Text, letter

Description automatically generated

Text, letter

Description automatically generated

Please refer to the code attached. To run the code on GPU you should below on Grace logins node with GPU (I ran it on Grace2.HPRC.TAMU.EDU).

module load CUDA

nvcc majorProject.cu -o majorProject.exe;./majorProject.exe

Also, I did develop the code for CPU version and it can be run using:

nvcc majorProjectCPU.cu -o majorProjectCPU.exe;./majorProjectCPU.exe

Text, letter

Description automatically generated

The GPU allows for better performance if and only if we can do a same task repeatedly on different element of a matrix. Also, it should be noted that GPU suffers from not having branch prediction and therefore, does not have a good relationship with if statements. Hence, we should avoid if statement in our code as much as possible.

My code (both the GPU and CPU based) starts with initializing some constants including the prediction point coordinates (xR, yR), fineness of the mesh discretization (m), normalization factors (L1 and L2), noise (t) and the memory allocation for arrays (initialization function).

Subsequently, we have to initialize x and y coordinates of the points in the map and their function values f. Next, we need to find the correlation between points (K) and correlation between points and the prediction points (K\*). To speed up the process, I did all of these initializations on the GPU as it can be done over the matrix and vector elements repeatedly (cudaInitXY, cudaInitF, cudaInitK, cudaInitKStar functions).

Next, we have to call the main part of the program which is the solver function. It has to first decompose the (tI+K) matrix into L and U (LU decomposition) and then solve and fid the result of (tI+K)-1F. The first part of this function is done by LUDecompositionPrep function while the latter one is handled by LUSolverPrep. 1) LUDecompositionPrep calls four kernel function in a loop (over al rows) to find the LU decomposition. To avoid synchronization and race condition, I find the L[i][j]\* U[j][k] in cudaMatrixMul (L, U) and then fed the result to cudaUFactorization which updates the upper triangular matrix. As the values of U[j][k] is changed, I re-calculated their multipilication (L[i][j]\* U[j][k]) and pass the results to cudaLFactorization to update the lower triangular matrix. After doing these steps for all of the rows, we have to copy the matrix values from GPU to RAM. It should be noted that my LUFactorization code is based on the one available at geeksforgeeks.com but I dissected it into three main parts to enhance the penalization (finding the reulst of L\*U, diving each row by diagonal elements in cudaLUNormalizer function and finally finding the actual lower and upper triangular parts) . 2) LUSolverPrep uses L, U and F matrices to find (tI+K)-1F. How? Assume that A=tI+K and A== L\*U (using LU decomposition). Now, we have to solve A-1F=Q to find Q as we know A and F. This resembles F=AQ and F=LUQ. Now, assume W=UQ🡪F=LW where W can be found using back-substation (cudaLSolver function). Similarly, we can find Q in W=UQ (cudaUSolver function).

Finally, in the last step, we have to multiply K\* into the result of (tI+K)-1F (which we found in the previous step) to get the predicted value for the point.

It should be noted that my code can still get enhanced by removing the redundancy in memcopy calls. But I am really out of time for these final touchups.

Text, letter

Description automatically generated

I am out of time to run the program.