2017

Simulation software similar to SPICE

Saba Fatima

ABSTRACT

SIMULATION SOFTWARE SIMILAR TO SPICE

Saba Fatima, M.S.

Department of Electrical Engineering

Northern Illinois University, 2017

Reza Hashemian, Director

A simulation program is generated in this thesis. The results are shown to match with those obtained through circuit simulation (SPICE). The program, written in C#.NET programming language, involves circuit analysis using nodal method, transient and AC analysis, and non-linearity of BJT and MOSFET, also involving the design and development of Graphical User Interface (GUI) made user friendly.

NORTHERN ILLINOIS UNIVERSITY

DEKALB, ILLINOIS

MAY 2017

SIMULATION SOFTWARE SIMILAR TO SPICE

BY

SABA FATIMA

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

Thesis Director:

Dr Reza Hashemian

ACKNOWLEDGEMENTS

At the very outset, I would like to express my sincere gratitude to my advisor, Dr. Reza Hashemian, for his continuous support, patience, motivation, enthusiasm and immense knowledge. I could not have imagined having a better advisor and mentor for my MS study. Theoretical work for the DC, transient and AC analysis has been done by Dr. Hashemian.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Donald Zinger, the department chair, and Dr. Lichuan Liu for their continuous support. I would like to express my special thanks and deepest gratitude to my parents, Syed KhaderMohiuddinand Habeeba Begum for their unconditional love and support.

DEDICATION

This thesis is dedicated to the Almighty!

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

There are several software products out there in the market which are being used for circuit simulations. The original SPICE program, SPICE1, was developed at Universityof California, Berkeley, and released for public use in May 1972. By 1975, after the next major release, called SPICE2, SPICE was in widespread use and adopted by most integrated circuit manufacturers. Most of the circuit simulation software is developed using the SPICE engine. Their main focus would be SPICE specific schematic entry, which is the ability to define and save number of analyses, and integrated graphing of simulation results. They might allow the inclusion of SPICE/PSPICE models from a user-expandable library. The focus is on analog circuit analysis and design at the component level and some of them may be dealing with mixed signals. They would help system-level interfaces to be tested with actual electrical designs emulating real-world applications. The ongoing research within developing these software applications mostly focuses on

- Improving the simulation times, reliability, and convergence on larger designs
- Improving the speed without loss of accuracy via integrated analog and event-driven digital simulations
- Exploring the circuit behavior using basic DC, AC, noise and transient analyses including various environmental variables that affect the behavior of the circuit
- Adding more libraries of analog and mixed-signal models and also devices dealing with short channels

- Exploring design relationships with “what if” scenarios before committing to hardware
- Identifying and simulating functional blocks of complex circuitry using mathematical expressions, functions, and behavioral devices[1]

My thesis work concentrates on designing simulation software similar to SPICE. This software would be open to researchers to improvise it and add new functionalities such as the use of fixator-norator Pairs (FNPs). The FNPs concept is not extensively used in this field at present. Dealing with nonlinear circuits with FNPs is very much the same as in a regular circuit analysis, except it may become much smoother and the convergence can be reached faster. The reason for this is because the design specs provide clues as to what to expect from the circuit in terms of the regions where nonlinear devices operate.

1.1 Necessity

There is always a necessity to optimize the existing software to make some aspect of it work more efficiently or use fewer resources.

1.2 DC Analysis

1.2.1 Circuit Analysis Using Nodal Method

A typical analog circuit design starts with a topology, circuit components, and a series of analysis/simulations that tracks the circuit performance. In circuit analysis we are often interested in finding out the voltages and currents at various locations in a circuit when sources are applied[2].

1.2.2 Nodal Analysis

The first step of node analysis is to count the number of nodes of the circuit. First, we pick arbitrarily a reference node. This reference node is usually called the datum node. Node analysis of a circuit with (n + 1) nodes is based on writing n KCL equations at n nodes in terms of a set of n node-pair voltages. From an analytical point of view we may describe the oriented graph of a network by listing all branches and nodes and indicating which branch is entering and leaving which node. This is conveniently done by writing down a matrix. Suppose that the oriented graph is made up of b branches and n nodes. Suppose also that we number arbitrarily all the branches and all the nodes of this graph. We call the node-to-branch incidence matrix Aa, a rectangular matrix of n rows and b columns whose (i, k)th element aik is defined by

a_{ik} = 1 if branch k leaves node (i)

a_{ik} = -1 if branch k enters node (i)

a_{ik} = 0 if branch k is not incident with node(i)

Obviously, A is obtainable from Aa by deleting the row corresponding to the datum node. Ais therefore called the reduced incidence matrix.

Conductance matrix G is obtained by multiplying (Reduced incidence matrix)*(Element conductance matrix)*(Transpose of reduced incidence matrix)

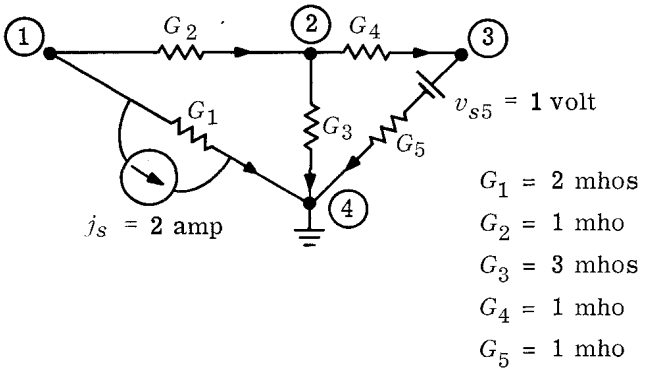
$$G = A\,G_e A^{\mathsf{T}}$$

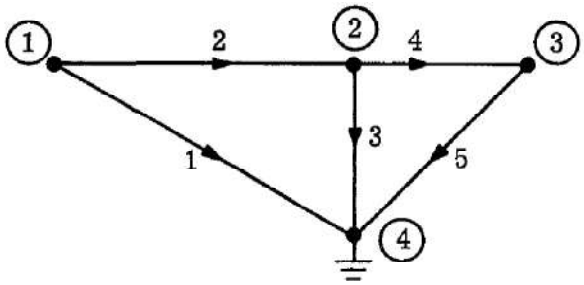
where G_e is the element conductance matrix which is a diagonal matrix with all the conductance of the circuit in its diagonal.

Involve active elements in the matrix by adding the conductance of active elements toG_e. The node voltages can be calculated by using the equation

$$e = G^{-1} I\ ,$$

where G is the final conductance matrix[2]. For example, consider the following network and its oriented graph:





The incidence matrix is given by

$$\mathbf{A} = \begin{matrix} \\ \\ \\ \\ \text{Branch} \end{matrix} \begin{matrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \\ \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \\ 1 \quad 2 \quad 3 \quad 4 \quad 5 \end{matrix} \begin{matrix} \text{Node} \\ \leftarrow ① \\ \leftarrow ② \\ \leftarrow ③ \end{matrix}$$

Its transpose is given by

$$\mathbf{A}^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Branch ← 1, ← 2, ← 3, ← 4, ← 5

Node ① ② ③

The $G_e$matrix of the above network is given by

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

By multiplying the above three matrices ($A\ G_e A^T$) , we get the final conductance matrix G.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Now invert the above matrix G and multiply it with the current vector to get the node voltages.

$$\frac{1}{25}\begin{bmatrix}9 & 2 & 1\\ 2 & 6 & 3\\ 1 & 3 & 14\end{bmatrix}\begin{bmatrix}-2\\ 0\\ 1\end{bmatrix}=\frac{1}{25}\begin{bmatrix}-17\\ -1\\ 12\end{bmatrix}$$

Branch voltages are obtained from the node voltages by the equation

$v = A^T e$

where A^T is the transpose of the reduces incidence matrix and e is the node voltage matrix.

$$\mathbf{v} = \mathbf{A}^T\mathbf{e} = \begin{bmatrix}1 & 0 & 0\\ 1 & -1 & 0\\ 0 & 1 & 0\\ 0 & 1 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix}$$

1.3 Frequency Domain Analysis

The analysis which uses directly the transformed variables is often referred to as analysis in the "frequency domain" in contrast to the analysis with differential equations, which is called analysis in the "time domain". In this thesis, the frequency response is achieved by using the symbolic analysis of analog circuits[2].

1.3.1 Symbolic Analysis of Analog Circuit

- For a resistive network N with (n+1) nodes and m capacitors, node admittance matrix, Y is constructed with n rows and n columns where each entry of Y is a vector of (m+1) dimension.

- Initially each entry of Y matrix can only have two nonzero entries:$y_{ij},0 = g_{ij}$ representing the conductance term and $y_{ij},1 = c_{ij}$ representing the capacitance term.
- Now the expression we have is of type : Y*V(s) = I*I(s), where V(s) is the output voltage, I(s) is the input current and I is identity matrix of order *n* by *n*.
- Apply LU decomposition method to the matrices Y and I using convolution of vectors in the Y matrix. The Y matrix is made upper matrix and the I matrix is made lower matrix at the end of LU factorization.
- Now, suppose the input current signal I(s) is applied to node 1 (i = 0) and the output voltage V(s) is received from the last node m (i = m-1). Then the transfer function V(s)/I(s) is obtained as

$$T(s) = \frac{V(s)}{I(s)} = \frac{N(s)}{D(s)} = \frac{u_{(m-1)(m-1)}(s)}{l_{00}(s)}$$

where, L = $\{l_{ij}\}$ and U = $\{u_{ij}\}$, for i, j = 0, 1, … m-1.

- $l_{ij}(s)$ and $u_{ij}(s)$ are functions of s so that

$$k_{ij}(s) = \sum_{j=0}^{j=m-1} k_{ij,h} s^h$$

where, $k_{ij}(s)$ is either $l_{ij}(s)$ or $u_{ij}(s)$, and $k_{ij,h}$ is the h+1 entry of vector k_{ij}.

- The convolution of two m-dimensional vectors A = $\{a_i\}$ and B = $\{b_i\}$ for i = 0, 1, …, m-1 are represented by C = $\{c_i\}$ for i = 0, 1, … 2m-2 so that

$$c_i = \sum_{j=0}^{j=m-1} a_j b_{i-j}$$

- The roots of the polynomial in D(s) give the poles and the roots of polynomial in N(s) give us the zeros of the network[3].

1.4 Time Domain Analysis

Since transient analysis is dependent on time, it uses different analysis algorithms, control options with different convergence-related issues and different initialization parameters than DC analysis.

1.4.1 Procedure

- It is important that the admittance matrix is fixed during time. Then replace each capacitance by a conductance G_c =(2*C)/delta. Each capacitor port has a current source, $I_c = i_c(t) + G_c v_c(t)$.
- Change the node admittance matrix to the branch admittance matrix.
- Next, add G_c = (2*C)/delta to each capacitor branch. This is the final branch admittance matrix G.
- There are three branch current sources :
  a) The fixed (DC) current sources I
  b) Variable (sin) current sources J
  c) The capacitor current sources I_c
- I is fixed. J is calculated by the following equation:

  $J_i = A_i \sin(2*pi*i/m)$ for i =0 to (m*d)-1, where d is the number of cycles, m is slices per cycle, and G_c = (2*C)/delta, where delta = 1/(frequency * m) (or) G_{ci} = $2*c_i*m*frequency$.
- Finally, the capacitor branch current is calculated as

$I_c(t) = 2*G_c*v(t) – I_c(t\text{-delta})$. This is done (m*d) number of times and displayed either as branch voltages or node voltages.

- In each iteration the capacitor currents $i_c(t)$ is calculated.

  $i_c(t+delta) = (G_c*v(t+delta)) – I_c(t)$ [3]

CHAPTER 2: LITERATURE REVIEW

SPICE (Simulation Program with Integrated Circuit Emphasis) is a general-purpose, open-source analog electronic circuit simulator. It is a program used in integrated circuit and board-level design to check the integrity of circuit designs and to predict circuit behavior. It is a computer program that accepts acircuit schematic as input and outputs the simulated circuit behaviors. The simulationcan be performed under the nonlinear DC, nonlinear transient and linearized AC operating conditions[4].

2.1 Introduction

Unlike board-level designs composed of discrete parts, it is not practical to breadboard integrated circuits before manufacture. Further, the high costs of photolithographic masks and other manufacturing prerequisites make it essential to design the circuit to be as close to perfect as possible before the integrated circuit is first built. Simulating the circuit with SPICE is the industry-standard way to verify circuit operation at the transistor level before committing to manufacturing an integrated circuit.

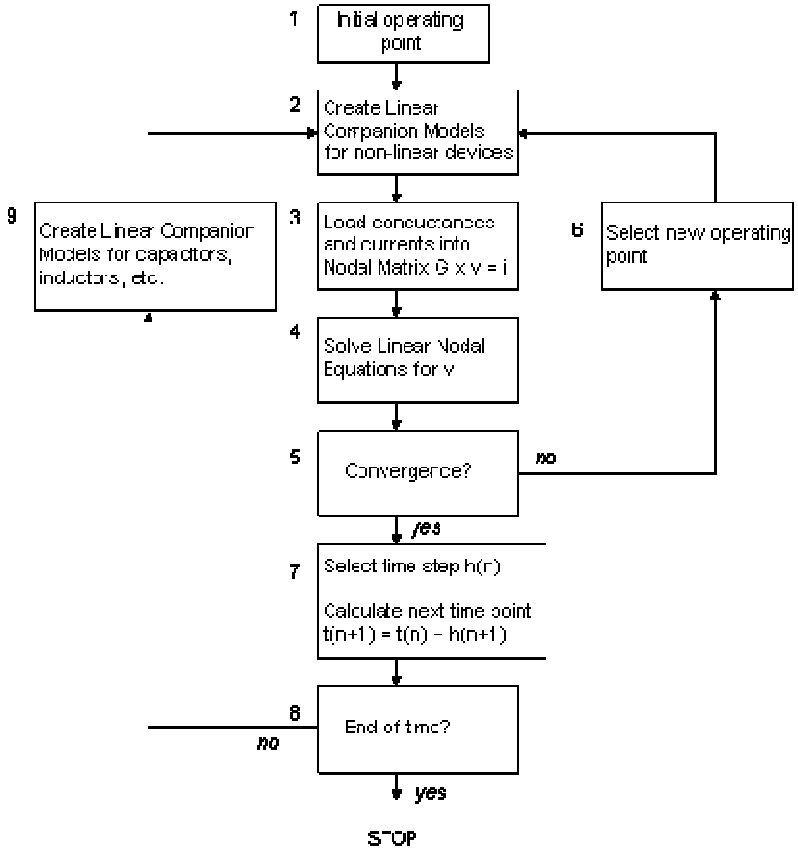
Board-level circuit designs can often be breadboarded for testing. Even with a breadboard, some circuit properties may not be accurate compared to the final printed wiring board, such as parasitic resistances and capacitances. These parasitic components can often be estimated more accurately using SPICE simulation. Also, designers may want more information about the circuit than is available from a single mock-up. For instance, circuit performance is affected by component manufacturing tolerances. In these cases it is common to use SPICE to

perform Monte Carlo simulations of the effect of component variations on performance, a task which is impractical using calculations by hand for a circuit of any appreciable complexity.

Circuit simulation programs, of which SPICE and derivatives are the most prominent, take a text netlist describing the circuit elements (transistors, resistors, capacitors, etc.) and their connections and translate this description into equations to be solved. The general equations produced are nonlineardifferential algebraic equations which are solved using implicit integration methods, Newton's method and sparse matrix techniques[4].

2.2 SPICE Simulator

Here's a simplified block diagram of the main SPICE program flow[1].



There are three key points about the algorithm:

Heart of the SPICE is **nodal analysis** (blocks 3 and 4) accomplished by formulating the nodal matrix and solving the nodal equations for the circuit voltages.

The inner loop (2 - 6) finds the solution for **nonlinear** circuits. Nonlinear devices are replaced by equivalent linear models. It may take much iteration before the calculations converge to a solution.

The outer loop (7 - 9) together with the inner loop performs a **transient analysis,** creating equivalent linear models for energy storage components for capacitors, inductors, etc., and selecting the best time points. Since transient analysis is dependent on time, it uses different analysis algorithms, control options with different convergence-related issues and different initialization parameters than DC analysis. However, since a transient analysis first performs a DC operating point analysis (unless the UIC option is specified in the .TRAN statement), most of the DC analysis algorithms, control options, and initialization and convergence issues apply to transient analysis.

2.3Algorithm in action

Overall, SPICE can do these following functions:

- DC Analysis
- Transient Analysis (Time Response)
- AC Analysis (Frequency Response)

Apart from these, SPICE also serves up some extended analysis like sensitivity, Fourier, noise, etc. [1].

2.3.1 DC Analysis

DC analysis is performed either to calculate a DC operating point or a DC sweep. An operating point is required for the initial solution to a transient analysis or as a bias point for an AC analysis. The DC sweep does exactly as the name implies - performs the DC analysis multiple times as you sweep a selected component parameter across a defined range. All energy storage components like capacitors, inductors and semiconductor charge mechanisms are ignored for this analysis.

One of the simpler tasks for SPICE is performing DC analysis on linear circuits. Only two of the blocks are really needed: load the nodal matrix (3) and solve the nodal equations (4) using Gaussian elimination[1].

2.3.2 Transient Analysis

SPICE completes the outer loop only to perform a transient analysis for linear circuits, ignoring blocks 2 and 5. After the initial operating point is found, the energy-storage components (capacitors, inductors, semiconductor junctions) are transformed into linear companion models well suited for the nodal equation solver. SPICE completes the nonlinear loop (2-6) at each time point of the transient analysis.

SPICE dynamically adjusts the time step, h(n), for two reasons - to improve accuracy and reduce long simulation times. The time step will be reduced when circuit voltages and currents are changing rapidly. Under these conditions, accuracy generally improves with a smaller time step.

On the other hand, it increases the time step to avoid a long simulation if nothing much is happening dynamically in a circuit[1].

2.3.3 AC Analysis:

The AC analysis happens in two parts.

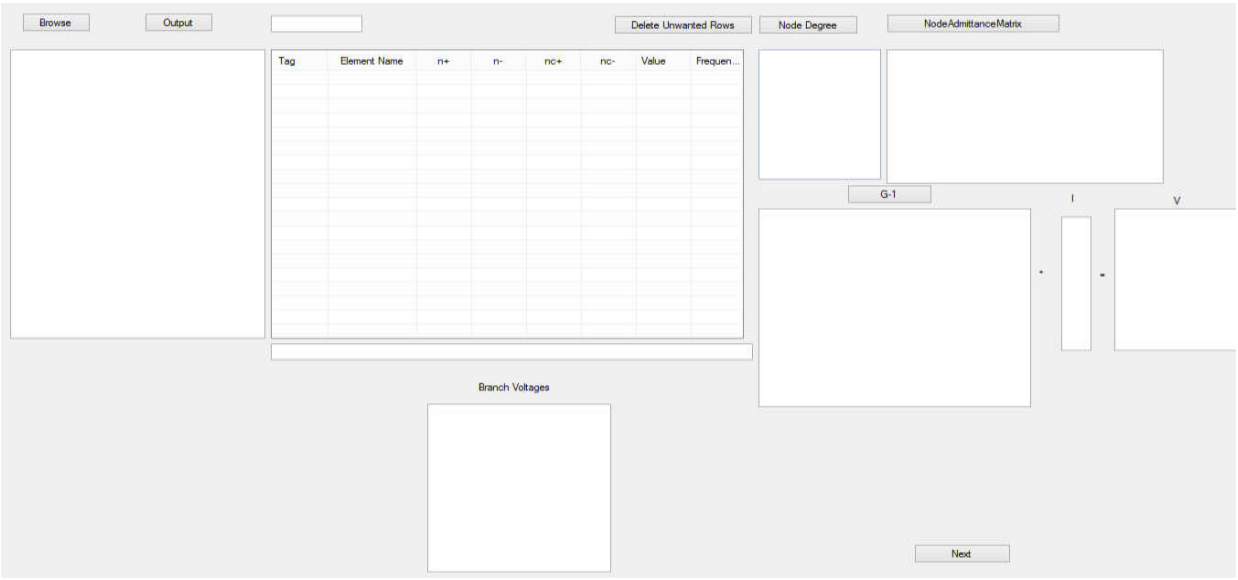
Part 1 SPICE performs the nonlinear circuit loop (1-6) to find the DC operating point. This loop ignores all capacitors, inductors and other energy storage components.

Part 2 This part performs blocks 3 and 4, but with slightly modified versions. SPICE loads the nodal matrix with the real and imaginary parts of impedances for the resistors, capacitors and inductors. For components like semiconductors, SPICE replaces nonlinear behaviors by their linearized small signal equivalents. Finally, power supplies are set to zero, the signal source is set to unity, and blocks 3 and 4 are repeated for all requested frequencies. Nodal analysis calculates the result in terms of magnitude and phase for each node voltage[1].

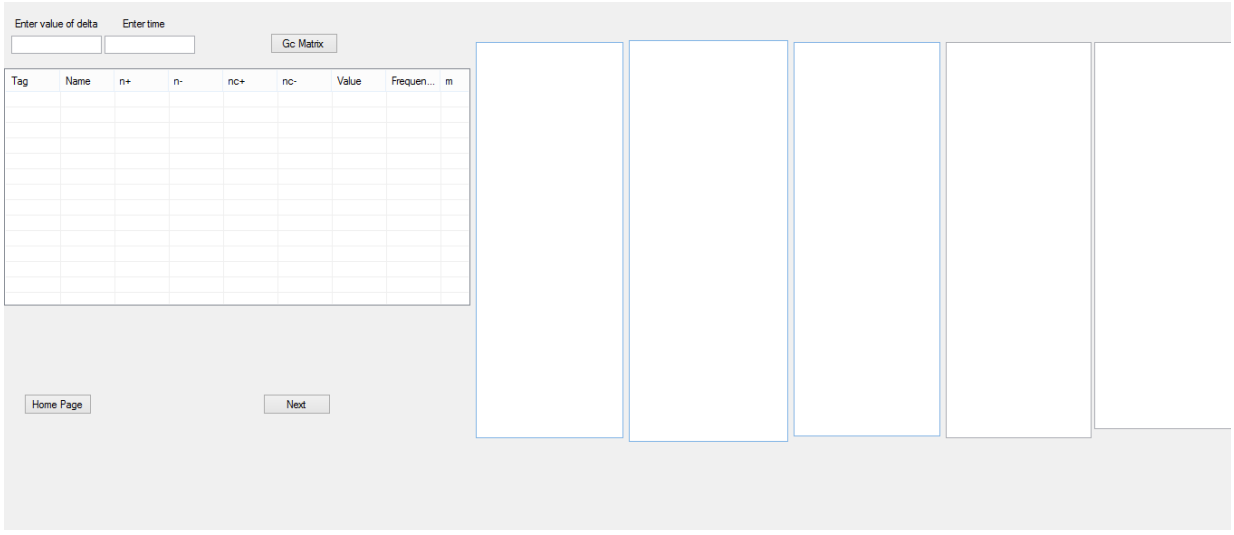
CHAPTER 3: APPLICATION DEVELOPMENT

3.1 GUI

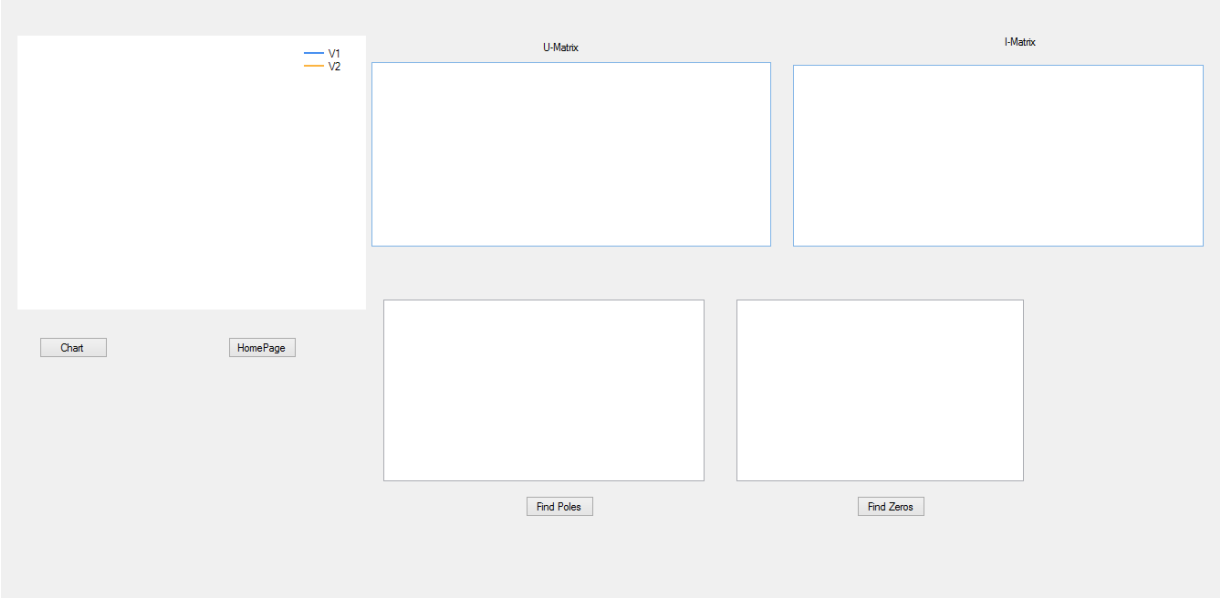
Page1



Page2



Page3



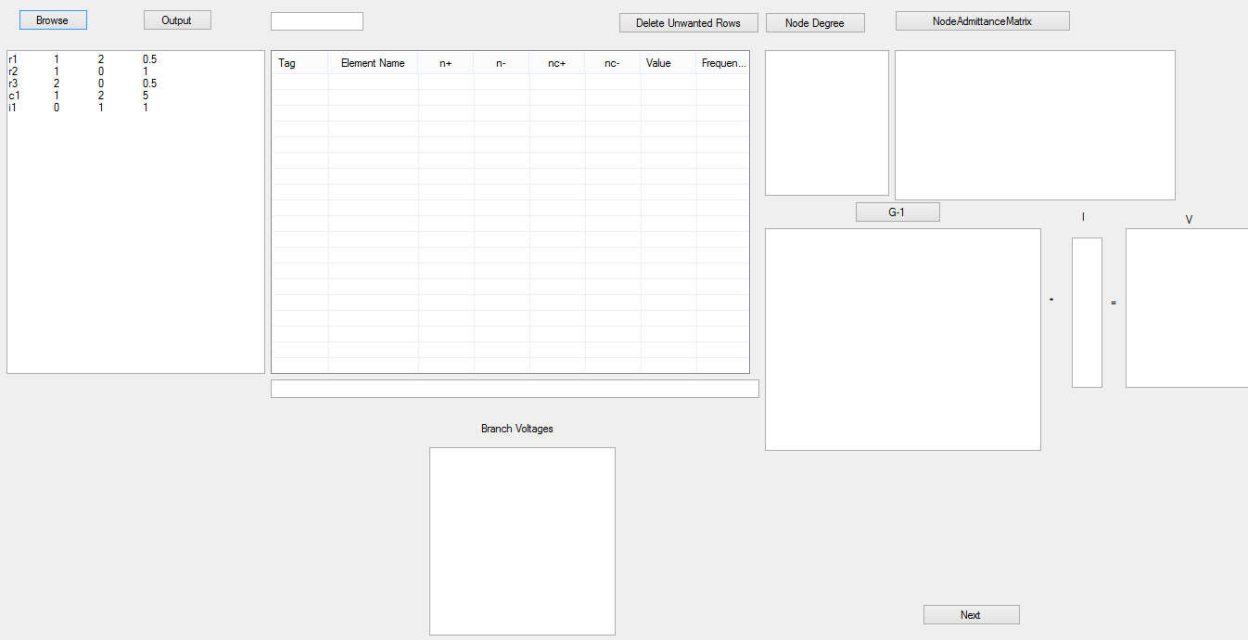
The software starts with browsing a .cir file and displaying the contents of the file in the textbox below the ‘browse’ button.

3.1.1 Example 1

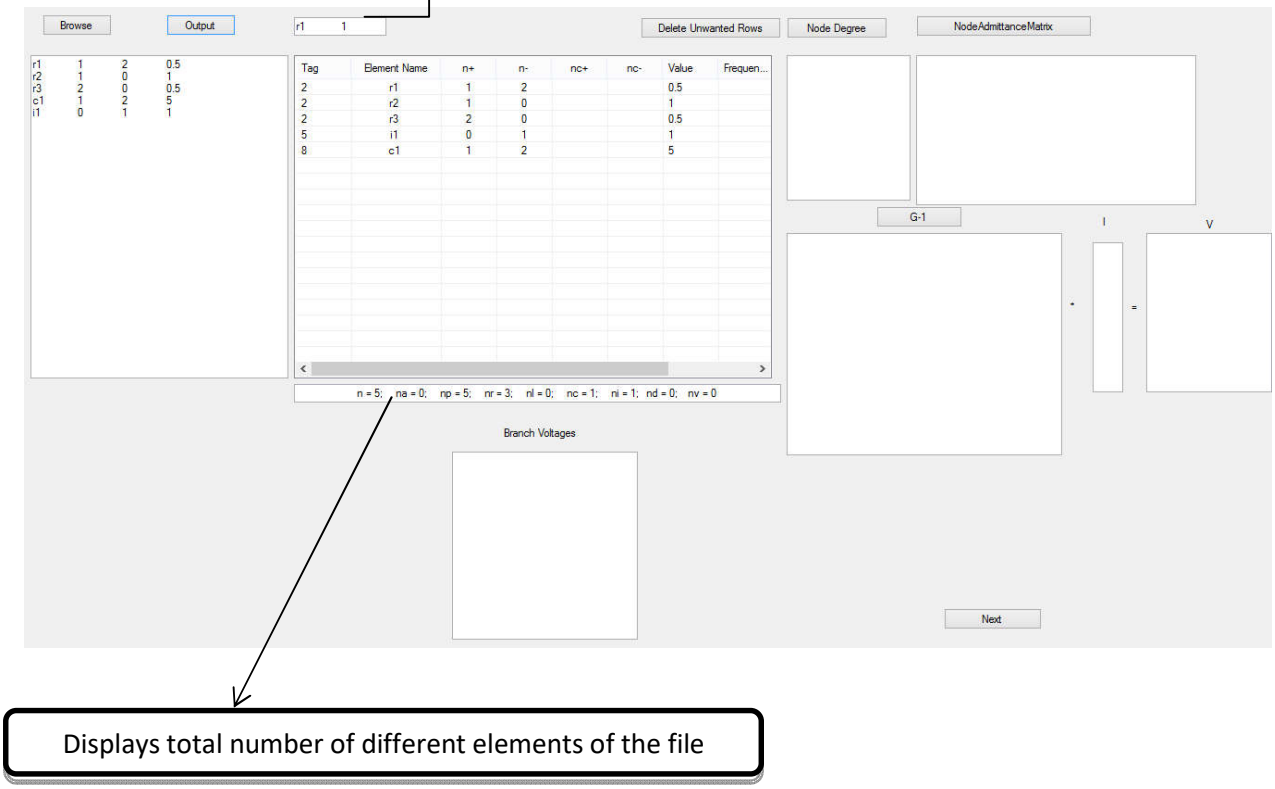
Consider cr1.cir file

r1	1	2	0.5			
r2	1	0	1			
r3	2	0	0.5			
c1	1	2	5			
i1	0	1	1			
j1	0	2	3	10	20	2

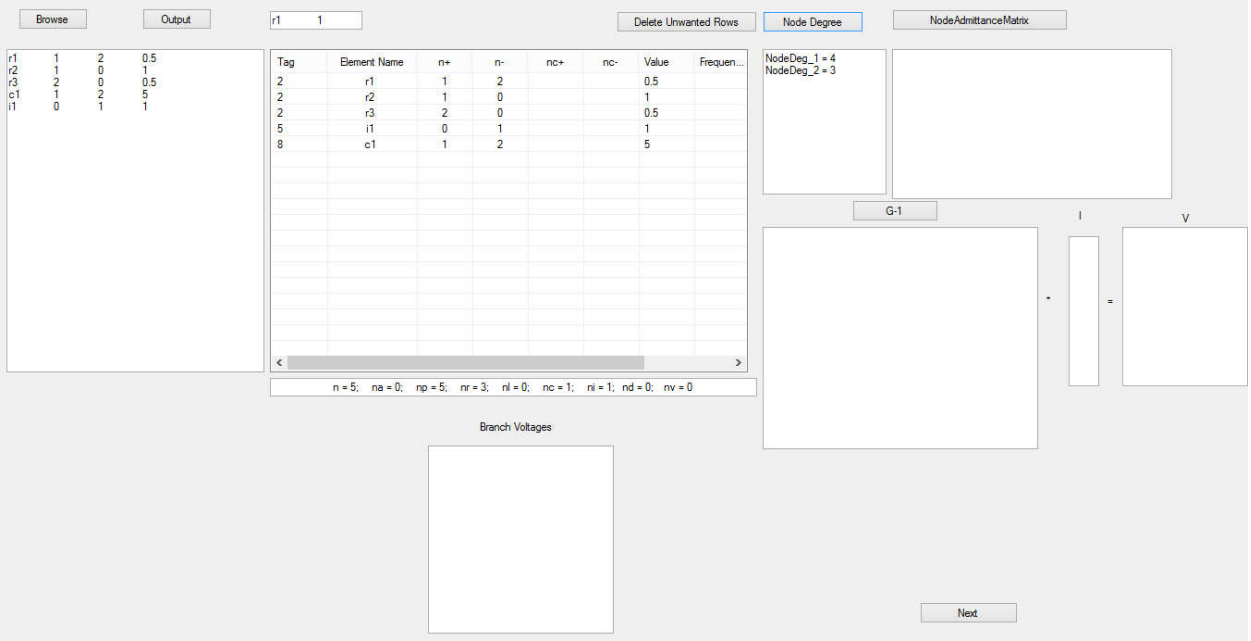
DC Analysis:



The output button separates the elements names from the nodes and value of the element and displays “not valid” for the elements that are not in proper format. This helps the user in locating and correcting any errors in the file.



'Node Degree' button displays in the textbox below it the degree of all the nodes involved in the circuit. The degree of a node is number of edges connected to a node.



Reduced incidence matrix A for the above circuit is constructed.

$$\begin{vmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{vmatrix}$$

Next, find the transpose of the reduced incidence matrix.

$$A^T = \begin{vmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Element conductance matrix, $G_{e,}$ is given by

$$\begin{vmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{vmatrix}$$

Conductance matrix is formed by multiplying the reduced incidence matrix to the product of element conductance matrix and transpose of reduced incidence matrix.

$$G = AG_eA^T$$

$$G = \begin{vmatrix} 3 & -2 \\ -2 & 4 \end{vmatrix}$$

There are no active elements in this circuit. Hence the active element matrix is all zeros. Thus, the node admittance matrix, *Y*, is then formed by adding the conductance matrix to the active element matrix. In this case it is just the conductance matrix.

Now, invert the node admittance matrix as shown in textbox below 'G^{-1}' label.

$$G^{-1} = \begin{vmatrix} 0.5 & 0.25 \\ 0.25 & 0.375 \end{vmatrix}$$

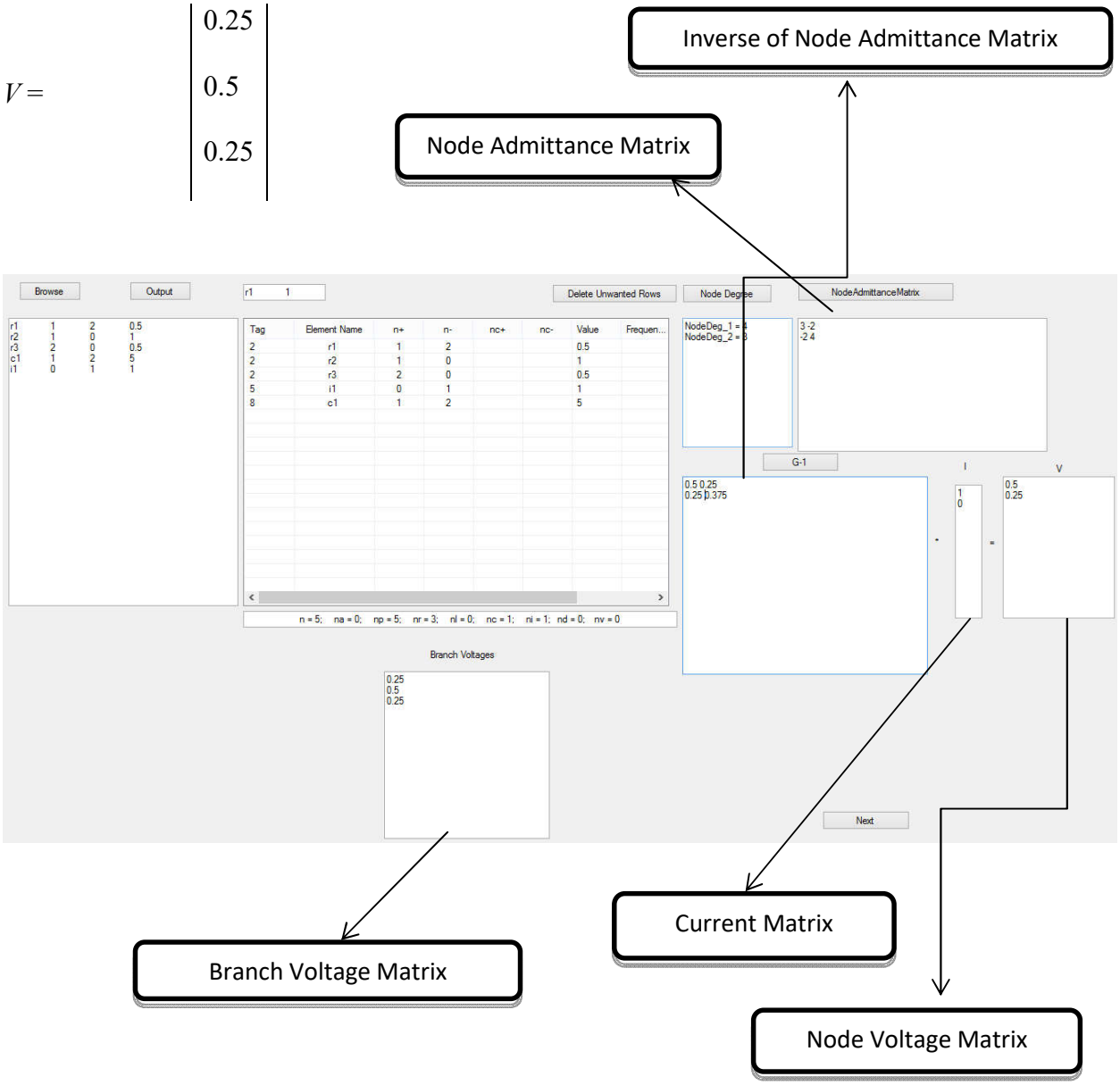
The current matrix *I* for this circuit would be $\begin{vmatrix} 1 \\ 0 \end{vmatrix}$

The node voltage matrix *e* is obtained by multiplying the inverse of node admittance matrix to the current matrix.

$$e = \begin{vmatrix} 0.5 \\ 0.25 \end{vmatrix}$$

To find the branch voltages, multiply the transpose of reduced incidence matrix to the node voltage matrix. Branch voltage matrix V is

$$V = \begin{vmatrix} 0.25 \\ 0.5 \\ 0.25 \end{vmatrix}$$



Compare to SPICE:

Node Voltages:

Values obtained from this program:	Values obtained from SPICE:
0.5 0.25	v(1) = 5.000000e-01 v(2) = 2.500000e-01

Branch Voltages:

Values obtained from this program:	Values obtained from SPICE:
0.25 0.5 0.25	(v(1)-v(2)) = 2.5000 00e-01 v(1) = 5.000000e-01 v(2) = 2.500000e-01

The node voltages and branch voltages from this program match with those obtained from SPICE.In DC analysis, the capacitors are short-circuited.

Frequency Domain Analysis:

Let us now discuss the frequency domain analysis, where the capacitances are added to the node admittance matrix, forming vectors. Each element of the node admittance matrix *Y* is an array of size *(m+1)* where *m* is the number of capacitors in the circuit.

Initially, there are only two elements in each array – one corresponding to the previous *Y* matrix and the other is the value of capacitance. The rest of all elements of the array are zeros.

The equation is *Y*V(s) = I*I(s)*, where *V(s)* corresponds to the output voltage, *I(s)* is the input current signal and *I* is identity matrix of order *n* by *n,* where *n* is the number of nodes in the circuit, excluding the datum node.

LU decomposition is applied to the *Y* matrix and corresponding operations are done to the identity matrix, resulting in the upper *Y* and lower *I* matrix. This factorization is done using the convolution of arrays in the node admittance matrix. The elements below the diagonal of *Y* matrix are made zero, thus called the upper decomposition. Correspondingly, the elements below the diagonal of *I* matrix are non-zero, leaving the elements above the diagonal to be zeros. This is called lower decomposition.

For the example cr1.cir, the upper matrix formed is

(3,5)	(-2,-5)
(0,0)	(8,15)

The corresponding lower matrix formed is

(1,0)	(0,0)
(2,5)	(3,5)

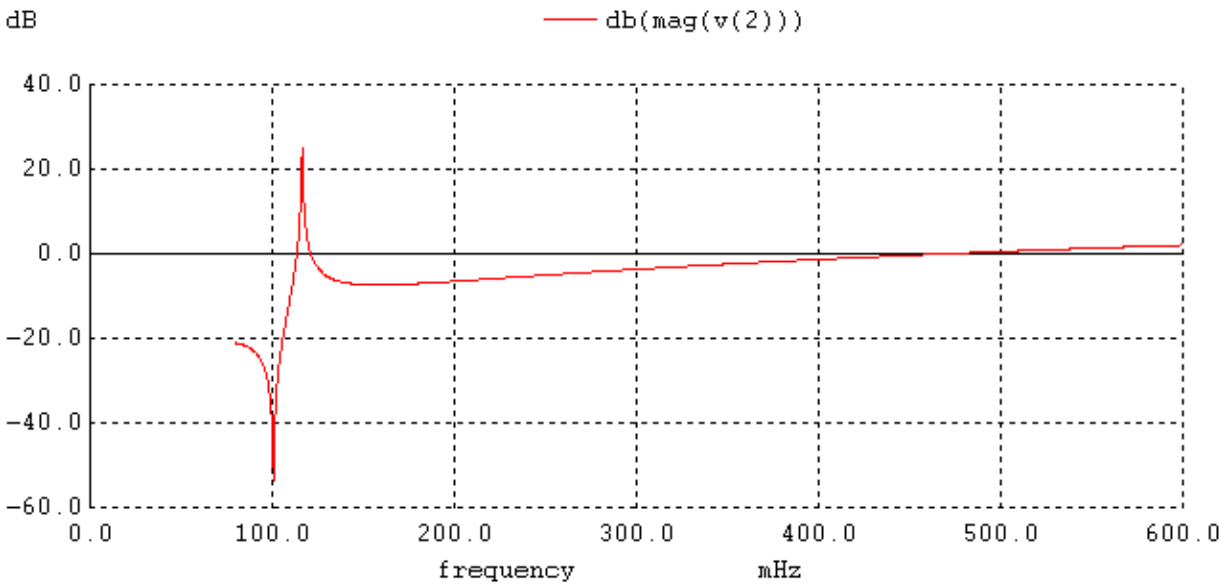
Poles and Zeros:

Poles are calculated by finding the roots of the polynomial formed by the array in upper matrix, and zeros are calculated by the polynomial in the lower matrix. Roots of the polynomials are found using the Jenkins-Traub algorithm. These values of poles and zeros are in ω. Divide it by (2π) to get the frequency in Hz.

Compareto SPICE:

Finding Poles and Zeros in SPICE

To locate poles and zeros, we use the bode plot. For the cr1.cir file, the bode plot looks something like this:



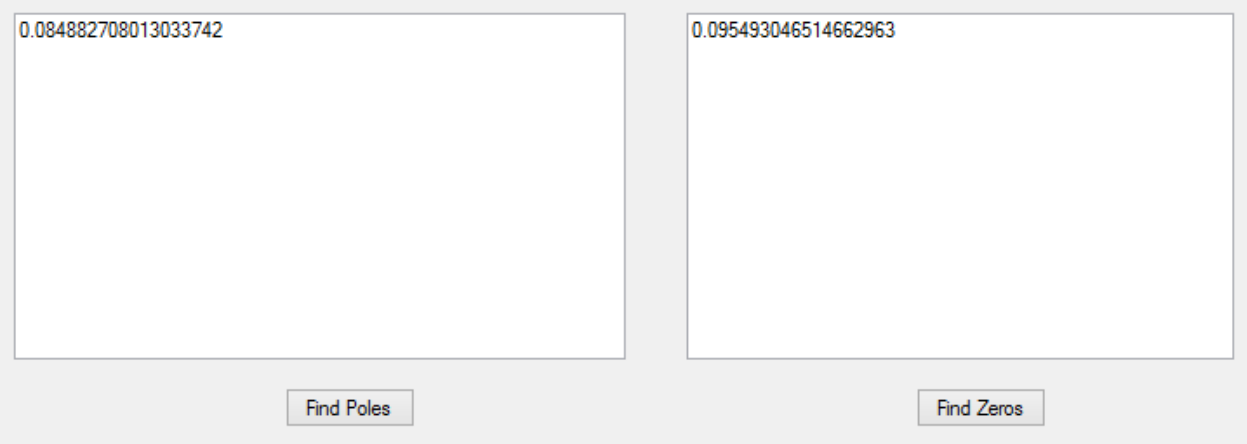
There is one pole and one zero.

Pole:

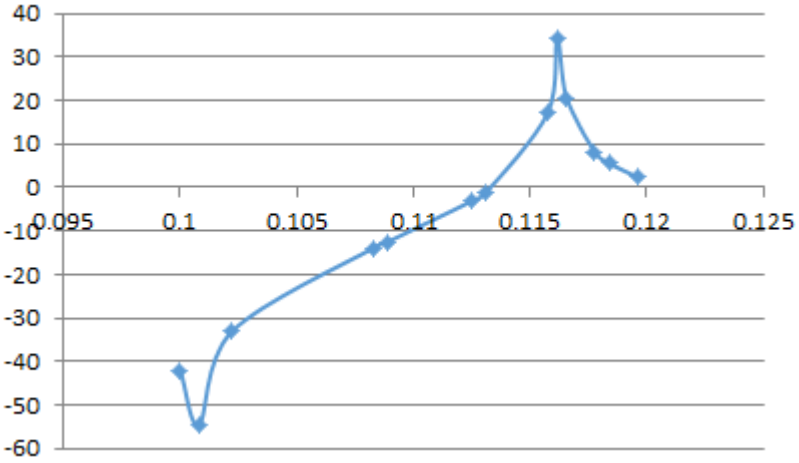
The pole is approximately at f_L = 116mHz. Square the value of f_L and multiply it by 2π to get the value of f_R.$f_L$is for inductances and f_R is for resistances. Here, f_R = 0.08454Hz.

Zero:

The zero is at f_L = 100.9mHz. This implies f_R = 0.0954Hz. Compare these value to the values of poles and zeros obtained from the program.



There is a single pole and a single zero with frequencies approximately the same as that obtained through SPICE. The magnitude plot obtained is



Time Domain Analysis:

For the time domain analysis, add the capacitive conductance to the node admittance matrix. This matrix is G_c; in this example there is only one capacitor. Hence the conductance added is g_c = (2*c)/delta, where c is the value of capacitance and delta = 1/(frequency*slices/cycle). For delta = 0.005:

g_c= (2*5)/0.005 = 2000. Hence $g_c$matrix is

$$\begin{vmatrix} 2000 & -2000 \\ -2000 & 2000 \end{vmatrix}$$

G_c=Y+$g_c$matrix.

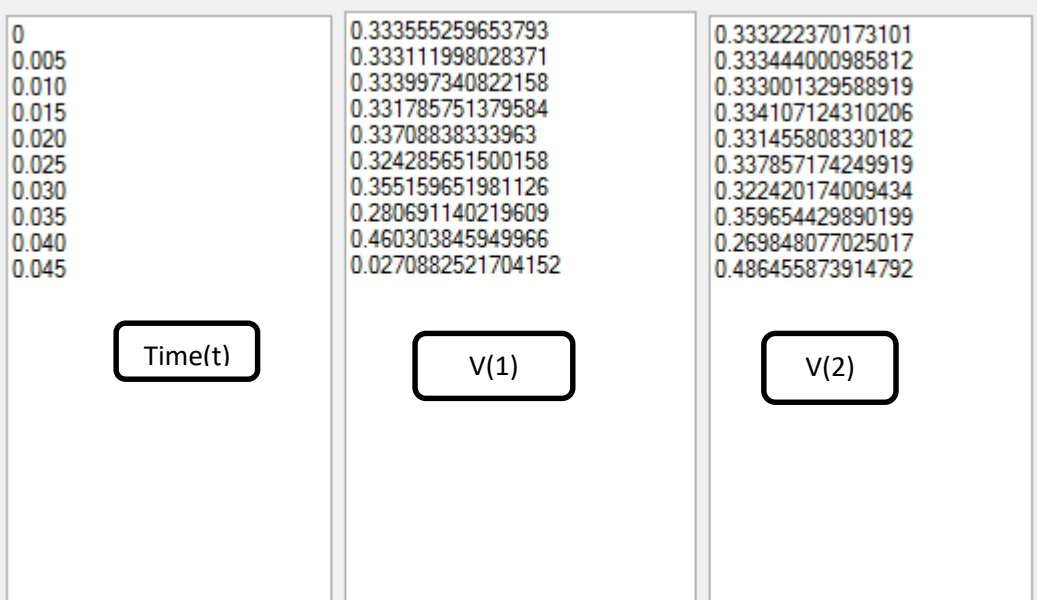
Therefore, G_c = $\begin{vmatrix} 2003 & -2002 \\ -2002 & 2004 \end{vmatrix}$

In every iteration, the capacitor current is added to the I matrix along with the variable sine current J. Initially, $I$is fixed, J is calculated by the following equation:

$J_i = A*sin(2\pi i/m)$ for i = 0 to (md-1)

$I_{ck}(t) = 2*g_{ck}*v_k(t) - I_{ck}(t\text{-}delta)$, where k = 0 to (Number of capacitors – 1) at any given time t. Add the latest $I_c(t)$ to the I matrix after every iteration. In this example, A = 3, frequency = 10, m = 20 and d = 2.

For time 0.05, the values of V(1) and V(2) are given as follows:



3.1.2 Example 2

Consider cr2.cir file

r1	1	0	2
r2	2	1	1
r3	2	0	1
r4	2	3	1
r5	0	3	1
c1	1	3	0.1
c2	2	3	0.1

c3 2 1 0.1

i1 3 1 5

j1 0 2 3 10 20 2

DC Analysis:

Node Voltages

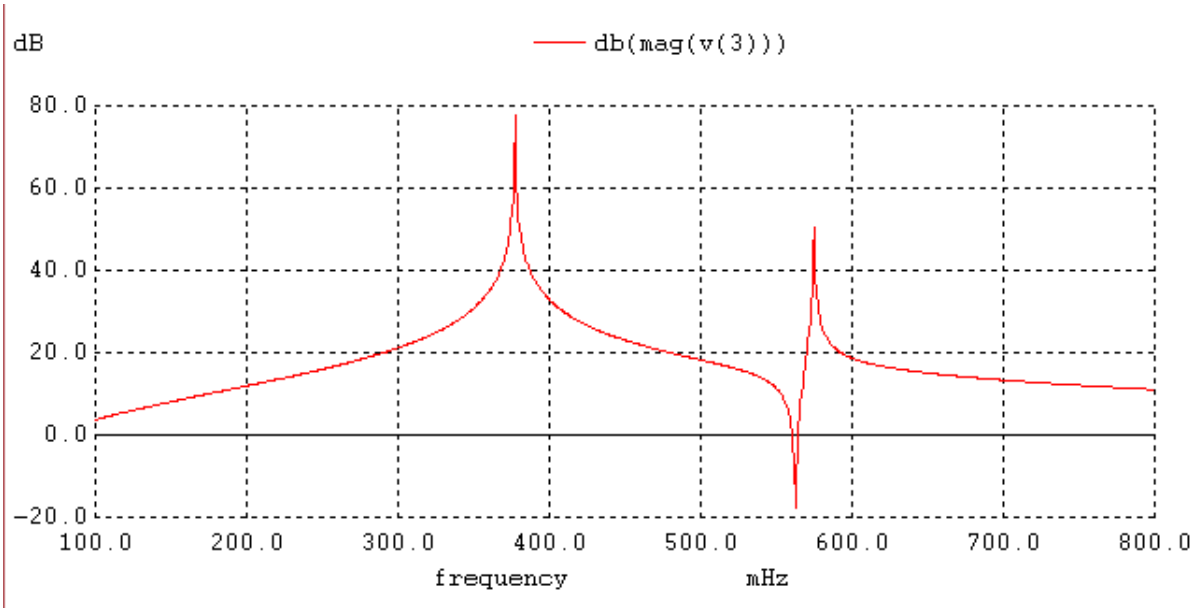
SPICE	This Program
v(1) = 3.636364e+00 v(2) = 4.545455e-01 v(3) = -2.27273e+00	V 3.63636363636364 0.454545454545454 -2.27272727272727

Branch Voltages

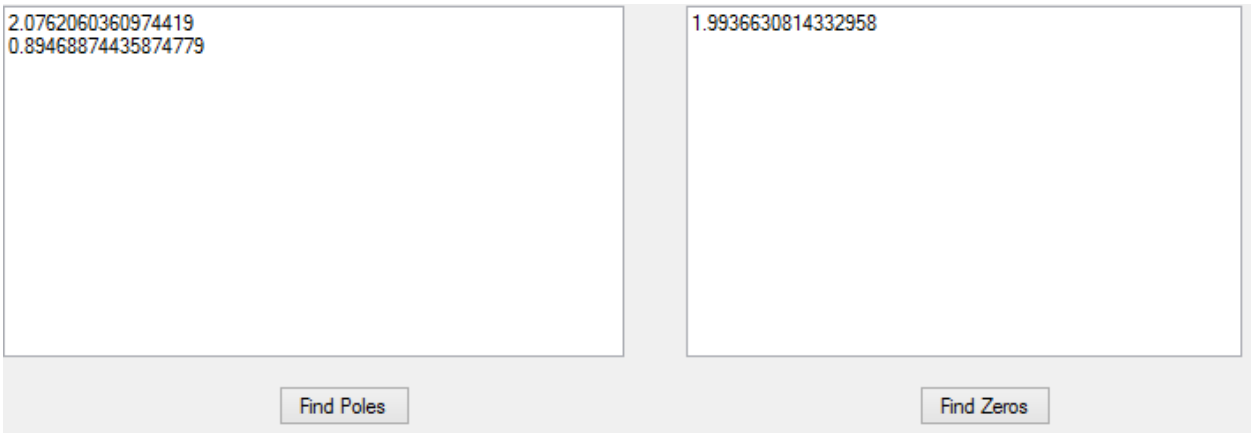
SPICE	This Program
v(1) = 3.636364e+00 v(2)-v(1) = -3.18182e+00 v(2) = 4.545455e-01 v(2)-v(3) = 2.727273e+00 v(3) = -2.27273e+00	Branch Voltages 3.63636363636364 -3.18181818181818 0.454545454545454 2.72727272727273 -2.27272727272727

Frequency Domain Analysis:

SPICE



Poles and zeros calculated through this program, after LU factorization, are



CHAPTER 4: CONCLUSION

A software application with Graphical User Interface is developed to work as a simulation program. This is similar to SPICE but is a step-by-step process and the GUI allows the user to view the simulation in detail at any stage. The program is developed in C#.Net programming language and involves the DC analysis, finding node and branch voltages; frequency domain analysis, finding poles and zeros; and time domain analysis, finding voltages and currents of network at any given time. The values obtained are shown to match with that obtained from SPICE. An example is explained and the SPICE values are verified with the program-generated values.

CHAPTER 5: FUTURE WORK

The program can be extended to nonlinear analysis. New features such as fixator-norator pairs (FNPs)[5] that can handle nonlinearity can be added to this simulation program. Dealing with nonlinear circuits with FNPs is very much the same as in a regular circuit analysis, except it may become much smoother and the convergence can be reached faster. The reason for this is because the design specs provide clues as to what to expect from the circuit in terms of the regions where nonlinear devices operate.

REFERENCES

1. Muralidhar, S., 2014, “New Approach for Direct Analog Circuit Design,” M.S. thesis, Northern Illinois University, DeKalb
2. Charles, A. D., and Ernest, S. K., 1969, *Basic Circuit Theory*, Mcgraw-Hill Book Company, New York, pp. 416-421
3. Theoritical work by Dr. Reza Hashemian
4. Wikipedia
5. Hashemian, R., “Application of Fixator-Norator Pairs in Designing Active Loads and Current Mirrors in Analog Integrated Circuits,” *IEEE Trans. Very Large Scale Int.(VLSI) Syst.,* vol. 20,no. 12, Dec 2012