CSCI 2033: Elementary Computational Linear Algebra (Spring 2020)

Assignment 2 (100 points)

Due date: March 27th, 2019 11:59pm

In this assignment, you will implement a MATLAB function to decompose a matrix into lower and upper triangular matrices ($\mathbf{L}$ and $\mathbf{U}$), i.e., $\mathbf{A} = \mathbf{LU}$, as outlined in Section 2. In addition, you must complete **either** Section 3 (Matlab implementation of LU with partial pivoting) **or** Section 4 (written questions). If you choose to complete both, your highest grade out of the two will be used to compute your final grade for this assignment.

For each function that you are asked to implement, you will need to complete the corresponding `.m` file with the same name that is already provided to you in the zip file. Failing to do so may result in points lost. In the end, you will zip up all your complete `.m` files and upload the zip file to "HW2 Programming" on **Gradescope**. For the written portion, please scan or compile your solution into a PDF and submit via "HW2 Written" on **Gradescope**. Note that you do not need to submit anything to "HW2 Written" if you wish to complete Section 3 of the assignment instead, but you have to submit your completed code for Section 2 since that part is **mandatory**.

Note on plagiarism A submission with any indication of plagiarism will be directly reported to University. Copying others' solutions or letting another person copy your solutions will be penalized equally. Protect your code!

1 Submission Guidelines

Your zip file should contain the following `.m` files on Gradescope (you only need to submit the first two files if you are doing the written portion):

- `replacement_lu.m`
- `my_lu.m`
- `interchange_lup.m`
- `my_lup.m`

You may use MATLAB's high-level array manipulation syntax, e.g. `size(A)`, `A(i,:)`, and `[A,B]`, but the built-in linear algebra functions such as `inv`, `lu`, `rref`, and `A\b` are *not* allowed. Please contact the TAs for further questions.

Please make sure that your functions comply with the expected input and output.

2 LU Decomposition **(60 points)**

In this part, you will write functions that perform LU decomposition without pivoting. We will deal with pivoting in the next part of the assignment.

First, you will implement a simple modular function that will help you with LU decomposition.

Notation: for subsequent sections, A_i indicates the i^{th} row of $\mathbf{A}$, and A_{ij} indicates the (i, j) entry of $\mathbf{A}$.

Specification:

`function [U_new, L_new] = replacement_lu(U, i, j, s, L)`
Input: two square matrices $\mathbf{U}$ and $\mathbf{L}$, two integers i and j, and a scalar s.
Output:

- $\mathbf{U}_{\text{new}}$: updated $\mathbf{U}$ by subtracting s times row j from row i, i.e. performing the row replacement operation $U_i \leftarrow U_i - sU_j$.
- $\mathbf{L}_{\text{new}}$: updated $\mathbf{L}$ by filling in its (i, j) entry with s, i.e., $L_{ij} \leftarrow s$.

Warning! If you are adapting your code from Assignment 1, be careful that we are now *subtracting* s times row j from row i instead of adding it.

Now, you will implement of the main portion of LU decomposition.

`function [L, U] = my_lu(A)`
Input: an $n \times n$ square matrix $\mathbf{A}$.
Output:

- $\mathbf{L}$: an $n \times n$ lower triangular matrix where the diagonal entries are all one, e.g., $\begin{bmatrix} 1 & 0 & 0 \\ * & 1 & 0 \\ * & * & 1 \end{bmatrix}$ where $*$ is a potentially nonzero element.
- $\mathbf{U}$: an $n \times n$ upper triangular matrix.

To get full credit, your function should handle the following case:

- Early termination: Due to round-off error, there may exist free variables whose pivots are extremely small but not precisely zero. You should terminate your LU decomposition if the absolute value of a pivot is less than 10^{-12}.

The process of LU decomposition uses Gaussian elimination that transforms $\mathbf{A}$ to an upper triangular matrix $\mathbf{U}$ while recording the pivot multipliers in a lower triangular matrix $\mathbf{L}$.

1. Initialize $\mathbf{L}$ to the identity matrix, and $\mathbf{U}$ to $\mathbf{A}$. You can use MATLAB's built-in function `eye(n)`.
2. At the i^{th} step, for each row k below the i^{th} row,

(a) Record the pivot multiplier to $\mathbf{L}$ at (k, i), i.e., $\mathbf{L}_{k,i} = \mathbf{U}_{k,i}/\mathbf{U}_{i,i}$ as shown in Figure 1. Note that this fills in the i^{th} column of $\mathbf{L}$.

(b) Reduce the k^{th} row of $\mathbf{U}$ using the pivot multiplier, i.e., $\mathbf{U}_k = \mathbf{U}_k - (\mathbf{U}_{k,i}/\mathbf{U}_{i,i})\mathbf{U}_i$ where $\mathbf{U}_i$ is the i^{th} row.

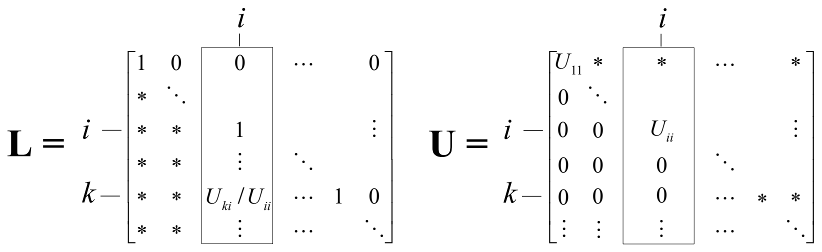

Figure 1: LU decomposition at the i^{th} step.

We provide pseudocode for LU decomposition in Algorithm 1.

Algorithm 1 LU decomposition of A

```
1: $n \leftarrow$ the number of columns of $A$
2: $L \leftarrow I_n$ where $I_n$ is $n \times n$ identity matrix.
3: $U \leftarrow A$
4: for $1 \le i \le n-1$ do
5:     if $U_{ii}$ is nearly zero* then
6:         return $L$, $U$                              ▷ Early termination
7:     end if
8:     for $i+1 \le k \le n$ do
9:         $p \leftarrow U_{ki}/U_{ii}$
10:        $[U, L]$ = replacement_lu($U$, $k$, $i$, $p$, $L$)
11:    end for
12: end for
13: return $L$, $U$
```

*Consider a number to be nearly zero if its absolute value is smaller than 10^{-12}.

Note: When terminating early, $\mathbf{L}$ is not *unique*, i.e., the order of rows corresponding to zero rows in $\mathbf{U}$ can be different. As long as the resulting decomposition satisfies $\mathbf{A} = \mathbf{LU}$, the solution $\mathbf{L}$ and $\mathbf{U}$ are valid.

Test cases:

- `[L,U] = my_lu([4,-2,2;-2,5,3;2,3,9])`

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.5 & 1 & 1 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 4 & -2 & 2 \\ 0 & 4 & 4 \\ 0 & 0 & 4 \end{bmatrix}$$

- Early termination:
  `[L,U] = my_lu([4,-2,2,1;-2,5,3,3;4,-2,2,1;-2,5,3,3])`
  $$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ -0.5 & 1 & 0 & 1 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 4 & -2 & 2 & 1 \\ 0 & 4 & 4 & 3.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$
  Without early termination, it will return **L** and **U** with `NaN` (Not-a-Number) entries.

- You can test your algorithm by constructing a random $n \times n$ matrix **A**, e.g., `A=rand(10,10)`, and testing whether multiplying **L** and **U** reconstructs **A**. You also need to check whether **L** is a lower triangular matrix with diagonal entries equal to 1, and **U** is an upper triangular matrix.

3 LU Decomposition with Partial Pivoting (40 points)

Based on your `my_lu`, you will write numerically stable LU decomposition with *partial pivoting*. At the i^{th} step of LU decomposition (i^{th} pivot column), you will find the row that has the largest absolute value in the pivot column (say row j), and swap the i^{th} and j^{th} rows of $\mathbf{U}$ as usual. Simultaneously, you will swap the *partial* entries of the i^{th} and j^{th} rows of $\mathbf{L}$, and record the row exchange in a permutation matrix $\mathbf{P}$. For further details, please see
`http://www.math.kent.edu/~reichel/courses/intr.num.comp.1/fall09/lecture9/lecture4.pdf`

First of all, similar to above, you will implement a simple modular function that will help you with LU decomposition with partial pivoting.

Specification:

`function [U_new, L_new, P_new] = interchange_lup(U, i, j, L, P)`
Input: three same size square matrices $\mathbf{U}$, $\mathbf{L}$, and $\mathbf{P}$, and two integers i and j.
Output:

- $\mathbf{U}_{\text{new}}$: updated $\mathbf{U}$ by swapping rows i and j, i.e. $U_i \leftrightarrow U_j$.
- $\mathbf{L}_{\text{new}}$: updated $\mathbf{L}$ by swapping *only* the first $(i-1)$ entries of rows i and j, i.e., $L_{i,1} \leftrightarrow L_{j,1}, \ldots, L_{i,(i-1)} \leftrightarrow L_{j,(i-1)}$ as shown in Figure 2.
- $\mathbf{P}_{\text{new}}$: updated $\mathbf{P}$ by swapping rows i and j, i.e. $P_i \leftrightarrow P_j$.

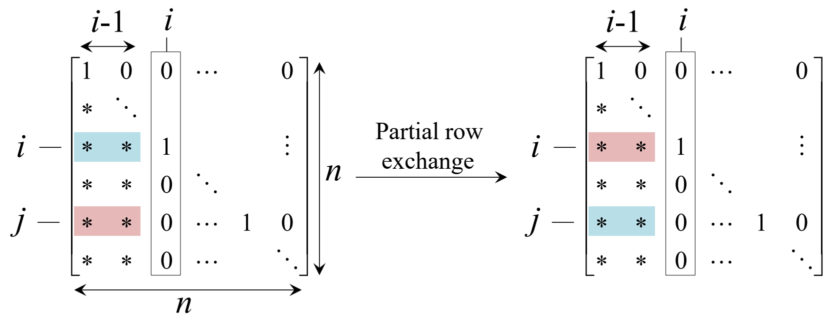

Figure 2: Partial row exchange in **L**.

Now, you will implement of the main portion of LU decomposition with partial pivoting.

`function [L, U, P] = my_lup(A)`
Input: an $n \times n$ square matrix $\mathbf{A}$.
Output:

- $\mathbf{L}$: an $n \times n$ lower triangular matrix where the diagonal entries are all 1.
- $\mathbf{U}$: an $n \times n$ upper triangular matrix.

- **P**: an $n \times n$ permutation matrix.

The process of LU decomposition with partial pivoting needs to compute an additional row permutation matrix **P**.

1. Initialize **L** and **P** to the identity matrix, and **U** to **A**. You can use MATLAB's built-in function `eye(n)`.
2. At the i^{th} step,
   (a) Similar to Assignment 1, perform partial pivoting in **U**.
   (b) Record the row exchange to the permutation matrix **P** by exchanging its corresponding rows.
   (c) Exchange the corresponding row entries in **L** to reflect the row exchange in **U**. Note that this exchange only involves with the row entries that have been recorded, i.e., not entire row exchange.
   (d) For each row k below the i^{th} row, record the pivot multiplier to **L** and replace the row in **U** using the pivot multiplier, like in the previous part of this assignment.

We provide pseudocode for LUP decomposition in Algorithm 2.

Algorithm 2 LUP decomposition of A

```
1:  $n \leftarrow$ the number of columns of $A$
2:  $L \leftarrow I_n$ where $I_n$ is $n \times n$ identity matrix.
3:  $P \leftarrow I_n$                                        ▷ Permutation initialization
4:  $U \leftarrow A$
5:  for $1 \leq i \leq n-1$ do
6:      $j \leftarrow$ the row index of the largest absolute value among rows $i$ to $n$ in the $i^{\text{th}}$ column of $U$
7:      $[U, L, P]$ = interchange_lup($U$, $i$, $j$, $L$, $P$)
8:      if $U_{ii}$ is nearly zero* then
9:          return $L$, $U$, and $P$                          ▷ Early termination
10:     end if
11:     for $i+1 \leq k \leq n$ do
12:         $p \leftarrow U_{ki}/U_{ii}$
13:         $[U, L]$ = replacement_lu($U$, $k$, $i$, $p$, $L$)
14:     end for
15: end for
16: return $L$, $U$, $P$
```

The red lines specify the major modification for partial pivoting.

*As before, consider a number to be nearly zero if its absolute value is smaller than 10^{-12}.

Note: When terminating early, $\mathbf{L}$ and $\mathbf{P}$ are not *unique*, i.e., the order of rows corresponding to zero rows in $\mathbf{U}$ can be different. As long as the resulting decomposition satisfies $\mathbf{PA} = \mathbf{LU}$, the solution $\mathbf{L}$ and $\mathbf{P}$ are valid.

Test cases:

- Partial pivoting:
  `[L,U,P] = my_lup([-2,5,3;2,3,9;4,-2,2])`
  $\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.5 & 1 & 1 \end{bmatrix}$, $\mathbf{U} = \begin{bmatrix} 4 & -2 & 2 \\ 0 & 4 & 8 \\ 0 & 0 & -4 \end{bmatrix}$, and $\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
- Early termination:
  `[L,U,P] = my_lup([4,-2,2;-2,5,3;4+5E-13,-2+2E-13,2+7E-13])`
  $\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$, $\mathbf{U} = \begin{bmatrix} 4 & -2 & 2 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \end{bmatrix}$, and $\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
- You can test your algorithm by comparing its results with MATLAB's built-in function, `[l, u, p] = lu(A)`, on a randomly generated $n \times n$ matrix A, e.g., `A=rand(10,10)`.

4 Written Problems **(40 points)**

1. Use LU factorization **<u>only</u>** to solve:

$$\begin{cases} 2x_1 + 2x_2 + 3x_3 + 4x_4 & = 8 \\ 2x_1 + 4x_2 + 9x_3 + 16x_4 & = 6 \\ 4x_1 + 8x_2 + 24x_3 + 63x_4 & = -26 \\ 6x_1 + 16x_2 + 51x_3 + 100x_4 & = 6 \end{cases}$$

   Using other approaches will result in partial credits only. Show all steps.

2. Let $\mathbf{A} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 5 & 6 \\ 2 & 11 & 5 \end{bmatrix}$. Find the elementary matrices $\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}$ such that $\mathbf{E_3 E_2 E_1 A} = \mathbf{U}$. Then, using **<u>only</u>** the elementary matrices you found, calculate $\mathbf{L}$. Do not compute $\mathbf{L}$ based on your reduction from $\mathbf{A}$ to $\mathbf{U}$.

   Note that $\mathbf{L}$ and $\mathbf{U}$ here refer to the LU factorization of $\mathbf{A}$. Show all steps.