Programming Assignment 1 Applied Linear Algebra 2 Spring 2024

The solutions are due on Canvas by 11:59 PM on Wednesday, January 31, 2024

General Task

This assignment is the first piece of code in the development of an LU factorization code that will be completed in the next programming assignment. This portion deals with triangular matrices.

Definitions

The matrix $L \in \mathbb{R}^{n\times n}$ is a unit lower triangular matrix, i.e.,

$$\lambda_{ij} = e_i^T L e_j \forall i = 1, \ldots, n, \quad \lambda_{ii} = e_i^T L e_i = 1, \quad \text{elements on the main diagonal are 1}$$
$$\forall i = 1, \ldots, n, \;\; j > i, \quad \lambda_{ij} = e_i^T L e_j = 0, \quad \text{elements strictly above the main diagonal are 0}$$

$$\forall i = 1, \ldots, n, \;\; j < i, \quad \lambda_{ij} = e_i^T L e_j, \quad \text{may or may not be 0.}$$

In other words all nonzeros other than the required 1's on the main diagonal must occur strictly below the main diagonal. Note that this matrix is, by definition, nonsingular.

The matrix $U \in \mathbb{R}^{n\times n}$ is a nonsingular upper triangular matrix, i.e.,

$$\mu_{ij} = e_i^T U e_j \forall i = 1, \ldots, n, \quad \mu_{ii} = e_i^T U e_i \neq 0, \quad \text{elements on the main diagonal are nonzero}$$
$$\forall i = 1, \ldots, n, \;\; j < i, \quad \mu_{ij} = e_i^T U e_j = 0, \quad \text{elements strictly below the main diagonal are 0}$$

$$\forall i = 1, \ldots, n, \;\; j > i, \quad \mu_{ij} = e_i^T U e_j, \quad \text{may or may not be 0.}$$

In other words all nonzeros other than the required nonzeros on the main diagonal must occur strictly above the main diagonal.

The matrix product $M = LU \in \mathbb{R}^{n\times n}$ is, in general, a dense square matrix, given vectors $v \in \mathbb{R}^n$ and $s \in \mathbb{R}^n$, the products $w = Lv$ and $z = Us$ are vectors in $\mathbb{R}^n$, and given vectors $b \in \mathbb{R}^n$ and $p \in \mathbb{R}^n$, the solutions to the triangular systems $Ly = b$ and $Ux = y$ are vectors in $\mathbb{R}^n$.

The Codes

The following subroutines must be implemented.

1. A subroutine that computes the product $w = Lv$ given the vector v and matrix L.

2. A subroutine that computes the product $z = Us$ given the vector s and matrix U.

3. A subroutine that computes the product $M = LU$ given the matrices L and U.

4. A subroutine that solves the triangular system $Ly = b$ given the vector b and the matrix L.

5. A subroutine that solves the triangular system $Ux = y$ given the vector y and the matrix U.

You may use any of the approaches described in the class notes, i.e., column-oriented, row-oriented, etc. You may implement these codes using a compiled typed language such as C, Fortran, C++ or in a coding/problem solving environment such as Matlab or Python. However, your coding style must be "basic", i.e., you must write code that exposes clearly the loop or vector control constructs and the singly and doubly indexed data structures required.

Your subroutines must be callable from a driver code that implements your testing and validation approaches.

You may use libaries and external routines in your test routines to generate solutions for comparisons, to generate historgrams, graphs and any other useful summary display mechanisms. You **may not use library routines inside your assigned subroutines**.

Data Structures

L and U will be stored in a doubly indexed array, e.g., ARRAY(1:RDIM,1:CDIM), where RDIM and CDIM are dimensions that are greater than or equal to n for the problems in a set you are testing and the colon notation indicates a range of index values (like Matlab or any vector supporting language). Note that this means in general that RDIM and CDIM will not be the same as n for most problems. This is encouraged since it is not unusual in practice.

When storing U the 0's below the main diagonal are not stored in ARRAY, i.e., ARRAY(I,J) $= \mu_{ij}$, with $I = i$, $J = j$ and $i \leq j$. Other elements of ARRAY must be viewed as undefined when computing with U.

Similarly, when storing L the 0's above the main diagonal and the 1's on the main diagonal are not stored in ARRAY, i.e., ARRAY(I,J) $= \lambda_{ij}$, with $I = i$, $J = j$ and $j < i$. Other elements of ARRAY must be viewed as undefined when computing with L.

When testing your matrix product routine to compute $M = LU$, L and U should both be stored in ARRAY and M should be placed in a second doubly indexed array MARRAY(1:MRDIM,1:MCDIM) that need not be the same size as ARRAY but of course must have row and column dimensions large than n.

Input and output vectors for the routines should be stored in appropiate singly index arrays.

Tests

Your testing must demonstrate the correctness of your codes. This does not mean you run them on small number of problems and verify manually the results. While such small simple cases are a useful part of the validation procedure you should also:

1. Run a range of problem sizes from $n = 10$ to n in the hundreds.

2. Generate problems for which you know the result analytically, i.e., not as the result of running a library code.

3. Generate problems for which you know the result and compare them to a library code, e.g., Matlab library routines.

4. Generate problems for which you know the result using your routines.For example, once you have validated your matrix vector prouct routines you can generate right-hand side vectors for your triangular system solving routines: for a chosen x, compute $b = Lx$ or $b = Ux$ and then use your system solving routines to solve $Lx = b$ or $Ux = b$ and compare the result to the chose x.

5. You may use library codes to generate matrices and vectors, e.g., random matrix and vector generators in Matlab.

6. When running a large number of problems over a range of n values and x and b vectors, you should not simply display the result of each subroutine. That is not acceptable and not useful. You should present your results in a summarizing fashion. For example, when comparing your computed results, x_{comp}, to true results, x_{true}, (however they are known) it is useful to examine the absolute and relative errors

$$\epsilon_{abs} = \|x_{comp} - x_{true}\| \quad \text{and} \quad l\epsilon_{rel} = \epsilon_{abs}/\|x_{true}\|,$$

   where $\|v\|$, is a chosen vector norm, and organizing their values over various sets, e.g., fixed L or U and many right-hand sides for solving systems or many input vectors for products. (Of course, for the relative error the norm of the true result should be kept away from a small number.) Histograms, means and variances of the error are useful. Selected plotting of trends can also be convincing. Note that this is one of the main lessons of the assignment – thinking about how to present in a coherent and convincing manner the correctness of your codes to a reader.

1 Submission of Results

You should submit a document that has the following form:

1. A section that describes you choices of algorithm and the organization of your testing routines and subroutines.

2. A section describing your validation strategies. This should include:

    - how your problems are generated;
    - the range of sizes and characteristics you are using in your problems;
    - how the correctness is being evaluated;
    - what information is displayed in your evaluation section and why it is appropriate for assessing the correctness of your codes.

3. One or more sections presenting your evidence of correctness of your subroutines and your arguments supporting the assertion of correctness given the evidence.

You should also submit your codes. You may be called upon to demonstrate their usage and to reproduce evidence contained in your document.