MATLAB assignment 6 solution

Introduction to Linear Algebra (Week 6) **Fall, 2019**

1. (*Programming in* MATLAB)

   In this problem, we want to program to compute a determinant of an $n \times n$ matrix by using a cofactor expansion along the first column without using the MATLAB command `det`. Actually there are many way to write code for our purposes, but at this time let's think about programming using **recursive method**. In order to do this, we need to know about followings

   - function file
   - recursive function call.

   A **function** begins with a function definition line, which has a well-defined list of inputs and outputs. The syntax of the function definition line is as follows:

   $$\texttt{function}\ [output\ variables]\ =\ function\_name(input\ variables)$$

   Problem (1). Write a function `mySum` which produces $a + b$ as an output when it takes two numbers a and b as inputs. Call the function in the command window by using a command:

   ```
   >> aPb = mySum(3, 5)
   ```

   solution.

   ```
   function aplusb = mySum(a,b)
   aplusb = a + b;
   end
   ```

   Problem (2). Write a function `myPM` which produces $a + b$, and $a - b$ as outputs when it takes two numbers a and b as inputs. Call the function in the command window by using a command:

   ```
   >> [aPb, aMb] = myPM(3, 5)
   ```

   solution.

   ```
   function [aplusb, aminusb] = myPM(a,b)
   aplusb = a + b;
   aminusb = a - b;
   end
   ```

In a function, the **recursive method** is used to produce 'more complex results' from 'the bottommost result'. For example, the factorial function can be defined recursively with the basic result

$$0! = 1$$

and a rule

$$n! = n \times (n-1)!, \quad \forall\, n > 0.$$

This is an example of a recursive factorial function. Observe that the function `myFactorial` is reused within its own code:

```
function NFac = myFactorial(N)
   if N==0
      NFac=1;
   else
      NFac=N*myFactorial(N-1);
   end
end
```

Considering the items below, write a function file `myDet.m` to compute the determinant of a matrix A by using the co-factor expansion along the first column for the smaller matrices.

- Make a new function file with a function name `myDet`. Make a matrix A an input to the function, and the determinant of A an output.
- Using the recursive functional call, compute $\det(A)$ by cofactor expansion along the first column. For a 2×2 matrix $A = (a_{ij})$, the determinant is $a_{11}a_{22} - a_{12}a_{21}$. You may use `if-else if-if` statement and `for` loop, symultaniously.
- On the MATLAB, execute the function file `myDet.m` by typing the command:

  ```
  >> myDet(A)
  ```

Problem (3). For a matrix

$$A = \begin{bmatrix} 2 & 3 & -1 & 1 \\ -3 & 2 & 0 & 3 \\ 3 & -2 & 1 & 0 \\ 3 & -2 & 1 & 4 \end{bmatrix},$$

find the determinant of A using `myDet`. Check your answer with the result of MATLAB command `det`.

solution.

```
function [D] = myDet(A)      % A: input matrix, D: output
                             % (determinant of input matrix A)
    [m,n]=size(A);           % the size of given input matrix A.
    D=0;                     % initialization of deteminant.

    if m~=n                  % if A is not a square matrix,
        % error message
        fprintf('Error: given matrix is not a square matrix!\n');
    else
        % othersize,
        if m==2              % if input matrix A is 2x2 matrix,
            % compute the determinant using ad-bc.
            D=A(1,1)*A(2,2)-A(1,2)*A(2,1);
        else
            if m>2           % if input matrix A is not 2x2 matrix,
            % In this for loop, we compute the determinant
            % of A by a cofactor expansion along the first
            % column of A using a recursive function call.
                for i=1:m
                D=D+(-1)^(i+1)*A(i,1)*myDet(A([1:i-1, i+1:m], 2:n));
                end
            end
        end
    end
end
```