Programming Assignment #3

Due Apr 1, 2020 by 11:59pm **Points** 4 **Submitting** a text entry box or a file upload
Available until Apr 8, 2020 at 11:59pm

This assignment was locked Apr 8, 2020 at 11:59pm.

Your third programming assignment will include the completion of two tasks

- Replacing the Matrix-Vector multiplication step from our latest optimization of a Laplace 3D solver (which used an explicitly constructed matrix) to a previously-seen version that performs this operation in a matrix-free fashion, and
- Replacing some of the hand-coded kernels involved in the Conjugate Gradients algorithm with library calls from the Intel Math Kernel Library & BLAS.

The code you will use as a basis of your implementation will be taken from ***LaplaceSolver/LaplaceSolver_1_4,*** and you will be reusing components from ***LaplaceSolver/LaplaceSolver_0_3*** for some of the code changes you are asked to perform.

Your tasks are as follows:

- The implementation of the Laplace solver in ***LaplaceSolver/LaplaceSolver_1_4*** is using an explicitly constructed matrix in the Conjugate Gradients routine; however, the only place where that matrix is being used, is in the ***ComputeLaplacian()*** call. We have seen, in previous implementations (in fact, that was the case for all examples in the directories ***LaplaceSolver/LaplaceSolver_0_?*** that this same Laplacian Computation can be done programmatically, without requiring the matrix to be explicitly constructed.

  Your task will be to replace the matrix-based implementation of ***ComputeLaplacian()*** in the version ***LaplaceSolver_1_4*** with a matrix-free implementation. You may use ***LaplaceSolver_0_3***, for example as the source of such a matrix-free implementation. Note that you are not asked to replace the construction or application of the preconditioner matrix; you can keep that matrix explicitly constructed and used, as before.

- The implementation in ***LaplaceSolver/LaplaceSolver_1_4*** demonstrated how routines in the Intel MKL library (specifically some BLAS Level 2 routines) can be used in replacement of some of the kernels we have implemented from scratch. For example, in this particular example we used the BLAS Level 2 routine **mkl_csblas_scsrgemv** (refer to the documentation **here (https://software.intel.com/en-us/mkl-developer-reference-c-mkl-cspblas-csrgemv)**) in replacement of our hand-coded Matrix-Vector multiplication routine for CSR matrices. Similarly, we used the BLAS Level 1 routine cblas_saxpy (documentation **here (https://software.intel.com/en-us/mkl-developer-**

reference-c-cblas-axpy)) in replacement of our hand-coded SAXPY routine (specifically, a special case of it that implements the operation $\boldsymbol{y} \leftarrow \boldsymbol{\alpha x} + \boldsymbol{y}$ as opposed to the more general operation $\boldsymbol{z} \leftarrow \boldsymbol{\alpha x} + \boldsymbol{y}$.

There are more opportunities for replacing some of our custom kernels with pre-packaged routines provided in the Intel MKL library. Those opportunities would include:

- The more general implementation of a SAXPY routine ($\boldsymbol{z} \leftarrow \boldsymbol{\alpha x} + \boldsymbol{y}$) which is needed at some other parts of our program,
- The reduction operations **InnerProduct()** and **Norm()**, and
- The **Copy()** routine in ***PointwiseOps.cpp***

You are asked to replace **two** (or more, if you desire) of these routines with the corresponding MKL routine (or routines; as some of these kernels might require more than one MKL routine when attempting to replace them). Once you perform this replacement, make sure that the behavior you get is identical to the previous implementation (one indication would be that you get exactly the same *residual norms* reported by the Conjugate Gradients algorithm). Optionally, conduct a performance comparison between the run time of the old kernel (the hand-coded one) and the MKL replacement by timing their execution time (with all cores available on your computer).

Hint: Look at MKL functions such as **cblas_scopy, cblas_ssdot, cblas_saxpy, cblas_isamax** (documentation **link** **(https://software.intel.com/en-us/mkl-developer-reference-c-blas-level-1-routines-and-functions)**).

For your deliverable, include both code and a brief report, (in PDF or text format) explaining the code modifications you made, and any (optional) timing comparisons.

Grading:

- You will be graded on a 0-3 scale (0 = very inadequate, 1 = quite problematic, 2 = buggy, or other minor issues, 3 = correct).
- A grade of 4 ("above and beyond") will be very rarely awarded for amazing work.

NEW - Updated Late Policy : Due to the special circumstances associated with remote instruction, a slightly more relaxed late submission policy will be instituted. Submissions will be accepted up to 1 week after the deadline; the late submission penalty will be at most 1 point in our grading scale