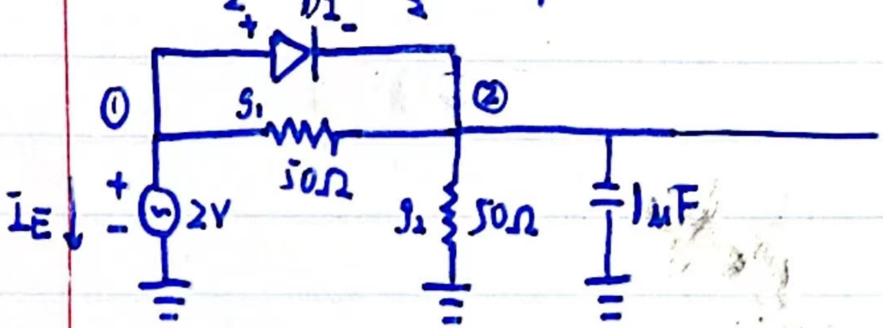
Deliverable 1

Q1)

$$Gx + sCx = b$$

$$2[\;]_2 \qquad 2[\;]_2 \qquad 2[\;]_1$$



$$I_{D1} = e^{-15}\left(e^{\frac{V_1 - V_2}{26e^{-3}}} - 1\right)$$

Q2)

$$\begin{array}{c} \\ ① \\ ② \\ ③ \end{array}\begin{array}{c} \begin{array}{ccc} ① & ② & ③ \end{array} \\ \begin{bmatrix} g_1 & -g_1 & +1 \\ -g_1 & s(+g_1+g_2 & 0 \\ +1 & 0 & 0 \end{bmatrix} \end{array} \begin{bmatrix} V_1 \\ V_2 \\ I_E \end{bmatrix} + C \overbrace{\begin{bmatrix} V_1' \\ V_2' \\ I_E' \end{bmatrix}}^{0} + \begin{bmatrix} -I_D \\ +I_D \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_E \end{bmatrix}$$

$$G \quad x(t) + C\,x'(t) + f(x(t)) = b$$

```
Xdc =

    2.0000
    1.2245
   -0.0245
```

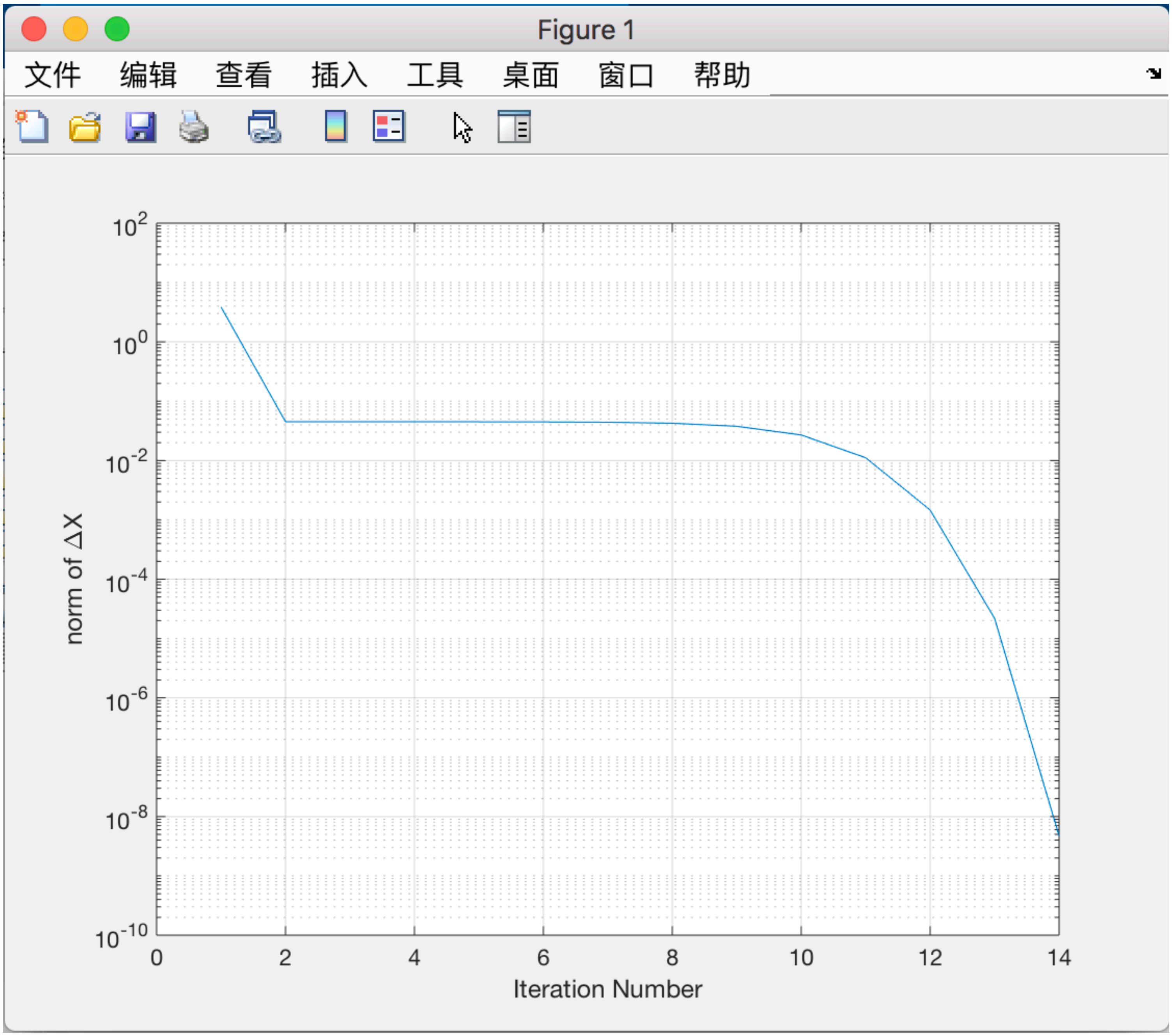


Figure 1
文件 编辑 查看 插入 工具 桌面 窗口 帮助
norm of ΔX
Iteration Number
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
2
4
6
8
10
12
14

```
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
S = size(G);
J = zeros(S);

Diodes = size(DIODE_LIST,2);

for I = 1:Diodes

    if (DIODE_LIST(I).node1 ~= 0) && (DIODE_LIST(I).node2 ~=0)
        v1 = X(DIODE_LIST(I).node1);
        v2 = X(DIODE_LIST(I).node2);
        Vt = DIODE_LIST(I).Vt;
        Is = DIODE_LIST(I).Is;

J(DIODE_LIST(I).node1,DIODE_LIST(I).node1)=J(DIODE_LIST(I).node1,DIODE_LIST(I).node1)+↙
(Is/Vt)*exp((v1-v2)/Vt);
J(DIODE_LIST(I).node1,DIODE_LIST(I).node2)=J(DIODE_LIST(I).node1,DIODE_LIST(I).node2)-↙
(Is/Vt)*exp((v1-v2)/Vt);
J(DIODE_LIST(I).node2,DIODE_LIST(I).node1)=J(DIODE_LIST(I).node2,DIODE_LIST(I).node1)-↙
(Is/Vt)*exp((v1-v2)/Vt);
J(DIODE_LIST(I).node2,DIODE_LIST(I).node2)=J(DIODE_LIST(I).node2,DIODE_LIST(I).node2)+↙
(Is/Vt)*exp((v1-v2)/Vt);

    elseif (DIODE_LIST(I).node1 == 0)
    v2 = X(DIODE_LIST(I).node2);
    Vt = DIODE_LIST(I).Vt;
    Is = DIODE_LIST(I).Is;
    J(DIODE_LIST(I).node2,DIODE_LIST(I).node2)=J(DIODE_LIST(I).node2,DIODE_LIST(I).node2)↙
+(Is/Vt)*exp(-v2/Vt);

    else (DIODE_LIST(I).node2 == 0)
    v1 = X(DIODE_LIST(I).node1);
    Vt = DIODE_LIST(I).Vt;
    Is = DIODE_LIST(I).Is;
    J(DIODE_LIST(I).node1,DIODE_LIST(I).node1)=J(DIODE_LIST(I).node1,DIODE_LIST(I).node1)↙
+(Is/Vt)*exp(v1/Vt);
    end
end
J=J+G;
%syms v1 v2 Ie
%global X eqn
%X = [v1;v2;Ie];
%eqn = [(-1e-15)*exp((v1-v2)/(26e-3));(1e-15)*exp((v1-v2)/(26e-3));0];
%J = jacobian(eqn,X);
%J = jacobian([(-1e-15)*exp((v1-v2)/(26e-3)),(1e-15)*exp((v1-v2)/(26e-3)),0],[v1,v2,Ie])
```

```
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
global G C b DIODE_LIST
g=1; %Xguess = [2,1.3,1];
f=f_vector(Xguess);
S=G.*Xguess+f-b;
J=nlJacobian(Xguess);
delta_x= -J\S;
Xguess=Xguess+delta_x;
dX(g,1) = norm(delta_x);

while dX>maxerr
g=g+1;
f=f_vector(Xguess);
S=G*Xguess+f-b;
J=nlJacobian(Xguess);
delta_x= -J\S;
Xguess=Xguess+delta_x;
dX(g,1) = norm(delta_x);
end

% Code exits when norm of DeltaX is less than maxerr
Xdc=Xguess;
% the correction solution
%Xguess = [2,1.3,1];
%G = [1/50,-1/50,1;-1/50,1/50,0;1,0,0];
%F = [(-1e-15).*exp((2-1.3)/(26e-3)),(1e-15).*exp((2-1.3)/(26e-3)),0];
%b = [0,0,2]
%theta = G.*Xguess+F-b;
% 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.
end
```