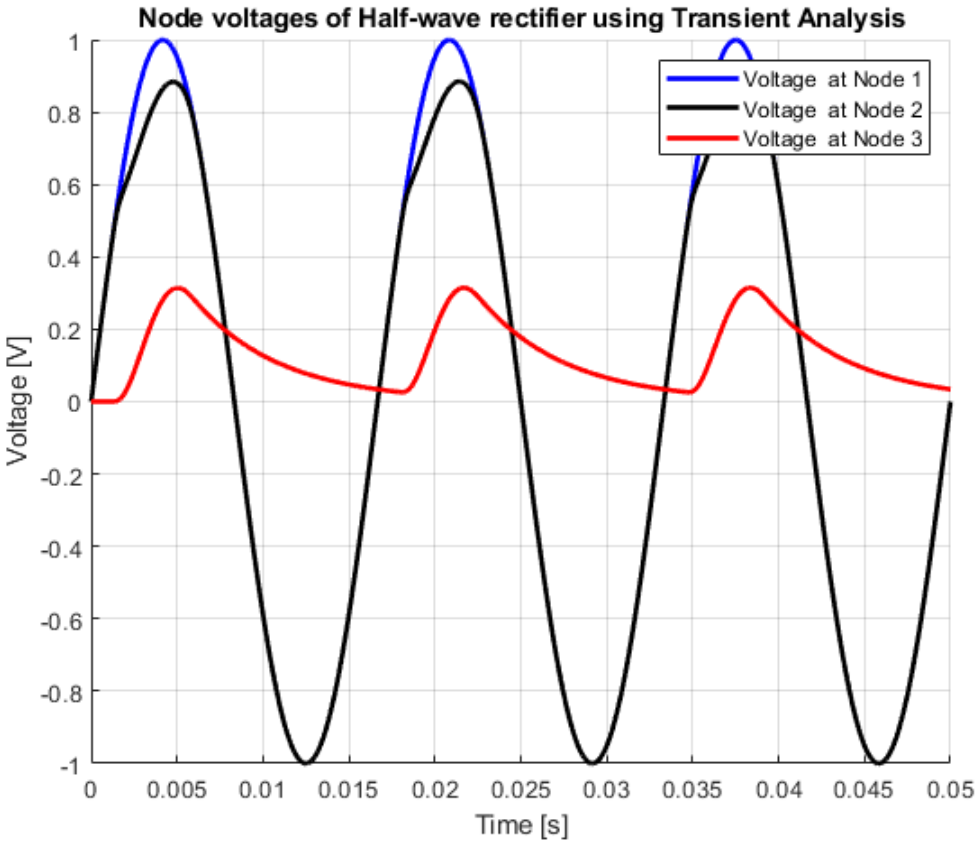
ECSE 597 Assignment 4

Yicheng Song 260763294

Q1

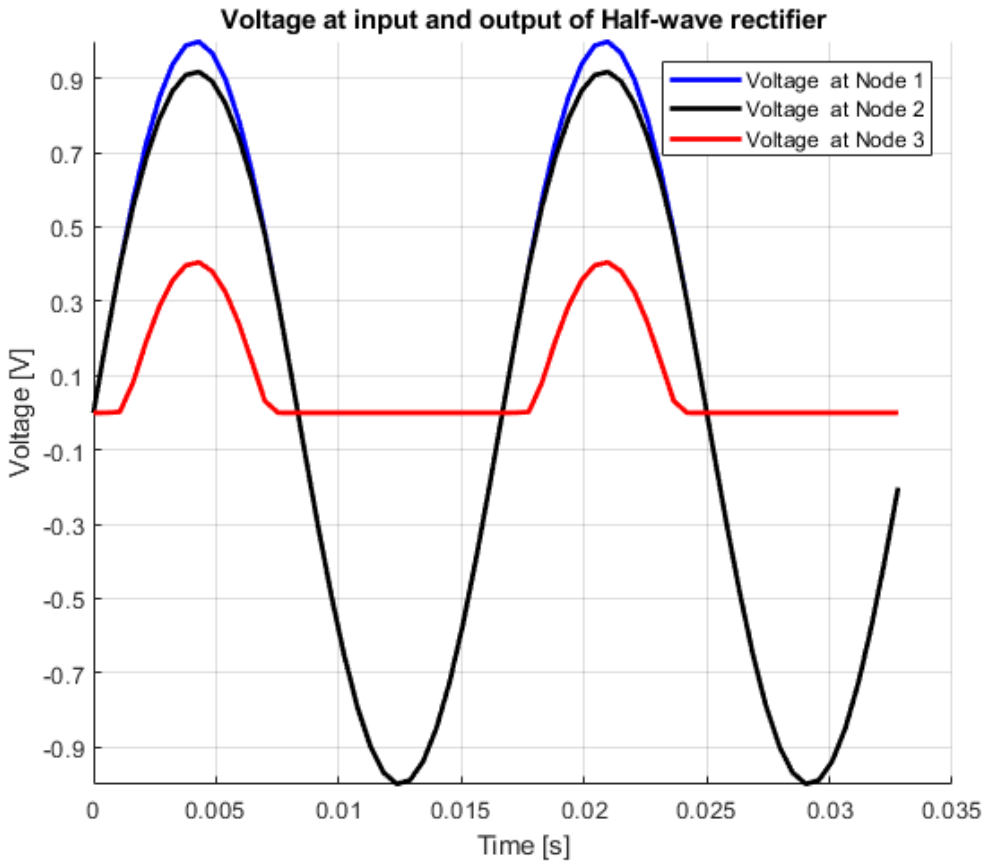


Code:

```
function [tpoints, X]= nl_BE_method(tEnd,h)

global elementList

maxerr = 1e-4;

tpoints = 0:h:tEnd;

Gmat = makeGmatrix;
Cmat = makeCmatrix;

A = Gmat + Cmat / h;

x_n = zeros(size(Gmat,2),1);
X = zeros(size(Gmat,2),length(tpoints));

for I=1:length(tpoints)
    b_n1 = makeBt(tpoints(I));
    Xguess = x_n;
    flag = 1;

    while flag
        F = makeFvect(Xguess);
        J = make_nlJacobian(Xguess);

        phi_x = A * Xguess + F - (b_n1 + Cmat * x_n / h);
        Dphi_x = A + J;
        delta_x = -Dphi_x\phi_x;
        Xguess = Xguess + delta_x;
        x_n1 = Xguess;
        if norm(delta_x) < maxerr
            flag = 0;
        end
    end
    X(:,I) = x_n1;
    x_n = x_n1;
end
end
```

Q2

Set H = 15, the plot is shown below:



Code:

makeGamma:

```
function gamma =  makeGamma(H)

Nh = 2*H+1; % number of fourier coefficients
gamma = zeros(Nh,Nh);

for I = 1:Nh
    gamma(I,1) = 1;
end

for I=1:H
    for J=1:Nh
        gamma(J,(2*I)) = cos(2*pi/Nh*I*(J-1));
        gamma(J,(2*I+1)) = sin(2*pi/Nh*I*(J-1));
    end

end
```

HB_fvect:

```
function Fbar = HB_fvect(Xs,H)
%Xs is in time-domain
global elementList

n = elementList.n;
Nh = 2*H+1;
nHB = Nh*n;

Is=1e-13;
Vt=0.025;

gamma = makeGamma(H);
F = zeros(nHB,1);

for K=1:elementList.Diodes.numElements
    nodes = elementList.Diodes.nodeNumbers(K,:);

    n1 = nodes(K,1);
    n2 = nodes(K,2);

    if(nodes(1)~=0 && nodes(2)~=0)
        for I=1:Nh
            Vd = Xs((n1-1)*Nh+I,1)-Xs((n2-1)*Nh+I,1);
            F((n1-1)*Nh+I,1) = F((n1-1)*Nh+I,1) + Is*(exp(Vd/Vt)-1);
            F((n2-1)*Nh+I,1) = F((n2-1)*Nh+I,1) - Is*(exp(Vd/Vt)-1);
        end
    elseif (nodes(1)==0 && nodes(2)~=0)
        for I=1:Nh
            Vd = -Xs((n2-1)*Nh+I,1);
            F((n2-1)*Nh+I,1) = F((n2-1)*Nh+I,1) - Is*(exp(Vd/Vt)-1);
        end
    elseif(nodes(1)~=0 && nodes(2)==0)
        for I=1:Nh
            Vd = Xs((n1-1)*Nh+I,1);
            F((n1-1)*Nh+I,1) = F((n1-1)*Nh+I,1) + Is*(exp(Vd/Vt)-1);
        end
    end
end

large_gamma = zeros(nHB,nHB);
for I=1:n
    large_gamma((I-1)*Nh+1:I*Nh,(I-1)*Nh+1:I*Nh) = gamma;
end

Fbar = large_gamma\F;
```

HB_nl_jacobian:

```
function J = HB_nl_jacobian(Xs,H)

%Xs is in time-domain

global elementList

n = elementList.n;
Nh = 2*H+1; % number of fourier coefficients.
nHB = Nh*n;

Is=1e-13;
Vt=0.025;

Gbar = makeHB_Gmat(H);
Cbar = makeHB_Cmat(H);
gamma = makeGamma(H);

large_gamma = zeros(nHB,nHB);
for I=1:n
    large_gamma((I-1)*Nh+1:I*Nh,(I-1)*Nh+1:I*Nh) = gamma;
end
```

```
%% Fill in the J for Diodes
J_t = zeros(nHB,nHB);

for I=1:elementList.Diodes.numElements
    nodes = elementList.Diodes.nodeNumbers(I,:);

    n1 = nodes(I,1);
    n2 = nodes(I,2);

    if(nodes(1)~=0 && nodes(2)~=0)
        for I=1:Nh
            Vd = Xs((n1-1)*Nh+I,1)-Xs((n2-1)*Nh+I,1);
            J_t((n1-1)*Nh+I,(n1-1)*Nh+I) = J_t((n1-1)*Nh+I,(n1-1)*Nh+I) + Is/Vt*exp(Vd/Vt);
            J_t((n2-1)*Nh+I,(n2-1)*Nh+I) = J_t((n2-1)*Nh+I,(n2-1)*Nh+I) + Is/Vt*exp(Vd/Vt);
        end
    elseif (nodes(1)==0 && nodes(2)~=0)
        for I=1:Nh
            Vd = -Xs((n2-1)*Nh+I,1);
            J_t((n2-1)*Nh+I,(n2-1)*Nh+I) = J_t((n2-1)*Nh+I,(n2-1)*Nh+I) + Is/Vt*exp(Vd/Vt);
        end
    elseif(nodes(1)~=0 && nodes(2)==0)
        for I=1:Nh
            Vd = Xs((n1-1)*Nh+I,1);
            J_t((n1-1)*Nh+I,(n1-1)*Nh+I) = J_t((n1-1)*Nh+I,(n1-1)*Nh+I) + Is/Vt*exp(Vd/Vt);
        end
    end
end

J = Gbar + Cbar + large_gamma\J_t*large_gamma;
```

hbsolve:

```
function [Xout] = hbsolve(Xguess,H)

global elementList

maxerr = 1e-8;

Nh = 2*H+1;
Gbar = makeHB_Gmat(H); % harmonic balance matrix Gbar
Cbar = makeHB_Cmat(H); % harmonic balance matrix Cbar
n = elementList.n; % size of regular MNA
nHB = Nh*n; % size of Harmonc Balance MNA
gamma = makeGamma(H);

large_gamma = zeros(nHB,nHB);
for I=1:n
    large_gamma((I-1)*Nh+1:I*Nh,(I-1)*Nh+1:I*Nh) = gamma;
end

[~,~,Bbar]= makeBvector(H);

while 1
    Fbar = HB_fvect((large_gamma*Xguess),H);
    J = HB_nl_jacobian((large_gamma*Xguess),H);
    phi_x = Gbar * Xguess + Cbar * Xguess + Fbar - Bbar;
    delta_x = -J\phi_x;
    Xguess = Xguess + delta_x;
    Xout = Xguess;
    if norm(delta_x) < maxerr
        break
    end

end
```