MAS3114 MATLAB Assignment 2

Name: Samantha Bennett

UF ID: ####-####

WARNING: You will recieve a zero on this assignment if you fail to enter your name, UF ID, and the seed in the random number generator.

```
clear
% Initialize the random number generator by typing the command
rng(########,'twister')
% where ######## must be your 8-digit UF ID number
```

Exercise 1 -- Linear Dependence

1A

```
% part(1)
A1 =  randi([-8,8],4,6)
```

```
A1 = 4×6
     6     5     3     2     3     8
    -4     2     1     5    -1    -2
     2     3     1    -3    -8     5
    -4    -1     4    -1     3     8
```

```
dependence(A1);
```

```
The set of the columns of the matrix is linearly dependent.
```

```
A2 =  randi([-8,8],4,6)
```

```
A2 = 4×6
    -2    -5     7    -3     5     8
    -7    -1     2     6     2    -1
    -6    -6    -6     4    -5    -5
    -3    -2     0    -7    -8     5
```

```
dependence(A2);
```

```
The set of the columns of the matrix is linearly dependent.
```

```
A3 =  randi([-8,8],4,6)
```

```
A3 = 4×6
    -5    -6    -1     1    -2     0
     7     8     2     7     5     1
    -6    -5    -8     0     0     6
    -2     7     2    -1    -7     8
```

```
dependence(A3);
```

```
The set of the columns of the matrix is linearly dependent.
```

```
% repeat the process for A2 and A3
```

```
% part(2)
B1 =  randi([-8,8],6,4)
```

```
B1 = 6×4
     8     2     0     4
    -7     5    -4    -3
    -6    -5     5    -7
     3     5     8    -2
    -8    -6     3    -3
    -8     0    -2     6
```

```
dependence(B1);
```

```
The set of the columns of the matrix is linearly independent.
```

```
B2 =  randi([-8,8],6,4)
```

```
B2 = 6×4
     6     7    -8    -6
    -3     5     7    -8
    -1    -6     6     1
    -7    -5    -1     2
    -1     3     2     2
     3     3    -7     0
```

```
dependence(B2);
```

```
The set of the columns of the matrix is linearly independent.
```

```
B3 =  randi([-8,8],6,4)
```

```
B3 = 6×4
    -7    -5     5    -4
     1     6     0    -6
    -7     4     6    -7
     3     6     0     4
    -3    -7     0    -6
     3     3     0     6
```

```
dependence(B3);
```

```
The set of the columns of the matrix is linearly independent.
```

```
% repeat the process for B2 and B3
```

1B Observe the result from part(1). For most cases, the columns of A are linearly dependent. Are the columns of any 4 x 6 matrix always linearly dependent? (If yes, provide a reason using the concepts in Linear Algebra. If not, provide an example.)

Yes, the columns of any 4 x 6 matrix are always going to be linearly dependent. This is because matrices are automatically linearly dependent when the matrix contains more columns than rows (more vectors than entries in each vector). So any matrix with the size 4 x 6 (4 rows < 6 columns) will be linearly dependent.

1C Observe the result from part(2). For most cases, the columns of B are linearly independent. Are the columns of any 6 x 4 matrix always linearly independent? (If yes, provide a reason using the concepts in Linear Algebra. If not, provide an example -- do not give a trivial example -- the zero matrix.)

No, the columns of any 6 x 4 matrix are not always going to be linearly independent. This is because the matrix has to have not only more columns than rows to be linearly independent. The matrix also has to not have a zero vector in it. A counter example to the claim of any 6 x 4 matrix is always linearly independent is:

$$\begin{matrix} 1 & 9 & 0 & 4 & 5 & 7 \\ 2 & 5 & 0 & 3 & 2 & 1 \\ 4 & 8 & 0 & 2 & 1 & 7 \\ 6 & 9 & 0 & 3 & 1 & 3 \end{matrix}$$

Due to the zero vector in the third column, this 6 x 4 matrix is linearly dependent.

Exercise 2 -- Onto/One-to-One Linear Transformations

2A (see the file transformation.m)

2B

```
% For T: R^6 --> R^4
disp('For linear transformation T(x) = Ax, where A=')
```

```
For linear transformation T(x) = Ax, where A=
```

```
A1
```

```
A1 = 4×6
     6     5     3     2     3     8
    -4     2     1     5    -1    -2
     2     3     1    -3    -8     5
    -4    -1     4    -1     3     8
```

```
transformation(A1)
```

```
The transformation is onto but not one-to-one
```

```
A2
```

```
A2 = 4×6
    -2    -5     7    -3     5     8
    -7    -1     2     6     2    -1
    -6    -6    -6     4    -5    -5
    -3    -2     0    -7    -8     5
```

```
transformation(A2)
```

```
The transformation is onto but not one-to-one
```

```
A3
```

```
A3 = 4×6
    -5    -6    -1     1    -2     0
     7     8     2     7     5     1
    -6    -5    -8     0     0     6
    -2     7     2    -1    -7     8
```

```
transformation(A3)
```

```
The transformation is onto but not one-to-one
```

```
% repeat the process for A2 and A3
%
% For T: R^4 --> R^6
disp('For linear transformation T(x) = Ax, where A=')
```

```
For linear transformation T(x) = Ax, where A=
```

```
B1
```

```
B1 = 6×4
     8     2     0     4
    -7     5    -4    -3
    -6    -5     5    -7
     3     5     8    -2
    -8    -6     3    -3
    -8     0    -2     6
```

```
transformation(B1)
```

```
The transformation is one-to-one but not onto
```

```
B2
```

```
B2 = 6×4
     6     7    -8    -6
    -3     5     7    -8
    -1    -6     6     1
    -7    -5    -1     2
    -1     3     2     2
     3     3    -7     0
```

```
transformation(B2)
```

```
The transformation is one-to-one but not onto
```

```
B3
```

```
B3 = 6×4
    -7    -5     5    -4
     1     6     0    -6
    -7     4     6    -7
     3     6     0     4
    -3    -7     0    -6
     3     3     0     6
```

```
transformation(B3)
```

```
The transformation is one-to-one but not onto
```

```
% repeat the process for B2 and B3
%
% For T: R^4 --> R^4
disp('For linear transformation T(x) = Ax, where A=')
```

```
For linear transformation T(x) = Ax, where A=
```

```
C1 = randi([-8,8],4,4)
```

```
C1 = 4×4
     1     2     4    -1
     0    -3    -7     8
     3    -7    -1     0
    -5     0     5    -5
```

```
transformation(C1)
```

```
The transformation is onto and one-to-one.
```

```
C2 = randi([-8,8],4,4)
```

```
C2 = 4×4
     7    -7     0    -5
    -2    -6    -3    -6
     0    -2     5     1
    -4     6     4     1
```

```
transformation(C2)
```

```
The transformation is onto and one-to-one.
```

```
C3 = randi([-8,8],4,4)
```

```
C3 = 4×4
    -4     8     2     8
     0     8    -6     3
    -4    -1    -1     1
    -1     2     7    -6
```

```
transformation(C3)
```

```
The transformation is onto and one-to-one.
```

```
% repeat the process for C2 and C3
```

2C Observe the result from 2B, and write your report below.

- When m < n: Never be one-to-one, the number of pivot positions are limited to the number of rows if the number of rows is greater than the number of columns. If being one-to-one means having the same number for the number of columns and number of pivot positions, it would be impposible to be one-to-one if the number of rows is always less than the number of columns. There would always be extra columns without pivot positions because there is not enough rows to have pivot positions in them.
- When m > n: Never be onto, the number of pivot positions are limited to the number of columns whent the number of columns is greater than the number of rows. If being onto means having the same number of rows and pivot positions, it would be immpossible to be onto if the number of columns is always less than the number of rows. There would always be extra rows in the matrix that do not have a pivot position because there are not enough columns to reach those rows.
- When m = n: the transformation $T(\mathbf{x}) = D\mathbf{x}$ is neither onto nor one-to-one.

```
% give an example of T(x) = Dx, where T is neither onto nor one-to-one and
% verify your example by calling transformation.
% Do not give a trivial example -- the zero matrix for D
D= [1,2,3,4; 1,2,3,4; 1,2,3,4; 0,0,0,0]
```

```
D = 4×4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     0     0     0     0
```

```
transformation(D)
```

```
The transformation is neither onto nor one-to-one
```

Extra Credit: When m = n, is it possible that a linear transformation

- $T : \mathbb{R}^n \to \mathbb{R}^n$ is onto but not one-to-one?

This is not possible because being onto means the number of rows equals the number of piviot positions. If the number of rows also equals the number of columns, it is impossible to have a only the number of rows equal to the number of pivot positions.

- $T : \mathbb{R}^n \to \mathbb{R}^n$ is one-to-one but not onto?

This is not possible because being one-to-one means the number of columns equals the number of piviot positions. If the number of columns also equals the number of rows, it is impossible to have a only the number of columns equal to the number of pivot positions.

(If yes, give an example. If not, provide a reason using Linear Algebra.)

Exercise 3 -- Matrix Operations

3A

```
n = randi([5,10]);  % The size of matrix A is n x n.
```

for loop (i)

```
A = zeros(n,n);
for i = 1:n
    for j = 1:n
        A(i,j) = i+j; % do not print each entry of A inside the loop
    end
end
A
```

```
A = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

for loop (ii)

```
A = zeros(n,n);
for i = 1:n
    for j = i:n
        A(i,j) = i+j;
        A(j,i) = A(i,j); % do not print each entry of A inside the loop
    end
end
A
```

```
A = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

- Do you have the same output A for loops (i) and (ii)? yes
- For an nxn matrix A, calculate the number of flops (additions) used in each for loop. (express your answer in terms of n) loop i = n^2; loop ii = $\frac{n^2}{2}$
- Which for loop uses fewer flops? loop ii

3B

while loop (i)

```
% rewrite for loop (i) using while loops only and display matrix A
% do not print each entry of A inside the loop
A = zeros(n,n);
i = 1;
while i <= n
    j = 1;
    while j <= n
        A(i,j) = i+j;
        j = j + 1;
    end
    i = i + 1;
end
A
```

```
A = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

while loop (ii)

```
% rewrite for loop (ii) using while loops only and display matrix A
% do not print each entry of A inside the loop
A = zeros(n,n);
i = 1;
while i <= n
    j = 1;
    while j <= n
        A(i,j) = i+j;
        A(j,i) = A(i,j);
        j = j + 1;
    end
    i = i + 1;
end
A
```

```
A = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

3C

```
B = randi([-8,8],n,n-2)
```

```
B = 6×4
    -2    -6     7    -8
     4     5    -7     6
     6     6     7     5
     1     4    -6    -4
    -1     0     0     3
    -1     4     8    -1
```

```
A*B
```

```
ans = 6×4
    24    75    47    13
    31    88    56    14
    38   101    65    15
    45   114    74    16
    52   127    83    17
    59   140    92    18
```

```
%B*A the number of columns in B do not match the number of rows in A, which
%means you cannot multiply these two matrices together in this order
```

(Only one matrix product works. Add a % sign to the command that does not work and give a reason why it does not work?)

```
C = randi([-8,8],n,n)
```

```
C = 6×6
    -4    -6    -6    -1     0     4
    -1     7     8     4     5    -7
```

```
    -1     6    -6     2     1     3
    -6    -2     3    -7    -2     1
    -5    -4    -7     4    -6     1
     7    -1     7     1    -1    -8
```

```
A*C
```

```
ans = 6×6
   -26    -8    10    14   -34   -46
   -36    -8     9    17   -37   -52
   -46    -8     8    20   -40   -58
   -56    -8     7    23   -43   -64
   -66    -8     6    26   -46   -70
   -76    -8     5    29   -49   -76
```

```
C*A
```

```
ans = 6×6
   -27   -40   -53   -66   -79   -92
    52    68    84   100   116   132
    29    34    39    44    49    54
   -46   -59   -72   -85   -98  -111
   -59   -76   -93  -110  -127  -144
   -18   -13    -8    -3     2     7
```

Do you have the same output from the matrix products? No

In general, the matrix multiplication is not commutative. (use the term that was mentioned in the lecture)

```
A*eye(n)
```

```
ans = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

```
eye(n)*A
```

```
ans = 6×6
     2     3     4     5     6     7
     3     4     5     6     7     8
     4     5     6     7     8     9
     5     6     7     8     9    10
     6     7     8     9    10    11
     7     8     9    10    11    12
```

Do you have the same output? Yes

A*I = I*A = A

3D

```
inv(A)
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  9.738798e-19.
ans = 6×6
10^15 ×
```

```
   -0.8386    2.4770   -1.2424   -1.2035    0.4193    0.3882
    2.4770    0.7318   -4.5036   -0.2815   -1.2385    2.8147
   -1.2424   -4.5036    6.8330    1.5530    0.6212   -3.2612
   -1.2035   -0.2815    1.5530    1.2230    0.0388   -1.3297
    0.4193   -1.2385    0.6212    0.0388    0.9162   -0.7571
    0.3882    2.8147   -3.2612   -1.3297   -0.7571    2.1450
```

Based on the warning message, is A invertible? No

```
% create matrix B
B = [1, 0, 2; 2, 1, 5; 0, 1, 2]
```

```
B = 3×3
     1     0     2
     2     1     5
     0     1     2
```

```
inv(B)
```

```
ans = 3×3
    -3     2    -2
    -4     2    -1
     2    -1     1
```

```
% Write the code using RREF (at most 2 lines) to find the inverse of B
G = rref([B eye(3)]);
G(:,4:6)
```

```
ans = 3×3
    -3     2    -2
    -4     2    -1
     2    -1     1
```

Note: You should get the same inverse of B for both methods.

```
inv(inv(B))
```

```
ans = 3×3
     1     0     2
     2     1     5
     0     1     2
```

$(B^{-1})^{-1} = B$

```
% create matrix C
C = [1, -1, 2; 0, 0, 1; 1, 1, -2]
```

```
C = 3×3
     1    -1     2
     0     0     1
     1     1    -2
```

```
inv(B*C)
```

```
ans = 3×3
   -0.5000    0.5000   -0.5000
   -5.5000    2.5000   -0.5000
   -4.0000    2.0000   -1.0000
```

```
inv(B)*inv(C)
```

```
ans = 3×3
   -2.5000    2.0000   -0.5000
   -3.0000    3.0000   -1.0000
    1.5000   -1.0000    0.5000
```

```
inv(C)*inv(B)
```

```
ans = 3×3
   -0.5000    0.5000   -0.5000
   -5.5000    2.5000   -0.5000
   -4.0000    2.0000   -1.0000
```

$(BC)^{-1} = B^{-1} * C^{-1}$

```
inv(B')
```

```
ans = 3×3
    -3    -4     2
     2     2    -1
    -2    -1     1
```

```
(inv(B))'
```

```
ans = 3×3
    -3    -4     2
     2     2    -1
    -2    -1     1
```

Is $(B^T)^{-1} = (B^{-1})^T$? Yes

Extra Credit: Display an n x n matrix A whose entries are random integers between -8 and 8, and use for loops to display the transpose of A ($B = A^T$).

Note: n is the random integer used in 3A.

```
A = randi([-8,8],n,n)
```

```
A = 6×6
     1     3    -4    -5    -7    -8
     5     5    -3    -8     3    -2
    -1     5    -4     7     1     2
     8    -7     7     1     2    -4
     7     5     1    -1     4    -4
     4     6     2     6    -1    -6
```

```
B = zeros(n,n); %initialize the matrix
% for loops
for i = 1:n
    for j = 1:n
        B(j,i) = A(i,j); % do not print each entry of A inside the loop
    end
end
% do not print each entry of B inside the loop
```

```
B %transpose of A
```

```
B = 6×6
     1     5    -1     8     7     4
     3     5     5    -7     5     6
    -4    -3    -4     7     1     2
    -5    -8     7     1    -1     6
    -7     3     1     2     4    -1
    -8    -2     2    -4    -4    -6
```