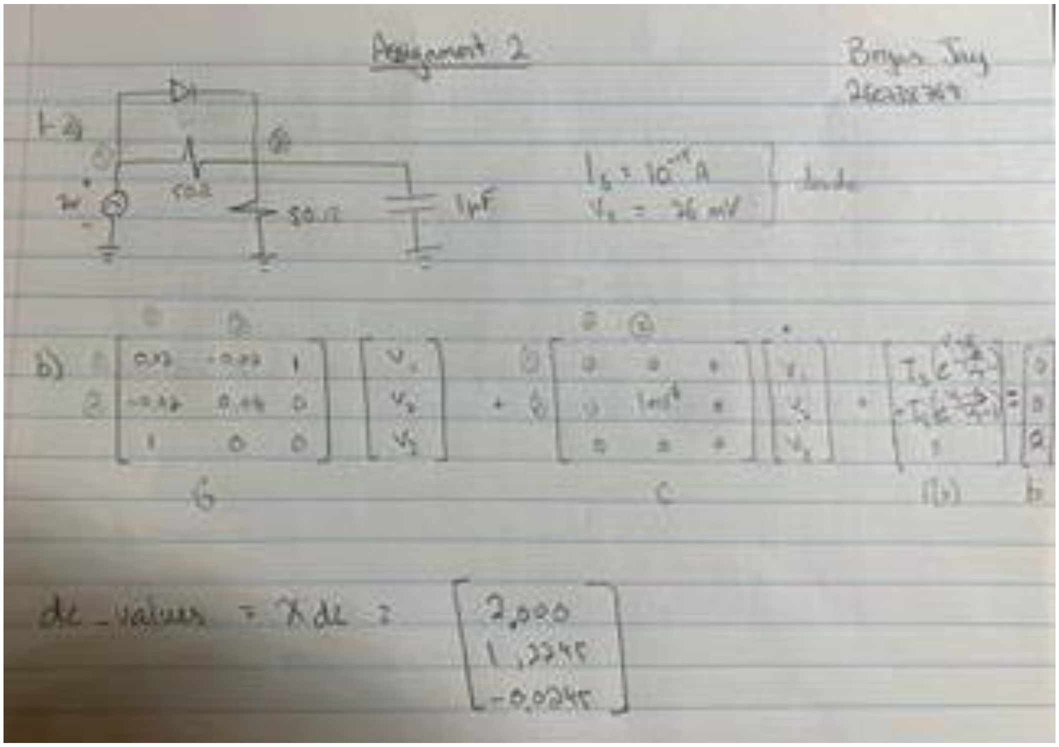
Bryan Jay
260738764

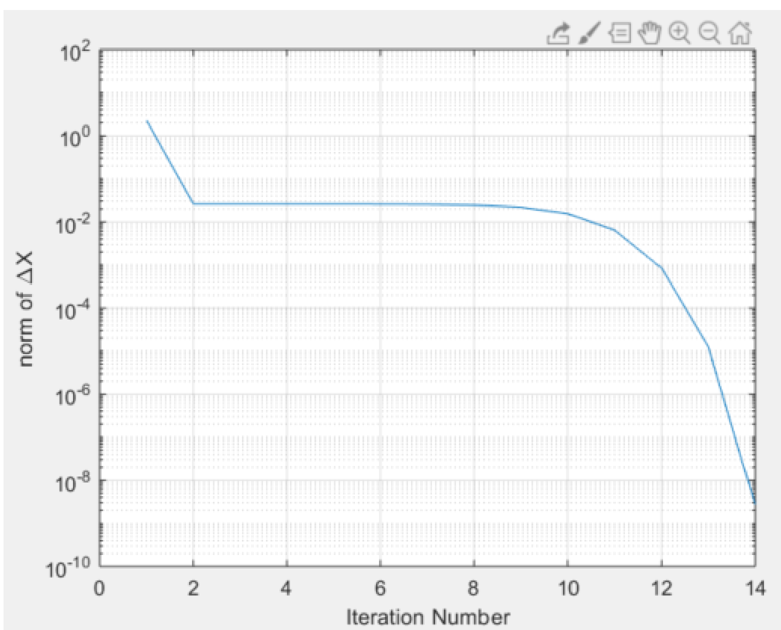
Assignment 2

1 – a)



b) dc values = Xdc = $\begin{bmatrix} 2.0000 \\ 1.2245 \\ -0.0245 \end{bmatrix}$

c) **Iteration Number vs Normal of delta x**



Bryan Jay
260738764

d) MATLAB Code:

nlJacobian.m Function:

```
function J = nlJacobian(X)
% Compute the jacobian of the nonlinear vector of the MNA equations as
a
% function of X
% input: X is the current value of the unknown vector.
% output: J is the jacobian of the nonlinear vector f(X) in the MNA
% equations. The size of J should be the same as the size of G.
global G DIODE_LIST

N = size(G);
f_d = zeros(N); % Initialize the f_d (derrivative) vector (same size
as G)

NbDiodes = size(DIODE_LIST,2);

% perform similar actions to the f_vector function
for I = 1:NbDiodes
    % fill the 3x3 matrix
    if (DIODE_LIST(I).node1 ~= 0) && (DIODE_LIST(I).node2 ~= 0)
        v1 = X(DIODE_LIST(I).node1); %nodal voltage at anode
        v2 = X(DIODE_LIST(I).node2); %nodal voltage at cathode
        Vt = DIODE_LIST(I).Vt; % Vt of diode (part of diode model)
        Is = DIODE_LIST(I).Is; % Is of Diode (part of diode model)

        % calculate the matrix based on having a diode with two nodes
        f_d = f_d + [ (Is/Vt)*exp((v1-v2)/Vt) (-1*Is/Vt)*exp((v1-
v2)/Vt) 0 ; (-1*Is/Vt)*exp((v1-v2)/Vt) (Is/Vt)*exp((v1-v2)/Vt) 0 ; 0 0
0 ];
    elseif (DIODE_LIST(I).node1 == 0)
        v2 = X(DIODE_LIST(I).node2); %nodal voltage at cathode
        Vt = DIODE_LIST(I).Vt; % Vt of diode (part of diode model)
        Is = DIODE_LIST(I).Is; % Is of Diode (part of diode model)

        f_d = f_d + [ 0 0 0 ; 0 (Is/Vt)*exp(-1*v2/Vt) 0 ; 0 0 0 ];
    elseif (DIODE_LIST(I).node2 == 0)
        v1 = X(DIODE_LIST(I).node1); %nodal voltage at anode
        Vt = DIODE_LIST(I).Vt; % Vt of diode (part of diode model)
        Is = DIODE_LIST(I).Is; % Is of Diode (part of diode model)

        % one node is connected to ground
        f_d = f_d + [ (Is/Vt)*exp(v1/Vt) 0 0 ; 0 0 0 ; 0 0 0 ];
    end
end

% return the Jacobian
J = G + f_d;
```

Bryan Jay
260738764

<u>dcsolve.m code:</u>

```
function [Xdc, dX] = dcsolve(Xguess,maxerr)
% Compute dc solution using newtwon iteration
% input: Xguess is the initial guess for the unknown vector.
%        It should be the correct size of the unknown vector.
%        maxerr is the maximum allowed error. Set your code to exit
the
%        newton iteration once the norm of DeltaX is less than maxerr
% Output: Xdc is the correction solution
%         dX is a vector containing the 2 norm of DeltaX used in the
%         newton Iteration. the size of dX should be the same as the
number
%         of Newton-Raphson iterations. See the help on the function
'norm'
%         in matlab.
global G C b

delta_x = intmax;
x_test = Xguess;
dX = [];

% since in DC this point is always 0
x_test_d = [0 ; 0 ; 0 ];

% continue iterating through until the threshold of maxerr is hit
while delta_x >= maxerr
    f = f_vector(x_test);
    phi = G*x_test + C*x_test_d + f - b;

    % Get the Jacobian matrix
    J = nlJacobian(x_test);

    % get delta_x matrix
    delta_x_m = -1 * inv(J) * phi;

    % caclulate the new point to test and get the normal of delta_x
    x_test = x_test + delta_x_m;
    delta_x = norm(delta_x_m);

    dX = [ dX , delta_x ];
end

Xdc = x_test;
```