MATLAB assignment 10

Introduction to Linear Algebra (The last assignment) **Fall, 2019**

1. In the assignment 7, we have dealt with a function file that changes a matrix A into the reduced row echelon form **only if** 'the row interchange operation' is **not** needed during the *rref* process.
   But in this assignment, we are going to take it one step further. We will write a function file `my_rref.m` to find the reduced row echelon form of a **general** $m \times n$ matrix A by performing Gauss-Jordan elimination. In other words, the `my_rref.m` can change A into rref$A$ even if the 'the row interchange operation' is necessary in the *rref* process.
   Check your result by applying your function for the augmented matrix given in the Example 5 of the Section 2.2 of the textbook.

   - To make the function file `my_rref.m` which is described as the problem, you should begin with the function defining line.
   - We have already learned how to implement the Gauss-Jordan elimination process in MATLAB. The only additional part is the 'row interchange' part.
     (See assignment 7.)
   - It should be very helpful to do the Gauss-Jordan process by your own hands and find **the circumstance** under which row interchange should occur during the process.
   - There is no guide code. Start with an empty script to write your code. Do not copy someone else's code. It's nice to reference each other, but if you submit exactly the same codes, you get a zero point.
   - To validate and score your code, we will run the code on several random matrices.

solution.

```
function rref_A = my_rref(A)
% Find the reduced row echelon form of A
% by performing Gauss-Jordan elimination with partial pivoting.
[m, n] = size(A);
rref_A = A;                  % Initialization of rref_A as A.

% Forward Phase with row interchanges.
rowIdx = 1;                  % Count row.
for colIdx = 1 : n
    % Find element to interchange two rows.
    [maxEntry, maxIdx] = max(abs(rref_A(rowIdx:m, colIdx)));

    if maxEntry >= 1E-10         % If This column is a non-zero vector,
        % Store index of pivot column.
        pivotCols(rowIdx) = colIdx;
        % Interchanging two rows.
        rref_A([rowIdx, maxIdx + rowIdx - 1], :) = rref_A([maxIdx + rowIdx - 1, rowIdx], :);
        % Normalizing current row.
        rref_A(rowIdx, :) = rref_A(rowIdx, :) / rref_A(rowIdx, colIdx);
        % Successive row operation.
        for r = rowIdx + 1 : m
            rref_A(r, :) = rref_A(r, :) - rref_A(r, colIdx) * rref_A(rowIdx, :);
        end
        rowIdx = rowIdx + 1;     % Move onto the next row.
    end
    if rowIdx > m                % If it was finished, stop.
        break
    end
end

% Backward phase.
for pc = pivotCols(end:-1:1)   % Find only the pivot columns.
    preLding1Row = 0;

    % This is looking for the index of row in which leading 1 is apeared.
    for lding1Row = m - preLding1Row : -1 : 1
        preLding1Row = preLding1Row + 1;
        if rref_A(lding1Row, pc) >= 1E-10
            break
        end
    end
    for r = (lding1Row - 1): -1 : 1
        % Add 'minus (r, pivotCol)-entry times the ith row' to the kth row.
        rref_A(r, :) = rref_A(r, :) - rref_A(r, pc) * rref_A(lding1Row, :);
    end
end
end
```