Lab2 by Cole Bardin Section: 62

Matrix Exploration

Question 1: Solve a Linear System by Row Reducing the Augmented Matrix

```
clc % Clear command window

A = [1, 1, 1; 1, 2, -3; 1, 1, -2]; % Coefficient matrix
b = [18; 0; 0]; % b matrix
AM = [A, b]; % Augmented matrix

RAM = rref(AM); % Fully reduce matrix to RREF

x = RAM(:,end); % Last column is the solution set to the system
disp(x); % Display the solution set
```

Question 2: Solve a Linear System with a Free Variable

```
clc % Clear command window

A = [1, 1, 1; 1, 1, -1; 1, 1, 0]; % Coefficient matrix
b = [4; 0; 2]; % b matrix
AM = [A, b]; % Augmented matrix

RAM = rref(AM); % Fully reduced matrix to RREF
disp(RAM); % Display the RREF matrix
```

Question 3: Matrices in general do not commute

```
clc % Clear command window

A = [1, 2; 2, 1]; % A matrix for operation

% 3a)
% Test matricies
B = [3, 4; 4, 3];
C = [1, 2; 2, 2];
D = [1, 2; 3, 4];

% 2x2 Array of zeros
z = zeros(2);

AcB = A*B - B*A; % Calculate A Commute B
AcC = A*C - C*A; % Calculate A Commute C
AcD = A*D - D*C; % Calculate A Commute D

% Display results of A commute B
```

```
if isequal(AcB, z)
    disp("A Commutes with B")
else
    disp("A does not Commute with B")
end

% Display results of A commute C
if isequal(AcC, z)
    disp("A Commutes with C")
else
    disp("A does not Commute with C")
end

% Display results of A commute D
if isequal(AcD, z)
    disp("A Commutes with D")
else
    disp("A does not Commute with D")
end

% 3b
% Declare symbols a b c d
syms a b c d
Bsym = [a, b; c, d]; % Make matrix with symbols

AmB = ( A*Bsym - Bsym*A ) * 0.5; % Commute A and Bsym
% Resultants are scaled by a factor of 1/2 to simplify
% Print results
fprintf("AmB =\n")
disp(AmB)

% 3c
F = [1, 2; -2, 1]; % F matrix
FmB = ( F*Bsym - Bsym*F ) * 0.5; % Commute G and Bsym
% Print results
fprintf("FmB =\n")
disp(FmB)
```

Question 4: Commuting Pairs of Matrices

```
clc % Clear the Command Window

count = 0; % Make count variable equal to 0
rng(2020) % Set random dumber generation to seed 2020

for i=1:10000 % Iterate 10000 times
    % Generate two random 2x2 matrices with values from -5 to 5
    A = randi([-5 5], 2);
    B = randi([-5 5], 2);
    % Check if A and B commute
    if isequal(A*B, B*A)
        count = count + 1;
    end
end

% Print number of commuting pairs
fprintf('4. The number of commuting pairs is %d\n', count)
```

Question 5: Trace of a matrix product AB

```
clc % Clear the command window
rng(2020); % Set the seed for random number generation to 2020

% 5a
A = magic(3);
TrA = trace(A);
fprintf('Trace(magic(3)) = %d\n', TrA)

% 5b
count_tr = 0;
count_tr_p = 0;
tolerance = 1e-10;
for i=1:10000 % Iterate 10000 times
    % Generate two random 2x2 matrices with values from -5 to 5
    A = randi([-5 5], 2);
    B = randi([-5 5], 2);

    % Check if Tr(AB) = Tr(BA)
    if abs(trace(A*B)-trace(B*A)) < tolerance
        count_tr = count_tr + 1;
    end
    % Check if Tr(AB) = Tr(A)*Tr(B)
    if abs(trace(A*B) - (trace(A)*trace(B))) < tolerance
        count_tr_p = count_tr_p + 1;
    end
end
% Print results
fprintf('5. The number of pairs satisfying Tr(AB) = Tr(BA) is %d\n', count_tr)
fprintf('5. The number of pairs satisfying Tr(AB) = Tr(A)*Tr(B) is %d\n', count_tr_p)
```

Question 6: Determinant of a matric product AB

```
clc % Clear the command window

count = 0;
tolerance = 1e-10;
for i=1:10000 % Iterate 10000 times
    % Generate two random 2x2 matrices with values from -5 to 5
    A = randi([-5 5], 2);
    B = randi([-5 5], 2);

    % Check if det(A*B) = det(A)*det(B)
    if abs(det(A*B)-det(A)*det(B)) < tolerance
        count = count +1;
    end
end
% Print results
fprintf('5. The number of pairs satisfying det(AB) = det(A)*det(B) is %d\n', count)
```

Question 7: Row Equivalent Matrices and the Reducing Matrix R

```
clc % Clear the command window

% 7a
rref_A = rref([1, 1, 10; 1, -1, 0]); % A matrix in RREf

% Matrices to test converted to RREF
rref_B = rref([2, 1, 15; 1, 2, 15]);
rref_C = rref([1, 1, 12; 1, -1, 0]);
rref_D = rref([1, 1, 10; 2, 2, 20]);

% Compare RREF of A to RREF of B
if isequal(rref_A, rref_B)
    disp("Matrix A is Row Equivalent to Matrix B")
else
    disp("Matrix A is NOT Row Equivalent to Matrix B")
end
% Compare RREF of A to RREF of C
if isequal(rref_A, rref_C)
    disp("Matrix A is Row Equivalent to Matrix C")
else
    disp("Matrix A is NOT Row Equivalent to Matrix C")
end
% Compare RREF of A to RREF of D
if isequal(rref_A, rref_D)
    disp("Matrix A is Row Equivalent to Matrix D")
else
    disp("Matrix A is NOT Row Equivalent to Matrix D")
end

% 7b
A = [1, 1, 10; 1, -1, 0]; % A Matrix
RA = rref(A); % Create verified RREF matrix
I = eye(2); % Create identity matrix

AM = [A, I]; % Concatenate A and I matrices

RAM = rref(AM); % Solve for RREF of [A, I]
R = RAM(:, [4,5]); % Retrieve all rows of the last two columns
```

```
% Test if R*A is equal to RA (RREF of A)
if isequal(R*A, RA)
    fprintf("R*A is equal to RA\nR =\n")
    disp(R)
else
    disp("R*A is NOT equal to RA\n")
end
```

Question 8: Solving Linear Systems Ax = b using linsolve(A, b)

```
clc % Clear command window

% 8a
A = [1, 1, 1; 1, 2, 3; 1, 1, -1]; % Create A matrix
b = [6; 14; 0]; % b matrix

XYZ = linsolve(A, b); % Use linsolve() method to generate solutions

% Display results
fprintf("Solution for system with linsolve():\nXYZ =\n")
disp(XYZ)

% 8b
AM = [A, b]; % Concatenate coefficient matrix with b matrix to form Augmented Matrix
RAM = rref(AM); % Use rref() to generate reduced solution

% Display results
fprintf("RREF of Augmented Matrix with rref():\n")
disp(RAM)
```

Questions 9-10: A Stoichiometry Example/Combustion of Propane

```
clc % Clear command window

% 9a
% Form: -X1*O + X2*W + X3*C = P
% [#C; #H; #O]
A = [0, 0, 1; 0, 2, 0; -2, 1, 2]; % Create A matrix for Propane reaction
b = [3; 8; 0]; % Create b matrix for Propane reaction

AM = [A, b]; % Create augmented matrix with A and b matrices
RAM = rref(AM); % Get RREF of the AM with rref()

% Display the RAM
fprintf("RAM =\n")
disp(RAM)
% Resulting: -5*O + 4*W + 3*C = P
```

Upgraded Challenge 1: Commuting pairs of 3x3 Matrices

```
clc % Clear the Command Window

count = 0; % Make count variable equal to 0
%rng(2020) % Set random dumber generation to seed 2020

for i=1:10000 % Iterate 10000 times
    % Generate two random 3x3 matrices with values from -5 to 5
    A = randi([-5 5], 3);
    B = randi([-5 5], 3);

    % Check if A and B commute
    if isequal(A*B, B*A)
        count = count + 1;
    end
end

% Print number of commuting pairs
fprintf('4. The number of commuting pairs is %d\n', count)
% Never got 1 commuting pair! Even with 10,000 iterations at a time
```

Upgraded Challenge 2: Combustion of Glucose

```
clc % Clear command window

% Form: -X1*O + X2*W + X3*C = G
% [#C; #H; #O]
A = [0, 0, 1; 0, 2, 0; -2, 1, 2]; % Create A matrix for Propane reaction
b = [6; 12; 6]; % Create b matrix for Propane reaction

AM = [A, b]; % Create augmented matrix with A and b matrices
RAM = rref(AM); % Get RREF of the AM with rref()

% Display the RAM
fprintf("RAM =\n")
disp(RAM)

% Resulting: -6*O + 6*W + 6*C = G
```