Improving a Electronic Circuit Simulator based on Homotopy Methods

Joo Erik de Andrade Melo*

[1]Informatics Center (CIn)
Federal University of Pernambuco (UFPE)
Recife - Pernambuco, Brazil

jeam2@cin.ufpe.br
Faculty advisor: Professor Ljiljana Trajkovic

1. Introduction

It is of great importance to industry and academia that works with electronic circuits determine voltages and currents, at many point of the circuit, that makes the electronic components operates properly, that is, with the behavior that makes the system works as desired. These specific voltages and currents are called DC operating points of the circuit. To find the DC operating points of a transistor circuit it is necessary solve a system of nonlinear algebraic equations that describe the DC behavior of this circuit. A common method used to find DC operating points is the Newton-Raphson method that require a initial point close enough to the solution, that sometimes is difficult to provide. Therefore, Homotopy methods, an alternative method to solve nonlinear system of equations, are being applied to find DC operating points of circuits. In this project we have used a software implementation of a homotopy method. This implementation is composed for two programs: a parser, developed by Edward Chan [2], that provide the system of equations from a circuit description. And a MATLAB script [1], written by Heath Hoffman, that implements the homotopy method to solve the system of equations. However, the Parser was not providing the equations in the correct form required by the MATLAB program. The main contribution of this project was figure out the most of the problems found in the Parser.

2. Fiding DC Operating Points

The flow to find the DC operating points on this project is described by the schema in the figure 1 bellow:

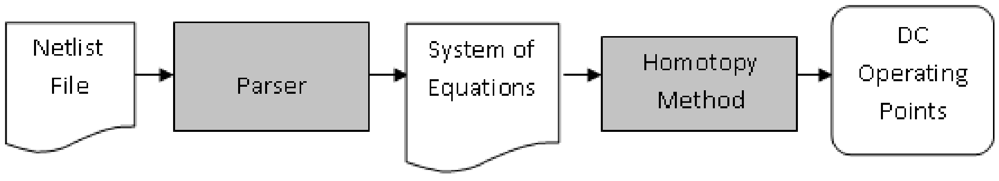

Figure 1. Flow to find DC Operating points

First the circuit is described by a netlist file, that is written in the well-known SPICE format. This file is used as input to the Parser program that produce the system of nodal or modified nodal equations in an symbolic format. These nonlinear equations are introduced in the homotopy algorithm that solve the system obtaining the DC operating points of the specified circuit. The next sections will give a quick explanation about the Homotopy method and the parser used in this project.

3. Homotopy Methods

Homotopy methods [7] [3] are methods to find zeros of a equation or system of equations. It is also called parameter embedding. This name comes from the fact that, in the homotopy methods, one new parameter is introduced in the system of equations.

The main idea behind the homotopy methods to solve a problem is create a new problem simpler than the original one and then deform this problem into the original one, computing a series of zeros until the zero(s) of the original problem.

The advantage of homotopy methods with relation to others methods like Newton-type method and Gauss-Jacobi is the fact that homotopy is a globally convergent. A method that has globally convergence is one that converge for a solution from almost any starting point [6]. Methods like Newton-type and Gauss-Jacobi are local convergent, what means that it cannot converge if the initial guess is not sufficiently closer to the solution. Let's see a example:

To solve a system of nodal equations given by:

$F(x) = 0$

We start creating a new function called Homotopy function $H(x, \lambda)$ where was introduced the embedded parameter λ. The Homotopy function $H(x, \lambda)$ is created in such form that it deforms continuously a system of equations $G(x)$ into $F(x)$ what means that $H(x, \lambda)$ is a continuous function where $H(x, 0) = G(x)$, $H(x, 1) = F(x)$ and $G(x)$ is a system of equation of which we know the solution or that is simple to solve.

An example of Homotopy function is:

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

where $G(x) = (x - a)$ for some starting point a.

The objective is obtain a set of pairs (x, λ), that we call $H^{-1}(0)$, such as

$H^{-1}(0) = \{(x, \lambda)|H(x, \lambda) = 0\}$

Inside this set we hope to find a continuous path (or curve) that connect zeros of our simple system $H(x, 0)$ to zeros of $H(x, 1)$, the desired system of equations. The method to find this path include differentiate the Homotopy function with respect to x and λ and then use a numerical method, as Euler or Runge-Kutta methods, to solve the differential equation(s) created [5].

4. Parser

As it was mentioned, the MATLAB algorithm that implements the homotopy method, require a system of equations in a symbolic form. The parser created for Edward Chan receives a netlist file as input and generate these equations. A netlist file is a text file that describe a circuit, its components and how they are connected. Bellow there is a example of Schmitt trigger circuit in the schematic (fig. 2) and netlist (fig. 3) representation.

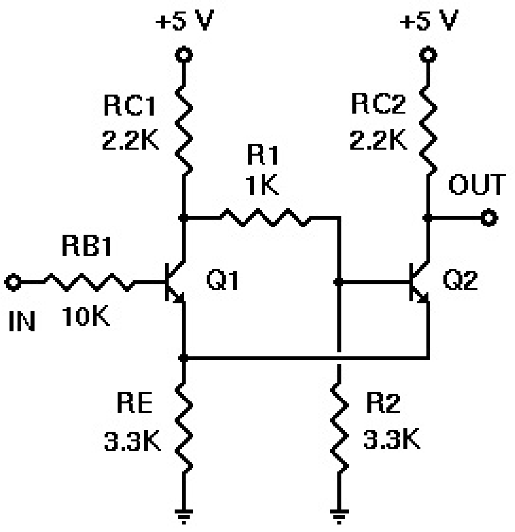

Figure 2. Schmitt Trigger Circuit (schematic)

```
Rc1  1 2 2.2K
R1   2 3 1K
Rc2  1 4 2.2K
Q1   2 5 6 Q2N2222A
Q2   4 3 6 Q2N2222A
Vin  5 0 5.0
RE   6 0 3.3K
R2   3 0 3.3K

.model Q2N2222A NPN BF=150 IS=1E-16 BR=7.5
```

Figure 3. Schmitt Trigger Circuit (netlist)

From the netlist file, the Parser gives as output the mathematical system of equations resulted of a nodal or modified nodal analysis. This equations describe the circuit behavior in terms of voltage and currents in each node, and its jacobian as well. A Jacobian is a square matrix of composed for the differentiation of each function with respect to each variable. This system of equations and its jaconbian is used as input of the Matlab algorithm, developed by Heath Hofmann, that uses Homotopy method for solving the system of equations and then find the DC operating points of the circuit.

4.1. Parser Bugs Correction

This section will present the solutions to the problems reported in the Andria Dyess' report [4]. The known bugs are related to error in the print of the equations and jacobian. It will be described all the changes in the code of the Parser made to correct its errors.

- When a Modified Nodal Analysis is performed the output is not in the correct format. For each voltage source (not floating) it is necessary add manually the jacobian of that voltage source, that is, the jacobian of the node that the source is connected in, for each unknown current.
- This problem was figured out by modifying the *speciaPrintJac()* function, member of *Component* class, to print these missing jacobians.

- Another problem was related to the Floating sources, that are the sources not connected to the ground (or reference node). This kind of source is not common in practical circuits, but they are important in theoretical analysis.
- In the Nodal Analysis, the parser did not print the *supernode* equations for each floating voltage source and its jacobian as well.
- A new function, member of *Component* class, named *printSuperNode()*, was included to figure out the problem with the missing equations. This function prints the supernodal equation for each floating voltage source in the circuit.
- One more problem that was figured out was about errors in the jacobians print for modified nodal equations. Sometimes, in the case of floating sources, the jacobians were -1 or 1 when it is suppose to be 0.

In additional, two functions that print the list of components and nodes of the circuit with their respective connections were created to make easier the maintenance of the code. And else, at the appendices of this report there is a brief tutorial about how to run the parser in the MS Windows command prompt.

The documentation of the Parser code has to be updated to include the description of the new member functions.

5. Conclusion

In the project it was possible to fix the most of the problems found in the parser. The problems are related to the print of the Nodal equations and Modified Nodal equations and their respective jacobians in specifics circuit configuration. Problems related with the equation of super node in case of floating voltage source was figured out. Some problems could not be figured out at the time of this report. The number of some equations in the modified nodal analysis are still wrong and their jacobians as well, in the case of floating voltage source. Therefore, more work have to be done to achieve a parser version that gives all the Nodal and Modified Nodal equations and the Jacobians in the correct format acceptable for the Matlab Homotopy algorithm.

References

[1] *Matlab Language Reference*. Mathworks Inc., 1996.

[2] E. Chan. Documentation on the parser program. Technical report, UC Berkeley, 1996.

[3] A. Dyes, E. Chan, H. Hofmann, W. Horia, and L. Trajkovic. Simple implementations of homotopy algorithms for finding dc solutions of nonlinear circuits. In *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, volume 6, pages 290–293 vol.6, Jul 1999.

[4] A. Dyess. Finding dc operating points of chua's circuit using homotopy methods. Technical report, University of Alabama, 1997.

[5] K. L. Judd. *Numerical Methods in Economics*. Massachusett Institute of Technology, 1998.

[6] W. H. Press, S. A. Teulkolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. , 1988.

[7] L. Trajkovic. Dc operating points of transistor circuits. *Nonlinear Theory and Its Applications, IEICE*, 2012.

Appendices

How to compile and run the parser on MS WINDOWS Command Prompt

1. Install the g++ compiler if you do not have yet. Normally it is installed in the directory C:\MinGW.
2. Open the Command Prompt of Windows (console).
3. Verify if the directory of g++ Compiler is in the PATH environmental variable. Use the command below:

   ```
   > echo  %PATH%
   ```

   If the Compiler directory is not the list showed continue to step 4. If the Compiler directory (C:\MinGW\bin) is showed, go to step 5.

   To get a list of all environment variables enter the command 'set'.

4. Add the Compiler directory to the PATH environmental variable ; Insert the command below:

   ```
   > set  PATH=%PATH%;C:\MinGW\bin
   ```

   This command takes the current path and sets PATH to it and adds C:\MinGW\bin to the PATH.
5. To compile the parser.cc (c++ file) use the command below in the directory where is the 'parser.cc' file:

   ```
   > g++  parser.cc  -o  parser
   ```

   '-o parser' set the name of the executable file to parser.exe

6. To run the parser the input file have be in the same directory of the 'parser.exe'. Then in this directory enter with the command below in the Console:

   ```
   > parser  -f  FILENAME -d datum -o OUTPUTFILENAME
   ```

   '-d datum' define the reference node. It is a optional argument, but it is recommend you set it