ME 318M Homework #5

Name: Jingsi Zhou

UT EID: jz24729

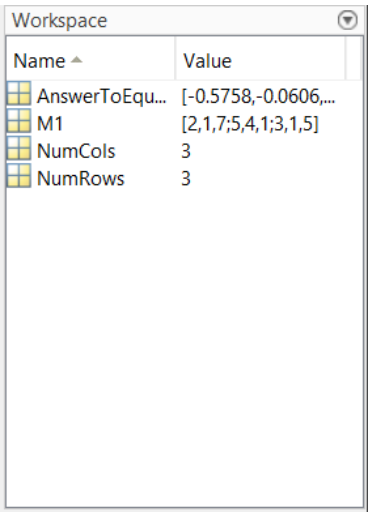
Section Number: 17460

Problem 1:

```
function [NumRows, NumCols, MatR] =
MatrixDescriptionAndOperator(inputMatrix);
%Input a Matrix M and the function will output number of rows, number of
%columns, and the solution to (inverse(transposed Matrix M * Matrix M)) *
%transposed Matrix M.
%Output format: [number of rows, number of columns, answer to equation]

[NumRows, NumCols] = size(inputMatrix);
MatR = inv((inputMatrix)' * inputMatrix)* (inputMatrix)';
```

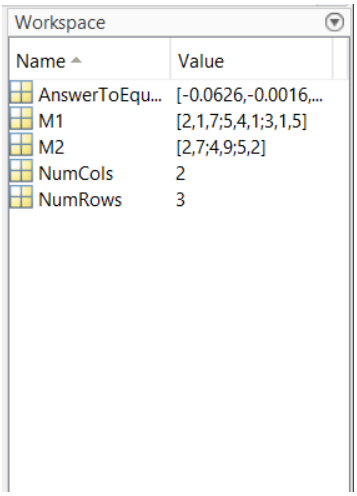
a)

```
>> M1 = [2,1,7;5,4,1;3,1,5];
>> [NumRows, NumCols, AnswerToEquation] = MatrixDescriptionAndOperator(M1)

NumRows =

     3


NumCols =

     3


AnswerToEquation =

   -0.5758   -0.0606    0.8182
    0.6667    0.3333   -1.0000
    0.2121   -0.0303   -0.0909
```



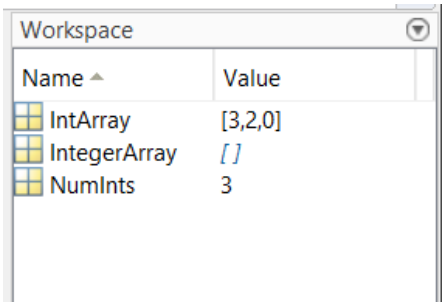
b)

```
>> M2 = [2,7;4,9;5,2];
>> [NumRows, NumCols, AnswerToEquation] = MatrixDescriptionAndOperator(M2)

NumRows =

     3


NumCols =

     2


AnswerToEquation =

   -0.0626   -0.0016    0.2263
    0.0802    0.0679   -0.0864
```



Problem 2:

```
function [IntegerArray, IsInteger] = IntArray(inputArray);
%Given an input array, the output array will only have integers. Also, the
%function will display the number of integers in the array.
%Sample Syntax: [IntegerArray, NumIntegers] = IntArray([1 2 3 4])

count = 0;
NewIndex = 0;
for i = 1:length(inputArray)
    if floor(inputArray(i)) == inputArray(i)
        NewIndex = NewIndex + 1;
        IntegerArray(NewIndex) = inputArray(i);
        IsInteger = NewIndex;
    end
    if floor(inputArray(i)) ~= inputArray(i)
        count = count + 1;
    end
end

if count == length(inputArray)
    IntegerArray = [];
    IsInteger = 0;
End
```



Command Window

```
>> [IntegerArray, NumInts] = IntArray([6.6, 4.4, 34.3])

IntegerArray =

     []


NumInts =

     0

>> [IntArray, NumInts] = IntArray([6.3, 3, 3.6, 2, 0, 6.343])

IntArray =

     3     2     0


NumInts =

     3
```

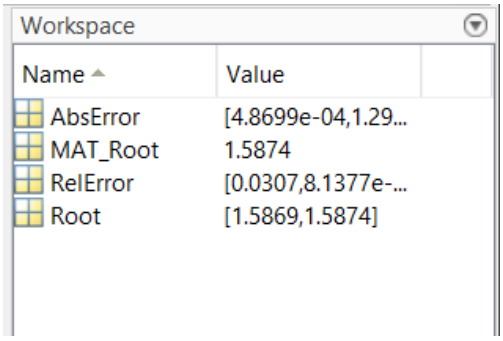
Problem 3:

```
function [Root1] = BisectRoot(Polynomial, a, b);
%The BisectRoot function finds a single root of a polynomial using the
%Bisection Method. The input must be a polynomial in array form.
%Example: [Root1] = BisectRoot([3 0 2], 0, 3), where [3 0 2] can also be
written
%as y(x) = 3x^3 + 2 and the root is to be found between y(0) and y(3). The
%output of this function gives the root between two intervals as an array.

LeftSol = polyval(Polynomial, a);
RightSol = polyval(Polynomial, b);

if LeftSol*RightSol < 0
    LeftX = a;
    RightX = b;
    while abs(RightX - LeftX) > 0.0005
        MidX = (LeftX + RightX)/2;
        Middle = polyval(Polynomial, MidX);
        if Middle*RightSol < 0
            LeftX = MidX;
        else
            RightX = MidX;
        end
    end
else
    disp(['There is no root between ', num2str(a), ' and ', num2str(b)])
end

Root1 = [LeftX, RightX];
```

Command Window

Note: The relative error is calculated in percent form.

```
>> Root = BisectRoot([1 0 0 -4], 1, 2)

Root =

    1.5869    1.5874

>> MAT_Root = 4^(1/3)

MAT_Root =

    1.5874

>> AbsError = abs(MAT_Root - Root)

AbsError =

   1.0e-03 *

    0.4870    0.0013
```



```
>> RelError = (AbsError/MAT_Root)*100

RelError =

    0.0307    0.0001
```

Problem 4:

a) The researcher probably made a table of integer x values and evaluated the function at those values. The researcher is convinced that one root of the equation is between 0 and 1 because f(0)*f(1) < 0.
b) 10 iterations
c) Edited **BisectRoot** function:

```
function [Root1, NumIterations] = BisectRoot(Polynomial, a, b);
%The BisectRoot function finds a single root of a polynomial using the
%Bisection Method. The input must be a polynomial in array form.
%Example: [Root1] = BisectRoot([3 0 2], 0, 3), where [3 0 2] can also be
written
%as y(x) = 3x^3 + 2 and the root is to be found between y(0) and y(3). The
%output of this function gives the root between two intervals as an array.

Error = input('What do you need your maximum absolute error to be? ');
LeftSol = polyval(Polynomial, a);
RightSol = polyval(Polynomial, b);

if LeftSol*RightSol < 0
    LeftX = a;
    RightX = b;
    count = 0;
    while abs(RightX - LeftX) > Error
        MidX = (LeftX + RightX)/2;
        Middle = polyval(Polynomial, MidX);
        if Middle*RightSol < 0
            LeftX = MidX;
        else
            RightX = MidX;
        end
        count = count + 1;
        disp(['iteration ', num2str(count), ': ', num2str(LeftX), ', ',
num2str(RightX)])
    end
else
    disp(['There is no root between ', num2str(a), ' and ', num2str(b)])
end

Root1 = (['The root of the equation is located between ',
num2str(LeftX), ...
    ' and ', num2str(RightX)]);
NumIterations = count;
```

Command Window:

```
>> [Root, Count] = BisectRoot([1 -4.9 6.73 -2.011], 0, 1)
What do you need your maximum absolute error to be? 0.001
iteration 1: 0, 0.5
iteration 2: 0.25, 0.5
iteration 3: 0.375, 0.5
iteration 4: 0.375, 0.4375
iteration 5: 0.40625, 0.4375
iteration 6: 0.40625, 0.42188
iteration 7: 0.40625, 0.41406
iteration 8: 0.41016, 0.41406
iteration 9: 0.41016, 0.41211
iteration 10: 0.41113, 0.41211

Root =

    'The root of the equation is located between 0.41113 and 0.41211'


Count =

    10
```

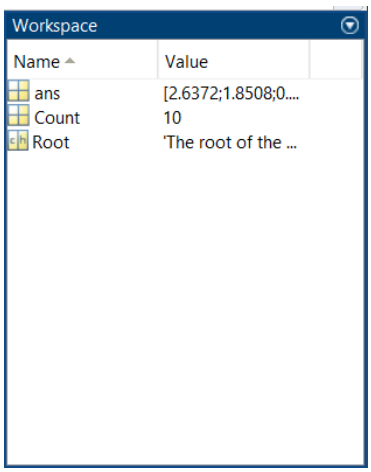


Table:

Count	Left x value	Right x value
1	0	0.5
2	0.25	0.5
3	0.375	0.5
4	0.375	0.4375
5	0.40625	0.4375
6	0.40625	0.42188
7	0.40625	0.41406
8	0.41016	0.41406
9	0.41016	0.41211
10	0.41113	0.41211

The root of the equation is located between 0.41113 and 0.41211.