_\max, j\_\max)$. Let $\text{core}_{\min}$ be the number of the processing core (with minimum index) that first generated the number $C_{\min}$ in `MPI_COMM_WORLD`. Let the location of $C_{\min}$ in C be $(i\_\min, j\_\min)$.

The main tasks in this MPI programming assignment include the `master` to search and find the pairs $(C_{\max}, \text{core}_{\max})$ and $(C_{\min}, \text{core}_{\min})$ in `MPI_COMM_WORLD` and hence identify the pairs $(i\_\max, j\_\max)$ and $(i\_\min, j\_\min)$.

Your (Fortran90+ or C or C++) code should be such that it will be easier for you/grader to choose $J = 2$ and run your code with $P = 4$ to validate the code. Only for this special case, your code should be such that the `master` should print

- the matrix C (one row per line and easy to read to validate the vectors below);
- the vectors $(C_{\max}, \text{core}_{\max})$, $(C_{\min}, \text{core}_{\min})$, $(i\_\max, j\_\max)$, and $(i\_\min, j\_\min)$.

If $J \neq 2$ or $P \neq 4$, your code should be such that the `master` should print

- the vectors $(C_{\max}, \text{core}_{\max})$, $(C_{\min}, \text{core}_{\min})$, $(i\_\max, j\_\max)$, and $(i\_\min, j\_\min)$;
- the MPI walltime required to execute lines between `MPI_Init` and `MPI_Finalize`.

Use suitable format and headings for your print so that your output can be easily read and identified. Submit your output for the following choices of J and P:
(i) $J = 2, P = 4$; (ii) $J = 100, P = 16$; (iii) $J = 50, P = 32$; and (iv) $J = 25, P = 64$.

The main (memory) constraint in your parallel program with MPI to achieve the above tasks is that at any point in time of executing your code (for any choice of J and P), each processing core should contain at most J consecutive rows of the covariance matrix C. (Only exception in your code is for $J = 2, P = 4$ case so that the `master` can gather and print C.)

Generate the random weighted covariant matrix C in parallel using the following steps. (Using various seeds in processing cores, first harvest random numbers in $[0, 1)$ and then transplant the harvest to generate random numbers in appropriate range values given below.)

Step 1:

- For $k = 1, \cdots, P$, the k-th processing core should first generate a J-dimensional random weight vector $\mathbf{r}_k$ so that, for $i = 1, \cdots, J$,
  the i-th entry of $\mathbf{r}_k$ is a random number $r_{k,i}$ in the range $[k/i, i * k + 1)$.
- The `master` in `MPI_COMM_WORLD` should gather these random weight numbers and
  - compute $S = \sum_{k=1}^{P} \sum_{i=1}^{J} r_{k,i}$;
  - compute an L-dimensional normalized random weight vector $\mathbf{w}$ with first J entries in $\mathbf{w}$ being $r_{1,i}/S,\ i = 1, \cdots, J$; the $J + 1, \cdots, 2J$ entries of $\mathbf{w}$ being $r_{2,i}/S,\ i = 1, \cdots, J$, etc. so that the last J entries of $\mathbf{w}$ are given by $r_{P,i}/S,\ i = 1, \cdots, J$;
  - and then broadcast the normalized weight vector $\mathbf{w}$ to all processing cores in `MPI_COMM_WORLD`. (The entries of $\mathbf{w}$ are denoted below by $w_j, j = 1, \cdots, L$.)

Note that $\sum_{j=1}^{L} w_j = 1$. Use this fact as a check in your code to validate **Step 1**. (Recall round-off error and the machine epsilon.) If this condition does not hold (with appropriate accuracy), then your code should use `MPI_Abort` and exit with an appropriate error message.

Step 2:
Let X be an $L \times L$ random sample matrix with (i, j)-th entry, $X_{i,j}$, being a random number in the range $[-i * j, i/j)$, for $i, j = 1, \cdots, L$. The L sample means of X are given by

$$\bar{x}_i = \frac{1}{L} \sum_{k=1}^{L} X_{i,k}, \qquad i = 1, \cdots, L. \tag{1}$$

Using **Step 1** and X, the (i, j)-th entry of the weighted $L \times L$ covariance matrix C is defined as

$$C_{i,j} = \frac{\sum_{k=1}^{L} w_k \left(X_{i,k} - \bar{x}_i\right) \left(X_{j,k} - \bar{x}_j\right)}{1 - \sum_{n=1}^{L} w_n^2}, \qquad i, j = 1, \cdots, L. \tag{2}$$

Write-down elements of a new $L \times L$ matrix Y (a scaled version of X) so that $C = YY^T$. (That is, identify C in (2) as a matrix-matrix multiplication of Y and Y^T.)

Your parallel (Fortran90+ or C or C++) program with MPI, in addition to implementing **Step 1**, should include the following constraints to achieve the tasks described earlier:

- Each core should allocate two $J \times L$ matrices for creating the J rows of X and C.
- The first processing core should create the first J rows of X; the second processing core should generate rows $J+1, \cdots, 2J$ of X etc. so that the last J rows of X are generated by the last processing core in `MPI_COMM_WORLD`.
- Each processing core should compute J sample means (see (1)) corresponding to the J rows of X that it owns.
- Using the formula for elements of Y, the first processing core should create the first J rows of C; the second processing core should create rows $J+1, \cdots, 2J$ of C etc. so that the last J rows of C are created by the last processing core in `MPI_COMM_WORLD`.
- Each processing core should compute the maximum and minimum entries of its $J \times L$ block of C and also identify the corresponding locations in the block.
- The `master` core should use the MPI pair search command to first get the pair $(C_{\max}, \text{core}_{\max})$ and then get the pair $(C_{\min}, \text{core}_{\min})$.
- Using these pair values, the `master` core should get $(i\_\max, j\_\max)$ and $(i\_\min, j\_\min)$.