ENSC 895: SPECIAL TOPICS: THEORY, ANALYSIS, AND SIMULATION OF NONLINEAR CIRCUITS

Finding DC Operating Points of Nonlinear Circuits using Homotopy Method

Fall 2007

FINAL PROJECT

Mohamadreza Najiminaini (mna28@cs.sfu.ca)

Anna Seung (anseung@sfu.ca)

Website: www.sfu.ca/~mna28

Abstract

Finding operating points of nonlinear circuits is one of the most significant steps for circuit designers when simulating electronic circuits. Homotopy methods have been developed over the last few decades to help solve nonlinear circuits and equations. Three feedback structured circuits (i.e. flip flop, Schmitt trigger, and Chua's circuit) are exercised in this project using homotopy method to solve nonlinear circuits. Matlab, in which homotopy method is implemented, and PSPICE are also utilized to simulate the result of multiple operating point of each nonlinear circuit. Throughout this paper, the back ground of homotopy method, its implantation and several circuit experiments will be demonstrated and discussed.

Table Of Content

ABSTRACT .... *I*

1 INTRODUCTION *1*

2 THEORY *2*

2.1 CONVERGENCE OF NEWTON RAPHSON(NR) METHOD *2*

2.2 HOMOTOPY METHOD *4*

3 IMPLEMENTATION *6*

3.1 SOLVING NONLINEAR CIRCUITS USING PCHOMOTOPY TOOL IN MATLAB *6*

3.2 SCHMITT TRIGGER *7*

3.3 FLIP-FLOP *9*

3.4 CHUA'S CIRCUIT *11*

4 CONCLUSION *14*

5 REFERENCES *15*

APPENDIX A: SNAPSHOT FOR SINGULARITY OF JACOBEAN MATRIX *16*

APPENDIX B: MATLAB CODE *17*

1 Introduction

Newton Raphson(NR) method is the common method implemented in SPICE-like circuit simulators. Due to the fact that Newton Raphson method does not always converge for solution, difficulties in computing dc operating point of nonlinear circuit system using SPICE-like circuit simulators often arise. Even though various techniques are introduced to help convergence of NR method, promising results are hardly obtained when the nonlinear circuit is associated with bifurcation.

There have been many trials to resolve computational difficulties using homotopy method, another type of the continuation and parameter embedding method, faced in other methods in solving non-linear equations.

Our goal, throughout the project, is to find dc operating points of three nonlinear circuits of the following:

1. Schmitt trigger circuit
2. Flip-flop circuit
3. Chua's circuit (nine operating point)

In this paper, the convergence of Newton Raphson method, background and different methods of homotopy are thoroughly explained in section 3, and overall implementation and simulation using Matlab and PSPICE will be demonstrated in section 4. Finally we conclude this project in section 5.

2 Theory

2.1 Convergence of Newton Raphson(NR) Method

Finding DC operating points is a fundamental task in circuit simulation. SPICE circuit simulators has become most common tool to compute dc operate points. However, the Newton Raphson(NR) method, implemented in SPICE simulator, often results in inaccuracy or failure of dc solutions due to the convergence problem. Newton Raphson guarantees quadratic convergence only when a starting point is chosen to be close to the solution. In other words, in order to achieve convergence in simulator, one has to predict a good starting point so that the computation becomes very easy. However, it is not always easy to provide a good starting point. However, several ad hoc techniques had been developed and applied to help convergence of NR methods: some of the common used techniques are called source-stepping, temperature sweeping, and G_{min} stepping.

Ramping function, an elementary unary real function, is employed for the sources and dc operating point is continuously computed by the simulator until the point where the response from the original set of driving voltage is reached [1,2,3]. In temperature sweeping procedure, the initial value for temperature is set to zero and dc operating point of circuit is calculated. Then, this dc operating point is a good starting point for the next increased temperature circuit equations. This process is repeated recursively until the dc operating points are calculated for the desired temperature. Another technique, known as G_{min} stepping, is performed with the initial condition that every circuit node and ground are connected by small conductance. By choosing large conductance as a starting point, convergence of NR method is well defined. Then, the operating points are computed and they are used for initial node voltage of the next step with decreased conductance. This process continues until the all conductance become near to zero, thus, dc operating points are found approximately.

The above explained techniques are based on the notion of embedding or continuation methods in which the same procedure is repeated with a range-varying parameter until the desired solution is reached [3]. However, these ad-hoc methods also encounter some problems when the continuation path is associated with bifurcation or turning points as shown in Figure 1.

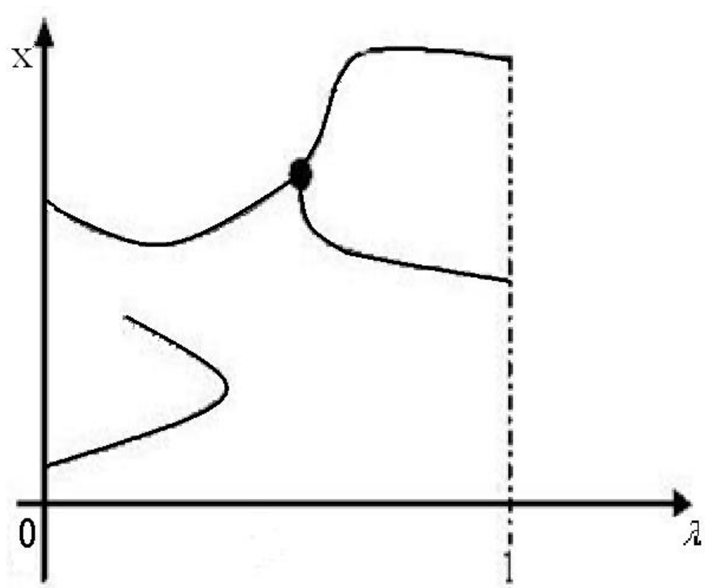


Figure 1.Bifurecation and turning point

2.2 Homotopy method

Several homotopy methods have been proposed and they are based on embedding parameter and continuation method [3, 4, and 5]. An example of simple homotopy is:

$$H(x,\lambda) = (1-\lambda)(x-a) + \lambda F(x) \tag{1}$$

Where $\lambda \in R^1$ is the continuation parameter, $a \in R^n$ is the starting vector for homotopy path, and $F(x) = 0$ is the nonlinear equation to be solved [3].

One of the first homotopy methods is fixed point homotopy as follows:

$$H(x,\lambda) = (1-\lambda)G(x-a) + \lambda F(x) \tag{2}$$

where a is random vector and diagonal matrix $G \in R^n \times R^n$ is embedded. If nonlinear equations satisfy coercivity conditions, homotopy methods can be globally convergent and bifurcation free, in the other hand, they will converge to a solution from an arbitrary starting point. Thus, by checking the passivity and no-gain properties of the circuit elements, circuit equations show that they satisfy such a condition. At $\lambda = 0$, homotopy path starts and the result is $x = a$, thus, when λ increases, the calculated result is a starting point for the next iteration. This process continues until $\lambda = 1$ which is the result for the nonlinear equation, $F(x) = 0$.

The second homotopy method is variable-stimulus homotopy as follows:

$$H(x,\lambda) = (1-\lambda)G(x-a) + F(x,\lambda) \tag{3}$$

where the node voltages of nonlinear elements are multiplied by λ. The procedure is similar to fixed point homotopy method but all voltages across the nonlinear elements are zero for the starting point of the homotopy method.

The fast converging homotopy for bipolar circuit is the variable-gain homotopy:

$$H(x,\lambda) = (1-\lambda)G(x-a) + F(x,\lambda\alpha) \qquad (4)$$

where α is a vector of forward and reverse current gains of transistors. These current gains are multiplied by λ. It starts by setting $\lambda = 0$ so that it forces all current gains equal to zero at the beginning of path, thus, the only things left are diodes and resistors and such a circuit has a unique operating point and can be solved easily[3,6 and 7].

The hybrid method is a combination of variable stimulus and variable-gain homotopy methods. For instance, using variable stimulus homotopy is to find starting point for variable gain homotopy which makes it efficient to acquire dc operating points of circuits.

"There is an important issue for finding dc operating points of circuit that characterize nonlinear circuit elements. Most homotopy algorithms require that these functions be at least C^2 continuous" in reference [7]. Besides, the passivity and no-gain property of the models of semiconductors should be preserved since it is necessary conditions required by the path following algorithms.

3 Implementation

3.1 Solving nonlinear circuits using pchomotopy tool in Matlab

In this section, we show the computing of dc operating points of three nonlinear circuits (i.e. Schmitt trigger and flip-flop, and Chua's circuit) using pchomotopy tool in Matlab. The implementation of pchomotopy in Matlab has been done by Hoffman. The pchomotopy tool uses the predictor-corrector method to solve the differential equations which describes the homotopy path. However, there are other methods such as NR and ODE which are not implemented in pchomotopy tool. We used nodal analysis to write equations for both circuits and the Jacobean matrices are calculated easily from their equations. The inputs for pchomotopy tool are circuits' equations and the Jacobean matrix for each case. We used the Ebers-Moll model for bipolar junction transistor.

$$\begin{pmatrix} i_e \\ i_c \end{pmatrix} = \begin{pmatrix} 1 & -\alpha_r \\ -\alpha_f & 1 \end{pmatrix} \begin{pmatrix} f_e(v_e) \\ f_c(v_c) \end{pmatrix} \tag{5}$$

where

$f_k(x) = m_k(e^{-n_k x} - 1)$ for k=1,…,4.

3.2 Schmitt trigger

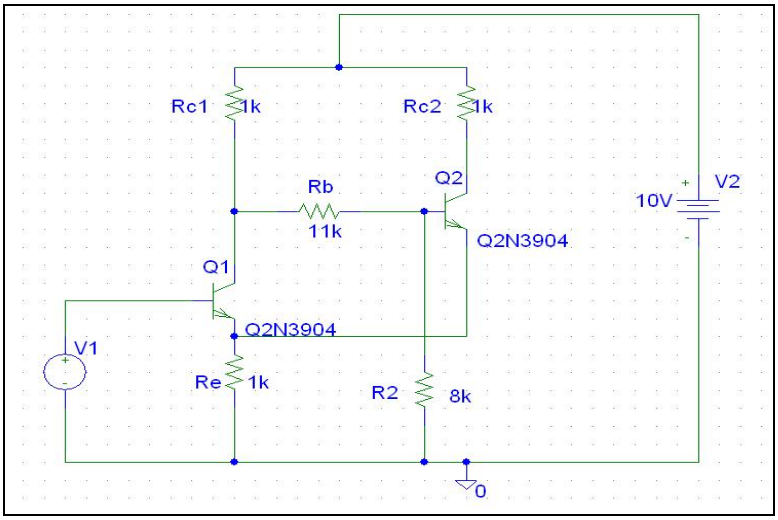


Figure 2: Schmitt trigger

Figure 2 above is the Schmitt trigger circuit used for PSPICE simulation. Depending on the value of Re, different triggering behavior is observed. By choosing the Re value equals to 1k Ohm, a good triggering behavior is obtained as shown in Figure 3.

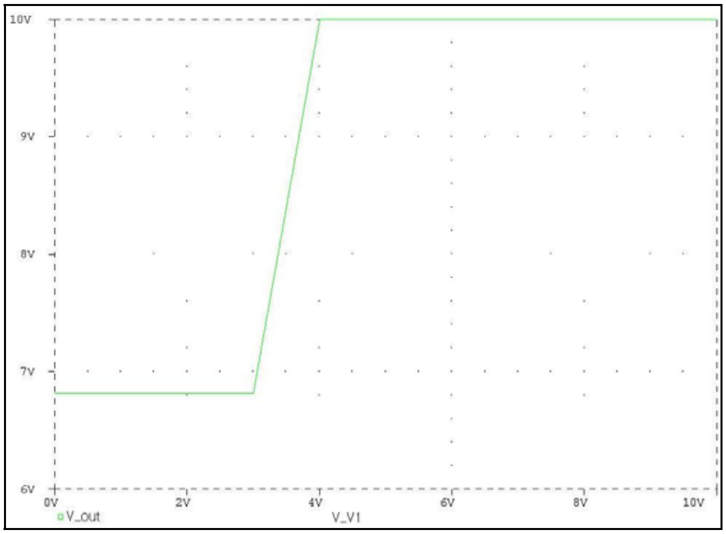


Figure 3: Triggering behavior of Schmitt trigger circuit (Re = 1k Ohm)

PSPICE simulation result of the Schmitt trigger circuit (Figure 2) is shown in Figure 4. Since it exhibits N-type NDR behavior, having more than one dc operating is secured.

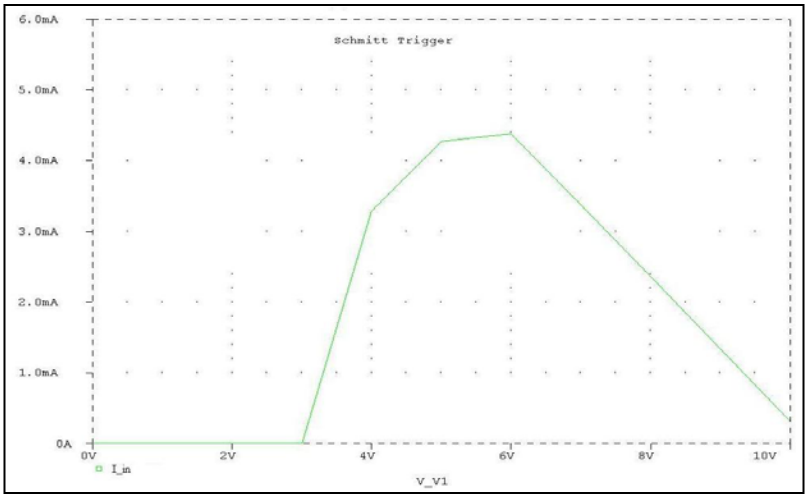


Figure 4: PSIPCE simulation of Schmitt trigger

Therefore, as a result of Matlab simulation, it shows that Schmitt trigger has three operating points for all nodes except node 6 whenever the path for each node passes $\lambda = 1$ as shown in figure 5.

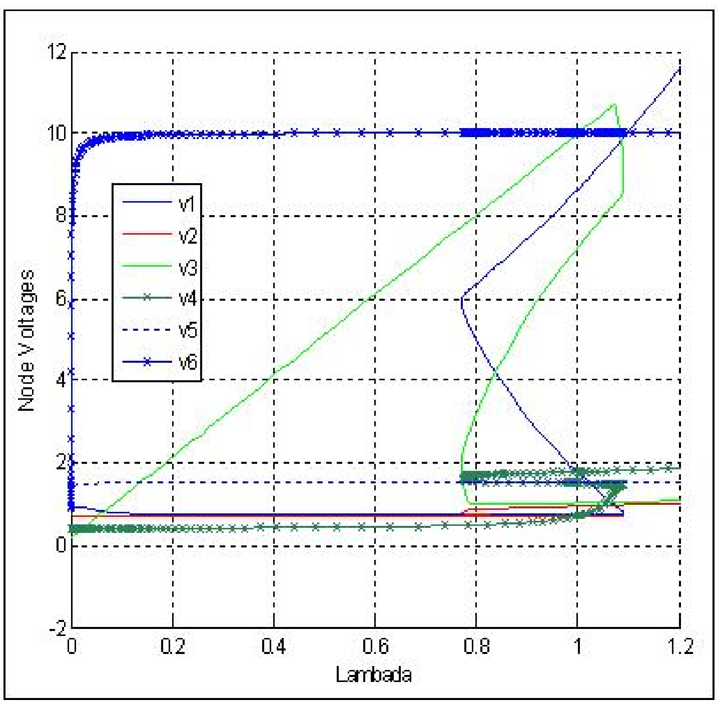


Figure 5: Matlab simulation of Schmitt trigger

3.3 Flip-flop

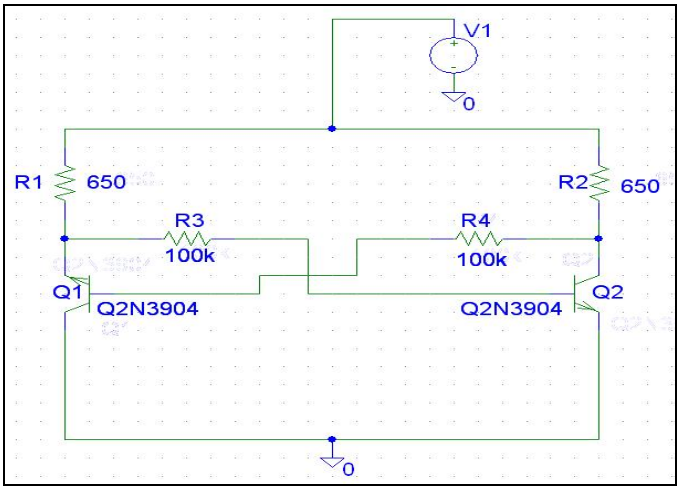


Figure 6: Flip-flop

Figure 6 is the flip-flop circuit used to simulate in PSPICE. During the simulation, different shape of N-type NDR behavior is observed, ensuring multiple dc operating point, with different values of collector resistors, R1 and R2. Choosing R1 and R2 to be 650 ohm, the following N-type NDR behaviour is obtained:

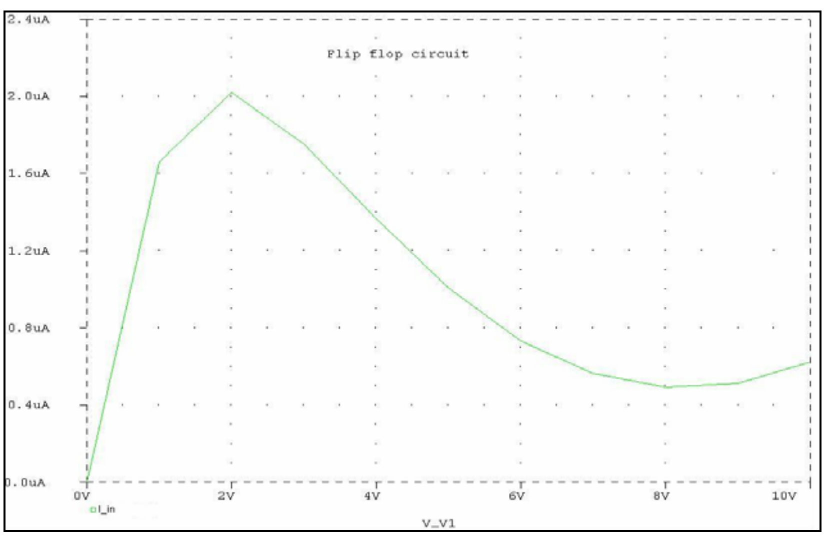


Figure 7: PSPICE simulation of flip-flop

However, in Matlab simulation, since the predictor-corrector method to solve differential equations makes Jacobean matrix singular when λ is close to one so that the solution for flip-flop is not achievable and the path rotates back when it gets close to $\lambda = 1$ as shown in figure 8, and a snapshot is taken from Matlab in appendix A to show singularity of Jacobin matrix while program is running. Therefore, another option is to use the other methods to solve differential equations such as Newton Raphson method or ODE (ordinary differential equations).

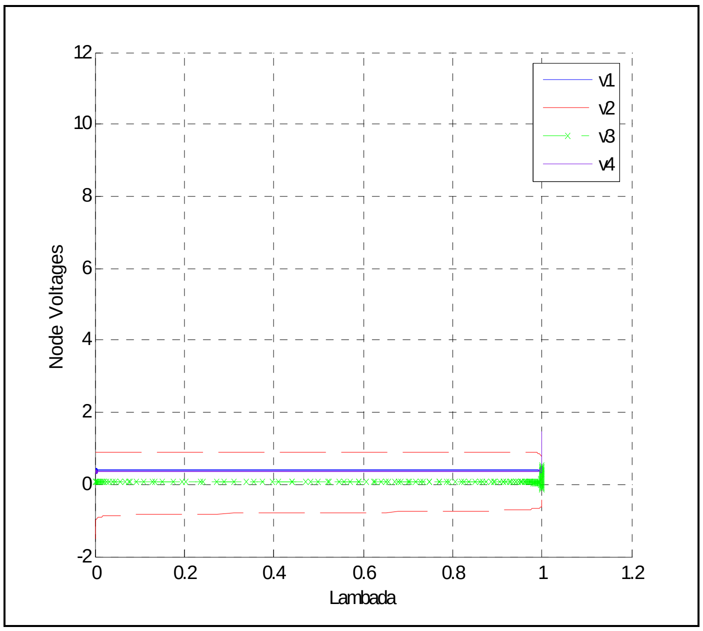


Figure 8: Matlab simulation of flip-flop

3.4 Chua's circuit

To practice more complicated nonlinear circuit with multiple operating points, Chua's circuit (Figure 9) with nine operating points is simulated by taking a reference to the paper [1]. For SPICE simulation, N-type NDR behaviour is also obtained as shown in Figure 10. In Matlab simulation, the same homotopy method (pchomotopy) used in Schmitt trigger is applied and the result is shown in Figure 11 and 12 for voltages and currents respectively.

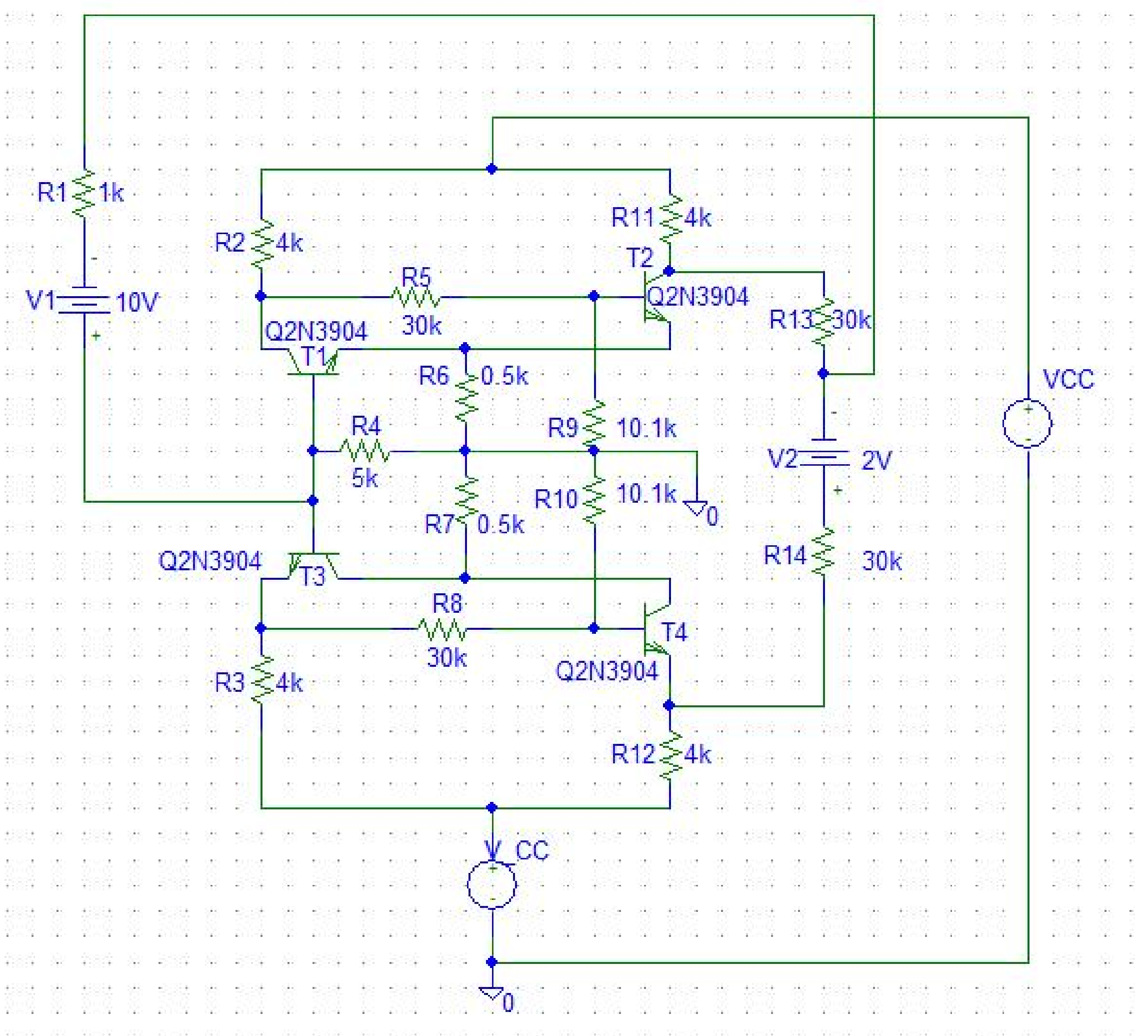


Figure 9: Chua's circuit (R1=1K, R2=R3=4K, R4=5K, R5=R8=30K, R6=R7=0.5K, R9=R10=10.1K,R11=R12=4K, R13=R14=30K, V1=10V, V2=2V)

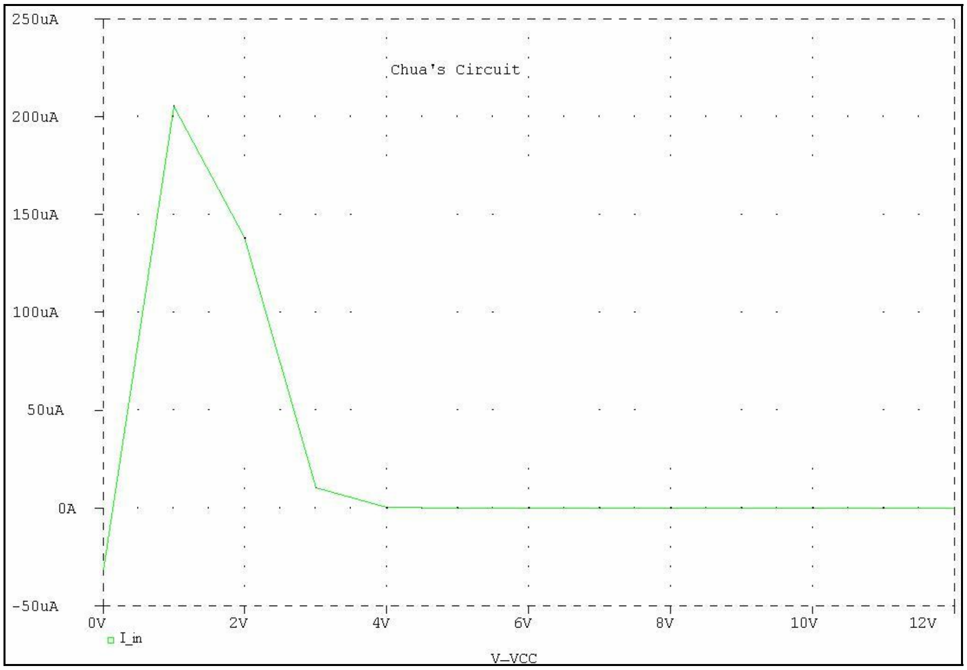


Figure 10: PSIPCE simulation of Chua's circuit

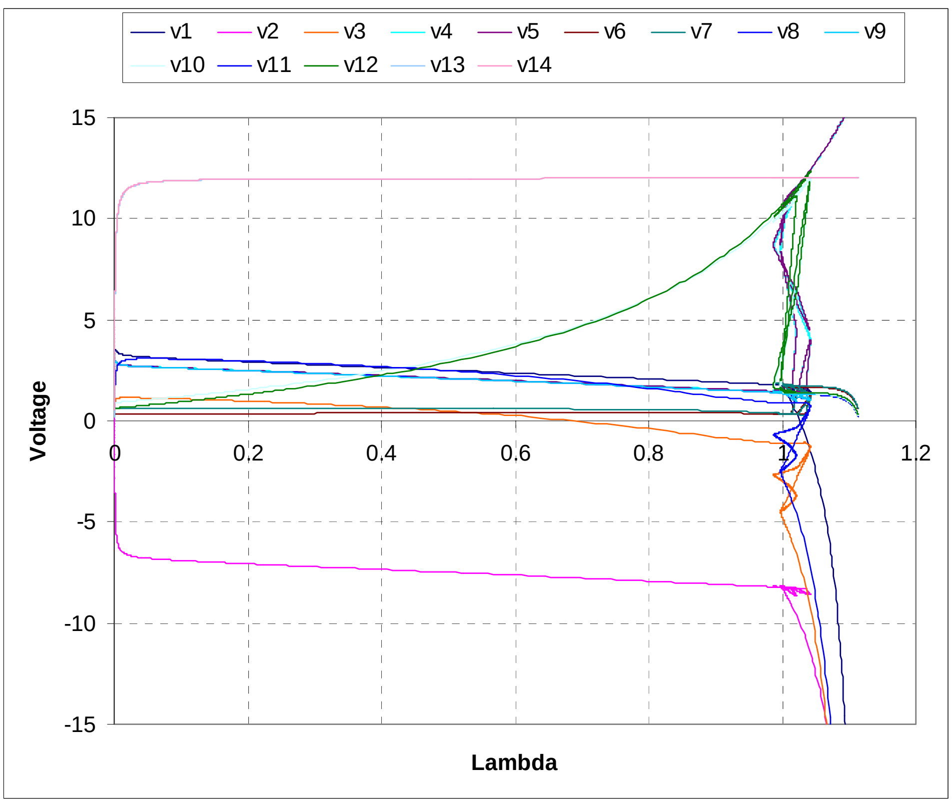


Figure 11: Matlab simulation of Chua's circuit for node voltages

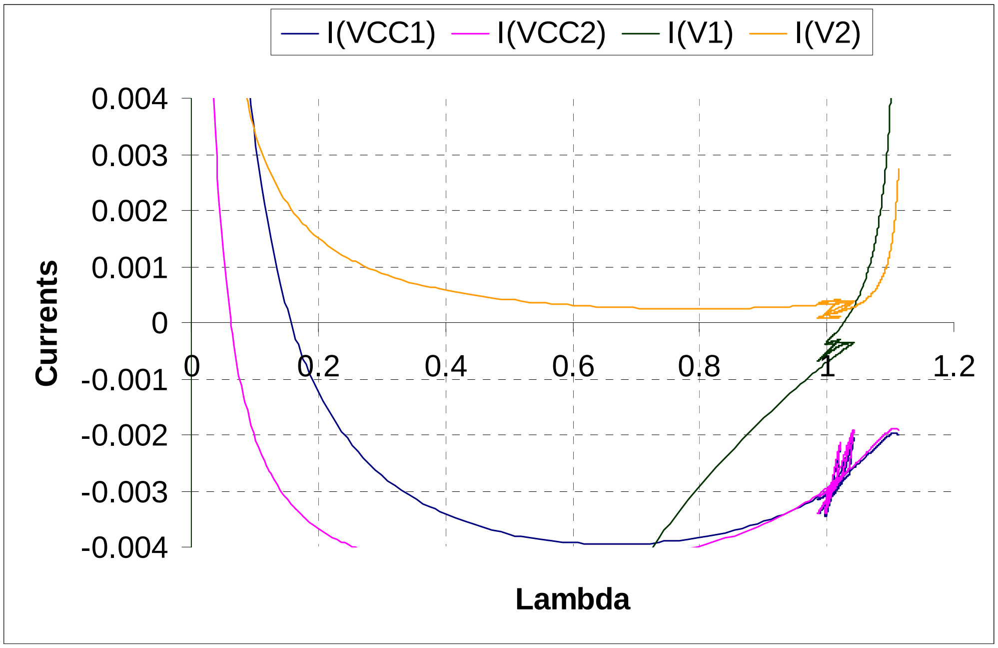


Figure 12: Matlab simulation of Chua's circuit for currents

4 Conclusion

We discussed about different embedding, continuation and homotopy methods. The coercivity condition was necessary to attain solutions for nonlinear circuits. As a result, using pchomotopy tool in Matlab, we found dc operating points of Schmitt trigger circuit and the result of flip flop circuit showed that the different methods should be used to solve its differential equations in order to acquire result for flip flop circuit.

5 References

[1] A. Dyess, E. Chan, H. Hofmann, W. Horia, and Lj. Trajkovic, ``Simple implementations of homotopy algorithms for finding dc solutions of nonlinear circuits," *Proc. IEEE Int. Symp. Circuits and Systems*, Orlando, FL, May 1999, vol. VI, pp. 290-293.

[2] W. Mathis, Lj. Trajkovic, M. Koch, and U. Feldmann, ``Parameter embedding methods for finding dc operating points of transistor circuits," *Proc. NDES '95*, Dublin, Ireland, July 1995, pp. 147-150.

[3] Lj. Trajkovic, ``Homotopy methods for computing dc operating points," invited contribution to the *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. New York: John Wiley & Sons, 1999, vol. 9, pp. 171-176.

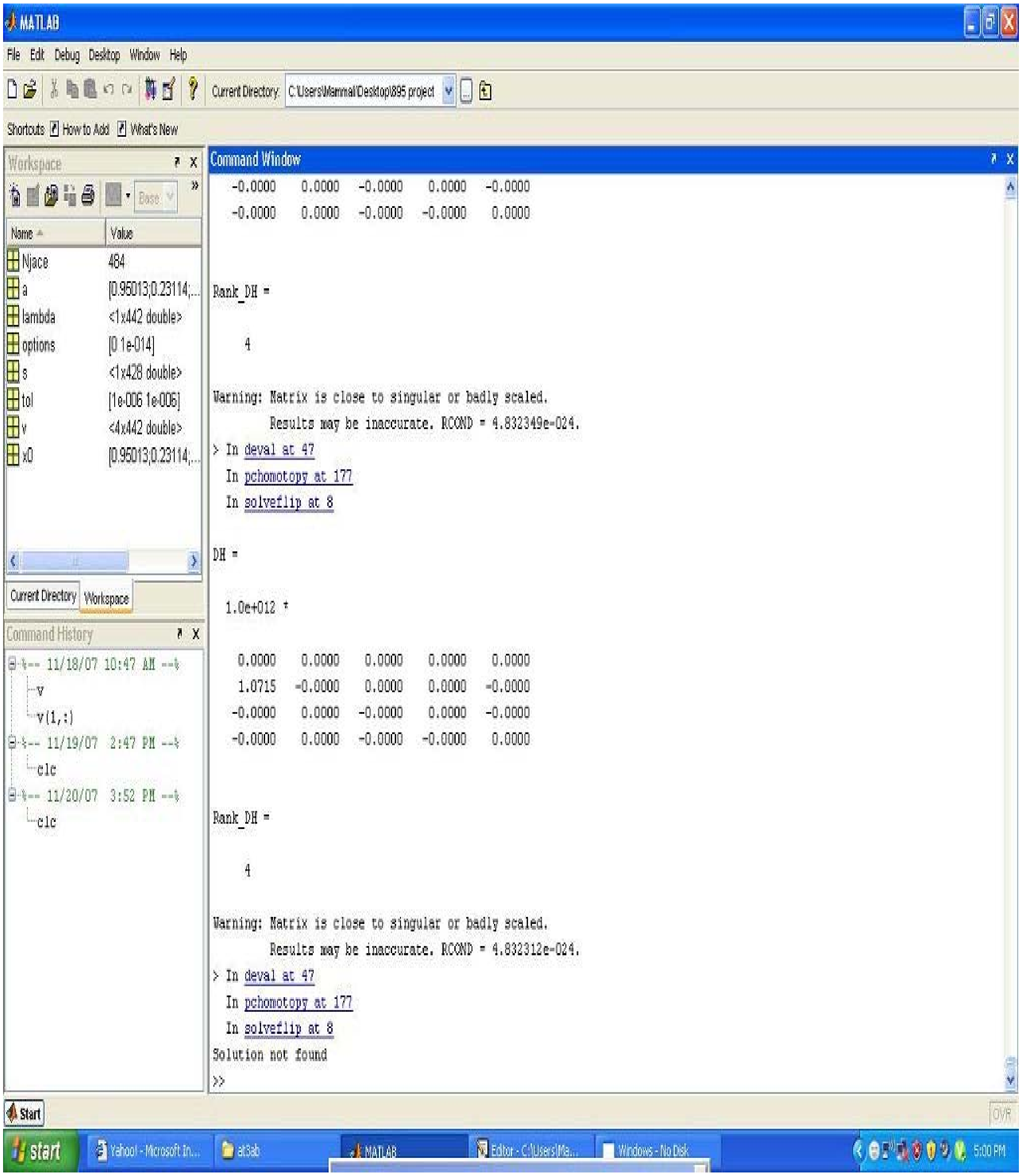
[4] R. C. Melville, Lj. Trajkovic, S. C. Fang, and L. T. Watson, ``Artificial parameter homotopy methods for the dc operating point problem," *IEEE Trans. on CAD*, vol. 12, pp. 861-877, June 1993.

[5] Lj. Trajkovic and W. Mathis, ``Parameter embedding methods for finding dc operating points: formulation and implementation," *Proc. NOLTA '95*, Las Vegas, NE, Dec. 1995, pp. 1159-1164.

[6] Trajkovic, L., Melville, R.C., Fang, S.-C. "Passivity and no-gain properties establish globalconvergence of a homotopy method for DC operating points," *Circuits and Systems, 1990., IEEE International Symposium,* 1-3 May 1990, Page(s):914 - 917 vol.2

[7] Lj. Trajkovic, ``Homotopy methods for computing dc operating points," invited contribution to the *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. New York: John Wiley & Sons, 1999, vol. 9, pp. 171-176.

Appendix A: Snapshot for Singularity of Jacobean Matrix



Appendix B: MATLAB Code

Schmitt Trigger Code:

Schmitt Trigger Circuit equation File

```
% func.m

function i = Schmittinit(X);

% FUNC Nonlinear Schmitt Trigger Circuit equation to be solved via homotopy

%Embedded parameter
lambda = 0;

%  Leakage voltage vector
global a


% This is schmitt1.Pout
q1IS = 1e-16;
q1BF = 100;
q1BR = 1;
q1N = 1.0/0.0257888;
q2IS = 1e-16;
q2BF = 100;
q2BR = 1;
q2N = 1.0/0.0257888;
rc1 = 2000;
rc2 = 1000;
r3 = 10000;
re = 100;

%Voltage sources
Vin = 1.5;
Ecc = 10;

Gleak = 1e-3;


% Circuit Equations  For Schmitt Trigger Circuit:


F(1) = (q1IS)*(exp(-q1N*(X(2)-X(5)))-1)+(-q1IS /q1BR )*(1+q1BR )*(exp(-q1N*(X(1)-X(5))) -1) +
(X(1)-X(6))/rc1 + (X(1)-X(4))/r3 ;

F(2) =  (-q1IS /q1BF )*(1+q1BF )*(exp(-q1N*(X(2)-X(5))) -1) - (- q1IS )*(exp(-q1N* (X(1)-X(5))) -1) +
(-q2IS /q2BF )*(1+q2BF )*(exp(-q2N*(X(2)-X(4))) -1) - (- q2IS )*(exp(-q2N*(X(3)-X(4))) -1) +
(X(2))/re ;


F(3) =  (q2IS )*(exp(-q2N*(X(2)-X(4))) -1) + (-q2IS /q2BR )*(1+q2BR )*(exp(-q2N*(X(3)-X(4))) -1) +
(X(3)-X(6))/rc2 ;
```

```
F(4) =  (-q2IS )*(exp(-q2N*(X(2)-X(4))) -1) + (q2IS /q2BR )*(1+q2BR )*(exp(-q2N* (X(3)-X(4))) -1) +
(q2IS /q2BF )*(1+q2BF )*(exp(-q2N* (X(2)-X(4))) -1) - ( q2IS )*(exp(-q2N*(X(3)-X(4))) -1) + (X(4)-
X(1))/r3 ;

F(6) =  (X(6)) -10;

F(5) =  (X(5)) -1.5;

F=F';

% This is homotopy function:

i = (1-lambda)*Gleak*(X-a) + lambda*F;

end
```

Schmitt Trigger Jacobean File

```
% Jacobian.m

function DH = SchmittJac(lambda,X)

% Homotopy Jacobian Function for Schmitt Trigger Circuit
%
% INPUTS:
% lambda - Embedded homotopy parameter
% v      - Voltages
%

%Voltage vector

global a;

Vin = 1.5;
Ecc = 10;



% This is schmitt trigger paramters
q1IS = 1e-16;
q1BF = 100;
q1BR = 1;
q1N = 1.0/0.0257888;
q2IS = 1e-16;
q2BF = 100;
q2BR = 1;
q2N = 1.0/0.0257888;
rc1 =2000;

rc2 = 1000;
r3 = 10000;
re = 100;

%Voltage sources
Vin = 1.5;
Ecc = 10;

Gleak = 1e-3;



% Circuit Equations for Schmitt Trigger:


F(1) = (q1IS)*(exp(-q1N*(X(2)-X(5)))-1)+(-q1IS /q1BR )*(1+q1BR )*(exp(-q1N*(X(1)-X(5))) -1) +
(X(1)-X(6))/rc1 + (X(1)-X(4))/r3 ;
```

```
F(2) =  (-q1IS /q1BF )*(1+q1BF )*(exp(-q1N*(X(2)-X(5))) -1) - (- q1IS )*(exp(-q1N* (X(1)-X(5))) -1) +
(-q2IS /q2BF )*(1+q2BF )*(exp(-q2N*(X(2)-X(4))) -1) - (- q2IS )*(exp(-q2N*(X(3)-X(4))) -1) +
(X(2))/re ;


F(3) =  (q2IS )*(exp(-q2N*(X(2)-X(4))) -1) + (-q2IS /q2BR )*(1+q2BR )*(exp(-q2N*(X(3)-X(4))) -1) +
(X(3)-X(6))/rc2 ;

F(4) =  (-q2IS )*(exp(-q2N*(X(2)-X(4))) -1) + (q2IS /q2BR )*(1+q2BR )*(exp(-q2N* (X(3)-X(4))) -1) +
(q2IS /q2BF )*(1+q2BF )*(exp(-q2N* (X(2)-X(4))) -1) - ( q2IS )*(exp(-q2N*(X(3)-X(4))) -1) + (X(4)-
X(1))/r3 ;

F(6) =  (X(6)) -10;

F(5) =  (X(5)) -1.5;

F=F';



% Jacobian


%  Jacobians for schmitt Trigger:

JAC(1, 1) =  (q1IS * q1N / q1BR)*(1 + q1BR)*(exp( - q1N *(X(1) - X(5)))) + 1/rc1 + 1/r3 ;

JAC(1, 5) =  (q1IS * q1N)*(exp( - q1N *(X(2) - X(5))))  - (q1IS * q1N / q1BR)*(1 + q1BR)*(exp( - q1N
*(X(1)-X(5)))) + 0 + 0 ;

JAC(1, 2) =  ( - q1IS * q1N)*(exp( - q1N *(X(2) - X(5)))) + 0 + 0 ;

JAC(1, 3) =  0 + 0 + 0 ;

JAC(1, 4) =  0 + 0 + (-1/r3) ;

JAC(1, 6) =  0 + (-1/rc1) + 0 ;

JAC(2, 1) =  ( -  q1IS * q1N)*(exp(-q1N*(X(1) - X(5)))) + 0 + 0 ;

JAC(2, 5) =  ( - q1IS * q1N / q1BF)*(1 + q1BF)*(exp(-q1N *(X(2)-X(5)))) + (q1IS * q1N)*(exp(-q1N
*(X(1) - X(5)))) + 0 + 0 ;

JAC(2, 2) =  (q1IS * q1N / q1BF)*(1 + q1BF)*(exp(-q1N *(X(2) - X(5)))) + (q2IS * q2N / q2BF)*(1 +
q2BF)*(exp(-q2N *(X(2) - X(4)))) + 1/re ;

JAC(2, 3) =  0 + ( -  q2IS * q2N)*(exp(-q2N *(X(3) - X(4)))) + 0 ;

JAC(2, 4) =  0 + ( - q2IS * q2N / q2BF)*(1 + q2BF)*(exp(-q2N *(X(2)-X(4)))) + (q2IS * q2N)*(exp(-q2N
*(X(3) - X(4)))) + 0 ;
```

JAC(2, 6) = 0 + 0 + 0 ;

JAC(3, 1) = 0 + 0 ;

JAC(3, 5) = 0 + 0 ;

JAC(3, 2) = (- q2IS * q2N)*(exp(- q2N *(X(2) - X(4)))) + 0 ;

JAC(3, 3) = (q2IS * q2N / q2BR)*(1 + q2BR)*(exp(- q2N *(X(3) - X(4)))) + 1/rc2 ;

JAC(3, 4) = (q2IS * q2N)*(exp(- q2N *(X(2) - X(4)))) - (q2IS * q2N / q2BR)*(1 + q2BR)*(exp(- q2N *(X(3)-X(4)))) + 0 ;

JAC(3, 6) = 0 + (-1/rc2) ;

JAC(4, 1) = 0 + (-1/r3) ;

JAC(4, 5) = 0 + 0 ;

JAC(4, 2) = (q2IS * q2N)*(exp(-q2N *(X(2) - X(4)))) - (q2IS * q2N / q2BF)*(1 + q2BF)*(exp(-q2N *(X(2) - X(4)))) + 0 ;

JAC(4, 3) = (- q2IS * q2N / q2BR)*(1 + q2BR)*(exp(-q2N *(X(3) - X(4)))) + (q2IS * q2N)*(exp(-q2N *(X(3) - X(4)))) + 0 ;

JAC(4, 4) = (- q2IS * q2N)*(exp(-q2N *(X(2) - X(4)))) + (q2IS * q2N / q2BR)*(1 + q2BR)*(exp(-q2N *(X(3)-X(4)))) + (q2IS * q2N / q2BF)*(1 + q2BF)*(exp(-q2N *(X(2) - X(4)))) - (q2IS * q2N)*(exp(-q2N *(X(3) - X(4)))) + 1/r3 ;

JAC(4, 6) = 0 + 0 ;

JAC(6, 1) = 0;

JAC(6, 5) = 0;

JAC(6, 2) = 0;

JAC(6, 3) = 0;

JAC(6, 4) = 0;

JAC(6, 6) = 1;

JAC(5, 1) = 0;

JAC(5, 5) = 1;

JAC(5, 2) = 0;

JAC(5, 3) = 0;

```
JAC(5, 4) = 0;

JAC(5, 6) = 0;


% WILLY HOMOTOPY FUNCTION
% Homotopy function (3): H(X,lambda)= (1-lambda)*Gleak*(X-a) + lambda*(F);
dHdv = (1-lambda)*Gleak*eye(6)+lambda*JAC;
dHdlambda = -Gleak*(X-a)+F ;


DH = [dHdlambda dHdv]
Rank_DH = rank(DH, 1.0e-16)

% stop
end
```

Schmitt Trigger Solve File

```
global a
%Randomly starting point
a=rand(6,1);
options(2)=1e-14;
%Finding result of equations based on starting  vector
x0 = fsolve('Schmittinit',a,options)
tol(1) = 1.e-6
tol(2) = 1.e-6

%Send Jacobean to pchomotopy tool to find solution for nonlinear equations
%based on homotopy method
%[xsolout, sout, lambdaout, xout, NJaceval]
[xs,s,lambda,v,Njace]=pchomotopy('SchmittJac',x0,50,tol,4);
axis ([0 2 0 10])
hold on
plot(lambda,v(1,:))
plot(lambda,v(2,:),'r')
plot(lambda,v(3,:),'g')
plot(lambda,v(4,:),'y')
plot(lambda,v(5,:),'b')
plot(lambda,v(6,:),'o')

axis([0 1.2 -2 12])
```

Flip flop Code:

Flip flop Circuit equation File

```
function i = flipflop(X);


lambda = 0;


global a

%Circuit Parameter For Flip-Flop

q1IS = -1e-16;
q1BF = 200;
q1BR = 1;
q1af=q1BF/(q1BF+1);
q1ar=q1BR/(q1BR+1);
q1N = 1.0/0.0257888;

q2IS = -1e-16;
q2BF = 200;
q2BR = 1;
q2af=q2BF/(q2BF+1);
q2ar=q2BR/(q2BR+1);
q2N = 1.0/0.0257888;

g1=10^(-7);
g2=.5*10^(-12);

%Voltage sources

Ecc = 20;

Gleak = 1e-3;

% Cicuit Equations


T=[ 1 -q1ar 0 0; -q1af 1 0 0; 0 0 -q2ar 1; 0 0 1 -q2af];
G=[ 2*g1+g2 -(g1+g2) -2*g1 g1; -(g1+g2) (g1+g2) g1 0 ; -2*g1 g1 2*g1+g2 (-g1-g2) ; g1 0 -g1-g2
g1+g2];
F=[(q1IS)*(exp(-q1N*(X(1)))-1); (q1IS)*(exp(-q1N*(X(2)))-1); (q2IS)*(exp(-q2N*(X(3)))-
1);(q2IS)*(exp(-q2N*(X(4)))-1)];
c=[ -g2*Ecc; g2*Ecc; -g2*Ecc; g2*Ecc];
H=F+inv(T)*G*[X(1);X(2);X(3);X(4)]+inv(T)*c;


% This is homotopy function:
```

```
i = (1-lambda)*Gleak*(X-a) + lambda*H;

end
```

Flip flop Jacobean File

```
% Jacobian.m

function DH = FlipJAC(lambda,X)

% Homotopy Jacobian Function for Flip-Flop Circuit
%
% INPUTS:
% lambda - Embedded homotopy parameter
% v      - Voltages
%

%Voltage vector

global a;

%Circuit Parameter For Flip-Flop



q1IS = -1e-16;
q1BF = 200;
q1BR = 1;
q1af=q1BF/(q1BF+1);
q1ar=q1BR/(q1BR+1);
q1N = 1.0/0.0257888;

q2IS = -1e-16;
q2BF = 200;
q2BR = 1;
q2af=q2BF/(q2BF+1);
q2ar=q2BR/(q2BR+1);
q2N = 1.0/0.0257888;

g1=10^(-7);
g2=.5*10^(-12);

%Voltage sources

Ecc = 20;

Gleak = 1e-3;


%Circuit Equations For Flip-Flop

T=[ 1 -q1ar 0 0; -q1af 1 0 0; 0 0 -q2ar 1; 0 0 1 -q2af];
G=[ 2*g1+g2 -(g1+g2) -2*g1 g1; -(g1+g2) (g1+g2) g1 0 ; -2*g1 g1 2*g1+g2 (-g1-g2) ; g1 0 -g1-g2
g1+g2];
F=[(q1IS)*(exp(-q1N*(X(1)))-1); (q1IS)*(exp(-q1N*(X(2)))-1); (q2IS)*(exp(-q2N*(X(3)))-
1);(q2IS)*(exp(-q2N*(X(4)))-1)];
c=[ -g2*Ecc; g2*Ecc; -g2*Ecc; g2*Ecc];
```

```
H=F+inv(T)*G*[X(1);X(2);X(3);X(4)]+inv(T)*c;

%Jaccobean For Flip Flop

JAC(1, 1) =   (-1/(-1+q1af*q1ar)*(2*g1+g2)-q1ar/(-1+q1af*q1ar)*(-g1-g2))-q1N*(q1IS)*(exp(-
q1N*(X(1))));

JAC(1, 2) =  (-1/(-1+q1af*q1ar)*(-g1-g2)-q1ar/(-1+q1af*q1ar)*(g1+g2)) ;

JAC(1, 3) =  (2/(-1+q1af*q1ar)*g1-q1ar/(-1+q1af*q1ar)*g1) ;

JAC(1, 4) =  1/(-1+q1af*q1ar)*g1;


JAC(2, 1) =  (-q1af/(-1+q1af*q1ar)*(2*g1+g2)-1/(-1+q1af*q1ar)*(-g1-g2)) ;

JAC(2, 2) = (-q1af/(-1+q1af*q1ar)*(-g1-g2)-1/(-1+q1af*q1ar)*(g1+g2))-q1N*(q1IS)*(exp(-q1N*(X(2)))) ;

JAC(2, 3) =  (2*q1af/(-1+q1af*q1ar)*g1-1/(-1+q1af*q1ar)*g1) ;

JAC(2, 4) =  -q1af/(-1+q1af*q1ar)*g1;



JAC(3, 1) = (2*q2af/(q2ar*q2af-1)*g1-1/(q2ar*q2af-1)*g1) ;

JAC(3, 2) =  -q2af/(q2ar*q2af-1)*g1;

JAC(3, 3) =  (-q2af/(q2ar*q2af-1)*(2*g1+g2)-1/(q2ar*q2af-1)*(-g1-g2))-q2N*(q2IS)*(exp(-q2N*(X(3))));

JAC(3, 4) =  (-q2af/(q2ar*q2af-1)*(-g1-g2)-1/(q2ar*q2af-1)*(g1+g2)) ;



JAC(4, 1) =  (2/(q2ar*q2af-1)*g1-q2ar/(q2ar*q2af-1)*g1) ;

JAC(4, 2) =  -1/(q2ar*q2af-1)*g1;

JAC(4, 3) =  (-1/(q2ar*q2af-1)*(2*g1+g2)-q2ar/(q2ar*q2af-1)*(-g1-g2)) ;

JAC(4, 4) =  (-1/(q2ar*q2af-1)*(-g1-g2)-q2ar/(q2ar*q2af-1)*(g1+g2))-q2N*(q2IS)*(exp(-q2N*(X(4))));



% WILLY HOMOTOPY FUNCTION
% Homotopy function (3): H(X,lambda)= (1-lambda)*Gleak*(X-a) + lambda*(F);
dHdv = (1-lambda)*Gleak*eye(4)+lambda*JAC;
```

```
dHdlambda = -Gleak*(X-a)+H ;


DH = [dHdlambda dHdv]
Rank_DH = rank(DH, 1.0e-16)

% stop
end
```

Flip flop Solve File

```
global a

%random vector of starting point

a=rand(4,1);
options(2)=1e-14;
%solving equations using a random vector

x0 = fsolve('flipflop',a,options)
tol(1) = 1.e-6
tol(2) = 1.e-6
% send Jacobean to pchomotopy tool in matlab to find solution for non
% linear equations
%[xsolout, sout, lambdaout, xout, NJaceval]

[xs,s,lambda,v,Njace]=pchomotopy('FlipJAC',x0,100000);
axis ([0 2 0 10])
hold on
plot(lambda,v(1,:))
plot(lambda,v(2,:),'r')
plot(lambda,v(3,:),'g')
plot(lambda,v(4,:),'y')

grid
axis([0 1.2 -2 12])
```

Chua Circuit equation File

```
% func.m

function i = chaucircuit(X);

% FUNC Nonlinear Chuacircuit Circuit equation to be solved via homotopy

%Embedded parameter
lambda = 0;

%  Leakage voltage vector
global a;


R1 = 10000;
R2 = 4000;
R3 = 4000;
R4 = 5000;
R5 = 30000;
R6 = 500;
R7 = 500;
R8 = 30000;
R9 = 10100;
R10 = 10100;
R11 = 4000;
R12 = 4000;
R13 = 30000;
R14 = 30000;
Q1IS = 1e-09;
Q1BF = 100;
Q1BR = 1;
Q1N = 38.7766;
Q2IS = 1e-09;
Q2BF = 100;
Q2BR = 1;
Q2N = 38.7766;
Q3IS = 1e-09;
Q3BF = 100;
Q3BR = 1;
Q3N = 38.7766;
Q4IS = 1e-09;
Q4BF = 100;
Q4BR = 1;
Q4N = 38.7766;

%Voltage sources
VCC1 = 12;
VCC2 = 12;
V1 = 10;
V2 = 2;

%Chua Circuit Equations
```

```
F(4) =  (X(4)-X(13))/R2 + (X(4)-X(6))/R5 + (Q1IS )*(exp(-Q1N*(X(8)-X(1))) -1) + (-Q1IS
/Q1BR )*(1+Q1BR )*(exp(-Q1N*(X(4)-X(1))) -1) ;

F(5) =  (X(5)-X(14))/R3 + (X(5)-X(7))/R8 + (Q3IS )*(exp(-Q3N*(X(9)-X(1))) -1) + (-Q3IS
/Q3BR )*(1+Q3BR )*(exp(-Q3N*(X(5)-X(1))) -1) ;

F(6) =  (X(6)-X(4))/R5 + (X(6))/R9 + (-Q2IS )*(exp(-Q2N*(X(8)-X(6))) -1) + (Q2IS
/Q2BR )*(1+Q2BR )*(exp(-Q2N*(X(10)-X(6))) -1) + (Q2IS /Q2BF )*(1+Q2BF )*(exp(-Q2N*(X(8)-
X(6))) -1) - ( Q2IS )*(exp(-Q2N*(X(10)-X(6))) -1) ;

F(8) =  (X(8))/R6 + (-Q1IS /Q1BF )*(1+Q1BF )*(exp(-Q1N*(X(8)-X(1))) -1) - (- Q1IS )*(exp(-
Q1N*(X(4)-X(1))) -1) + (-Q2IS /Q2BF )*(1+Q2BF )*(exp(-Q2N*(X(8)-X(6))) -1) - (- Q2IS )*(exp(-
Q2N*(X(10)-X(6))) -1) ;

F(9) =  (X(9))/R7 + (-Q3IS /Q3BF )*(1+Q3BF )*(exp(-Q3N*(X(9)-X(1))) -1) - (- Q3IS )*(exp(-
Q3N*(X(5)-X(1))) -1) + (-Q4IS /Q4BF )*(1+Q4BF )*(exp(-Q4N*(X(9)-X(7))) -1) - (- Q4IS )*(exp(-
Q4N*(X(12)-X(7))) -1) ;

F(7) =  (X(7)-X(5))/R8 + (X(7))/R10 + (-Q4IS )*(exp(-Q4N*(X(9)-X(7))) -1) + (Q4IS
/Q4BR )*(1+Q4BR )*(exp(-Q4N*(X(12)-X(7))) -1) + (Q4IS /Q4BF )*(1+Q4BF )*(exp(-Q4N*(X(9)-
X(7))) -1) - ( Q4IS )*(exp(-Q4N*(X(12)-X(7))) -1) ;

F(10) =  (X(10)-X(13))/R11 + (X(10)-X(3))/R13 + (Q2IS )*(exp(-Q2N*(X(8)-X(6))) -1) + (-Q2IS
/Q2BR )*(1+Q2BR )*(exp(-Q2N*(X(10)-X(6))) -1) ;

F(12) =  (X(12)-X(14))/R12 + (X(12)-X(11))/R14 + (Q4IS )*(exp(-Q4N*(X(9)-X(7))) -1) + (-Q4IS
/Q4BR )*(1+Q4BR )*(exp(-Q4N*(X(12)-X(7))) -1) ;

F(13) =  (X(13)) -12;

F(14) =  (X(14)) -12;

F(1) =  (X(1)-X(2)) -10;

F(11) =  (X(11)-X(3)) -2;

F(15)= X(15)+ (X(13)-X(4))/R2 + (X(13)-X(10))/R11 ;

F(16)= X(16)+ (X(14)-X(5))/R3 + (X(14)-X(12))/R12 ;

F(17)= X(17)+ (X(1))/R4 + (-Q1IS )*(exp(-Q1N*(X(8)-X(1))) -1) + (Q1IS /Q1BR )*(1+Q1BR )*(exp(-
Q1N*(X(4)-X(1))) -1) + (Q1IS /Q1BF )*(1+Q1BF )*(exp(-Q1N*(X(8)-X(1))) -1) - ( Q1IS )*(exp(-
Q1N*(X(4)-X(1))) -1) + (-Q3IS )*(exp(-Q3N*(X(9)-X(1))) -1) + (Q3IS /Q3BR )*(1+Q3BR )*(exp(-
Q3N*(X(5)-X(1))) -1) + (Q3IS /Q3BF )*(1+Q3BF )*(exp(-Q3N*(X(9)-X(1))) -1) - ( Q3IS )*(exp(-
Q3N*(X(5)-X(1))) -1) ;

F(2)= (-X(17) )+ (X(2)-X(3))/R1 ;

F(18)= X(18)+ (X(11)-X(12))/R14 ;

F(3)= (-X(18) )+ (X(3)-X(2))/R1 + (X(3)-X(10))/R13 ;
```

```
F=F';

Gleak = 1e-3;

% This is homotopy function:

i = (1-lambda)*Gleak*(X-a) + lambda*F;

end
```

Chua Jacobean File

```
% Jacobian.m

function DH = chauJac(lambda,X)

% Homotopy Jacobian Function for Chua Circuit
%
% INPUTS:

%Voltage vector
global a;
R1 = 10000;
R2 = 4000;
R3 = 4000;
R4 = 5000;
R5 = 30000;
R6 = 500;
R7 = 500;
R8 = 30000;
R9 = 10100;
R10 = 10100;
R11 = 4000;
R12 = 4000;
R13 = 30000;
R14 = 30000;
Q1IS = 1e-09;
Q1BF = 100;
Q1BR = 1;
Q1N = 38.7766;
Q2IS = 1e-09;
Q2BF = 100;
Q2BR = 1;
Q2N = 38.7766;
Q3IS = 1e-09;
Q3BF = 100;
Q3BR = 1;
Q3N = 38.7766;
Q4IS = 1e-09;
Q4BF = 100;
Q4BR = 1;
Q4N = 38.7766;

%Voltage sources
VCC1 = 12;
VCC2 = 12;
V1 = 10;
V2 = 2;
 %Circuit Equation and Jacobean Matrix

F(4) =  (X(4)-X(13))/R2 + (X(4)-X(6))/R5 + (Q1IS )*(exp(-Q1N*(X(8)-X(1))) -1) + (-Q1IS
/Q1BR )*(1+Q1BR )*(exp(-Q1N*(X(4)-X(1))) -1) ;
```

```
F(5) =  (X(5)-X(14))/R3 + (X(5)-X(7))/R8 + (Q3IS )*(exp(-Q3N*(X(9)-X(1))) -1) + (-Q3IS
/Q3BR )*(1+Q3BR )*(exp(-Q3N*(X(5)-X(1))) -1) ;

F(6) =  (X(6)-X(4))/R5 + (X(6))/R9 + (-Q2IS )*(exp(-Q2N*(X(8)-X(6))) -1) + (Q2IS
/Q2BR )*(1+Q2BR )*(exp(-Q2N*(X(10)-X(6))) -1) + (Q2IS /Q2BF )*(1+Q2BF )*(exp(-Q2N*(X(8)-
X(6))) -1) - ( Q2IS )*(exp(-Q2N*(X(10)-X(6))) -1) ;

F(8) =  (X(8))/R6 + (-Q1IS /Q1BF )*(1+Q1BF )*(exp(-Q1N*(X(8)-X(1))) -1) - (- Q1IS )*(exp(-
Q1N*(X(4)-X(1))) -1) + (-Q2IS /Q2BF )*(1+Q2BF )*(exp(-Q2N*(X(8)-X(6))) -1) - (- Q2IS )*(exp(-
Q2N*(X(10)-X(6))) -1) ;

F(9) =  (X(9))/R7 + (-Q3IS /Q3BF )*(1+Q3BF )*(exp(-Q3N*(X(9)-X(1))) -1) - (- Q3IS )*(exp(-
Q3N*(X(5)-X(1))) -1) + (-Q4IS /Q4BF )*(1+Q4BF )*(exp(-Q4N*(X(9)-X(7))) -1) - (- Q4IS )*(exp(-
Q4N*(X(12)-X(7))) -1) ;

F(7) =  (X(7)-X(5))/R8 + (X(7))/R10 + (-Q4IS )*(exp(-Q4N*(X(9)-X(7))) -1) + (Q4IS
/Q4BR )*(1+Q4BR )*(exp(-Q4N*(X(12)-X(7))) -1) + (Q4IS /Q4BF )*(1+Q4BF )*(exp(-Q4N*(X(9)-
X(7))) -1) - ( Q4IS )*(exp(-Q4N*(X(12)-X(7))) -1) ;

F(10) =  (X(10)-X(13))/R11 + (X(10)-X(3))/R13 + (Q2IS )*(exp(-Q2N*(X(8)-X(6))) -1) + (-Q2IS
/Q2BR )*(1+Q2BR )*(exp(-Q2N*(X(10)-X(6))) -1) ;

F(12) =  (X(12)-X(14))/R12 + (X(12)-X(11))/R14 + (Q4IS )*(exp(-Q4N*(X(9)-X(7))) -1) + (-Q4IS
/Q4BR )*(1+Q4BR )*(exp(-Q4N*(X(12)-X(7))) -1) ;

F(13) =  (X(13)) -12;

F(14) =  (X(14)) -12;

F(1) =  (X(1)-X(2)) -10;

F(11) =  (X(11)-X(3)) -2;

F(15)= X(15)+ (X(13)-X(4))/R2 + (X(13)-X(10))/R11 ;

F(16)= X(16)+ (X(14)-X(5))/R3 + (X(14)-X(12))/R12 ;

F(17)= X(17)+ (X(1))/R4 + (-Q1IS )*(exp(-Q1N*(X(8)-X(1))) -1) + (Q1IS /Q1BR )*(1+Q1BR )*(exp(-
Q1N*(X(4)-X(1))) -1) + (Q1IS /Q1BF )*(1+Q1BF )*(exp(-Q1N*(X(8)-X(1))) -1) - ( Q1IS )*(exp(-
Q1N*(X(4)-X(1))) -1) + (-Q3IS )*(exp(-Q3N*(X(9)-X(1))) -1) + (Q3IS /Q3BR )*(1+Q3BR )*(exp(-
Q3N*(X(5)-X(1))) -1) + (Q3IS /Q3BF )*(1+Q3BF )*(exp(-Q3N*(X(9)-X(1))) -1) - ( Q3IS )*(exp(-
Q3N*(X(5)-X(1))) -1) ;

F(2)= (-X(17) )+ (X(2)-X(3))/R1 ;

F(18)= X(18)+ (X(11)-X(12))/R14 ;

F(3)= (-X(18) )+ (X(3)-X(2))/R1 + (X(3)-X(10))/R13 ;
F=F';


JAC(4, 13) =  (-1/R2) + 0 + 0 ;
```

```
JAC(4, 15) = 0;

JAC(4, 14) =  0 + 0 + 0 ;

JAC(4, 16) = 0;

JAC(4, 1) =  0 + 0 + (Q1IS * Q1N)*(exp( - Q1N*(X(8) - X(1))))  - (Q1IS * Q1N / Q1BR)*(1 +
Q1BR)*(exp( - Q1N*(X(4)-X(1)))) ;

JAC(4, 2) =  0 + 0 + 0 ;

JAC(4, 17) = 0;

JAC(4, 11) =  0 + 0 + 0 ;

JAC(4, 3) =  0 + 0 + 0 ;

JAC(4, 18) = 0;

JAC(4, 4) =  1/R2 + 1/R5 + (Q1IS * Q1N / Q1BR)*(1 + Q1BR)*(exp( - Q1N*(X(4) - X(1)))) ;

JAC(4, 5) =  0 + 0 + 0 ;

JAC(4, 6) =  0 + (-1/R5) + 0 ;

JAC(4, 8) =  0 + 0 + ( - Q1IS * Q1N)*(exp( - Q1N*(X(8) - X(1)))) ;

JAC(4, 9) =  0 + 0 + 0 ;

JAC(4, 7) =  0 + 0 + 0 ;

JAC(4, 10) =  0 + 0 + 0 ;

JAC(4, 12) =  0 + 0 + 0 ;

JAC(5, 13) =  0 + 0 + 0 ;

JAC(5, 15) = 0;

JAC(5, 14) =  (-1/R3) + 0 + 0 ;

JAC(5, 16) = 0;

JAC(5, 1) =  0 + 0 + (Q3IS * Q3N)*(exp( - Q3N*(X(9) - X(1))))  - (Q3IS * Q3N / Q3BR)*(1 +
Q3BR)*(exp( - Q3N*(X(5)-X(1)))) ;

JAC(5, 2) =  0 + 0 + 0 ;

JAC(5, 17) = 0;
```

```
JAC(5, 11) =  0 + 0 + 0 ;

JAC(5, 3) =  0 + 0 + 0 ;

JAC(5, 18) = 0;

JAC(5, 4) =  0 + 0 + 0 ;

JAC(5, 5) =  1/R3 + 1/R8 + (Q3IS * Q3N / Q3BR)*(1 + Q3BR)*(exp( - Q3N*(X(5) - X(1)))) ;

JAC(5, 6) =  0 + 0 + 0 ;

JAC(5, 8) =  0 + 0 + 0 ;

JAC(5, 9) =  0 + 0 + ( - Q3IS * Q3N)*(exp( - Q3N*(X(9) - X(1)))) ;

JAC(5, 7) =  0 + (-1/R8) + 0 ;

JAC(5, 10) =  0 + 0 + 0 ;

JAC(5, 12) =  0 + 0 + 0 ;

JAC(6, 13) =  0 + 0 + 0 ;

JAC(6, 15) = 0;

JAC(6, 14) =  0 + 0 + 0 ;

JAC(6, 16) = 0;

JAC(6, 1) =  0 + 0 + 0 ;

JAC(6, 2) =  0 + 0 + 0 ;

JAC(6, 17) = 0;

JAC(6, 11) =  0 + 0 + 0 ;

JAC(6, 3) =  0 + 0 + 0 ;

JAC(6, 18) = 0;

JAC(6, 4) =  (-1/R5) + 0 + 0 ;

JAC(6, 5) =  0 + 0 + 0 ;

JAC(6, 6) =  1/R5 + 1/R9 + ( - Q2IS * Q2N)*(exp(-Q2N*(X(8) - X(6)))) + (Q2IS * Q2N / Q2BR )*(1 +
Q2BR )*(exp(-Q2N*(X(10)-X(6)))) + (Q2IS * Q2N / Q2BF )*(1 + Q2BF )*(exp(-Q2N*(X(8) - X(6)))) -
(Q2IS * Q2N)*(exp(-Q2N*(X(10) - X(6)))) ;
```

JAC(6, 8) = 0 + 0 + (Q2IS * Q2N)*(exp(-Q2N*(X(8) - X(6)))) - (Q2IS * Q2N / Q2BF)*(1 + Q2BF)*(exp(-Q2N*(X(8) - X(6)))) ;

JAC(6, 9) = 0 + 0 + 0 ;

JAC(6, 7) = 0 + 0 + 0 ;

JAC(6, 10) = 0 + 0 + (- Q2IS * Q2N / Q2BR)*(1 + Q2BR)*(exp(-Q2N*(X(10) - X(6)))) + (Q2IS * Q2N)*(exp(-Q2N*(X(10) - X(6)))) ;

JAC(6, 12) = 0 + 0 + 0 ;

JAC(8, 13) = 0 + 0 + 0 ;

JAC(8, 15) = 0;

JAC(8, 14) = 0 + 0 + 0 ;

JAC(8, 16) = 0;

JAC(8, 1) = 0 + (- Q1IS * Q1N / Q1BF)*(1 + Q1BF)*(exp(-Q1N*(X(8)-X(1)))) + (Q1IS * Q1N)*(exp(-Q1N*(X(4) - X(1)))) + 0 ;

JAC(8, 2) = 0 + 0 + 0 ;

JAC(8, 17) = 0;

JAC(8, 11) = 0 + 0 + 0 ;

JAC(8, 3) = 0 + 0 + 0 ;

JAC(8, 18) = 0;

JAC(8, 4) = 0 + (- Q1IS * Q1N)*(exp(-Q1N*(X(4) - X(1)))) + 0 ;

JAC(8, 5) = 0 + 0 + 0 ;

JAC(8, 6) = 0 + 0 + (- Q2IS * Q2N / Q2BF)*(1 + Q2BF)*(exp(-Q2N*(X(8)-X(6)))) + (Q2IS * Q2N)*(exp(-Q2N*(X(10) - X(6)))) ;

JAC(8, 8) = 1/R6 + (Q1IS * Q1N / Q1BF)*(1 + Q1BF)*(exp(-Q1N*(X(8) - X(1)))) + (Q2IS * Q2N / Q2BF)*(1 + Q2BF)*(exp(-Q2N*(X(8) - X(6)))) ;

JAC(8, 9) = 0 + 0 + 0 ;

JAC(8, 7) = 0 + 0 + 0 ;

JAC(8, 10) = 0 + 0 + (- Q2IS * Q2N)*(exp(-Q2N*(X(10) - X(6)))) ;

JAC(8, 12) = 0 + 0 + 0 ;

JAC(9, 13) = 0 + 0 + 0 ;

JAC(9, 15) = 0;

JAC(9, 14) = 0 + 0 + 0 ;

JAC(9, 16) = 0;

JAC(9, 1) = 0 + (- Q3IS * Q3N / Q3BF)*(1 + Q3BF)*(exp(-Q3N*(X(9)-X(1)))) + (Q3IS * Q3N)*(exp(-Q3N*(X(5) - X(1)))) + 0 ;

JAC(9, 2) = 0 + 0 + 0 ;

JAC(9, 17) = 0;

JAC(9, 11) = 0 + 0 + 0 ;

JAC(9, 3) = 0 + 0 + 0 ;

JAC(9, 18) = 0;

JAC(9, 4) = 0 + 0 + 0 ;

JAC(9, 5) = 0 + (- Q3IS * Q3N)*(exp(-Q3N*(X(5) - X(1)))) + 0 ;

JAC(9, 6) = 0 + 0 + 0 ;

JAC(9, 8) = 0 + 0 + 0 ;

JAC(9, 9) = 1/R7 + (Q3IS * Q3N / Q3BF)*(1 + Q3BF)*(exp(-Q3N*(X(9) - X(1)))) + (Q4IS * Q4N / Q4BF)*(1 + Q4BF)*(exp(-Q4N*(X(9) - X(7)))) ;

JAC(9, 7) = 0 + 0 + (- Q4IS * Q4N / Q4BF)*(1 + Q4BF)*(exp(-Q4N*(X(9)-X(7)))) + (Q4IS * Q4N)*(exp(-Q4N*(X(12) - X(7)))) ;

JAC(9, 10) = 0 + 0 + 0 ;

JAC(9, 12) = 0 + 0 + (- Q4IS * Q4N)*(exp(-Q4N*(X(12) - X(7)))) ;

JAC(7, 13) = 0 + 0 + 0 ;

JAC(7, 15) = 0;

JAC(7, 14) = 0 + 0 + 0 ;

JAC(7, 16) = 0;

JAC(7, 1) = 0 + 0 + 0 ;

JAC(7, 2) = 0 + 0 + 0 ;

```
JAC(7, 17) = 0;

JAC(7, 11) =  0 + 0 + 0 ;

JAC(7, 3) =  0 + 0 + 0 ;

JAC(7, 18) = 0;

JAC(7, 4) =  0 + 0 + 0 ;

JAC(7, 5) =  (-1/R8) + 0 + 0 ;

JAC(7, 6) =  0 + 0 + 0 ;

JAC(7, 8) =  0 + 0 + 0 ;

JAC(7, 9) =  0 + 0 + (Q4IS * Q4N)*(exp(-Q4N*(X(9) - X(7)))) - (Q4IS * Q4N / Q4BF)*(1 +
Q4BF)*(exp(-Q4N*(X(9) - X(7)))) ;

JAC(7, 7) =  1/R8 + 1/R10 + ( - Q4IS * Q4N)*(exp(-Q4N*(X(9) - X(7)))) + (Q4IS * Q4N / Q4BR )*(1 +
Q4BR )*(exp(-Q4N*(X(12)-X(7)))) + (Q4IS * Q4N / Q4BF )*(1 + Q4BF )*(exp(-Q4N*(X(9) - X(7)))) -
(Q4IS * Q4N)*(exp(-Q4N*(X(12) - X(7)))) ;

JAC(7, 10) =  0 + 0 + 0 ;

JAC(7, 12) =  0 + 0 + ( - Q4IS * Q4N / Q4BR)*(1 + Q4BR)*(exp(-Q4N*(X(12) - X(7)))) + (Q4IS *
Q4N)*(exp(-Q4N*(X(12) - X(7)))) ;

JAC(10, 13) =  (-1/R11) + 0 + 0 ;

JAC(10, 15) = 0;

JAC(10, 14) =  0 + 0 + 0 ;

JAC(10, 16) = 0;

JAC(10, 1) =  0 + 0 + 0 ;

JAC(10, 2) =  0 + 0 + 0 ;

JAC(10, 17) = 0;

JAC(10, 11) =  0 + 0 + 0 ;

JAC(10, 3) =  0 + (-1/R13) + 0 ;

JAC(10, 18) = 0;

JAC(10, 4) =  0 + 0 + 0 ;

JAC(10, 5) =  0 + 0 + 0 ;
```

```
JAC(10, 6) =  0 + 0 + (Q2IS * Q2N)*(exp( - Q2N*(X(8) - X(6))))  - (Q2IS * Q2N / Q2BR)*(1 +
Q2BR)*(exp( - Q2N*(X(10)-X(6)))) ;

JAC(10, 8) =  0 + 0 + ( - Q2IS * Q2N)*(exp( - Q2N*(X(8) - X(6)))) ;

JAC(10, 9) =  0 + 0 + 0 ;

JAC(10, 7) =  0 + 0 + 0 ;

JAC(10, 10) =  1/R11 + 1/R13 + (Q2IS * Q2N / Q2BR)*(1 + Q2BR)*(exp( - Q2N*(X(10) - X(6)))) ;

JAC(10, 12) =  0 + 0 + 0 ;

JAC(12, 13) =  0 + 0 + 0 ;

JAC(12, 15) = 0;

JAC(12, 14) =  (-1/R12) + 0 + 0 ;

JAC(12, 16) = 0;

JAC(12, 1) =  0 + 0 + 0 ;

JAC(12, 2) =  0 + 0 + 0 ;

JAC(12, 17) = 0;

JAC(12, 11) =  0 + (-1/R14) + 0 ;

JAC(12, 3) =  0 + 0 + 0 ;

JAC(12, 18) = 0;

JAC(12, 4) =  0 + 0 + 0 ;

JAC(12, 5) =  0 + 0 + 0 ;

JAC(12, 6) =  0 + 0 + 0 ;

JAC(12, 8) =  0 + 0 + 0 ;

JAC(12, 9) =  0 + 0 + ( - Q4IS * Q4N)*(exp( - Q4N*(X(9) - X(7)))) ;

JAC(12, 7) =  0 + 0 + (Q4IS * Q4N)*(exp( - Q4N*(X(9) - X(7))))  - (Q4IS * Q4N / Q4BR)*(1 +
Q4BR)*(exp( - Q4N*(X(12)-X(7)))) ;

JAC(12, 10) =  0 + 0 + 0 ;

JAC(12, 12) =  1/R12 + 1/R14 + (Q4IS * Q4N / Q4BR)*(1 + Q4BR)*(exp( - Q4N*(X(12) - X(7)))) ;
```

```
JAC(13, 13) = 1;

JAC(13, 15) = 0;

JAC(13, 14) = 0;

JAC(13, 16) = 0;

JAC(13, 1) = 0;

JAC(13, 2) = 0;

JAC(13, 17) = 0;

JAC(13, 11) = 0;

JAC(13, 3) = 0;

JAC(13, 18) = 0;

JAC(13, 4) = 0;

JAC(13, 5) = 0;

JAC(13, 6) = 0;

JAC(13, 8) = 0;

JAC(13, 9) = 0;

JAC(13, 7) = 0;

JAC(13, 10) = 0;

JAC(13, 12) = 0;

JAC(14, 13) = 0;

JAC(14, 15) = 0;

JAC(14, 14) = 1;

JAC(14, 16) = 0;

JAC(14, 1) = 0;

JAC(14, 2) = 0;

JAC(14, 17) = 0;

JAC(14, 11) = 0;
```

JAC(14, 3) = 0;

JAC(14, 18) = 0;

JAC(14, 4) = 0;

JAC(14, 5) = 0;

JAC(14, 6) = 0;

JAC(14, 8) = 0;

JAC(14, 9) = 0;

JAC(14, 7) = 0;

JAC(14, 10) = 0;

JAC(14, 12) = 0;

JAC(1, 13) = 0;

JAC(1, 15) = 0;

JAC(1, 14) = 0;

JAC(1, 16) = 0;

JAC(1, 1) = 1;

JAC(1, 2) = -1;

JAC(1, 17) = 0;

JAC(1, 11) = 0;

JAC(1, 3) = 0;

JAC(1, 18) = 0;

JAC(1, 4) = 0;

JAC(1, 5) = 0;

JAC(1, 6) = 0;

JAC(1, 8) = 0;

JAC(1, 9) = 0;

JAC(1, 7) = 0;

JAC(1, 10) = 0;

JAC(1, 12) = 0;

JAC(11, 13) = 0;

JAC(11, 15) = 0;

JAC(11, 14) = 0;

JAC(11, 16) = 0;

JAC(11, 1) = 0;

JAC(11, 2) = 0;

JAC(11, 17) = 0;

JAC(11, 11) = 1;

JAC(11, 3) = -1;

JAC(11, 18) = 0;

JAC(11, 4) = 0;

JAC(11, 5) = 0;

JAC(11, 6) = 0;

JAC(11, 8) = 0;

JAC(11, 9) = 0;

JAC(11, 7) = 0;

JAC(11, 10) = 0;

JAC(11, 12) = 0;

JAC(15, 13) = 1/R2 + 1/R11 ;

JAC(15, 15) = 1;

JAC(15, 14) = 0 + 0 + 0 ;

JAC(15, 16) = 0;

JAC(15, 1) = 0 + 0 + 0 ;

JAC(15, 2) = 0 + 0 + 0 ;

JAC(15, 17) = 0;

JAC(15, 11) = 0 + 0 + 0 ;

JAC(15, 3) = 0 + 0 + 0 ;

JAC(15, 18) = 0;

JAC(15, 4) = 0 + (-1/R2) + 0 ;

JAC(15, 5) = 0 + 0 + 0 ;

JAC(15, 6) = 0 + 0 + 0 ;

JAC(15, 8) = 0 + 0 + 0 ;

JAC(15, 9) = 0 + 0 + 0 ;

JAC(15, 7) = 0 + 0 + 0 ;

JAC(15, 10) = 0 + 0 + (-1/R11) ;

JAC(15, 12) = 0 + 0 + 0 ;

JAC(16, 13) = 0 + 0 + 0 ;

JAC(16, 15) = 0;

JAC(16, 14) = 1/R3 + 1/R12 ;

JAC(16, 16) = 1;

JAC(16, 1) = 0 + 0 + 0 ;

JAC(16, 2) = 0 + 0 + 0 ;

JAC(16, 17) = 0;

JAC(16, 11) = 0 + 0 + 0 ;

JAC(16, 3) = 0 + 0 + 0 ;

JAC(16, 18) = 0;

JAC(16, 4) = 0 + 0 + 0 ;

JAC(16, 5) = 0 + (-1/R3) + 0 ;

JAC(16, 6) = 0 + 0 + 0 ;

JAC(16, 8) = 0 + 0 + 0 ;

JAC(16, 9) = 0 + 0 + 0 ;

JAC(16, 7) = 0 + 0 + 0 ;

JAC(16, 10) = 0 + 0 + 0 ;

JAC(16, 12) = 0 + 0 + (-1/R12) ;

JAC(17, 13) = 0 + 0 + 0 + 0 ;

JAC(17, 15) = 0;

JAC(17, 14) = 0 + 0 + 0 + 0 ;

JAC(17, 16) = 0;

JAC(17, 1) = 1/R4 + (- Q1IS * Q1N)*(exp(-Q1N*(X(8) - X(1)))) + (Q1IS * Q1N / Q1BR)*(1 + Q1BR)*(exp(-Q1N*(X(4)-X(1)))) + (Q1IS * Q1N / Q1BF)*(1 + Q1BF)*(exp(-Q1N*(X(8) - X(1)))) - (Q1IS * Q1N)*(exp(-Q1N*(X(4) - X(1)))) + (- Q3IS * Q3N)*(exp(-Q3N*(X(9) - X(1)))) + (Q3IS * Q3N / Q3BR)*(1 + Q3BR)*(exp(-Q3N*(X(5)-X(1)))) + (Q3IS * Q3N / Q3BF)*(1 + Q3BF)*(exp(-Q3N*(X(9) - X(1)))) - (Q3IS * Q3N)*(exp(-Q3N*(X(5) - X(1)))) ;

JAC(17, 2) = 0 + 0 + 0 ;

JAC(17, 17) = 1;

JAC(17, 11) = 0 + 0 + 0 + 0 ;

JAC(17, 3) = 0 + 0 + 0 + 0 ;

JAC(17, 18) = 0;

JAC(17, 4) = 0 + 0 + (- Q1IS * Q1N / Q1BR)*(1 + Q1BR)*(exp(-Q1N*(X(4) - X(1)))) + (Q1IS * Q1N)*(exp(-Q1N*(X(4) - X(1)))) + 0 ;

JAC(17, 5) = 0 + 0 + 0 + (- Q3IS * Q3N / Q3BR)*(1 + Q3BR)*(exp(-Q3N*(X(5) - X(1)))) + (Q3IS * Q3N)*(exp(-Q3N*(X(5) - X(1)))) ;

JAC(17, 6) = 0 + 0 + 0 + 0 ;

JAC(17, 8) = 0 + 0 + (Q1IS * Q1N)*(exp(-Q1N*(X(8) - X(1)))) - (Q1IS * Q1N / Q1BF)*(1 + Q1BF)*(exp(-Q1N*(X(8) - X(1)))) + 0 ;

JAC(17, 9) = 0 + 0 + 0 + (Q3IS * Q3N)*(exp(-Q3N*(X(9) - X(1)))) - (Q3IS * Q3N / Q3BF)*(1 + Q3BF)*(exp(-Q3N*(X(9) - X(1)))) ;

JAC(17, 7) = 0 + 0 + 0 + 0 ;

```
JAC(17, 10) =  0 + 0 + 0 + 0 ;

JAC(17, 12) =  0 + 0 + 0 + 0 ;

JAC(2, 13) =  0 + 0 ;

JAC(2, 15) = 0;

JAC(2, 14) =  0 + 0 ;

JAC(2, 16) = 0;

JAC(2, 1) = 0 ;

JAC(2, 2) = 1/R1 ;

JAC(2, 17) = -1;

JAC(2, 11) =  0 + 0 ;

JAC(2, 3) =  0 + (-1/R1) ;

JAC(2, 18) = 0;

JAC(2, 4) =  0 + 0 ;

JAC(2, 5) =  0 + 0 ;

JAC(2, 6) =  0 + 0 ;

JAC(2, 8) =  0 + 0 ;

JAC(2, 9) =  0 + 0 ;

JAC(2, 7) =  0 + 0 ;

JAC(2, 10) =  0 + 0 ;

JAC(2, 12) =  0 + 0 ;

JAC(18, 13) =  0 + 0 ;

JAC(18, 15) = 0;

JAC(18, 14) =  0 + 0 ;

JAC(18, 16) = 0;

JAC(18, 1) =  0 + 0 ;
```

JAC(18, 2) = 0 + 0 ;

JAC(18, 17) = 0;

JAC(18, 11) = 1/R14 ;

JAC(18, 3) = 0 ;

JAC(18, 18) = 1;

JAC(18, 4) = 0 + 0 ;

JAC(18, 5) = 0 + 0 ;

JAC(18, 6) = 0 + 0 ;

JAC(18, 8) = 0 + 0 ;

JAC(18, 9) = 0 + 0 ;

JAC(18, 7) = 0 + 0 ;

JAC(18, 10) = 0 + 0 ;

JAC(18, 12) = 0 + (-1/R14) ;

JAC(3, 13) = 0 + 0 + 0 ;

JAC(3, 15) = 0;

JAC(3, 14) = 0 + 0 + 0 ;

JAC(3, 16) = 0;

JAC(3, 1) = 0 + 0 + 0 ;

JAC(3, 2) = 0 + (-1/R1) + 0 ;

JAC(3, 17) = 0;

JAC(3, 11) = 0 + 0 ;

JAC(3, 3) = 1/R1 + 1/R13 ;

JAC(3, 18) = -1;

JAC(3, 4) = 0 + 0 + 0 ;

JAC(3, 5) = 0 + 0 + 0 ;

JAC(3, 6) = 0 + 0 + 0 ;

```
JAC(3, 8) =  0 + 0 + 0 ;

JAC(3, 9) =  0 + 0 + 0 ;

JAC(3, 7) =  0 + 0 + 0 ;

JAC(3, 10) =  0 + 0 + (-1/R13) ;

JAC(3, 12) =  0 + 0 + 0 ;


Gleak = 1e-3;


% WILLY HOMOTOPY FUNCTION
% Homotopy function (3): H(X,lambda)= (1-lambda)*Gleak*(X-a) + lambda*(F);
dHdv = (1-lambda)*Gleak*eye(18)+lambda*JAC;
dHdlambda = -Gleak*(X-a)+F;

DH = [dHdlambda dHdv]
Rank_DH = rank(DH, 1.0e-16)

% stop
end
```

Chua Solve File

```
global a

%Random vector for starting point
a=rand(18,1);
options(2)=1e-16;
%Find solution for random vector
X0 = fsolve('chaucircuit',a,options)
%[xsolout, sout, lambdaout, xout, NJaceval]
%Send Jacobean to Pchomotopy tool for solving nonlinear equations
[xs, vs,s,lambda,v]=pchomotopy('ChauJac',X0,250,[1e-8  1e-8]);
axis ([0 2 -1 10])
hold on
plot(lambda,v)
```