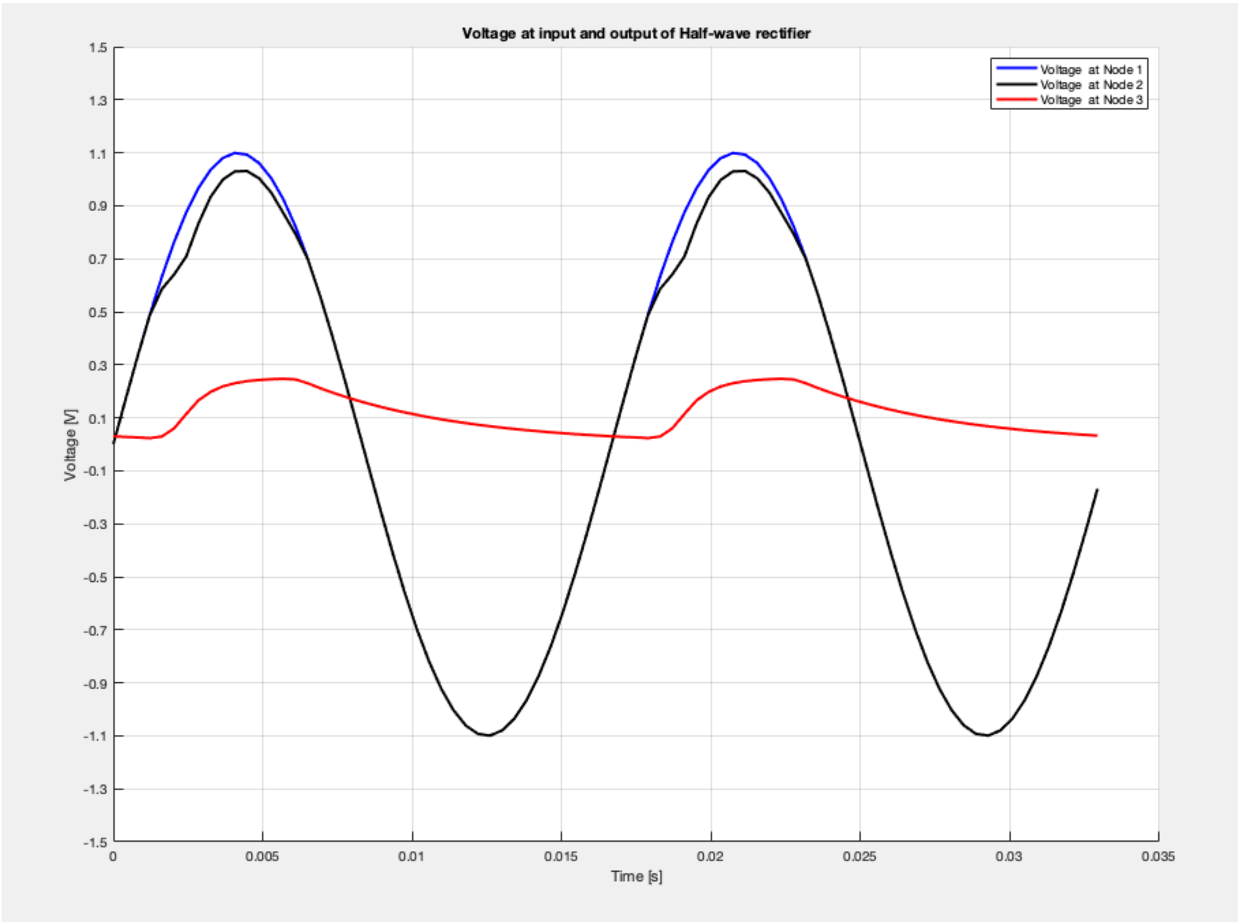
ECSE 597: Circuit Simulation and Modelling
Assignment 4

Student Name: Dafne Culha
Student ID: 260785524

Department of Electrical and Computer Engineering
McGill University, Canada
December, 2021

TestBenchNew output figure:



1. makeGamma.m

```
function Gamma =  makeGamma(H)
%This function takes number of harmonics H as the input and returns
%the Direct Fourier Transform (DFT) matrix, Gamma, as the output.
%gamma is the direct fourier transform matrix

%Input: H is the number of harmonics
% Output: Gamma is the DFT matrix
Nh = 2*H+1;                      % number of fourier coefficients


%% Write your code here.
% Slides I00 33/44

for n = 0: (Nh-1)
    %for k = 0 : H
    for j = 1 : (Nh)
        i = n + 1;

        % starts at 0
        k = fix (j / 2);
        theta = k * n * 2 * pi / Nh;

        if ((rem (j,2))==1)
            Gamma(i, j) = sin(theta);
        end


        if ((rem (j,2))==0)
            Gamma(i, j) = cos(theta);
        end
       if (j == 1)
           Gamma(i,j) = 1;
       end
    end
end
```

2. HB_nljacobian.m

```
function J = HB_nljacobian( Xbar_guess,H)

%  this function takes the vector containing Fourier Coefficients for nodal
%  voltages/currentsXs as input and returns  Jacobian J  also in "FREQUENCY" domain.

% Inputs: 1.  Xbar_guess is the Newton-Raphson guess vector. This vector is
%             in FREQUENCY DOMAIN as it contains the Fourier Coefficients for nodal
%             voltages/currents.
%        2.   H is the number of harmonics

% Output: J is the Jacobian in FREQUNECY DOMAIN.


global G DIODE_LIST

H1 = H
n = size(G,1);
Nh = 2*H+1; % number of fourier coefficients.
J = zeros(n*Nh); % Initialize the f vector (same size as b)
SIZEJ = length(J)
NbDiodes = size(DIODE_LIST,2);
Gamma = makeGamma(H)
d_diode = zeros (Nh, Nh);


%% Fill in the Frequnecy Domain Jacobian for Diodes
```

```
for I = 1:NbDiodes

    node_i = DIODE_LIST(I).node1;
    node_j = DIODE_LIST(I).node2;
    Vt = DIODE_LIST(I).Vt; % Vt of diode (part of diode model)
    Is = DIODE_LIST(I).Is; % Is of Diode (part of diode model)

    if (node_i ~= 0) && (node_j ~= 0)
        v1_f = Xbar_guess((node_i-1) * Nh + 1 : node_i * Nh); %nodal voltage at anode
        v2_f = Xbar_guess((node_j-1) * Nh + 1 : node_j * Nh); %nodal voltage at cathode
        v1_t = Gamma * v1_f;
        v2_t = Gamma * v2_f;
        g = (Is / Vt) * exp ((v1_t - v2_t) / Vt);

        g = diag(g);

        g= Gamma \ g * Gamma;
        disp(g)

        %J = J+[g -g; -g g]
        %d_diode = d_diode + [g -g; g -g]
        J (((node_i-1) * Nh + 1 : node_i * Nh),((node_i-1) * Nh + 1 : node_i * Nh)) =  + g;
        J (((node_i-1) * Nh + 1 : node_i * Nh),((node_j-1) * Nh + 1 : node_j * Nh))  = - g;
        J (((node_j-1) * Nh + 1 : node_j * Nh),((node_i-1) * Nh + 1 : node_i * Nh)) =- g;
        J (((node_j-1) * Nh + 1 : node_j * Nh),((node_j-1) * Nh + 1 : node_j * Nh))=  + g;



        %f(node_i) = f(node_i) + diode_current;
        %f(node_j) = f(node_j) - diode_current;
    end

    %elseif (DIODE_LIST(I).node1== 0) && (DIODE_LIST(I).node2 ~= 0)
        % You can omit this part, as this is not needed for this assignment


    %elseif (DIODE_LIST(I).node1~= 0) && (DIODE_LIST(I).node2 == 0)
         % You can omit this part, as this is not needed for this assignment

end
%Gbar = makeHB_Gmat(H)  % make Gbar
%Cbar = makeHB_Cmat(H)  % make Cbar

%J = Gbar + Cbar + J;
```

3. *HB_f_vector.m*

```
function Fb = HB_f_vector(Xbar_guess,H)

%  this function takes the FREQUENCY domain vector Xb as input and returns
%  Fb  as output in FREQUNECY DOMAIN.

% Inputs: 1.Xbar_guess is the Newton-Raphson guess vector. This vector is
%             in FREQUENCY DOMAIN as it contains the Fourier Coefficients for nodal
%             voltages/currents.
%        2. H is the number of harmonics

% Output: Fb is the nonlinear vector in "frequnecy" domain ( it will
% contain the fourier coefficients for nonlinearity.)


global G DIODE_LIST

n = size(G,1);
Nh = 2*H+1; % number of fourier coefficients.
Fb = zeros(n*Nh,1); % Initialize the Fb vector
```

```
f = zeros(n*Nh,1);
Gamma = makeGamma(H);
NbDiodes = size(DIODE_LIST,2);
%Xbar_guess = Gamma .* Xbar_guess;

%% Fill in the fs for Diodes
for I = 1:NbDiodes
    node_i = DIODE_LIST(I).node1;
    node_j = DIODE_LIST(I).node2;
    Vt = DIODE_LIST(I).Vt; % Vt of diode (part of diode model)
    Is = DIODE_LIST(I).Is; % Is of Diode (part of diode model)

    if (node_i ~= 0) && (node_j ~= 0)
        v1_f = Xbar_guess((node_i-1) * Nh + 1 : node_i * Nh); %nodal voltage at anode
        v2_f = Xbar_guess((node_j-1) * Nh + 1 : node_j * Nh); %nodal voltage at cathode
        v1_t = Gamma * v1_f;
        v2_t = Gamma * v2_f;
        diode_current_f = Is*(exp((v1_t-v2_t)/Vt)-1);
        diode_current_t = Gamma \ diode_current_f;

        Fb ((node_i-1) * Nh + 1 : node_i * Nh) = Fb ((node_i-1) * Nh + 1 : node_i * Nh)+diode_current_t;
        Fb ((node_j-1) * Nh + 1 : node_j * Nh) = Fb ((node_j-1) * Nh + 1 : node_j * Nh)-diode_current_t;
        %f(node_i) = f(node_i) + diode_current;
        %f(node_j) = f(node_j) - diode_current;
    end
end
%
%Fb = Gamma\f
%Fb = Gamma \ f;
%Fb((Z-1)*Nh +1 : Z*Nh)


% Hint: To fill up the Fb with the Fourier Coeffeicnts for node Z
% you can access the the suitable indices using,
% Fb((Z-1)*Nh +1 : Z*Nh)



    %elseif (DIODE_LIST(I).node1== 0) && (DIODE_LIST(I).node2 ~= 0)
         % You can omit this part, as this is not needed for this assignment
    %elseif (DIODE_LIST(I).node1~= 0) && (DIODE_LIST(I).node2 == 0)
        % You can omit this part, as this is not needed for this assignment
    %end
end
```

4. *HBSolve.m*

```
function [Xout] = HBsolve(Xguess,H)
% This function takes a initial guess vector, Xguess as input and Number of
% Harmonics, H, as the input and then solves the Harmonic Balance system of
% equations using Newton-Raphson's method.

%Inputs: 1. Xguess: Is the Intial Guess for the harmonic balance.
%        2.  H :    Is the numberof Harmonics.

%Output: Xout: Is the solution vector (it is a vector of Fourier
%coefficients of nodal voltages/currents)

global G  C Bbar HBsources_LIST  DIODE_LIST

Nh = 2*H+1; % number of fourier coefficients
n = size(G,1); % size of the MNA matrices.

maxerr = 1;
% The  variables that you may need are here.
```

```
frequency  = HBsources_LIST(1).freq;  % frequency.
Gbar = makeHB_Gmat(H);  % make Gbar
Cbar = makeHB_Cmat(H);  % make Cbar
% The variable Babr is made for you below.
Bbar = zeros(Nh*n,1);
%% make Bbar
for I = 1:size(HBsources_LIST,1)
    Bbar(Nh*(HBsources_LIST(I).CurrentIdx-1)+3) =  HBsources_LIST(I).val;
end

%% write your code here.

% Slides I00 20/41
%Gbar*Xbar + Cbar*Xbar + F(Xbar) = Bbar



sz = length(G);
dX = [];

error_tolerable = false;

iteration = 0;


% slides B00
while (~error_tolerable)
    iteration = iteration +1;

    F = HB_f_vector(Xguess, H);


    phi = Gbar * Xguess + 2 * pi * frequency * Cbar * Xguess + F - Bbar;

    J = HB_nljacobian(Xguess, H);

    %disp(f);

    delta_x = (Gbar + 2 * pi * frequency * Cbar + J) \  phi;

    Xguess = Xguess - delta_x;

    if ((norm(delta_x) <= maxerr) && (norm(phi) <=maxerr))
        error_tolerable = true;
    end

    dX(iteration) = norm(delta_x);

end

Xout = Xguess;
```