ECSE 597: Circuit Simulations and Modeling

Assignment 4, October 10, 2019
Wenjie Wei, 260685967

1 Results of Circuit Simulation

Figure 1 shows the result curve of the testing circuit. The results of the three testing points: -10V, -2V, and 8V are indicated in the plot:

- $V_i = -10V, \quad V_o = -3.43V;$
- $V_i = -1.92V, \quad V_o = -1.92V;$
- $V_i = 3.28V, \quad V_o = -7.98V;$

Because of the nature of the MATLAB function `linspace()`, only the results at the nearest points are shown.

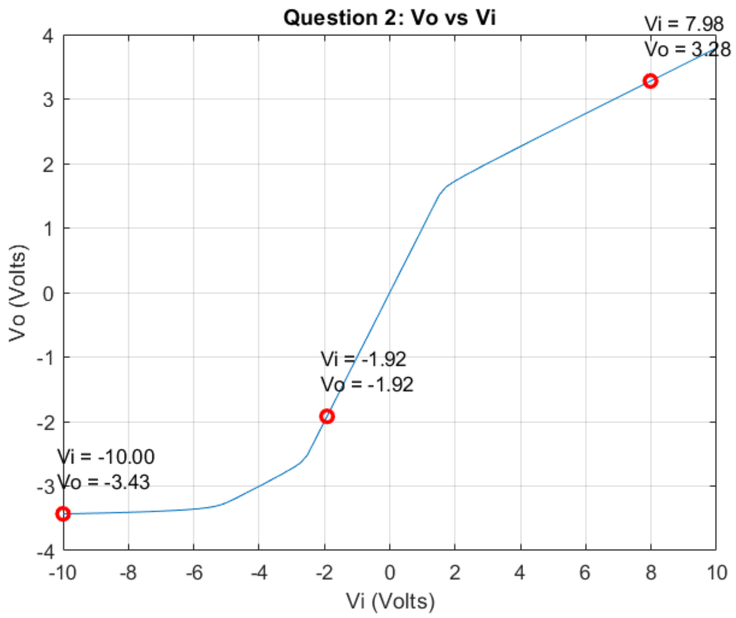

Figure 1: Results of the Test Circuit Indicating Three Test Points

A Code Listings

Listing 1: MATLAB Function to Compute the DC Solution (`dcsolvealpha.m`).

```
function Xdc = dcsolvealpha(Xguess,alpha,maxerr)
    % Compute dc solution using newtwon iteration for the augmented system
    % G*X + f(X) = alpha*b
    % Inputs:
    % Xguess is the initial guess for Newton Iteration
    % alpha is a paramter (see definition in augmented system above)
    % maxerr defined the stopping criterion from newton iteration: Stop the
    % iteration when norm(deltaX)<maxerr
    % Oupputs:
    % Xdc is a vector containing the solution of the augmented system
    global G b

    Xdc = Xguess;

    converged = false;
    while ~converged
        Phi = G * Xguess + f_vector(Xdc) - alpha .* b;
        J = nlJacobian(Xdc);

        %dX = [dX (-inv(J) * Phi)];
        %Xdc = Xdc + dX(:, iteration + 1);
        %Xguess = Xguess + dX(:, iteration + 1);
        dX = -inv(J) * Phi;
        Xdc = Xdc + dX;
        Xguess = Xguess + dX;

        if norm(dX, 2) < maxerr
            converged = true;
        end
    end
end

%% Function to compute the Jacobian during Newton-Ralphson Iterations.
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.

    % Diode curve: I = Is(exp(V/VT) - 1)
    global G DIODE_LIST

    % Create the Jacobian matrix F of f(x).
    F = zeros(size(G, 1), size(G, 2));

    for i = 1:size(DIODE_LIST, 2)
        diode = DIODE_LIST(i);

        v1 = X(diode.node1);
        v2 = X(diode.node2);
        n1 = diode.node1;
        n2 = diode.node2;

        dF = (diode.Is / diode.Vt) * exp((v1 - v2) / diode.Vt);

        if diode.node1 ~= 0
            F(n1, n1) = F(n1, n1) + dF;
        end

        if diode.node2 ~= 0
            F(n2, n2) = F(n2, n2) + dF;
        end

        if diode.node1 ~= 0 && diode.node2 ~= 0
```

```
            F(n1, n2) = F(n1, n2) - dF;
            F(n2, n1) = F(n2, n1) - dF;
        end
    end

    J = G + F;
end
```

Listing 2: MATLAB Function to Compute the DC Solution Using Power Ramping (`dcsolvecont.m`).

```
function Xdc = dcsolvecont(n_steps,maxerr)
    % Compute dc solution using newtwon iteration and continuation method
    % (power ramping approach)
    % inputs:
    % n_steps is the number of continuation steps between zero and one that are
    % to be taken. For the purposes of this assigments the steps should be
    % linearly spaced (the matlab function "linspace" may be useful).
    % maxerr is the stopping criterion for newton iteration (stop iteration
    % when norm(deltaX)<maxerr
    global G

    Xguess = zeros(size(G, 1), 1);
    alpha = linspace(0, 1, n_steps);

    for i = 1:size(alpha, 2)
        Xdc = dcsolvealpha(Xguess, alpha(i), maxerr);

        Xguess = Xdc;
    end
end
```