Graded Homework 3 Foundations of Computational Math 2 Spring 2021

The solutions are due by 11:59PM on March 12, 2021

Written Exercises

There are no written problems in this assignment.

Programming Exercise

General Task

Consider the case where $A \in \mathbb{R}^{n \times n}$ is nonsingular and the system $Ax = b$ This assignment is to empirically evaluate the predictions you make about the structure of the factors and the general performance of the method across a large set of randomly generated problems when using LU factorization to solve the problems.

The Codes

1. Implement a code that computes the LU factorization of a matrix A without pivoting, using partial pivoting or using complete pivoting. The code that is capable of performing these three tasks, based on a user selection. Your code should also detect situations where the factorization may not proceed and exit gracefully. Clearly, this is the case when the candidate pivot set contains no acceptable value. In exact arithmetic this means they are all 0. Your code should also allow the detection of a set of candidate pivots that are all "too small" and warn the user. Your factorization routine, which must be a separate routine from the driver/tester routine, should accept the matrix A stored in a simple 2-dimensional array-like data structure and other relevant parameters such as n and a flag indicating what form of the factorization should be attempted, e.g., no pivoting, partial pivoting, or complete pivoting. The routine should return the matrices L and U stored within the array that contained A on input, i.e., you should implement the in-place algorithm strategy described in the class notes and lectures. You should also keep a copy of A in an additional data structure for correctness checking. The 1 values on the diagonal of L, the 0 values in the upper triangular portion of L and the 0 values in the lower triangular portion of U should not be stored at any point in the code. The routine should also return the permutation matrices P and/or Q appropriate to the pivoting strategy used. These permutation matrices should not be stored as full matrices within any array. The matrices, or equivalently the set of el-

ementary permutations that are their factors, can be represented by at most n integers each.

2. Implement a routine that accepts as input the 1-dimensional arrays specifying P and Q and applies them to vectors and matrices stored in 1 dimensional and 2 dimensional arrays.

3. Implement a solution routine that accepts as input a vector b stored in a simple 1-dimensional array. The routine should also accept as input the 2-dimensional array containing the information specifying the L and U matrices and the 1 dimensional arrays specifying P and Q if appropriate. Any application of these permutations should used the permutation routine listed earlier. The forward and backward solves $Ly = b$ and $Ux = y$ must also work only with the information specificed in the 2-dimensional array and not expanded version of L and U. The routine should return the solution x $Ax = b$.

4. Implement a matrix multiplicaiton routine that accepts as input the 2-dimensional array containing the information specifying the L and U matrices and return in a separate 2-dimensional array the product $M = LU$ **or** the product $M = |L||U|$ based on user selection indicated by an input flag. Note that these matrix multiplications must use the information specifying L and U within the data structure. It **must not** expand them into separate arrays that include the 1 and 0 values that are not stored in the input array.

5. You will also need various test routines designed to evaluate and validate the correctness of the code and accomplish the tasks described below. You may code in any compiled and typed language you wish although C, C++, Julia, and Fortran are preferred. In all cases, however, you may not use standard libraries such as LAPACK or built-in matrix routines for pieces of your routines implementing the computations described above.

6. The test routines should include the computation of $\|W\|$ where W is a matrix. The norms used should be finite in computation, i.e., $\|W\|_1$, $\|W\|_\infty$, or $\|W\|_F$. You may use library routines to compute $\|W\|_2$ if you wish but you are definitely not expected to generate the code for the 2-norm. The matrix norm code can also be used to compute the associated vector norms if $W \in \mathbb{R}^n$ since the formulas can handle a $n \times 1$ matrix. The vector 2-norm of course is easily implemented using a simple inner product.

Library Codes

You may use libaries and exterrnal routines in your test routines to generate solutions for comparisons, to generate historgrams, graphs and any other useful summary display mechanisms. Make sure when using library routines as part of your empirical analysis, you carefully check, e.g., the P, Q, L and U generated by your routines and the library, e.g., MATLAB.

These factors are not unique given that pivoting choices are not unique in general. Note that the textbook has relevant MATLAB coding examples. Programs 1, 2, and 3 in Chapter 3 implement triangular system solving when L and U are given in a 2-dimensional array. Note that some thought must be given when adapting these to the current assignment since L and U will be stored together in a single 2-dimensional array resulting from the factorization algorithm.

Programs 4, 5, and 6 give in-place versions of LU without pivoting. This style is the form expected for the solution to the assignment, i.e., no matrix operations are implemented directly in terms of full dense matrices. These routines do not solve the assignment, however, since they lack the required pivoting capabilities.

Program 9 implements complete pivoting and therefore performs one of the assigned tasks. However, its style is one that yields clarity and ease of implementation. It is **not acceptable** for the solution to the assignment since it is wasteful of computations and space. Notice how an entire matrix is used to represent the accumulated row and column permutations. The accumulation of Gauss transforms and permutations is accomplished by full matrix multiplication, each requiring $2n^3 + O(n^2)$ operations despite the fact that the matrices have structure that we have seen to reduced complexity substantially. A solution program using this style will not receive credit.

Metrics

There are several important metrics to use when assessing the code's correctness. These metrics should be computed in double precision espeicially if you have run your routines in single precision. Some suggestions follow:

1. When comparing matrices use more than one matrix norm, e.g., the finitely computable ones, $\|M\|_1$, $\|M\|_\infty$ and $\|M\|_F$.

2. When comparing vectors use more than one norm, e.g., $\|v\|_1$, $\|v\|_\infty$ and $\|v\|_2$.

3. Check the factorization accuracy

$$\frac{\|PAQ - LU\|}{\|A\|}$$

   where $\|A\| \geq 1$, i.e., relative error for large A.

4. If the solution is known by design of the problem check

$$\frac{\|x - \tilde{x}\|}{\|x\|}$$

   where $\tilde{x}$ is the computed solution and $\|x\| \geq 1$.

5. You should check the accuracy via the residual $b - A\tilde{x}$ and

$$\frac{\|b - A\tilde{x}\|}{\|b\|}$$

assuming $\|b\| \geq 1$ for all attempts to solve a system, i.e., whether or not you know the true solution.

6. You should compute the growth factor

$$\gamma_\epsilon = \frac{\|\ |L_\epsilon||U_\epsilon|\ \|}{\|A\|}$$

where $L_\epsilon U_\epsilon = PAQ$ is the **computed** factorization of PAQ using the selected pivoting strategy.

7. If you have access to a standard library you may also use the results of its LU factorization algorithms. However, as noted above care must be taken since details of pivoting strategies may yield differences in the factors and permutations. The library routines are very useful when you generate a system by choosing A and b and need a reliable way of generating x to compare with your computed solution.

Generation of Test Problems

A key consideration in this assignment is the generation of test problems. Some suggestions follow:

1. Generate L and U so they are nonsingular unit lower triangular and upper triangular matrices respectively. Evaluate their product as A. This is useful for both small and large values of n. For small values you can also constrain L and U to have integer values so A will have integer values. Take care with the magnitude of the elements of L and U. Nicely conditioned problems tend to be specified by off-diagonal elements in L smaller than 1 in magnitude and diagonal elements in U that are not too small compared to off-diagonal elements in U.

2. Remember even if A is generated from L and U when you run your routine with partial or complete pivoting nontrivial permutation matrices P and/or Q may result and you may return $\tilde{L}$ and $\tilde{U}$ such that $PAQ = \tilde{L}\tilde{U}$.

3. Randomly generated matrices tend to be nonsingular and reasonably conditioned. You can make sure any matrix is nonsingular by adding to the diagonal elements until the matrix is diagonally dominant by rows, columns or both.[1] This guarantees success of

[1]A matrix is strictly diagonally dominant by rows (columns) if the magnitude of each diagonal element is strictly larger than the sum of the magnitudes of all off-diagonal elements in the same row (column). A matrix may of course be diagonally dominant by rows and columns simultaneously, e.g., the identity.

the factorization if run without pivoting. As noted above, if you allow pivoting the routine may so do even for a diagonally dominant matrix depending on the form of dominance and the pivoting strategy used. It is also useful to note that after generating such a matrix, $\tilde{A}$, you can apply random permutations P_L and Q_R to generate a new test matrix $A = P_L\tilde{A}Q_R$ that will not be diagonally dominant but will still be nonsingular. Of course, P_L and Q_R are not necessarily the permuations that will be generated by applying partial or complete pivoting to A.

4. You can generate a symmetric positive definite A from a lower triangular $\tilde{L}$ with positive diagonal elements (not necessarily 1) via $A = \tilde{L}\tilde{L}^T$. A is nonsingular by definition and factorization will succeed without pivoting. Note that your code will still produce an LU factorization since your factorization routine is not designed to exploit symmetry. However, there is a relationship between L, U and $\tilde{L}$. This is probed in the first set of structured factorization tasks.

5. Matrices with known structures that influence the structure or magnitude pattern of their factorizations are also useful. This is especially true if the patterns scale in a known way with n. Recall the example in the notes/homework Probelm 2.8. Consider what should happen when no pivoting is used, partial pivoting and complete pivoting for various n values. Also, nonzero patterns such as banded matrices for A should produce specific nonzero patterns in L and U. Structure is the subject of the first set of empirical tasks discussed below.

6. Given a matrix one can also generate a vector b based on a chosen solution x. That is given A choose x and generate b via the matrix vector product Ax.

7. Make sure that the matrices you use to check pivoting **actually require some pivoting when factored**.

8. You should run a range of problem sizes for each algorithm and problem type you evaluate.

9. Do not simply report the computed solution and/or the factors for a small number of small systems. Think about how you would report the results of testing each of the routines with many matrices including those of sizes too large to display for any useful effect.

Empirical Tasks Set 1 : Structure

You are strongly encouraged to think about this set of experiments before Exam 1.

Consider the following structured matrices, predict the structure of their factors and factorization, and verify them empirically. Your solutions must include an explanation of your observations and justification for the conclusion that your predictions are correct.

1. Consider a matrix $A \in \mathbb{R}^{n\times n}$ that is is diagonal with positive elements, i.e., $\alpha_{ij} = e_i^T A e_j = 0$ for all $i \neq j$ and $\alpha_{ii} > 0$ for $1 \leq i, j \leq n$. For example, let $\alpha_{ii} = i$ or $\alpha_{ii} = n - i + 1$

$$n = 5 \rightarrow A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

   Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors as a function of your choice of structure in the positive diagonal elements? What is the growth factor for your chosen problems and the pivoting choices?

2. Consider a matrix $A \in \mathbb{R}^{n\times n}$ that is is antidiagonal with positive elements, i.e., $\alpha_{1,n} > 0, \alpha_{2,n-1} > 0, \ldots, \alpha_{n-1,2} > 0, \alpha_{n,1} > 0$. For example,

$$n = 5 \rightarrow A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

   Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors as a function of your choice of structure in the positive antidiagonal elements? What is the growth factor for your chosen problems and pivoting choices?

3. Consider a matrix $A \in \mathbb{R}^{n\times n}$ that is is the sum of a diagonal matrix and an antidiagonal matrix with positive elements, i.e., and X nonzero pattern. Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors as a function of your choice of structure in the positive antidiagonal elements? If complete pivoting is there any structure that follows for a particular choice of pivot elements?

4. Consider a matrix $A \in \mathbb{R}^{n\times n}$ that is unit lower triangular and the elements in the strict lower part all have magnitude less than 1, i.e., $|\lambda_{ij}| < 1$ for $i > j$, $\lambda_{ij} = 0$ for $i < j$,

and $\lambda_{ii} = 1$. Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors? (Note that your code does not know that A is unit lower triangular and will eliminate the elements below the diagonal by applying Gause transforms.)

5. Consider a lower triangular matrix A again but this time let the diagonal elements be positive and not 1 and the elements in the strictly lower triangular part be larger than 1, e.g.,
$$A = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 \\ 4 & 3 & 2 & 0 & 0 \\ 5 & 4 & 0 & 2 & 0 \\ 6 & 5 & 4 & 3 & 2 \end{pmatrix}.$$
Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors?

6. Consider a matrix $A \in \mathbb{R}^{n \times n}$ that is tridiagonal with all elements on the main diagonal and the first super and subdiagonals nonzero, i.e., $\alpha_{ii} \neq 0$, $\alpha_{i+1,i} \neq 0$, and $\alpha_{i,i+1} \neq 0$. Of course, these must be such that the matrix is nonsingular.

   Suppose A is, additionally, strictly diagonally dominant by rows and columns. Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors? Suppose A is such that when using partial pivoting by rows it requires a row interchange **on every elimination step**. What is the structure of the factors of L, U and P?

7. Consider a matrix $A \in \mathbb{R}^{n \times n}$ with
   - $\alpha_{ij} = e_i^T A e_j = -1$ when $i > j$, i.e., all elements strictly below the diagonal are -1;
   - $\alpha_{ii} = e_i^T A e_i = 1$, i.e., all elements on the diagonal are 1;
   - $\alpha_{in} = e_i^T A e_n = 1$, i.e., all elements in the last column of the matrix are 1;
   - all other elements are 0

   e.g., for $n = 4$ we have
$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$
   Consider factoring with no pivoting, partial row pivoting and complete pivoting. What can be said about the L, U, P and Q factors? What is the growth factor for the different pivoting choices?

8. Suppose $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite generated from a lower triangular $\tilde{L}$ with positive diagonal elements (not necessarily 1) via $A = \tilde{L}\tilde{L}^T$. A is nonsingular by definition and factorization will succeed without pivoting. Note that your code will still produce an LU factorization since your factorization routine is not designed to exploit symmetry. What is the relationship between L, U and $\tilde{L}$?

Empirical Tasks Set 2 : General Trends

This set requries the generation of a large set of problems grouped and empifically analyzed by problem size n and, if appropriate the class of matrix problem considered.

For each problem size and class of problem, generate many example problems and evaluate the various metrics discussed earlier but in particular the relative size of the residuals, the relative error in the factorization, the growth factor, and, if the true solution is known, the relative error in the computed solution $\tilde{x}$. You should present your results in a form appropriate to characterize these metrics over a large data set, i.e., too large to look at each problem individually. This can be done, for example, by graphs and histograms. The latter is particularly useful for detecting outliers in the performancesuch as large error or residuals. These outliers can be discussed in more detail and explained if you wish.

You should also include empirical evidence of the $O(n^3)$ complexity giving the appropriate trend in time required to factor a matrix. It is recommended that this is done with a matrix that does not require pivoting such as a matrix that is strictly diagonally dominant by row and column.