Project Expectations

The project must be completed by a team of two students. A single hard copy of each project should be submitted by the project deadline. A cover sheet should be included with the project submission. The cover sheet should include the names of the two students on the project team. It should also include a brief summary of the work completed by each student. Project team members need to sign the cover sheet indicating their agreement with the summary of work they completed. In addition, MATLAB m-files should be e-mailed to shontz@ku.edu by the project deadline. Please use ‘EECS 639: Project 2’ as the subject line of your e-mail. Do not submit work unless you have adhered to the principles of academic integrity as described in the syllabus. Points will be deducted by poorly commented code, redundant computation that seriously effects efficiency, and failure to use features of MATLAB that are part of the course syllabus. In particular, use vector operations whenever possible. In addition, you should make an effort to minimize the error in your computations (where possible).

Development of a Numerical Library for Solving Linear Systems of Equations

In this project, you will develop a numerical library for solving linear systems of equations using MATLAB. In particular, you will develop numerical algorithms and software for solving linear systems involving various types of matrices.

Part A: Implementation of Linear Solvers

Write MATLAB functions for the following algorithms:

- Backward Substitution
- Forward Substitution
- Gaussian Elimination
- LU Factorization
- Gaussian Elimination with Partial Pivoting
- Cholesky Factorization
- Backward Substitution - specialized for an upper triangular matrix with p bands
- Backward Substitution - specialized for an upper triangular block-structured matrix with \( m \times m \) blocks
  (i.e., the blocks will have an upper triangular structure, although the full matrix will not necessarily be upper triangular)
- Forward Substitution - specialized for a lower triangular matrix with p bands
- Forward Substitution - specialized for a lower triangular block-structured matrix with \( m \times m \) blocks.

Part B: Demonstration of Algorithm Correctness

Once you have implemented the above set of linear solvers, debug them on small matrices (e.g., 10 \( \times \) 10 matrices before proceeding with the numerical experiments which will be performed on larger matrices). (You do not need to turn-in anything for this step of the project.)

Next, demonstrate that your algorithms work on matrices of various structures: upper triangular, lower triangular, banded, block-structured, symmetric positive definite, etc. To do so, visit the MatrixMarket website at http://math.nist.gov/MatrixMarket/ and select several matrices of size 1000 \( \times \) 1000 having various structures stemming from an array of applications. The following websites should prove useful for these purposes: http://math.nist.gov/MatrixMarket/applications.html and http://math.nist.gov/
MatrixMarket/deli/. If a particular matrix structure does not happen to be available from MatrixMarket, you should generate your own matrix using Matlab. Use the mmread.m function at http://math.nist.gov/MatrixMarket/mmio/matlab/mmiomatlab.html to read the matrices into Matlab.

For each matrix used in your project, list the application where the matrix arose, report on relevant statistics (e.g., condition number, number of nonzeros, matrix structure, etc.), and generate a spyplot of the matrix.

Here are some helpful Matlab commands:

\[
\begin{align*}
\text{cond}(A,1) & \text{ and } \text{cond}(A,\text{'inf'}) - \text{ computes the condition number of } A \text{ using the 1-norm and the infinity-norm, respectively} \\
\text{nnz}(A) & - \text{ returns the number of nonzeros of } A \\
\text{spy}(A) & - \text{ generates a spyplot of } A
\end{align*}
\]

Part C: Floating Point Analysis

Perform a detailed floating-point analysis for each linear solver for which we did not discuss the computational complexity in class. (In other words, you will need to determine the computational complexity for four of the linear solvers.)

Part D: Scaling Study

Perform a scaling study for each linear solver. To do so, download matrices of various sizes (i.e., for matrices ranging in size from 1000 \( \times \) 1000 to 10,000 \( \times \) 10,000) and determine how the run time of the algorithm changes as a function of matrix size. Plot the run time as a function of matrix size. Compare your findings with the theoretical floating point analysis for the algorithm.

Items to Submit: Submit your Matlab code, relevant plots and analysis, and a succinct discussion of your results.

Grading Criteria: Your project will be graded on the basis of: accuracy and efficiency of the numerical algorithms, relevance of the matrix statistics, correctness of the analysis, thoroughness of the results, and the writing itself. The project is worth a total of 100 points.