Iterative Block Cholesky Decomposition

PROGRAMMING PROBLEM I

CS 3513

Due: February 24, 4:30 p.m.

Description: The programming problem is to implement an iterative block Cholesky decomposition of a Gram matrix for solving the normal equations.

Details: Your program should be calll `ibchk` and take two or three commandline parameters:

`ibchk` *infileA outFileL* $\{b\}$

which are the filename of the data files containing the values for input matrix $\boldsymbol{A}$; the location to store lower triangular matrix $\boldsymbol{L}$ which is the Cholesky decomposition of Gram matrix $\boldsymbol{G}$=$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$= $\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$; and an optional blocksize parameter b. Use a default b=8 if not given. The filenames will be placed as commandline arguments. **Do not prompt for filenames.**

- The iterative block Cholesky decomposition processes b columns and rows of the remaining submatrix an each iteration. Looking at the first iteration for an $n \times n$ symmetric matrix $\boldsymbol{G}$

$$\begin{bmatrix} \boldsymbol{G}_{00} & \boldsymbol{G}_{10}^T \\ \boldsymbol{G}_{10} & \boldsymbol{G}_{11} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{00} & \boldsymbol{0} \\ \boldsymbol{L}_{10} & \boldsymbol{L}_{11} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_{00}^T & \boldsymbol{L}_{10}^T \\ & \boldsymbol{0}\boldsymbol{L}_{11}^T \end{bmatrix}$$

  where $\boldsymbol{G}_{00}$ and $\boldsymbol{L}_{00}$ are $b \times b$ submatrices, $\boldsymbol{L}_{00}$ and $\boldsymbol{L}_{11}$ are lower triangular submatrices, $\boldsymbol{L}_{11}$ is $n-b \times n-b$ submatrix, and $\boldsymbol{L}_{10}$ is $b \times n-b$ regular submatrix.

- The small $b \times b$ problem $\boldsymbol{G}_{00}$=$\boldsymbol{L}_{00}\,\boldsymbol{L}_{00}^T$ is solved by using the standard (unblocked) Cholesky decomposition algorithm (write your own code to do this).

- $\boldsymbol{L}_{10}$ is solved by using $\boldsymbol{L}_{00}$ from the previous step and either forward substitution on $\boldsymbol{L}_{00}\,\boldsymbol{L}_{10}^T$=$\boldsymbol{G}_{10}^T$ or backsubstitution on $\boldsymbol{L}_{10}\,\boldsymbol{L}_{00}^T$=$\boldsymbol{G}_{10}$. Again, write your own code to do this.

- Since $\boldsymbol{G}_{11}$ - $\boldsymbol{L}_{10}\,\boldsymbol{L}_{10}^T$ = $\boldsymbol{L}_{11}\,\boldsymbol{L}_{11}^T$, the remaining submatrix $\boldsymbol{G}_{11}$ is updated by subtracting $\boldsymbol{L}_{10}\,\boldsymbol{L}_{10}^T$ and $\boldsymbol{L}_{11}$ is solved in the remaining iterations.

- Data files are line based text files with one row of the matrix per line. The files use whitespace separation. The matrix values will be floating point values. Each row should have the same number of values. Store the calculated matrix $\boldsymbol{L}$ in the same format as the input files.

Submit all source code files using subversion or D2L. Also submit a readme.txt file. The readme file should describe all known bugs that were not removed from the program and discuss problems encountered when developing and testing the program. The readme file should describe the programming language, complier version, OS environment, and all necessary steps to compile and execute your code. Non-standard libraries should be submitted with the code. You may assume

that numpy (for python), eigen (for C++), and jama (for java) are all ready installed. Any external sources of information should be explicitly mentioned in the readme.txt and in the comments of the source code.

EXTRA CREDIT COMPONENTS FOR PP 2

- (5%) Use the results of the iterative block Cholesky decomposition to solve for the least squares of $\boldsymbol{Ax}=\boldsymbol{b}$. Name this program *lsibchk* and it should take the parameters *infileA inFileB outFileX*. **Don't duplicate your code** from the main assignment, but call it (i.e., have the common code in a module usable by both programs).
- (20%) Create an alternative implementation that is a binary recursive block Cholesky decomposition. Split the matrix in the middle in each recursion. Solve the upper corner with the unblocked version when it gets to size b, update the off diagonals and the lower corner on the return from recursion, and then recurse on the lower corner.