# CSCE 689-600 Spring 2020

## Minor Project (OpenMP)

Due: 11:59pm Monday, March 16, 2020

Gaussian Process Regression can be used to predict the values of a function at a point from observations at other points in the domain. As an example, consider an m x m grid of points on a two-dimensional unit square. Node coordinates are given by

$$(x_i,y_j) = (ih,jh),\ i=1,..,m,\ j=1,\dots,m, \quad (1)$$

where h = 1/(m+1) is the mesh width. Observed data value at the point $r=(x_i,y_j)$ is given by the function

$$f(x_i,y_j) = 1-[(x_i-0.5)^2+(y_i-0.5)^2]+d_{ij},$$

(2)

where $d_{ij}$ is a random value between [–0.05,0.05]. The predicted value of the function at r*=(x,y) is given by

$$f(x,y) = k^T(tI+K)^{-1}f. \quad (3)$$

K is an n x n matrix with elements $K(r,s)=\exp(-||r-s||^2)$, where $||r-s||^2$ is the distance between grid points r and s, and $\exp(a)=e^a$ (e is the base of the natural logarithm). Note that for an m x m grid, $n=m^2$. Further, k is an n x 1 vector with elements $k(r) = \exp(-||r-r^*||^2)$, f is the vector of observed data values given by equation (2), and t is a noise parameter that is set to 0.01.

A Matlab file `GPR.m` is provided as an illustration.

1. (70 points) In this assignment, you have to develop an OpenMP-based parallel code to compute f(x,y) for a given point (x,y). The code should use a **single shared-memory** node on ADA or Terra, and should be parallelized to exploit all the cores on the processing node. The code should initialize the grid points and observed data values using equations (1) and (2). Next, the matrix K' = (tI+K) should be computed. This should be followed by LU factorization of K'. These factors should be used to compute the solution of the system K'z=f using the L and U factors obtained in the previous step. Finally, the predicted value should be computed as $f(x,y)=k^Tz$. You must develop your own code to compute the LU factors and to solve the triangular systems. LU factorization can be replaced by Cholesky factorization, which is a more efficient algorithm for symmetric positive definite matrices.
2. (20 points) Describe your strategy to parallelize the algorithm. Discuss any design choices you made to improve the parallel performance of the code.
3. (10 points) Compute the flop rate you achieve in the factorization routine and in the solver routine using all the cores. Compare this value with the peak flop rate achievable on a single core, and estimate the speedup obtained over one core and the corresponding efficiency/utilization of the cores on the node. You may choose appropriate values for the grid size to study the features of your implementation.

**Submission:** You need to upload the following to eCampus:

1. Submit the code you developed.
2. Submit a single PDF or MSWord document that includes the following.
    - Responses to Problem 1, 2, and 3. Response to 1 should consist of a brief description of how to compile and execute the code on the parallel computer

**Helpful Information:**

1. Source file(s) are available on ecampus.
2. Load the Intel software stack prior to compiling and executing the code. Use:

   ```
   module load intel/2017A
   ```
3. The run time of a code should be measured when it is executed in dedicated mode.