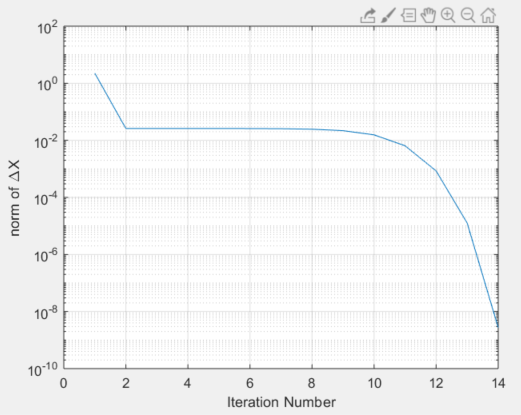
**Assignment 2**

**1 – a)**

**b)** dc values = Xdc =

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



**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;

**dcsolve.m code:**

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;