People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Electronics**

Option: **Computer Engineering**

Title:

Passive networks simulation Based on Graph theory

Presented by:

- **Hamadi Amel**

Supervisor:

Mr A. ZITOUNI

Registration Number:……/2020

DEDICATION

*Great thank goes to my lovely **parents**, especially my **MOM**, to the pure sprite of my grandmother, my sisters **Abir** and **Doaa**, to my brother **abdrahmane**,*

*Also, this work is dedicated to my husband **WALID**, who for me represent everything in my life, and who has supported me all the way since the beginning of my studies. To my daughters also **SERINE** and **MERIEM** who have been a great source of motivation and inspiration.*

*I warmly wish to extend my thanks to **Mr mohammed** and **Mr Nouredine** who have supported me during the period of this work, and I'm really honored to work in theirs offices.*

Finally, this work is dedicated to all my friends to E10 students, especially the computer option's Student, to those who believe in the richness of learning.

Amel

ACKNOWLEDGEMENT

First of all we thank the God, the Almighty, for giving us the strength and the will to better carry out this work covered by the induction training report.

Our deep reconnaissance, my very large and sincere gratitude also goes to ***Mr. ZITOUNI****, through my study for its ongoing advices, his daily availability, encouragement, and kindness during the accomplishment of my project.*

AMEL

Table of contents

Dedications……I

Acknowledgement ……II

Abstract……..III

Table of content…….IV

Introduction ……1

Chapter one …….. Topological network analysis

I.1.Introduction…….2

I.2 graph theory

I.2.1 Basic concepts and Definitions…….……...3

I.2.2 terms and definitions…….....5

I.2.3relation between nodes, links and branches…….....7

I.3.topological network analysis techniques…….8

I.4 Echelon Algorithm……12

I.5 Topological analysis methods……....19

I.6.modified nodal analysis…….27

I.7.the purpose of using MNA……28

I.8.Conclusion……....29

Chapter two….....……the implemented circuit

II.1.Introduction……..31

II.2 Simulator description……..31

II.3 Net-list format used in this project……..34

II.4 MNA model used for the algorithm implementation……..34

II.6. MATLAB……36

II.7 general flowchart of the algorithm……...36

II.8 conclusion…….....48

Chapter three …….testing and results

III.1 Introduction ……49

III.2 NGspice ……..……49

III.3 Simulation of a simple RC filter…..……49

III.4 Simulation results……..…..52

III.5 Simulation of a T-Network…….53

III.6 Conclusion …….56

Conclusion…….....57

APPENDIX…….……..……..VI

References……..…...XXI

Abstract

The aim of this project is to build a passive linear networks simulator for both time and frequency analysis including mutual effects that may occur between the mutual branches of the network.

After a quick survey on the different topological network analysis techniques, we based our work on the Modified Nodal Analysis (**MNA**) technique which has been selected to implement our algorithm, The final code is written in **MATLAB** programming language and presented as Graphical User Interface (GUI). The results obtained after simulation are displayed as plots or can be saved in a disk file for further processing.

Introduction

A computer simulation, a computer model, or a computational model is a computer program, run on a single computer, or a network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modeling of many natural systems in physics (computational physics), astrophysics, chemistry and biology, human systems in economics, psychology, social science, and engineering. Simulation of a system is represented as the running of the system's model. It can be used to explore and gain new insights into new technology and to estimate the performance of systems too complex for analytical solutions. Computer simulations vary from computer programs that run a few minutes to network-based groups of computers running for hours to ongoing simulations that run for days. The scale of events being simulated by computer simulations has far exceeded anything possible (or perhaps even imaginable) using traditional paper-and-pencil mathematical modeling. Electronic circuit simulation uses mathematical models to replicate the behavior of an actual electronic device or circuit. Simulation software allows for modeling of circuit operation and is an invaluable analysis tool Due to its highly accurate modeling capability. Simulating a circuit's behavior before actually building it can greatly improve design efficiency by making faulty designs known as such, and providing insight into the behavior of electronics circuit designs.

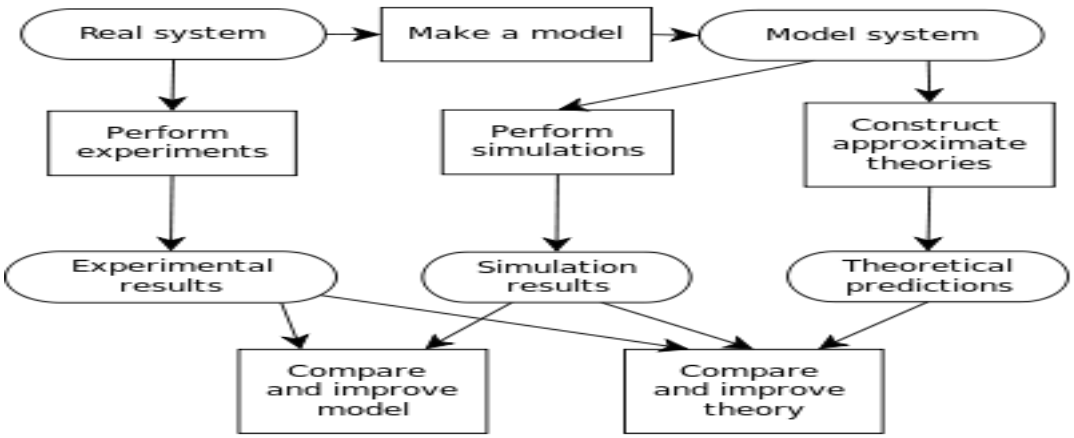


Figure i

History of simulation:

Computer simulation developed hand-in-hand with the rapid growth of the computer, following its first large-scale deployment during the Manhattan Project in World War II to model the process of nuclear detonation. It was a simulation of 12 hard spheres using a Monte Carlo algorithm. Computer simulation is often used as an adjunct to, or substitute for, modeling systems for which simple closed form analytic solutions are not possible. There are many types of computer simulations; their common feature is the attempt to generate a sample of representative scenarios for a model in which a complete enumeration of all possible states of the model would be prohibitive or impossible.

To simulate circuits on computer, I make use of a particular program called SPICE, which works by describing a circuit to the computer by means of a listing of text. In essence, this listing is a kind of computer program in itself, and must adhere to the syntactical rules of the SPICE language. The computer is then used to process, or "run," the SPICE program, which interprets the text listing describing the circuit and outputs the results of its detailed mathematical analysis, also in text form.

First, we need to have SPICE installed on our computer. As a free program, it is commonly available on the internet for download, and in formats appropriate for many different operating systems.

Project objectives and outlines

The main goal of our project is to build a passive circuit's simulator. Resistors, capacitors, inductors with mutual effect and independent sources are included to simulate, a text description (**netlist**) similar to the SPICE netlist will be used for circuit representation. The algorithm is implemented using the MATLAB programming language, based on the modified nodal analysis (**MNA**) technique used for electrical network analysis. The project report consists of 3 main chapters: chapter one is a background review about the different topological network analysis techniques, with a focus on the modified nodal analysis (MNA) technique. Chapter two deals with the algorithm description and the code implementation. Finally, in chapter three we will test our product by comparing the results obtained from simulating several circuits with those from the trustful SPICE software.

Chapter One

Topological Network Analysis

1.1 Introduction

Electrical network theory is an important and perhaps the oldest branch of electrical engineering. There are two aspects of electrical network theory, analysis and design. The behavior of a network is described by a set of equations determined by Ohm's law, KVL and KCL. Those equations depend on the structure or the graph of the network, thus the graph theory plays a key role in the study of the electrical networks. In this chapter, we will introduce some principles of the graph theory, and the different topological network analysis techniques derived from the results obtained from the graph theory, with a focus on the Modified Nodal Analysis technique.

1.2 Graph Theory

In mathematics and computer science, graph theory is the study of graphs, which are mathematical structures used to model pair-wise relations between objects. A "graph" in this context is made up of "vertices" or "nodes" and lines called edges that connect them. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another. In a network analysis of such a circuit from a topological point of view, the network nodes are the vertices of graph theory and the network branches are the edges of graph theory.

1.2.1 Basic concepts and Definitions

Graph: It is collection of branches and nodes in which each branch connects two nodes.

Graph of a Network: The diagram that gives network geometry and uses lines with dots at the ends to represent network element is usually called a graph of a given network. For example, figure 1.2 represents the network shown in figure 1.1.

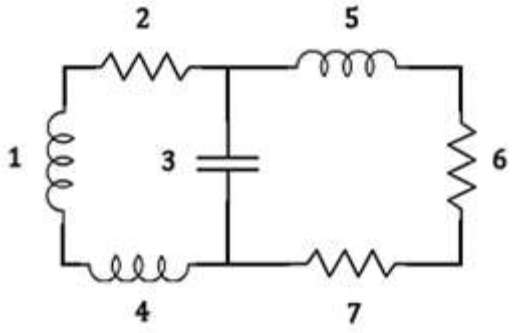


Figure 1.1 Network

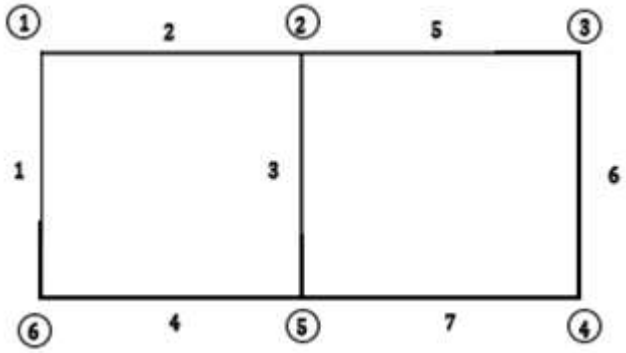


Figure 1.2 Graph

Sub-graph: A sub-graph is a subset of branches and nodes of a graph for example branches 1, 2, 3 & 4 forms a sub-graph. The sub-graph may be connected or unconnected. One sub- graph of graph shown in figure 1.2 is shown in figure 1.3.

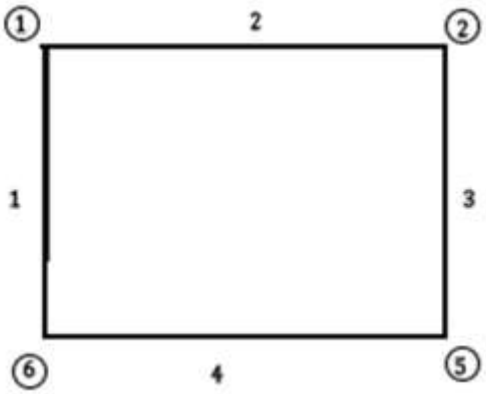


Figure 1.3 Sub-graph

Connected Graph: If there exists at least one path from each node to every other node, then the graph is said to be connected. In figure 1.4 is illustrated a connected graph.

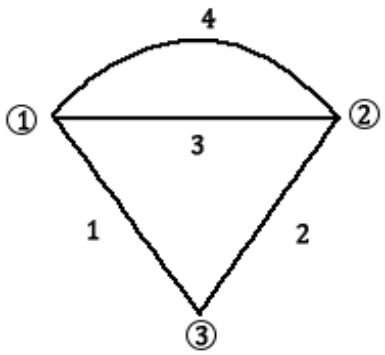


Figure 1.4 Connected graph

Un-connected Graph: If no path exists from each node to every other node, the graph is said to be un-connected graph. For example, the network of figure 1.5(a) containing a transformer (inductively coupled parts) its graph in figure 1.5 (b) could be un-connected.

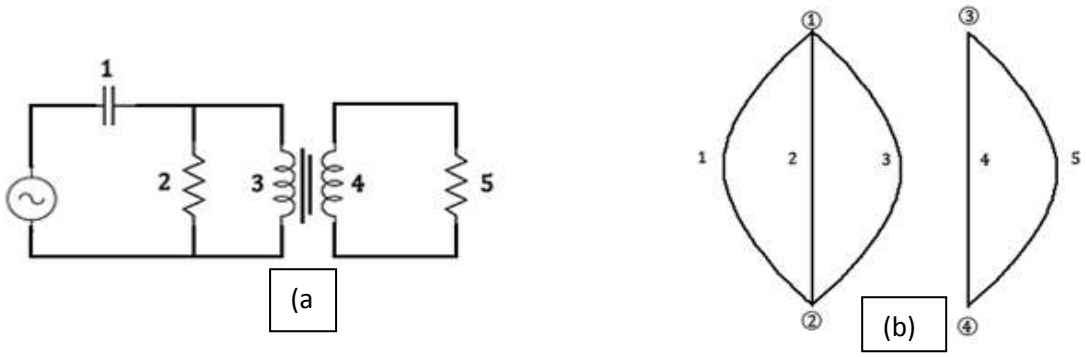


Figure 1.5 Network and its un-connected graph

Path: A sequence of branches going from one node to other is called path. The node once considered should not be again considered the same node.

Loop (Closed Path): Loop may be defined as a connected sub-graph of a graph, which has exactly two branches of the sub-graph connected to each of its node. For example, the branches1, 2 & 3 in figure 1.6 constitute a loop.

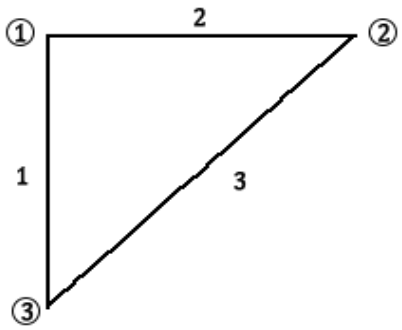


Figure 1.6 Loop (Closed Path)

Oriented Graph: The graph whose branches carry an orientation is called an oriented graph.

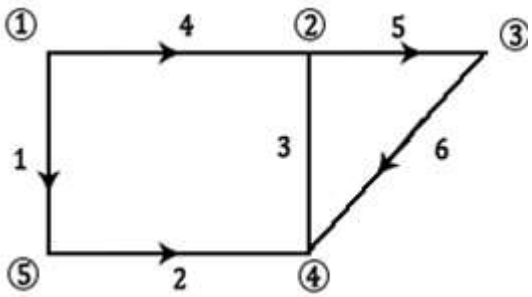


Figure 1.7 Oriented graph

The current and voltage references for a given branches are selected with a +ve sign at tail side and –ve sign at head

Tree: Tree of a connected graph is defined as any set of branches, which together connect all the nodes of the graph without forming any loops. The branches of a tree are called Twigs.

Co-tree: Remaining branches of a graph, which are not in the tree form a co-tree. The branches of a co-tree are called links.

The tree and co-tree for a given oriented graph shown in figure 1.8 (a) are shown in figure1.8 (b) and figure1.8(c)

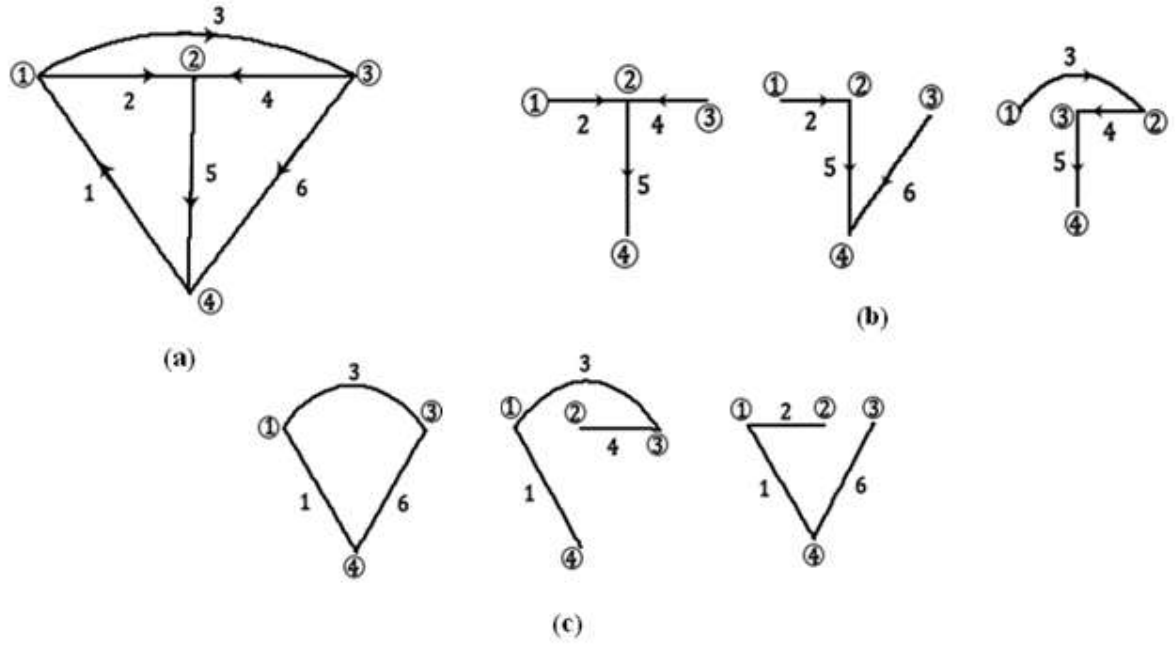


Figure 1.8: (a) Oriented graph, (b) Trees, (c) Co-Trees

Cut set: Cut set is a closed line which cuts the original graph into two distinct parts, the so obtained sub graphs must have all branches connected to one or two (distinct) nodes.

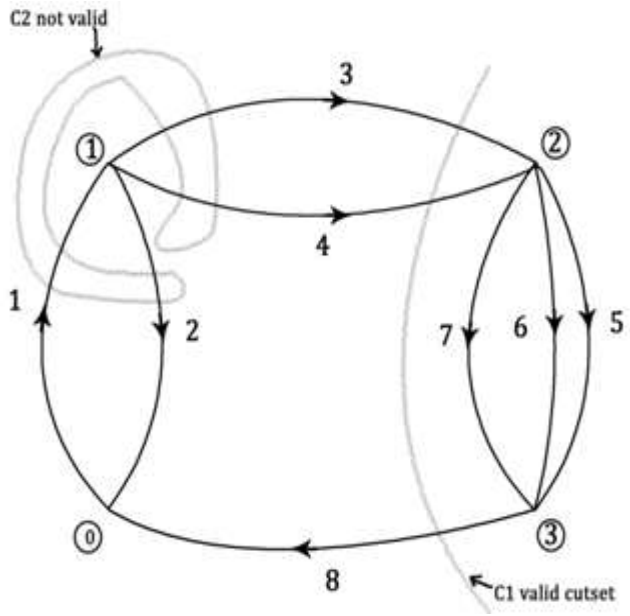


Figure 1.9 Cut-sets

1.2.2 Node to branch incidence matrix

Definition: The node to branch matrix $\mathbf{A_a}$ sets up a relation between the directed branches b and the nodes n of a graph and is a **(n × b)** matrix. The entries of this matrix A_a are defined as follows.

$$\mathbf{a_{ij}} = \begin{cases} +1 \cdots \text{when branch } \mathbf{b_j} \text{ is incident to node } \mathbf{n} \text{and is directed away from it} \\ \mathbf{-1} \cdots \text{when branch } \mathbf{b_j} \text{ is incident to node } \mathbf{n}_i \text{ and is directed towards it} \\ \mathbf{0} \cdots \text{when branch } \mathbf{b_j} \text{is not incident to node } \mathbf{n}_i \end{cases}$$

The sum of the elements along each column of $\mathbf{A_a}$ is zero, hence $\mathbf{A_a}$ can be reduced by one row (in general we take row (0) with no loss of information, the so obtained matrix $\mathbf{A}$ is known as the reduced node to branch incidence matrix of the graph.

Incidence Matrix and Kirchhoff's Current Law: Let$I = [i_1\ i_2 \ldots i_b]^T$,be the column vector of branch currents, Using$\mathbf{A}$ and $\mathbf{I}$, KCL can be written in a matrix formas:

$$\mathbf{A.I = 0}$$

The branch – node voltage transfer: Let's consider a graph of $\mathbf{n}$ nodes and $\mathbf{b}$ branches, let $\mathbf{V=[v_1\ v_2 \ldots v_b]^T}$, be the column vector of branch voltages and $\mathbf{E=[e_1\ e_2 \ldots e_{n-1}]^T}$ be the column vector of node voltages, so the relationship between the node voltages and the branch voltages is given by

$$\mathbf{V = A^T.\ E}$$

1.2.3Loop to branch incidence matrix

Definition: The fundamental loop matrix $\mathbf{B_f}$ sets up a relation between the directed branches $\mathbf{b}$ and the fundamental loops $\mathbf{L}$ of a graph and is a $\mathbf{(b-(n-1))\times b}$ matrix. The entries of this matrix $\mathbf{B_f}$ are calculated as follows.

$$\mathbf{b_{ij}} = \begin{cases} \mathbf{+1} \cdots \text{when branch } \mathbf{b_j} \text{is in loop } \mathbf{l_i} \text{ and has the same direction} \\ \mathbf{-1} \cdots \text{when branch } \mathbf{b_j} \text{ is in loop } \mathbf{l_i} \text{ and has opposite direction} \\ \mathbf{0} \cdots \text{when branch } \mathbf{b_j} \text{ is not in loop } \mathbf{l_i} \end{cases}$$

Loop to branch incidence matrix and Kirchhoff's Voltage Law: Let $\mathbf{V=[v_1\ v_2 \ldots v_b]^T}$, be the column vector of branch voltages. Using $\mathbf{B_f}$ and $\mathbf{V}$, KVL can be written in a matrix formas:

$$\mathbf{B_f.V = 0}$$

1.2.4Fundamental cut-set matrix

Definition: The fundamental cut-set matrix **D** sets up a relation between the directed branches **b** and the fundamental cut-sets **c** of a graph and is a **((n − 1)× b)** matrix. The entries of this matrix **D** are calculated as follows.

$$d_{ij}=\begin{cases}+1 & \cdots \text{when branch } \mathbf{b_j} \text{ is in the cut-set } \mathbf{c_i} \text{ and has the same direction}\\ -1 & \cdots \text{when branch } \mathbf{b_j} \text{ is in the cut-set } \mathbf{c_i} \text{and has opposite direction}\\ 0 & \cdots \text{when branch } \mathbf{b_j} \text{ is not in the cut-set } \mathbf{c_i}\end{cases}$$

Fundamental cut-set Matrix and Kirchhoff's Current Law: Let $\mathbf{I=[i_1\ i_2\ \dots\ i_b]^T}$,be the column vector of branch currents , Using **D** and **I**, KCL can be written in a matrix form as :

$$\mathbf{D.I = 0}$$

1.3 Topological network analysis techniques

Using the results obtained from the graph theory, a set of techniques are derived for solving (analyzing) linear electrical networks (the Nodal, Loop and Cut-set Analysis).

1.3.1 Generalized branch model

Assuming branch **k**which connects node **i**to node **j**as shown in Figure 1.10.

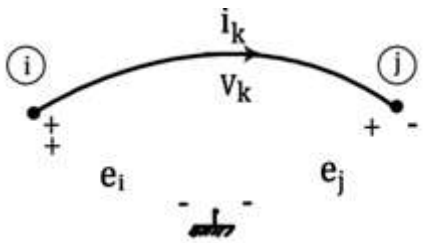


Figure 1.10

If the branch contains linear elements, then it can be modeled as shown in Figure 1.11.

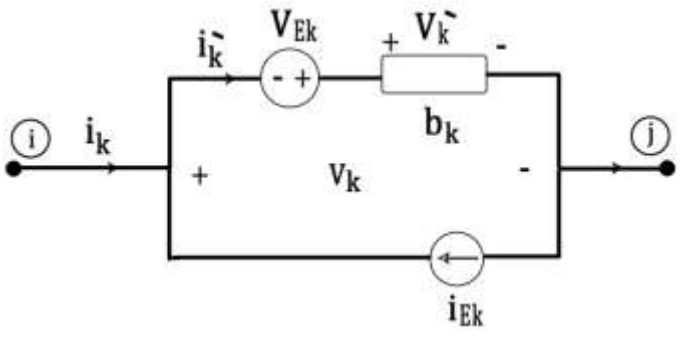


Figure 1.11 The generalized branch model

- $\mathbf{i_k}$ and $\mathbf{v_k}$ are the branch variables (current and voltage)
- $\mathbf{b_k}$ is the branch element (R, L, C or dependent sources …)
- $\mathbf{i_k'}$ and $\mathbf{v_k'}$ are the branch internal variables ($\mathbf{i_k'}$ the current through $\mathbf{b_k}$ and $\mathbf{v_k'}$ is the voltage across $\mathbf{b_k}$)
- $\mathbf{v_{Ek}}$ and $\mathbf{i_{Ek}}$ represent any voltage source or current source applied externally to the branch element (independent sources $\mathbf{(v_g,i_g)}$, or initially stored energy in C and L)

Required condition: $\mathbf{i_{ek}}$ and $\mathbf{v_{ek}}$ are taken to support the flow of the internal branch current $\mathbf{I_k'}$,Hence:

$$\mathbf{i_k' = i_k + i_{ek}}$$

$$\mathbf{v_k' = v_k + v_{ek}}$$

If we define the following vectors:

The column vector of the branch currents $\mathbf{I(s) = [i_{b1}(s)\ i_{b2}(s)\ \dots\ i_{bk}(s)]^T}$

The column vector of the branch voltages $\mathbf{V(s) = [v_{b1}(s)\ v_{b2}(s)\ \dots\ v_{bk}(s)]^T}$

The column vector of the internal branch currents $\mathbf{I'(s) = [i_{b1}(s)\ i_{b2}(s)\ \dots\ i_{bk}(s)]^T}$

The column vector of the internal branch voltages $\mathbf{V'(s) = [v_{b1}(s)\ v_{b2}(s)\ \dots\ v_{bk}(s)]^T}$

The column vector of the external branch currents source $\mathbf{I_E(s) = [i_{E1}(s)\ i_{E2}(s)\ \dots i_{Ek}(s)]^T}$

The column vector of the external branch voltage sources $\mathbf{V_E(s) = [v_{E1}(s)\ v_{E2}(s) \dots v_{Ek}(s)]^T}$

Then

$$\mathbf{I'(s) = I(s) + I_E(s)}$$

$$\mathbf{V'(s) = V(s) + V_E(s)}$$

1.3.2 Branch impedance matrix Z(s)

If **Z(s)** exists then **V'(s) = Z(s).I'(s)**, where **Z(s)** is a diagonal matrix that combines the respective bloc impedances.

Z(s) = $\begin{pmatrix} R & & \\ & (1/S)\ D & \\ & & \end{pmatrix}$

SL

R is a diagonal matrix, for which the main diagonal contains the resistances of all resistive branches of the network

$$R = \begin{bmatrix} \boldsymbol{R1} & 0 & 0 \\ 0 & \boldsymbol{R2} & 0 \\ 0 & 0 & \boldsymbol{R3} \end{bmatrix}$$

D is a diagonal matrix, for which the main diagonal contains the inverse of capacitances of all capacitive branches of the network

$$D = \begin{bmatrix} \boldsymbol{1/C1} & 0 & 0 \\ 0 & \boldsymbol{1/C2} & 0 \\ 0 & 0 & \boldsymbol{1/C3} \end{bmatrix}$$

L is a square matrix, which is not necessarily a diagonal matrix

$$L = \begin{bmatrix} \boldsymbol{L1} & 0 & 0 \\ 0 & \boldsymbol{L2} & 0 \\ 0 & 0 & \boldsymbol{L3} \end{bmatrix}$$

The main diagonal elements however **L** contains all inductances of the network

The off diagonal elements may correspond to mutual inductances

1.3.3 Branch admittance matrix Y(s)

If **Y(s)** exists then **I'(s) = Y(s).V'(s)** and if **Z(s)** is nonsingular then **Y(s) =Z^{-1}(s)**, where **Y(s)** is a diagonal matrix that combines the respective bloc admittances.

1.3.4 Nodal analysis

The nodal analysis is used to determine directly the node voltage matrix **E(s).**

We have: **A.I=0** and **I'(s)=I(s)+I$_E$(s)**, then **A.I'(s)=A.I$_E$(s)**

Assuming that **Y(s)** exists then**I'(s)=Y(s).V'(s)**

Hence: **A.(Y(s).V'(s))=A.I$_E$(s)**

Also we have: **A^T.E(s) = V(s) = V'(s)–V$_E$(s)**

Hence:**A.Y(s).(A^T.E(s)+V$_E$(s))=A.I$_E$(s) or (A.Y(s).A^T).E(s) = A.(I$_E$(s) – Y(s).V$_E$(s))**

Setting **Y$_N$(s) = A.Y(s).A^T**, and **I$_N$(s) = A.(I$_E$(s) – Y(s).V$_E$(s)),** then **Y$_N$(s).E(s) = I$_N$(s)**

So **the nodal analysis solution equation** is given by:

$$E(s) = Y_N^{-1}(s).I_N(s)$$

1.3.5 Loop analysis

The loop analysis is used to determine directly the branch current matrix I(s).

We have $B_f .V(s)=0$and $V'(s)=V(s)+V_E(s)$, then :$B_f .V'(s)=B_f.V_E(s)$

Assuming that **Z(s)** exists then $V'(s)=Z(s).I'(s)$

Hence, $B_f .Z(s).I'(s) = B_f. V_E(s)$

$I'(s)=I(s)+I_E(s)$ so, $B_f.Z(s).(I(s)+I_E(s))=B_f.V_E(s)$

We have $I(s) = B_f^T.I_C(s)$ so, $B_f.Z(s).(B_f^T.I_C(s)+I_E(s))=B_f.V_E(s)$ or

$(B_f.Z(s).B_f^T).I_C(s)=B_f.(V_E(s)-Z(s).I_E(s))$

Let $Z_L(s) = B_f .Z(s).B_f^T$ and $V_L(s)=B_f.(V_E(s)-Z(s).I_E(s))$ so, $Z_L(s).I_C(s) = V_L(s)$

Hence we get the **loop analytic equation**:

$$I_C(s) = Z^{-1}_L(s).V_L(s)$$

Where $I_C(s)$ is the link current matrix, hence the branch currents matrix is given by:

$$I(s)=B_f^T.I_C(s)$$

1.3.6 Cut-set analysis

The cut-set analysis is used to determine directly the branch voltage matrix **V(s)**.

We have D.I$(s)=0$ and $I'(s) = I(s) + I_E(s)$then: $D.I'(s)=D.I_E(s)$

Assuming that **Y(s)** exists then$I'(s)=Y(s).V'(s)$

Hence $D.(Y(s).V'(s))=D.I_E(s)$

$V'(s)=V(s)+V_E(s)$ so, $D.Y(s).(V(s)+V_E(s))=D.I_E(s)$

We have $V(s) = D^T.V_T(s)$ so, $D.Y(s).(D^T.V_T(s)+V_E(s))=D.I_E(s)$or

$(D.Y(s).D^T).V_T(s)= D.(I_E(s)-Y(s).V_E(s))$

Let $Y_{CUT}(s) = D .Y(s).D$ and $I_{CUT}(s) = D.(I_E(s)-Y(s).V_E(s))$

So, $\mathbf{Y_{CUT}(s).V_T(s) = I_{CUT}(s)}$

Hence we get the **cut-set analytic equation**:

$$\mathbf{V_T(s) = Y_{CUT}^{-1}(s).I_{CUT}(s)}$$

Where $\mathbf{V_T(s)}$ is the twig voltage matrix, hence the branch voltages matrix is given by:

$$\mathbf{V(s) = D.V_T(s).}$$

1.4 Echelon Algorithm

Definition to find a tree visually is not an especially difficult task for the circuit analyst programming a computer to stipulate a tree from only formulated matrix is a nontrivial problem. The program that effects tree definitions can be predicated on two fundamental properties that mathematically given tree

First, the tree submatrix square of order *N* in an (*N* + 1) node circuit, in the present case this means that the tree sub matrix of *A* is extracted from only 4of the 10 columns of *A.*

Second, the determinant of a tree submatrix is + 1 , this situation establishes the feasibility of searching *A* to find an *N* x *N* nonsingular sub matrix, From arguments provided earlier , one is assured that such a submatrix correspond without exception to a tree.

A programmable search procedure for a nonsingular *N* x *N* submatrix in a matrix of order M x P is afforded by the *Echelon* algorithm

A matrix is said to be in Echelon form if it is written as

$$\mathbf{Q_E} = \begin{pmatrix} \mathbf{1} & X & X & X \ldots X & X & X \ldots X \\ 0 & \mathbf{1} & X & X \ldots X & X & X \ldots X \\ 0 & 0 & \mathbf{1} & X \ldots X & X & X \ldots X \\ 0 & 0 & 0 & 0 \ldots 0 & \mathbf{1} & X \ldots X \\ \vdots & & & & & \\ 0 & 0 & 0 & 0 \ldots 0 & 0 & \mathbf{1} \ldots X \\ \vdots & & & & & \\ 0 & 0 & 0 & 0 \ldots 0 & 0 & 0 \ldots X \end{pmatrix}$$

N rows

M rows

N columns P- N column

Where X denotes a nonzero element (not necessarily the same nonzero element in every indicated position) .observe that below and to the left of the staircase. Only zero elements appear, whereas immediately above and immediately to the right of the staircase, only unity elements appear along the diagonal of a square submatrix of order N is upper triangular and nonsingular. If the column denote branches in an incidence matrix, these same columns must represent twigs of a tree.

A matrix can always be reduced to Echelon form through use of only elementary row operations. To illustrate the Echelon algorithm is stated below and exemplified with respect to (2.60)

Step 1. Scan the column of A from the left until a column that has nonzero entries is found, in (2.60) the first such column encountered is column 1.

Step 2. Scan the column just found from the top until a nonzero element is found, say in row r. If r =/= 1, interchange row r and row 1 this procedure is tantamount to node renumbering. In (2.60), r=1 in first column, and no row interchange operation is required.

Step 3. Subsequent to row interchange, multiply the resultant first row by the inverse of the first element in this row to obtain unity in the (1.1) position. This step is actually not necessary when dealing with network incidence matrices, although; formally, it is a part of the echelon algorithm.

Step 4. Reduce all elements below row 1 in the column upon which attention is focused to zero by elementary row operation. In (2.60); one can accomplish this prong of the procedure by replacing the second row by the sum of first and second rows. The result is a new matrix; say A_{E1} such that

$$A_{E1} = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

Step 5. Step 1 through 4 are now repeated on the submatrix formed by deleting the row and column upon which elementary operations have concluded. The procedure continues until the element in the Nth row and the Nth column is rendered unity.

From the A_{E1}matrix, the submatrix alluded to in step5 is

$$A_{E2} = \begin{array}{c} \begin{matrix} (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (2) \\ (3) \\ (4) \end{matrix}$$

Multiplication of the first row by (-1), followed by replacement of the second row by the second row minus the first row, leads to

$$A_{E3} = \begin{array}{c} \begin{matrix} (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (2) \\ (3) \\ (4) \end{matrix}$$

The new candidate for Echelon operation is

$$A_{E4} = \begin{array}{c} \begin{matrix} (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 1 & -1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (3) \\ (4) \end{matrix}$$

In this case, the first column is already in the proper format, and as a result, one can proceed to the next matrix.

$$A_{E4} = \begin{array}{c} \begin{matrix} (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (4) \end{matrix}$$

The first column processing a nonzero element is the second column, corresponding to branch 5. The element in this column in unity ands since there are no additional rows to consider, this column is in proper format. The final form of ***A*** is obtained from row 1 and column 1 from $\mathbf{A_{E1}}$ $\mathbf{A_{E3}}$ $\mathbf{A_{E4}}$ and $\mathbf{A_{E5}}$ namely

$$A_E = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

The most important result of interest is that the branch array {1, 2, 3, 5} is a tree for the linear graph of fig 2.8b. These branches are shown by heavy lines in the graph. In order to place (2.60) in the standard form of (2.4), one can now reorder the columns of (2.60) to display twigs first and then links. Thus,

$$A = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix} \qquad (2.62)$$

1.4.1 Voltage and current equations

The K V L and K C L equations for the determination of each branch voltage and each branch current utilize respectively the fundamental cutset and fundamental loop matrices.

In (2.62) replace the second row by the sum of the first and second rows so that the first modified ***A*** matrix is

$$A_1 = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

The second row is now multiplied by (-1) to obtain

$$A_2 = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

Matrix A_3 results from replacement of third row by the difference of third and second rows:

$$A_3 = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (5) & (4) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

If the third row of A3 is replaced by the sum of the third and fourth rows, the resultant matrix is

$$D = \left(\begin{array}{cccc|cccccc} (1) & (2) & (3) & (5) & (4) & (6) & (7) & (8) & (9) & (10) & \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & (1) \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & (2) \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & (3) \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 1 & 0 & (4) \end{array}\right)$$

(2.63)

Insertion of (2.63) into (2.54) fields

$$D = \left(\begin{array}{cccc|cccccc} (1) & (2) & (3) & (5) & (4) & (6) & (7) & (8) & (9) & (10) \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

And we have taken the next circuit in 1.12 as an example

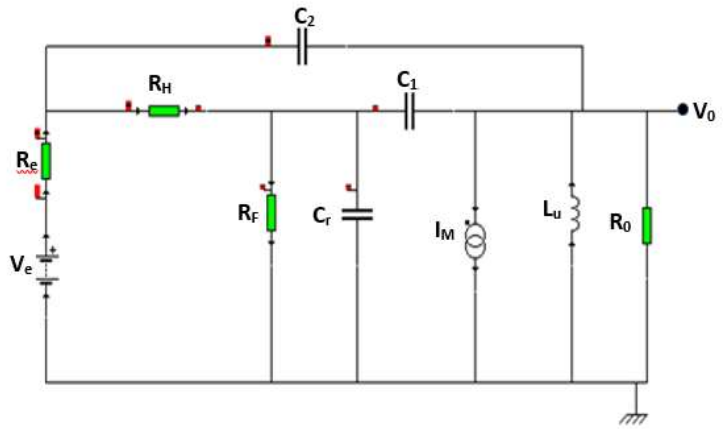


Fig 1.12 a Passive network used to illustrate application of network graph theory to computer aided analysis

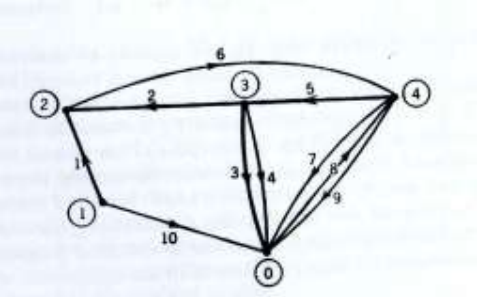


Fig 1.12 b linear graph network in (a)

Application: by using the implemented Matlab routine in appendix 01

We have to use the following incidence matrix as an input argument to the echelon function to obtain a tree and its corresponding cotree as show below.

$$A = \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

```
A =

     1     0     0     0     0     0     0     0     0     1
    -1    -1     0     0     0     1     0     0     0     0
     0     1     1     1    -1     0     0     0     0     0
     0     0     0     0     1    -1     1    -1     1     0

>> [ T C ]= echelon(A)

T =

     1     2     3     5


C =

     4     6     7     8     9    10
```

1.5 Topological analysis methods

1.5.1. Reformulation of the equations from graph theory using the concept of tree

If a tree is selected then all matrices and column vector can be partitioned as follows:

Incidence matrix node to branch: A = [A_t] A_c]

Fundamental incidence loop matrix: $B_f = [B_t|U]$

Fundamental cutset matrix :$D = [U|D_c]$

U denotes the identity matrix

Column vector of branch voltages : $V = \begin{bmatrix} V_t \\ \overline{V_c} \end{bmatrix}$

Column vector of branch currents : $I = \begin{bmatrix} I_t \\ \overline{I_c} \end{bmatrix}$

Column vector of node voltages with respect to the reference node "0" ground): E .

It can be shown that:

$$B_t = (A_t^{-1} A_c)^T$$

$$D_C = -B_t^T = -A_t^{-1} A_c$$

And That:

$$I = B_f^T I_c$$

$$V = D^T I_c$$

Moreover, according to the voltage transformation theorem we have: V = A^TE

Finally, according to the generalized branch model :

$$V' = V + V_E$$
$$I' = I + I_E$$
$$V' = ZI'$$
$$I' = YV'$$

1.5.2. Nodal analysis method

It consists in solving the following equation:

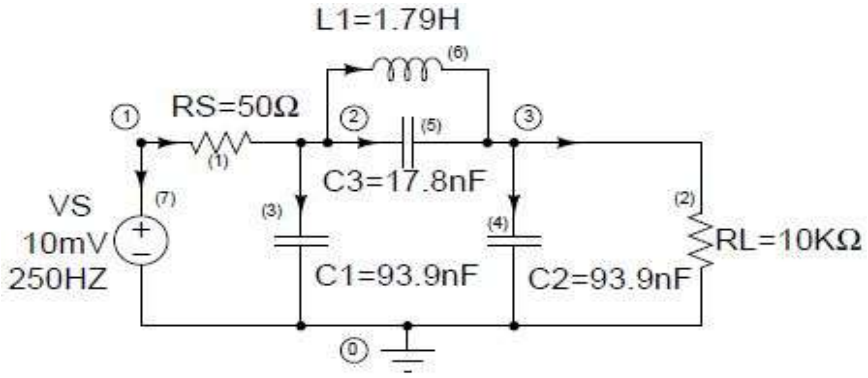
$$Y_N E = I_N$$
$$ou,$$
$$Y_N = AYA^T$$

$$I_N = A(I_E - YV_E)$$

Therefore, this method makes it possible to obtain the potentials of all the nodes of the circuit E with respect to the ground. It is also possible to determine V and I using the previous relations :

$$V = A^T E$$
$$I = Y(V + V_E) - I_E$$

To illustrate this technique we have used the elliptic filter network of the following figure and we have implemented this technique in the following MATLAB script:




```
% Nodal Analysis - Elliptic Network
%
% This Script implements in the time domain the response of an elliptical
filter
% at AC (sinusoidal) excitation.
```

```
%
% Inputs: - Source frequency (F)
% - The amplitude of the source (VM)
% - The load value (RL)
% - Number of cycles (periods) in the simulation interval (NC)
% - Number of simulation points per cycle (NS)
% Outputs: - Graphs of the excitation vs (t) is of the response vL (t) as a
% function of time
clear all;
clc; % Clear the screen

% Fixed circuit data

RS = 50; % Source resistance
rs = 1.0e-3;
C1 = 93.9e-9; % C1 = 93.9nF
C2 = 93.9e-9; % C2 = 93.9nF
C3 = 17.8e-9; % C3 = 17.8nF
L1 = 1.79; % L1 = 1.79H

% Get entries

F = input ('Enter the frequency value in Hz:');
T = 1 / F; % The source period
s = 1i * 2 * pi * F; % The complex variable
```

```
VM = input ('Enter the value of the source amplitude in V:');

RL = input ('Enter the value of the load in Ohm:');

NC = input ('Enter the number of cycles you want for the simulation:');

NS = input ('Enter the number of simulation points per cycle:');


% Determine the total number of simulation points


N = NC * NS;


% Create the simulation time interval


t = inspace (0, NC * T, N);


% Create the complex vector of the excitation source


VS = VM * exp (s * t);


% Build A
% In order to give Y the diagonal form Block, o reorders the branches
%
% so that the branch (7) becomes a resistive branch.


order_branches = [1, 2, 7, 3, 4, 5, 6];


A = A (:, order_branches);
```

```
% Construct the main diagonal of Y

G = diag ([1 / RS, 1 / RL, 1 / rs]);
C = diag ([C1, C2, C3]);
GAMMA = diag ([1 / L1]);

% Build Y, YN, IE and VE (initial)

Y = blkdiag (G, s * C, GAMMA / s);
YN = A * Y * A ';
IE = zeros (7.1);
VE = zeros (7.1);

E = []; % Initializes the solution matrix

for k = 1: N
    VE (3) = - VS (k); % update the 3rd element of VE which corresponds to
the
                % branch containing the excitation VS
    IN = A * (IE-Y * VE); % build IN
    E = [E, YN \ IN]; % Update the solution matrix by adding a
                % new column (the solution at time t (k))
end
```

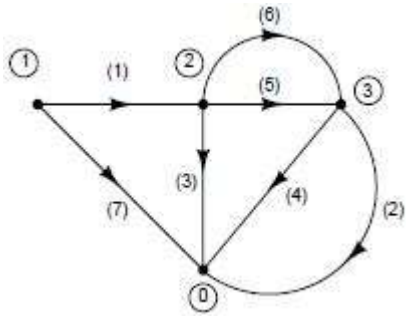
```
% Obtain the instantaneous values vs (t) of VS (excitation) and vl (t) of E
(3)
% (reply). It suffices to extract the real parts of the vectors
% complex


vl = imag (E (3, :));
vs = imag (VS);


% Draw the graphs
```

```
plot (t, vs, 'k -', t, vl, 'k--');
grid on
xlabel ('temp (sec)');
ylabel ('amplitude (v)')
title ('Response of the Elliptical filter to a sinusoidal excitation');
legend ('Excitation', 'Response');
```

Using the corresponding graph of the network



The node to branch incidence matrix can be obtained as:

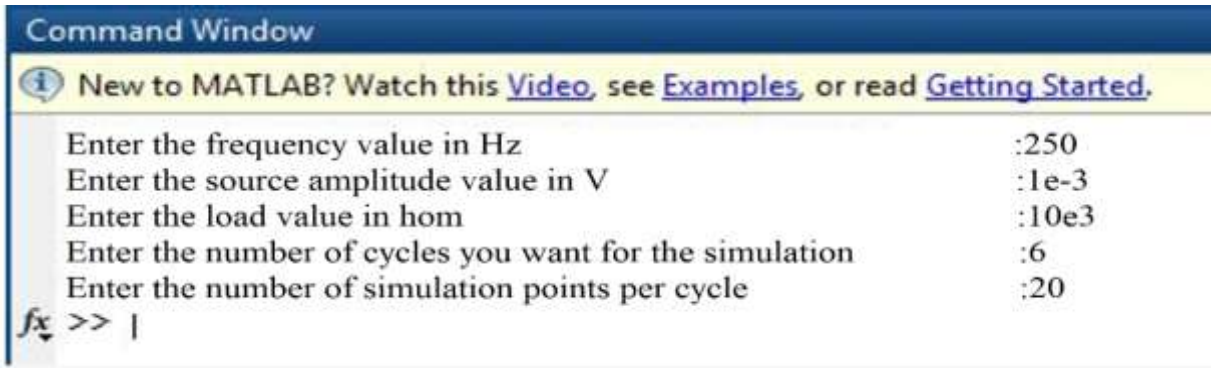
(2)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	
1	0	0	0	0	0	0	0	0	0	(1)

A= -1 -1 0 0 0 1 0 0 0 0 (2)

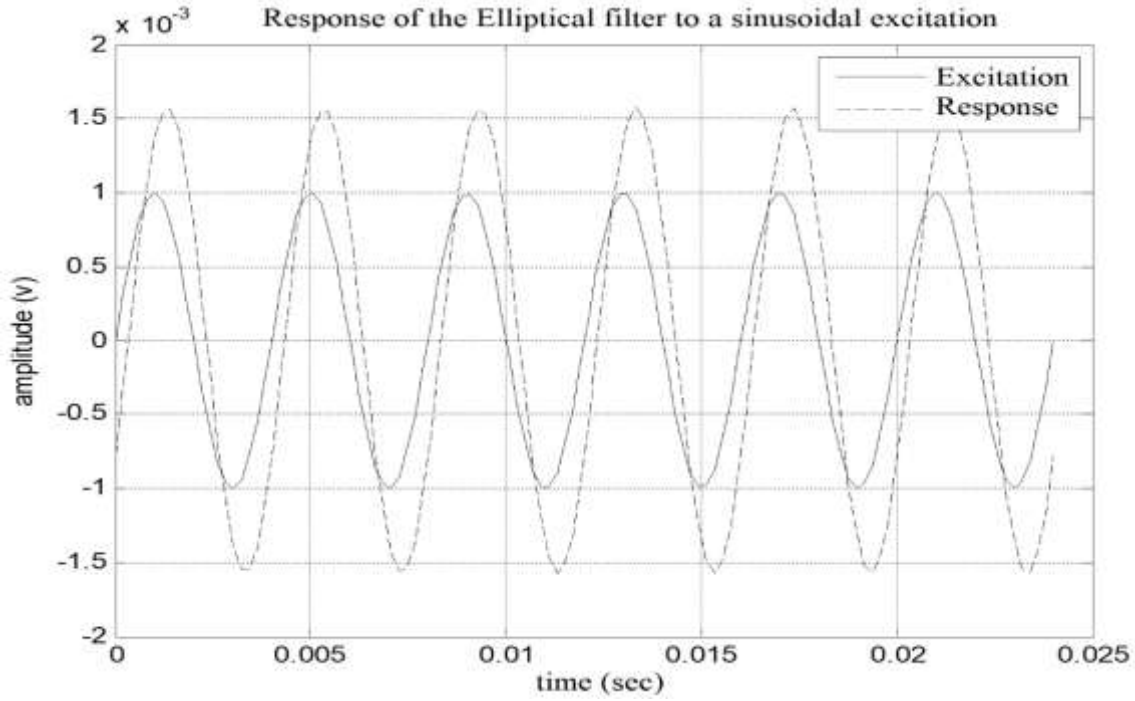
0 1 1 1 -1 0 0 0 0 0 (3)

0 0 0 0 1 -1 1 -1 1 0 (4)

The following figure shows how data are entered to conduct the simulation.



The result of the simulation is shown in the following figure



1.5.3. Fundamental mesh analysis method

It consists in solving the following equation:

$$B_L I_c = V_L$$

$$ou,$$

$$B_L = B_f Z B_f^T$$

$$V_L = B_f (V_E - Z I_E)$$

Therefore, this method makes it possible to obtain the currents of all the branches of the I_C co-tree.

It is also possible to determine V and I using the preceding relations:

$$I = B_f^T I_c$$
$$V = Z(I + I_E) - V_E$$

This method is implemented in a Matlab routine and gives the same results of the nodal analysis on the same circuit.

1.5.4. Fundamental cut analysis method

It consists in solving the following equation:

$$D_{cut} V_t = I_{cut}$$
$$ou,$$
$$D_{cut} = DYD^T$$
$$I_{cut} = D(I_E - YV_E)$$

Therefore, this method makes it possible to obtain the voltages of all the branches of the tree V_t.

It is also possible to determine V and I using the preceding relations :

$$V = D^T V_t$$
$$I = Y(V + V_E) - I_E$$

This method is implemented in a Matlab routine and gives the same results of the nodal analysis on the same circuit.

Remarks:

a. Unlike the methods of loop and cutset, the nodal analysis does not require the use of a tree for its application.

b. All the analysis methods presented here are based on the generalized branch model. However, if we consider the branch (7) of the circuit which contains the excitation source, it does not have a passive element as required by the model. So none of the topological methods can be applied.

To get around this obstacle, a trick consists in associating with this voltage source, an internal resistance (rs) of very low value (and therefore negligible), so as not to significantly affect the overall response of the circuit. Thus, the branch (7) becomes a resistive branch whose passive element is rs. In this precise case, we will choose the value of 1μΩ.

We will see later, another method known as "modified nodal", which accepts branches without passive elements and does not require the use of such a trick.

c. For the methods requiring the use of a tree, we used the echelon algorithm implemented in the Matlab file "echelon.m". This algorithm [see appendix #1] makes it possible as we have seen previously, to automatically determine from the incidence matrix ***A*** a tree of the graph.

d. For the methods of meshes and fundamental cuts, the matrices $\mathbf{B_f}$ and **D** were extracted directly from the matrix A, in accordance with the preceding equations

The Matlab scripts "nodal analysis.m", "loop analysis.m" and "cutset analysis.m", respectively implement each the topological analysis methods that we have exposed. It can be seen that the results obtained are exactly identical. In addition, these results are identical to that obtained by the SPICE simulator.

1.6. The Modified Nodal Analysis

The Modified Nodal Analysis or MNA is an extension of nodal analysis which not only determines the circuit's node voltages (as in classical nodal analysis), but also the branch currents.

The modified nodal analysis was formulated in the mid-1970s and developed subsequently for the analysis of analog filters and the simulation of electronic circuits; it is used in one form or another in many modern simulation packages, such as SPICE.

Let **A** be the reduced incidence matrix for a linear RLC network, for a convenient description of this approach, it is appropriate to decompose the reduced incidence matrix according to the element types of their branches. By definition of the reduce incidence matrix, every column of **A** corresponds to one branch. We assume that the branches are enumerated in the following order: first all resistive branches, then all capacitive branches, then all inductive branches,

then all voltage source branches, and finally all current source branches. Then A can be decomposed into block form:

$$\mathbf{A = [A_R, A_C, A_L, A_E, A_I]}$$

Where the index stands for resistive, capacitive, inductive, voltage source and current source branches, respectively .Using the characteristic equations as indicated above, we obtain the following system:

$$\mathbf{A_C \frac{d}{dt} C\, A_C' \, e + A_R\, G\, A_R' \, e + A_L i_L + A_E i_V = -A_I I_J}$$

$$\mathbf{\frac{d}{dt} L\, i_L - A_L' \, e = 0}$$

$$\mathbf{A_E' \, e = V_E}$$

Here, we have used the following denotations:

- $\mathbf{i_V}$ is the vector of all branch currents through voltage sources.
- $\mathbf{i_L}$ is the vector of all branch currents of inductive branches.
- $\mathbf{I_J}$ is the vector of the values of all current sources.
- $\mathbf{V_E}$ is the vector of the values of all voltage sources.
- **C** is the diagonal matrix containing all capacities
- **G** is the diagonal matrix containing all conductances (inverses of the resistances)
- **L** is the diagonal matrix containing all inductivities.

The unknowns in the classical MNA system are the node voltages **e,** the currents through voltage sources $\mathbf{i_V}$, and the currents through inductors $\mathbf{i_L}$. In matrix notation, the system can be written down as:

$$\begin{bmatrix} A_C C A_C^T & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{d}{dt} \begin{bmatrix} e \\ i_L \\ i_V \end{bmatrix} + \begin{bmatrix} A_R G A_R^T & A_L & A_E \\ -A_L^T & 0 & 0 \\ A_E^T & 0 & 0 \end{bmatrix} \begin{bmatrix} e \\ i_L \\ i_V \end{bmatrix} = \begin{bmatrix} -A_j I_j \\ 0 \\ V_E \end{bmatrix}$$

Such a system is called a differential-algebraic equation, and it's the general MNA model for passive linear network.

1.7 Why do we use the modified nodal analysis?

The topological network analysis techniques mentioned before (nodal, loop and cut-set analysis) have been widely used for formulating circuit equations in computer-aided network analysis and design programs. However, several limitations exist in those methods including the inability to process the current through voltage and current sources in a simple and efficient manner (adding a series resistor with the voltage sources and a parallel resistor across the current source are needed). A modified nodal analysis (MNA) method is proposed here in order to remove those limitations

1.8 Conclusion

In this chapter, we have described the different topological network analysis techniques and their use. We have also talked about the modified nodal analysis (MNA) technique, and its advantages over other topological network analysis techniques.

2.1 Introduction:

Computers can be powerful tools if used properly, especially in the realms of science and engineering. Software exists for the simulation of electric circuits by computer, and these programs can be very useful in helping circuit designers test ideas before actually building real circuits, saving much time and money.

These same programs can be fantastic aids to the beginning student of electronics, allowing the exploration of ideas quickly and easily with no assembly of real circuits required. Of course, there is no substitute for actually building and testing real circuits, but computer simulations certainly assist in the learning process by allowing us to experiment with changes and see the effects they have on circuits. Throughout this chapter, we will be incorporating computer printouts from circuit simulation frequently in order to illustrate important concepts. By observing the results of a computer simulation.

To simulate circuits on computer, we use a particular program called SPICE, which works by describing a circuit to the computer by means of a listing of text. In essence, this listing is a kind of computer program in itself, and must adhere to the syntactical rules of the SPICE language. The computer is then used to process, or "run," the SPICE program, which interprets the text listing describing the circuit and outputs the results of its detailed mathematical analysis, those wanting more

2.2 Simulator description:

As shown in Figure 2.1, the simulator implemented in this project performs both time and frequency analysis for passive linear circuits with harmonic independent sources. To describe a given circuit, a text file known as a net-list, which is similar to the SPICE net-list, is used.

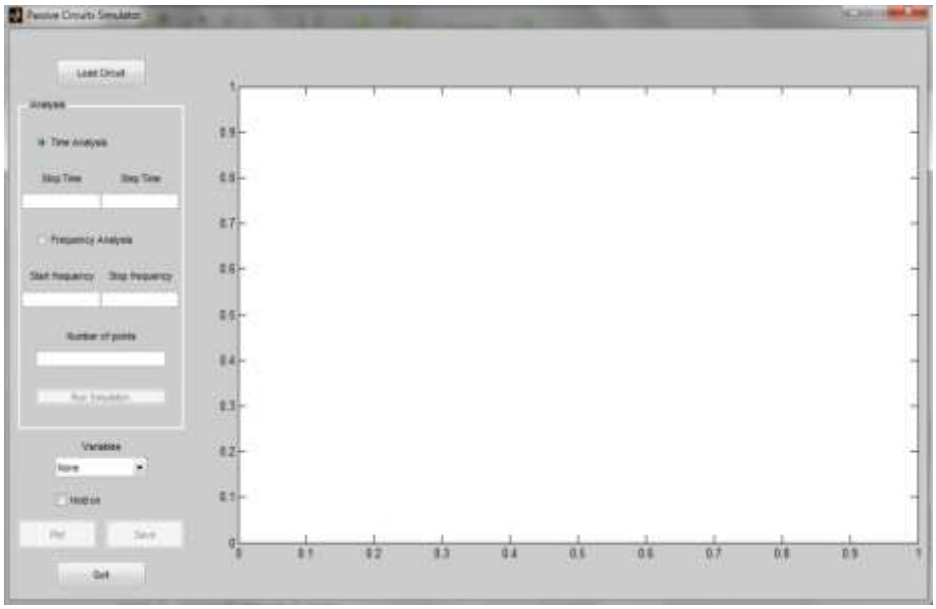


Figure *2.1* the simulator interface

2.2.1 SPICE:

SPICE was developed at the Electronics Research Laboratory of the University of California, Berkeley by Laurence Nagel with direction from his research advisor, Prof. Donald Pederson. SPICE1 was largely a derivative of the CANCER program, which Nagel had worked on under Prof. Ronald Rohrer.

SPICE1 was first presented at a conference in 1973. SPICE1 was coded in FORTRAN and used nodal analysis to construct the circuit equations. Nodal analysis has limitations in representing inductors, floating voltage sources and the various forms of controlled sources. SPICE1 had relatively few circuit elements available and used a fixed-timestep transient analysis. The real popularity of SPICE started with SPICE2 in 1975. SPICE2, also coded in FORTRAN, was a much-improved program with more circuit elements, variable timestep transient analysis using either the trapezoidal (second order Adams-Moulton method) or the Gear integration method (also known as BDF), equation formulation via modified nodal analysis (avoiding the limitations of nodal analysis), and an innovative FORTRAN-based memory allocation system developed by another graduate student, Ellis Cohen. The last FORTRAN version of SPICE was in 1983. SPICE3 was developed by Thomas Quarles (with A. Richard Newton as advisor) in 1989. It is written in C, uses the same netlist syntax, and added X Window System plotting.

SPICE (Simulated Program with Integrated Circuit Emphasis) is a general purpose software that simulates different circuits and can perform various analysis of electrical and electronic circuits including time domain response, small signal frequency response, total power dissipation, determination of nodal voltages, transient analysis…etc. This software is designed in such a way so that it can simulate different circuit operations.

2.2.2 SPICE Net-list

In SPICE, the circuit is first described to a computer by using a file called circuit file (netlist file). It contains the details of components and elements, the information about sources, the commands for the determination of the objective and the desirable results to be provided by the computer at the end. In circuit file, the user is to assign the node numbers while the nodes connect the circuit elements. From the description of circuits, SPICE develops the mathematical tool to solve for the network

The description and analysis of the circuit require the specification of the following:

- Element values
- Nodes
- Circuit elements and models
- Types of analysis
- Output variables
- Spice output commands
- Formats of circuit files and output files

The corresponding SPICE net-list for the RLC network shown in Figure 2.2 is:

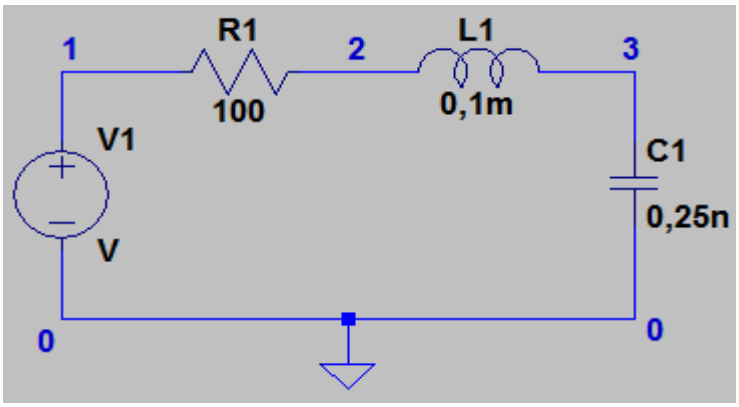


Figure 2.2:RLC circuit in series

V1 1 0 ac 12 sin
R1 1 2 100
L1 2 3 0,1m
C1 3 0 0,25n
.tran 0 10 1 1
.print dc v(0,1) v(1)
.end

2.3 Net-list format used in this project

For our project purposes, we have used a similar model of text file (net-list) to that of the SPICE, the following format is used to describe the different components:

Resistor: R<name><node 1><node 2> value

Inductor: L<name><node 1><node 2> value

Capacitor: C<name><node 1><node 2> value

Mutual effect: K<name><Lxx> <Lyy> value

Voltage source: V<name><node 1><node 2>amplitude_value phase_value

Current source: I<name><node 1><node 2>amplitude_valuephase_value

Frequency:F value

Then, the equivalent net-list for the network shown in Fig 2.2 is:

V1 1 0 12 0
L1 2 3 0,1m
R1 1 2 30
C1 2 0 100e-06
F 60

In order to use this simulator , we need to load the .crt circuit description file (netlist) , then we choose either time or frequency analysis , after clicking on simulate , all variables (node and branch voltages , branch currents) are computed , then we can choose one of them to plot according the type of analysis.

2.4 MNA model used for the algorithm implementation

For the purposes of this project, we will assume that all sources are harmonic and have the same frequency f, i.e., they follow the relations:

$$v(t) = V \sin(\omega t + \varphi) \qquad i(t) = I \sin(\omega t + \varphi)$$

Where $\omega = 2\pi f$ and φ is the phase angle

Under these assumptions it is very convenient to use complex quantities instead of the trigonometric functions (complex exponentials are applied).

The different components values in exponential form are given as follows:

$$v = Ve^{i(\omega t + \varphi)}$$

$$i = Ie^{i(\omega t + \varphi)}$$

The relation between v and i is given by: V= Z.I; where Z is known as the impedance.
The different branches impedances values in complex exponential form are given by:

Resistor: R ***Capacitor: 1/j ω C*** ***Inductor: j ω L***

Where R,C are diagonal matrices and L is not diagonal except in the case where mutual effect is zero.
Therefore, the general Modified Nodal Analysis model for the RLC network with harmonic sources is given by:

$$\begin{bmatrix} A_R Y_R A_R^T + A_C Y_C A_C^T + A_L Y_L A_L^T & A_E \\ A_E^T & 0 \end{bmatrix} \begin{bmatrix} e \\ i_v \end{bmatrix} = \begin{bmatrix} -A_j I_j \\ V_E \end{bmatrix}$$

OR

$$\begin{bmatrix} Y_N & A_E \\ A_E^T & 0 \end{bmatrix} \begin{bmatrix} e \\ i_v \end{bmatrix} = \begin{bmatrix} -A_j I_j \\ V_E \end{bmatrix}$$

Where $A_N = [A_R , A_C , A_L]$ and $Y_N = A_N Y A_N^T$ where Y is the inverse of the impedances (admittances), and it is the diagonal matrix containing Y_R, Y_C, Y_L , which denote diagonal matrices which contain the admittances of the resistive, capacitive and inductive branches, respectively.

2.6 MATLAB

MATLAB (**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

.

2.6.1History

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and found MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains.

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

2.7 general flowchart of the algorithm:

This algorithm consists of three main subroutines which are: load the text file and get circuit description subroutine, circuit analysis based on MNA technique subroutine and show the results obtained on the GUI subroutine.

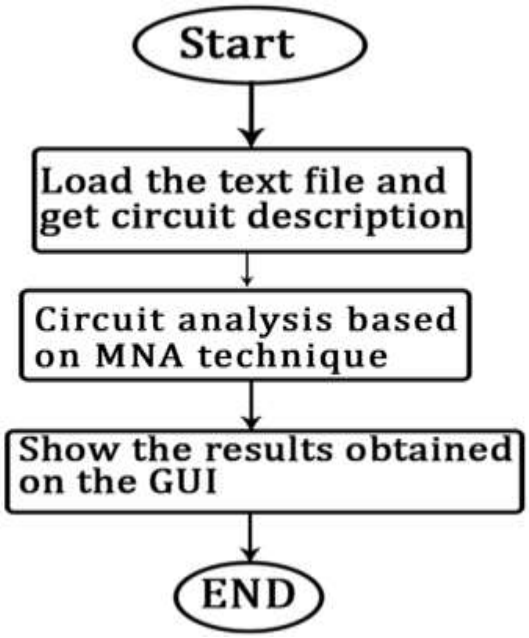


Figure2.3: general flowchart

2.7.1 Subroutines flowcharts:

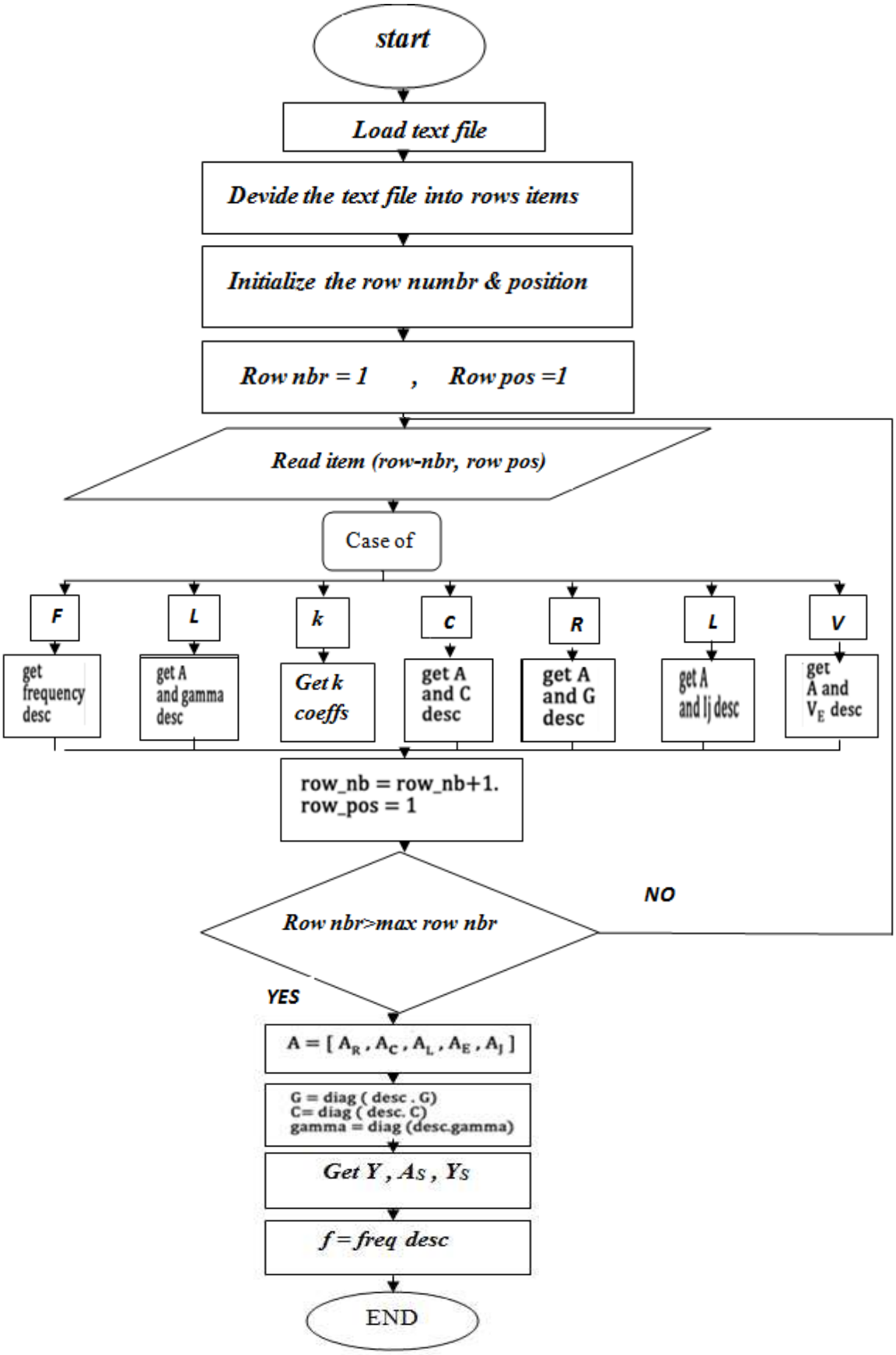
(A)Load text file and get circuit descriptions subroutine:

This subroutine reads the loaded text file (net-list) by the user, and divides it in set of rows, and then read each item of each row, for example, read item (2, 3), means read the third item of the second row of the net-list file, and then extract all the required parameters to performs the network analysis according to the MNA model described before***. (See flowchart 1).***

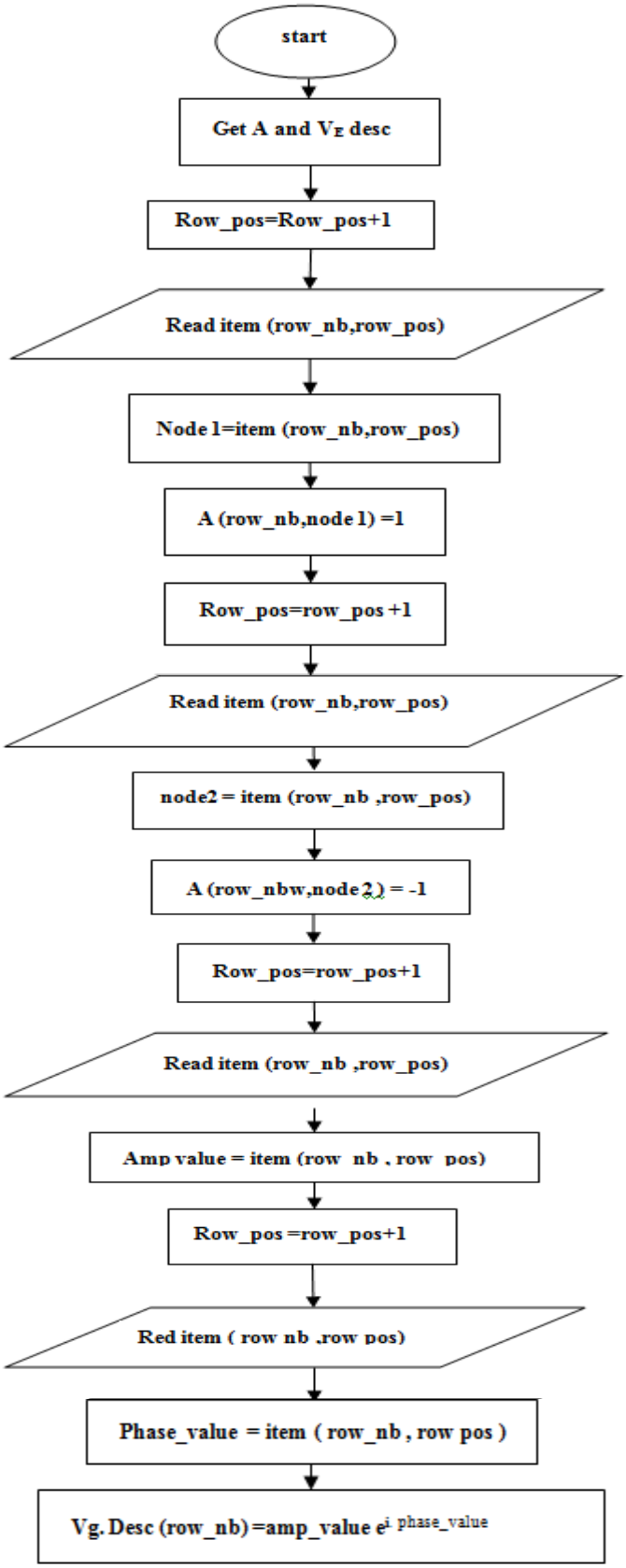
(a)Get A and V_E description subroutine: This subroutine tends to get the required reduced incidence matrix A, and the vector of the values of all voltage sources ***V*** ***(See flowchart 2)***

(b) Get A and I_J description subroutine: This subroutine tends to get the required reduced incidence matrix A,and the vector of the values of all current sources $\boldsymbol{I_J}$

(See flowchart 1)



Figur2.4: flowchart of *load text file and get circuit descriptions subroutine*



Figur2.5: Flowchart of Getting A and V_E descriptions subroutine

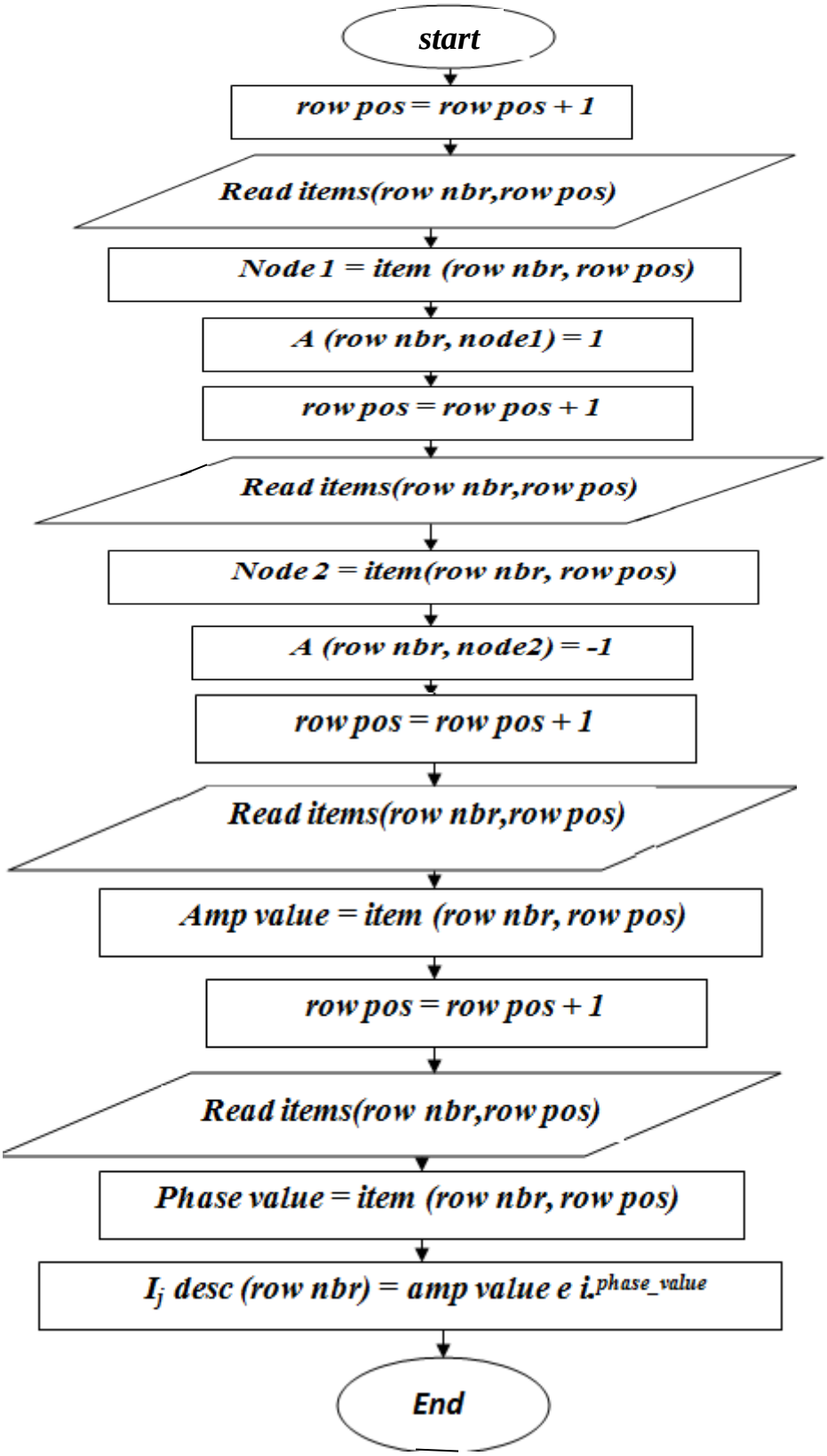


Figure 2.6: Flowchart of *Getting A and IJ description subroutine*

(c) Get A and G description subroutine: This subroutine tends to get the required reduced incidence matrix A, and the diagonal matrix containing all conductances G ***(See flowchart 4)***

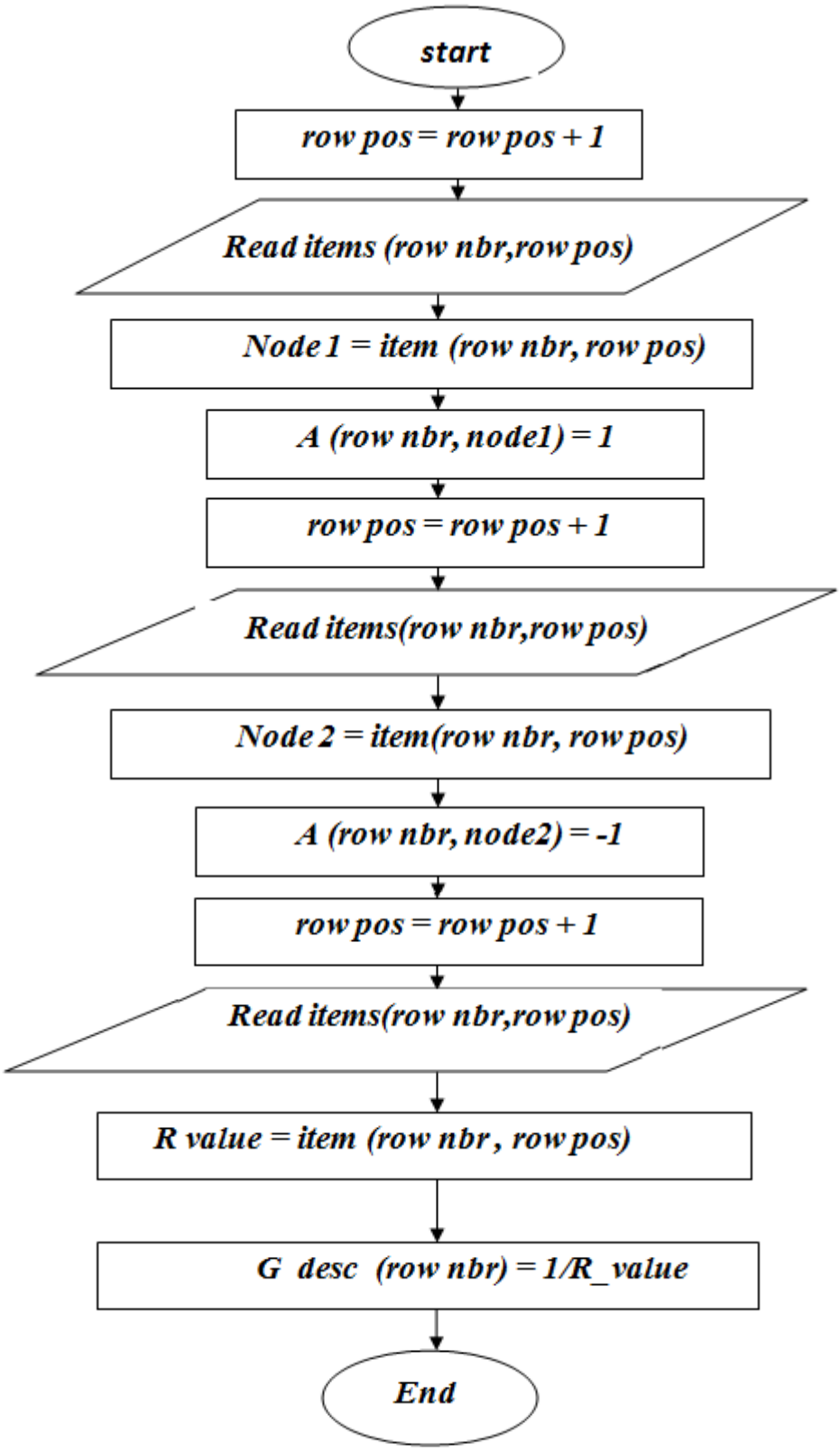


Figure2.7: Flowchart of Getting A and G description subroutine

(d)Get A and C description subroutine: This subroutine tends to get the required reduced incidence matrix A, and the diagonal matrix containing all capacitances ***C (See flowchart 5)***

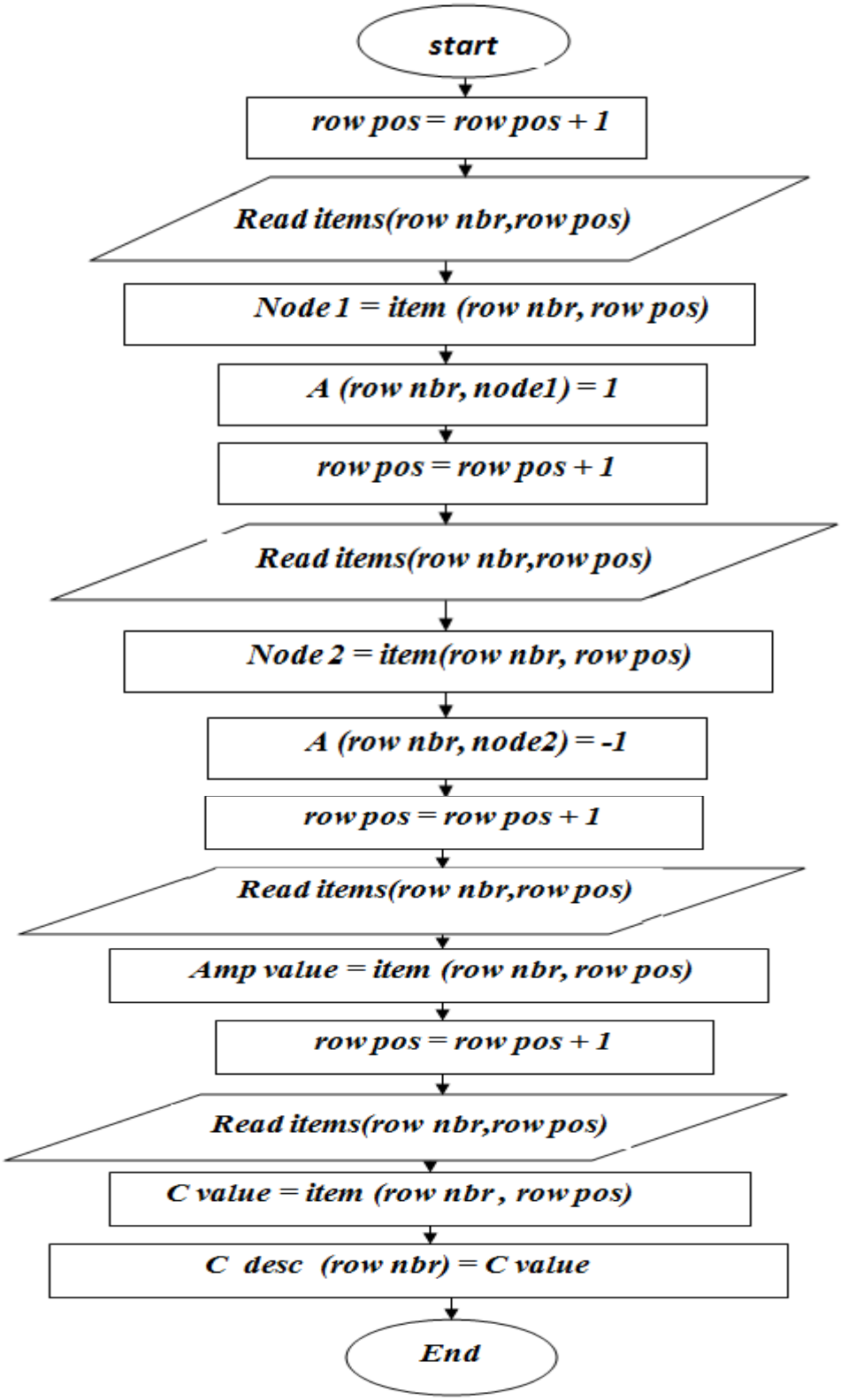


Figure 2.8:Flowchart of *Getting A and C description subroutine*

(e) Get A and gamma description subroutine: *This subroutine tends to get the required reduced incidence matrix A, and the diagonal matrix of inverse of inductivities gamma* ***(See flowchart 6)***

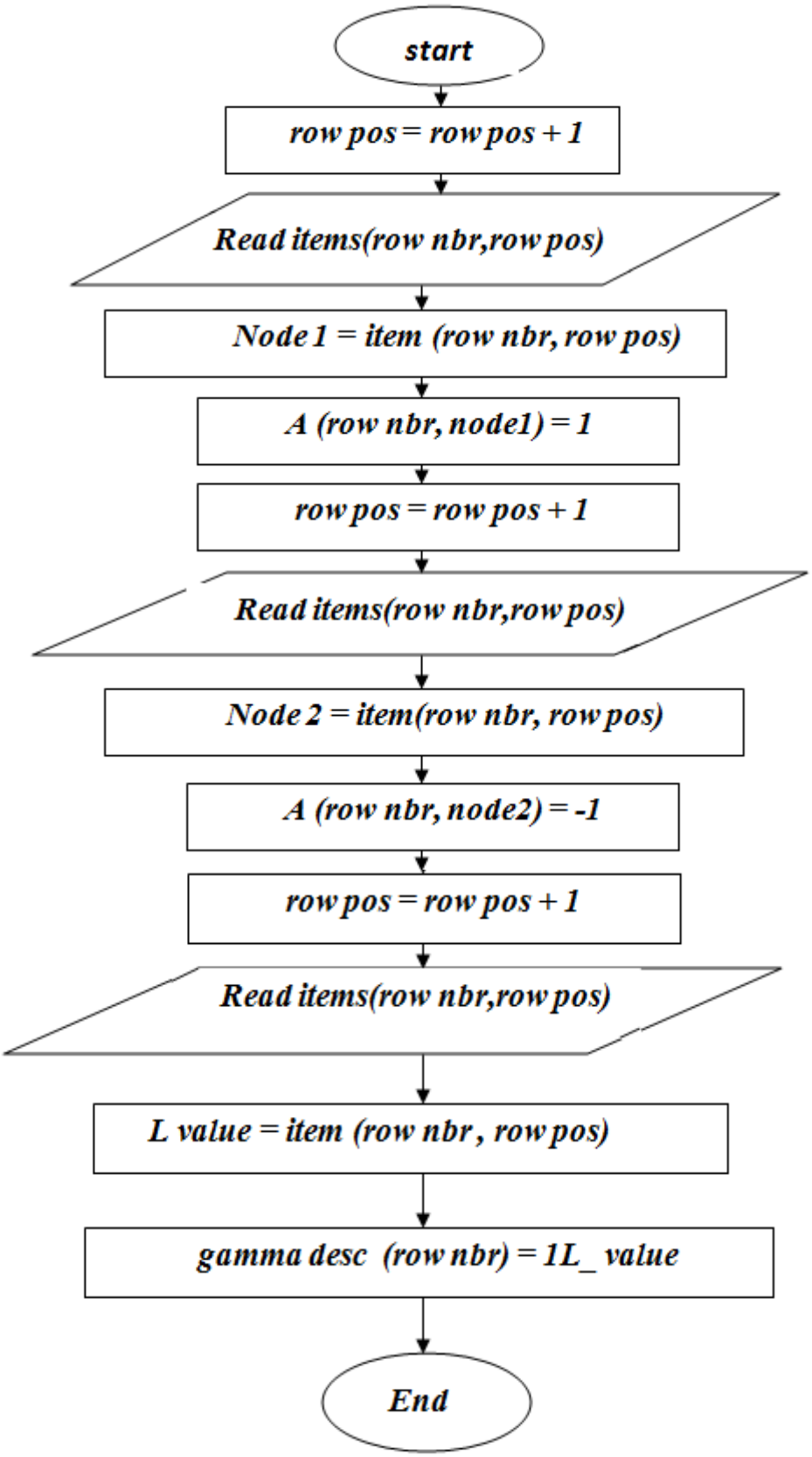


Figure 2.9: flowchart of *Getting A and gamma*

(e) Get K and gamma description subroutine:This subroutine tends to get the required mutual effect, and the diagonal matrix of inverse of gamma **(see flowchart 7)**

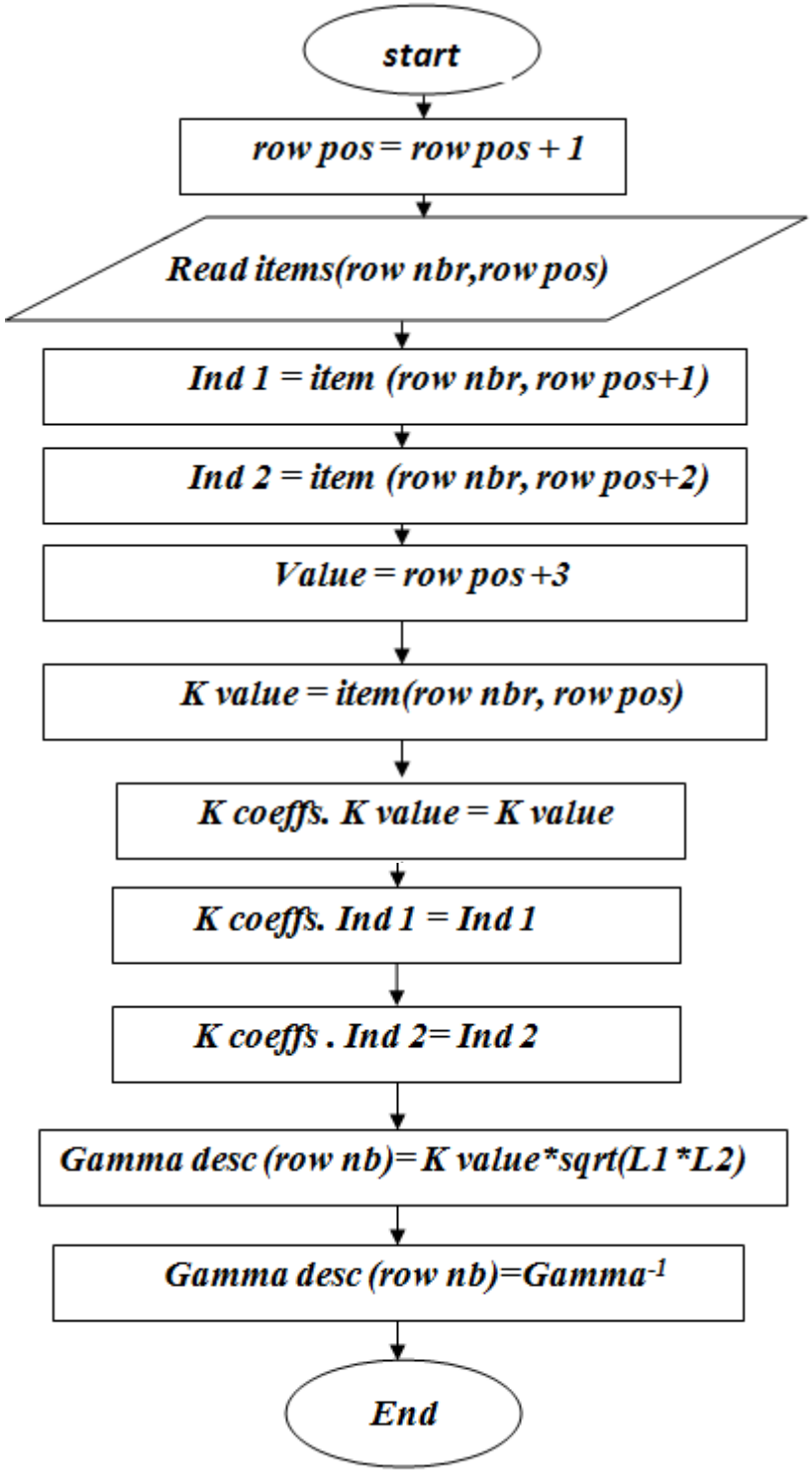


Figure 2.10: Flowchart of getting k and gamma

(f)Get frequency description subroutine: This subroutine tends to get the value of ***the*** frequency of the harmonic sources ***(See flowchart 8)***

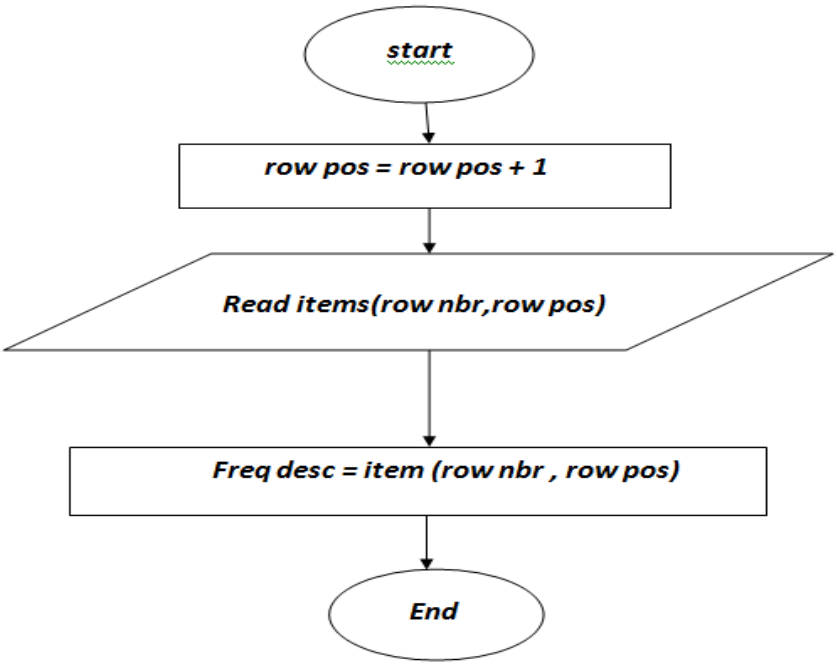


Figure 2.11: Flowchart of getting frequency description subroutine

(B) Circuit analysis based on MNA technique subroutine

This subroutine performs the time and frequency analysis of the network based on the MNA model described before ***(See flowchart 9)***

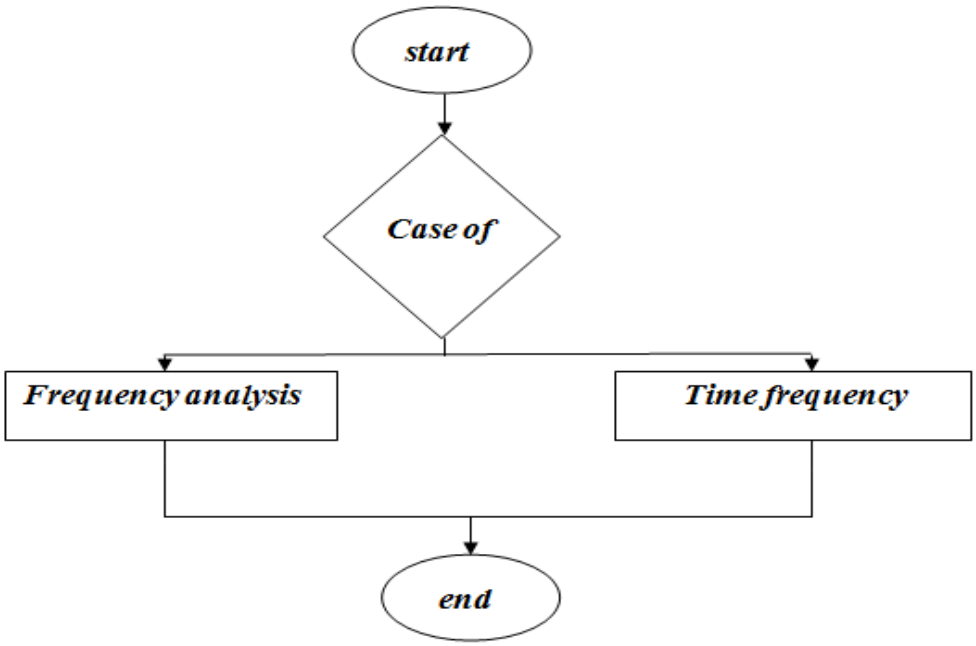


Figure 2.12: Flowchart of time analysis subroutine

(a)Time analysis subroutine: This subroutine solves the network in time domain ***(See flowchart 10)***

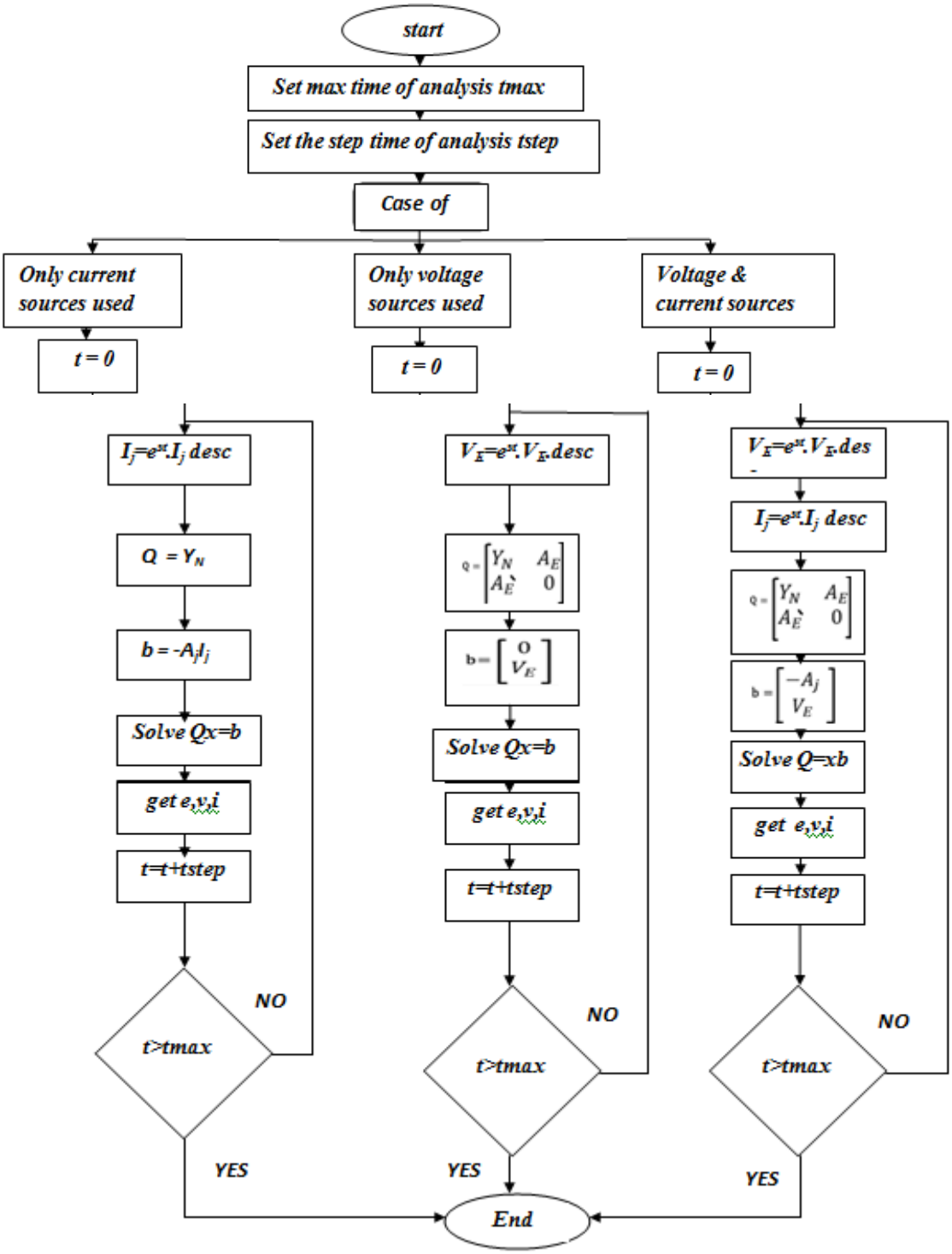


Figure 2.13 Flowchart of time analysis subroutine

(b) Frequency analysis subroutine: This subroutine solves the circuit in frequency domain ***(See flowchart 11)***

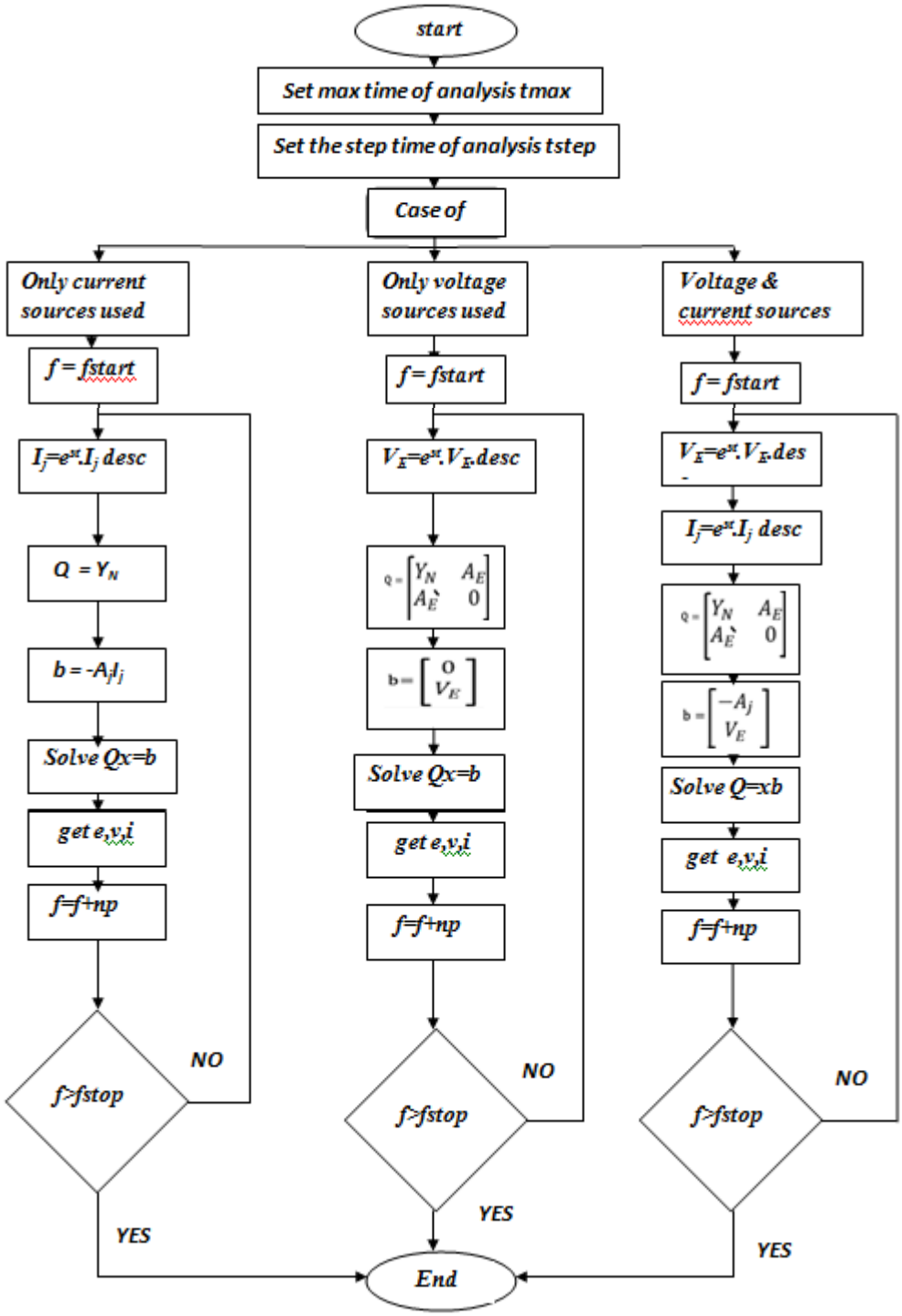


Figure2.14 Flowchart frequency analysis subroutine

(C)Subroutine that shows the results obtained on the GUI

This subroutine shows the results obtained after the network analysis process using the Graphical User Interface of MATLAB as shown in Fig 2.3.

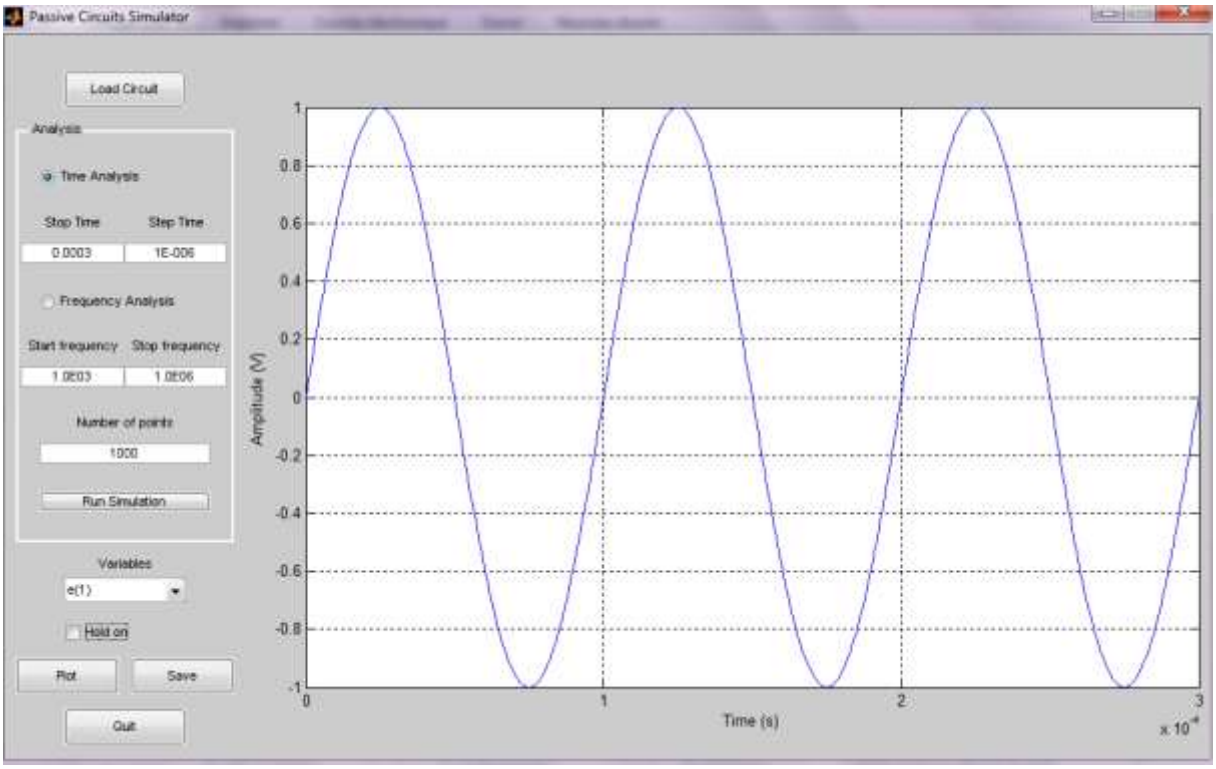


Figure2.15: the result obtained are shown in the GUI

2.8 Conclusion:

A detailed description of the implemented simulator was given in this chapter, starting from the way to use this simulator, then a description of the Modified Nodal Analysis model used and finally the algorithm implementation steps based on this model.

Testing and results

3.1 Introduction

In this chapter, we will test our implemented simulator "Netsim" by taking several passive networks as example and compare the obtained results with those obtained from another SPICE based simulator (ngspice).

3.2 ngspice

Ngspice is a general-purpose circuit simulation program for nonlinear and linear analyses. Circuits may contain resistors, capacitors, inductors, mutual inductors, independent or dependent voltage and current sources, loss-less and lossy transmission lines, switches, uniform distributed RC lines, and the five most common semiconductor devices: diodes, BJTs, JFETs, MESFETs, and MOSFETs.

Ngspice is an update of Spice3f5, the last Berkeley's release of Spice3 simulator family.

Ngspice is being developed to include new features to existing Spice3f5 and to fix its bugs.

3.3 Simulation of a simple RC filter

Let's take the RC network with a voltage source shown in Figure 3.1 and we will simulate it using our MATLAB based simulator and the ngspice then compare the obtained results.

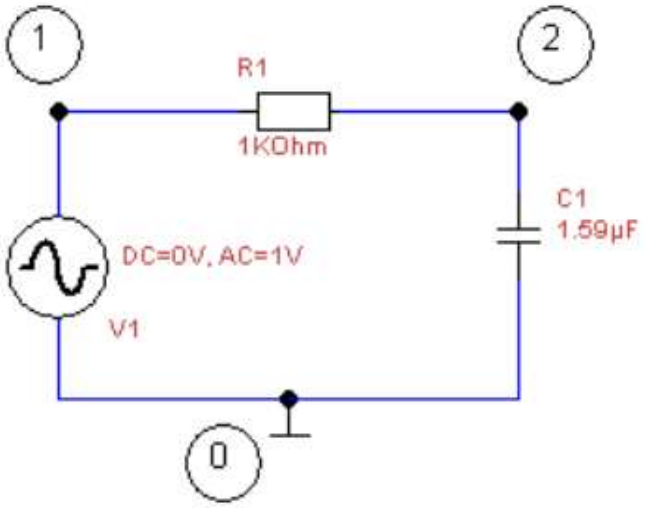


Figure 3.1 RC filter

3.3.1 Using our simulator

First let's write the equivalent netlist for this circuit:

```
VS      1       0       1       0
R1      1       2       1000
C1      2       0       1.59e-6
F       10e3
```

The resulting file has an extension of .cir as shown in Figure 3.2

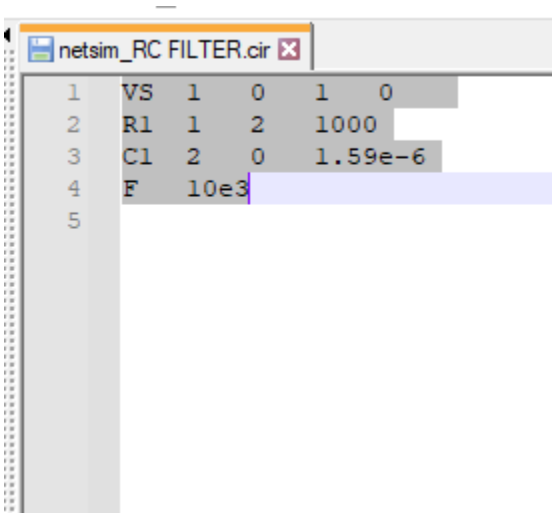


Figure 3.2 Netsim Netlist file of the RC filter

Then we load the file as shown in Figure 3.3

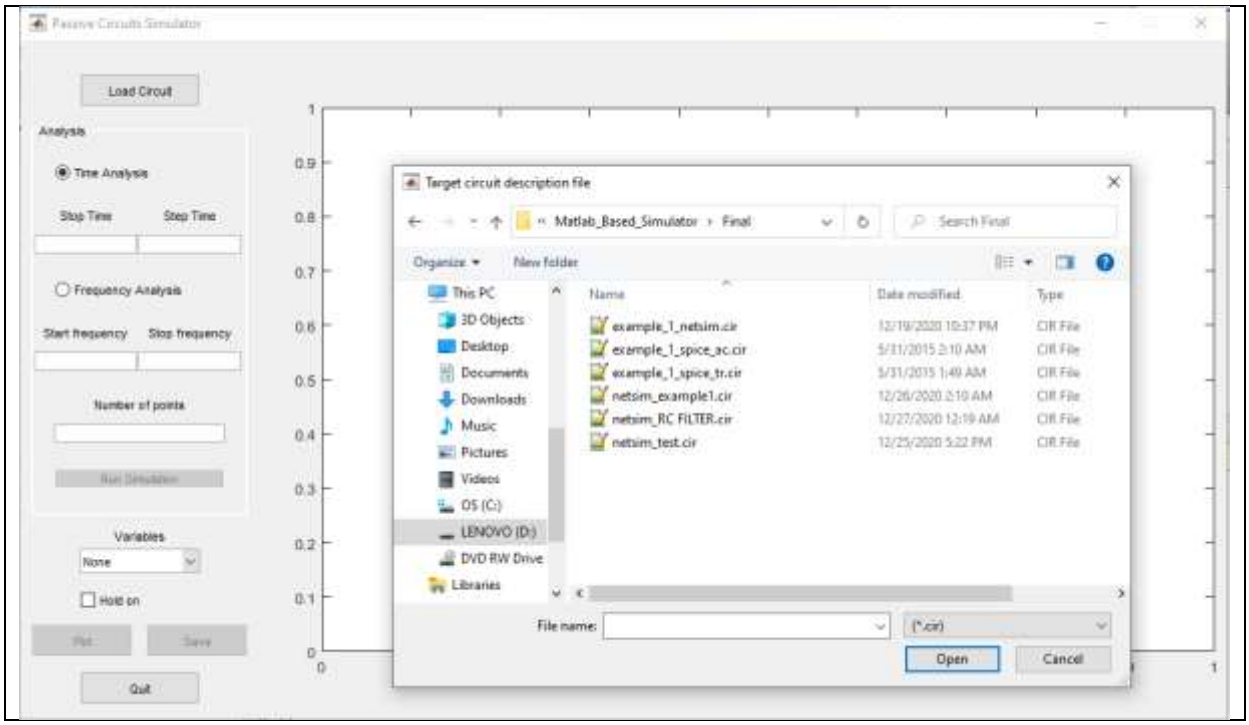


Figure 3.3 Loading the netlist file

Once the file is loaded, we can choose either time or frequency analysis mode of simulation and set the required parameter values accordingly.

3.3.2 Using ngspice

To simulate with gnspice, first we run the program, then we load the netlist file using the source command as shown in Figure 3.4. The netlist file is shown in figure 3.5.

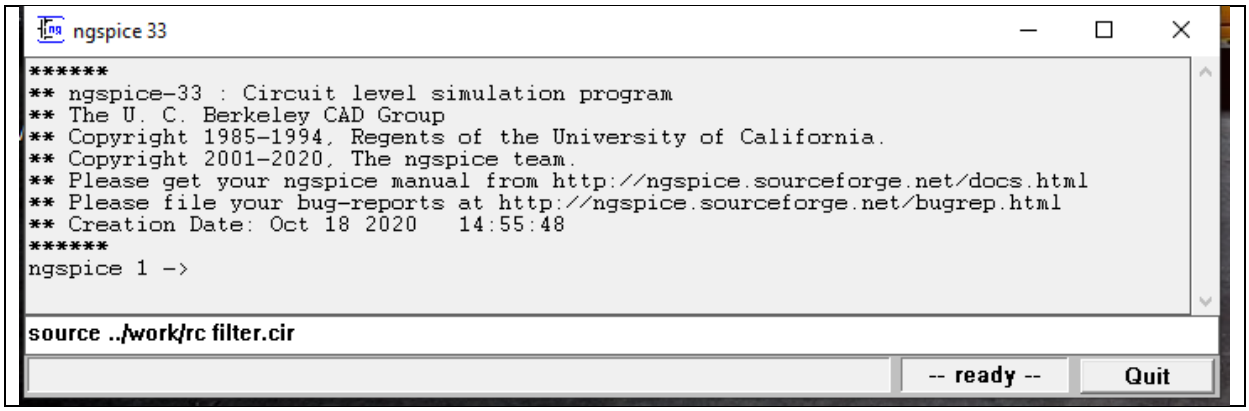


Figure 3.4 loading the netlist file using ngspice

rc filter.cir

```
1   * RC filter
2
3   V1   1    0    dc   0    ac 1
4   R1   1    2    1k
5   C1   2    0    1.59u
6   .control
7   ac   dec 41   10   100k
8   set color0=rgb:FF/FF/FF
9   plot vdb(2)
10  plot 180/PI*phase(v(2))
11  .endc
12  .end
```

Fig 3.5 the ngspice netlist file of the RC filter

The simulation runs automatically and results are given as plots.

3.3.3 Simulation results

Simulation in frequency domain is carried out using ngspice and netsim and results are presented on figure 3.6(a) and (b). As it is clearly shown on the figures the results are strictly similar to each others.

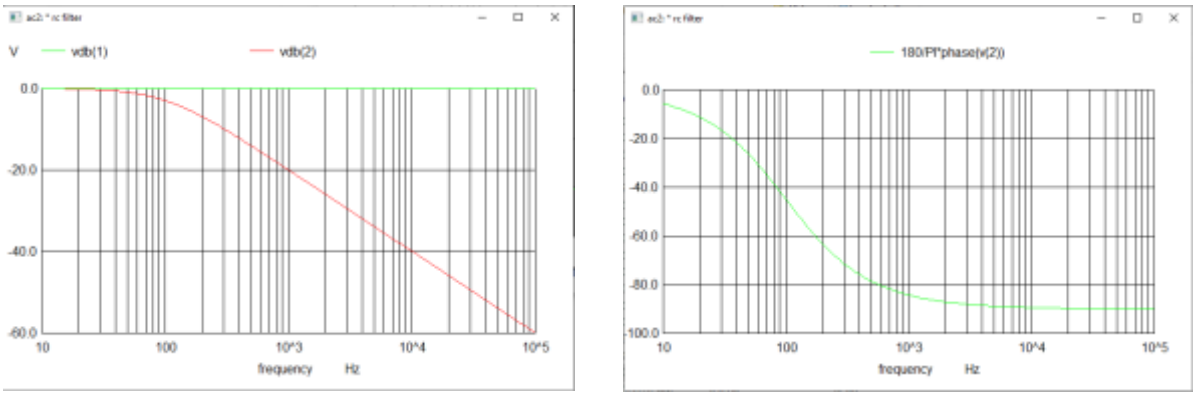


Figure 3.6 (a) RC filter frequency response simulation using ngspice

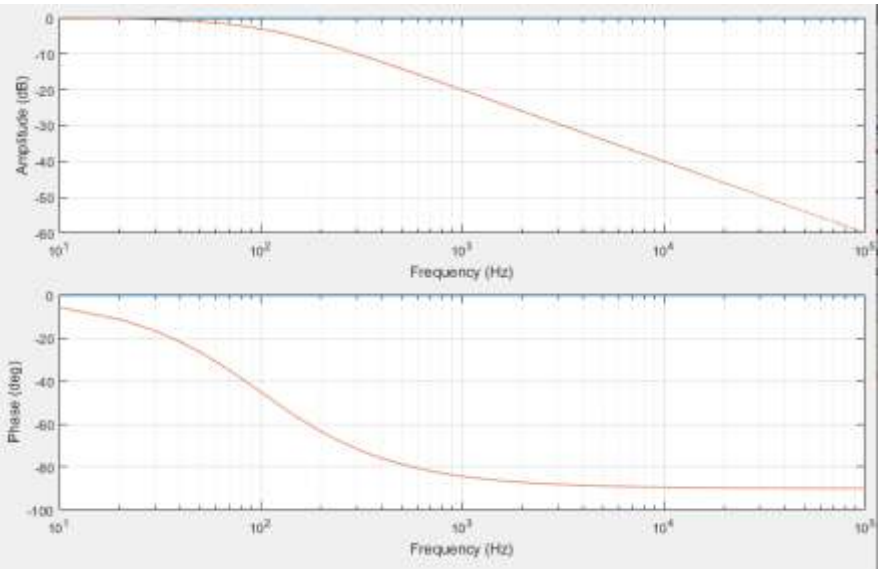


Figure 3.6 (b) RC filter frequency response simulation using netsim

3.4. Simulation of a T-Network

Let's take now the T-network with a voltage source as shown in Figure 3.7. As we did previously, we will simulate this network using our MATLAB based simulator and the ngspice in both time domain and frequency domain then compare the obtained results.

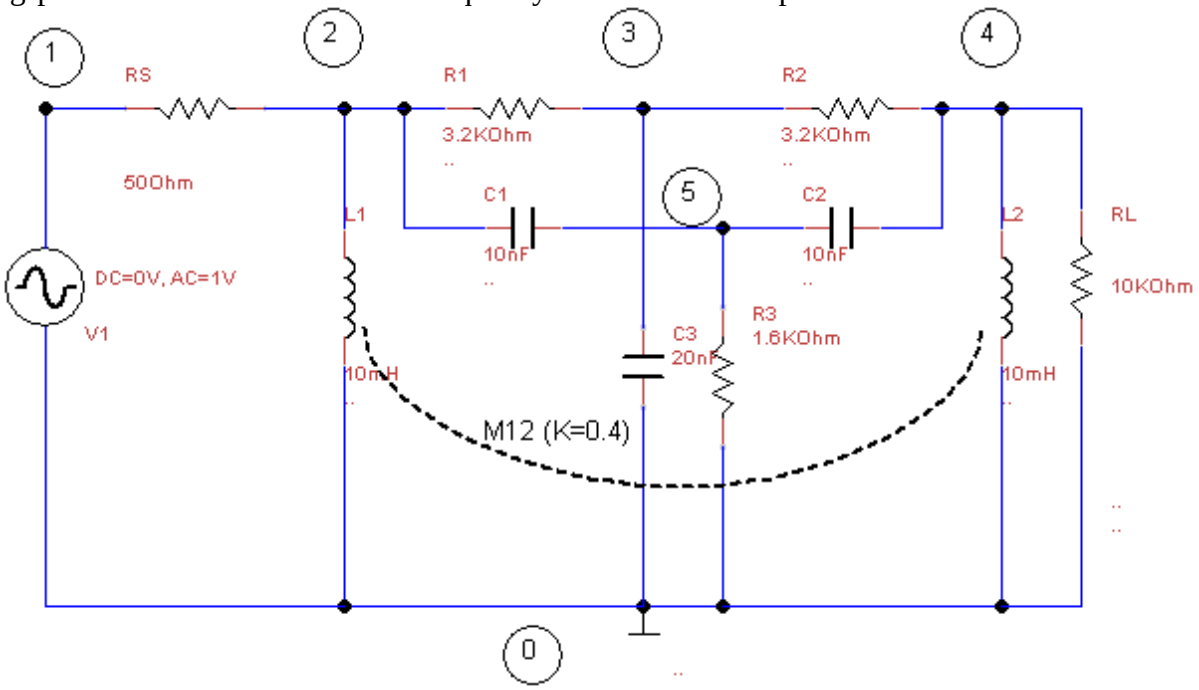


Figure 3.7. The T-Network

3.4.2.Timedomain analysis: The results obtained after a time domain analysis mode of simulation is performed are shown in the following figures:

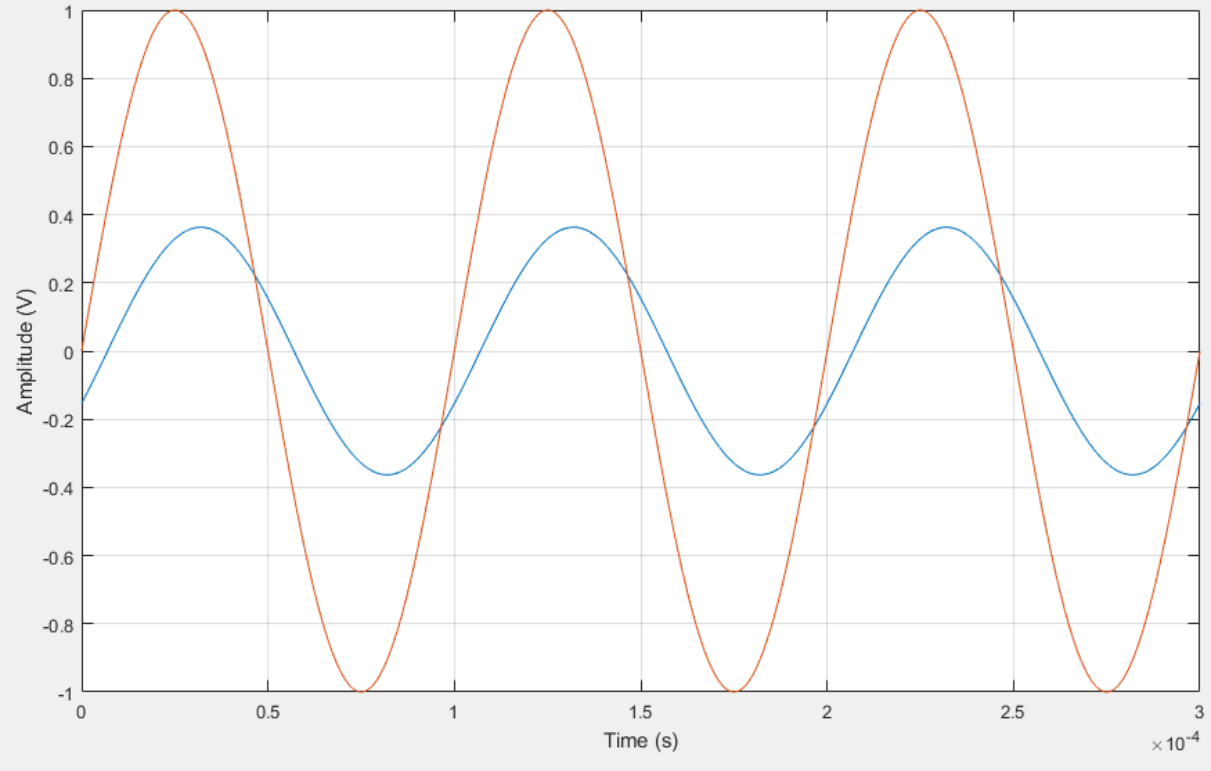


Figure 3.8. Transient analysis using Netsim. Excitation in Red, Response in Blue

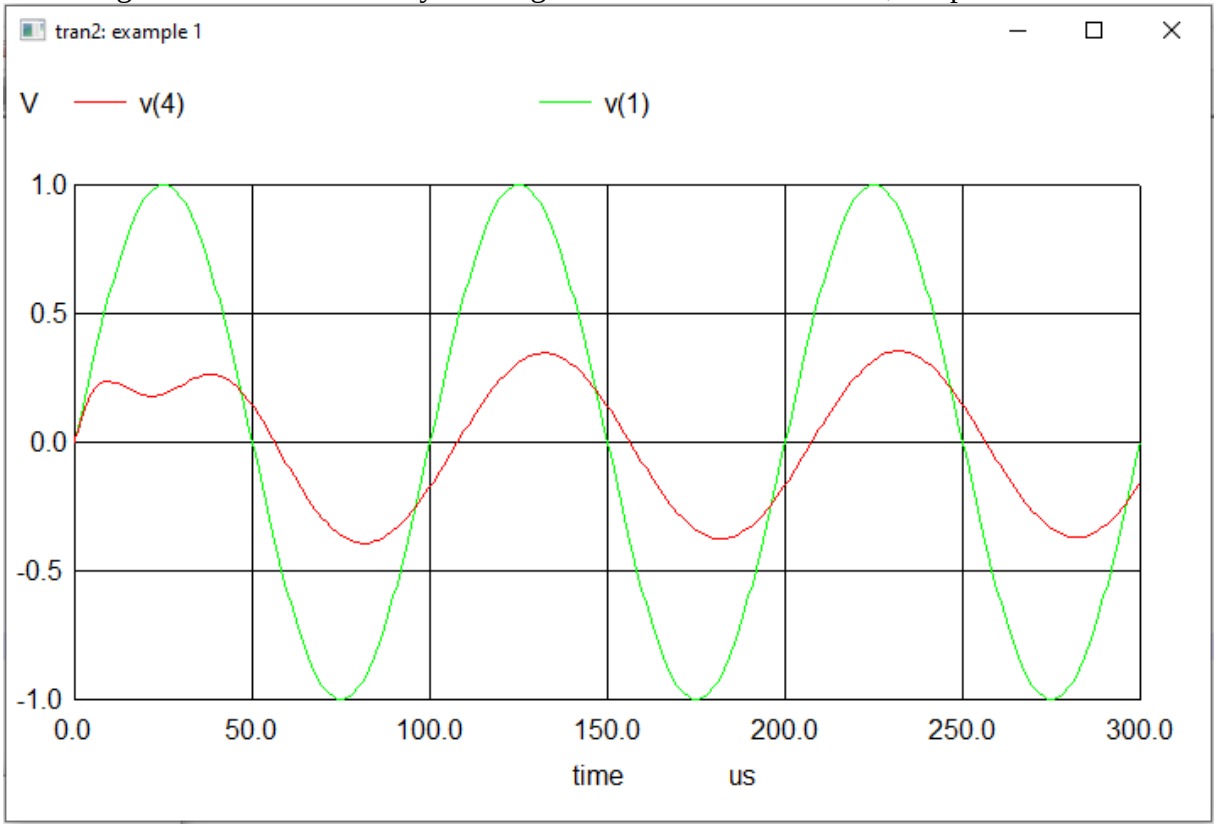


Figure 3.9. Transient analysis using ngspice. Excitation in Green, Response in Red

3.4.3. Frequency domain analysis: The results obtained after a frequency domain analysis mode of simulation is performed are shown in the Figures below:

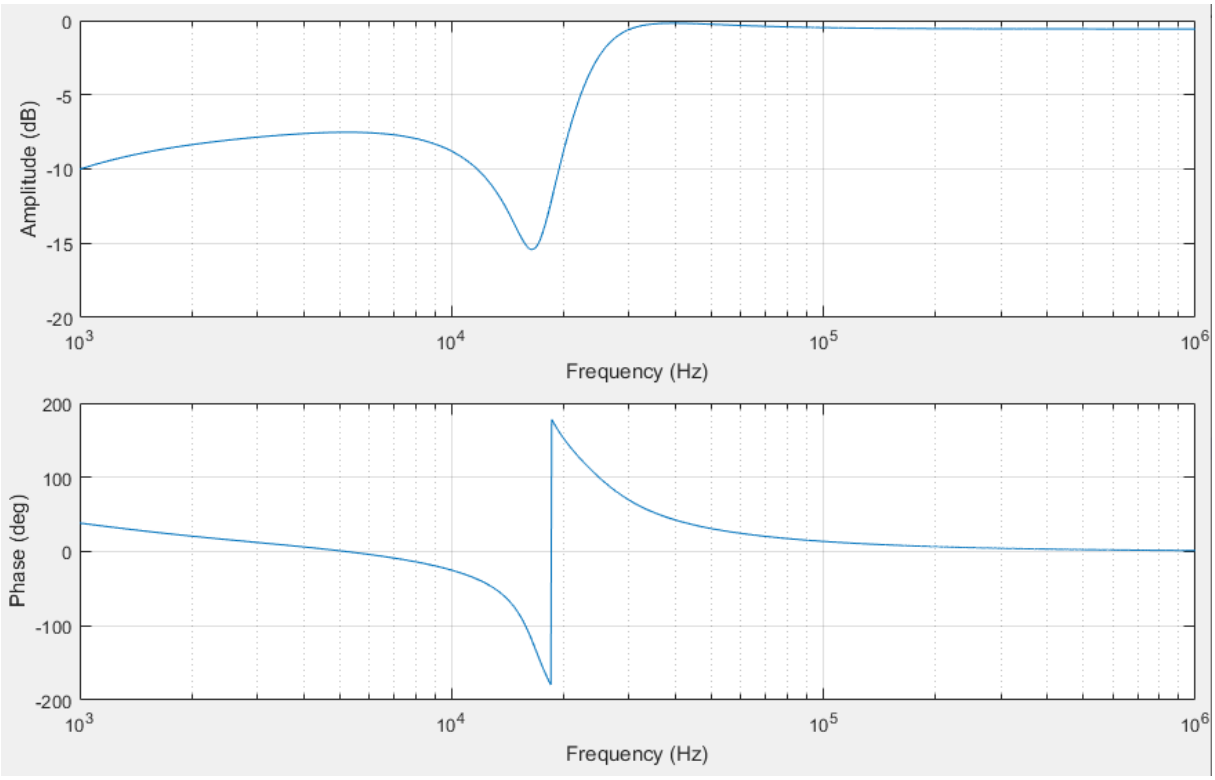


Figure 3.10. ac analysis using Netsim

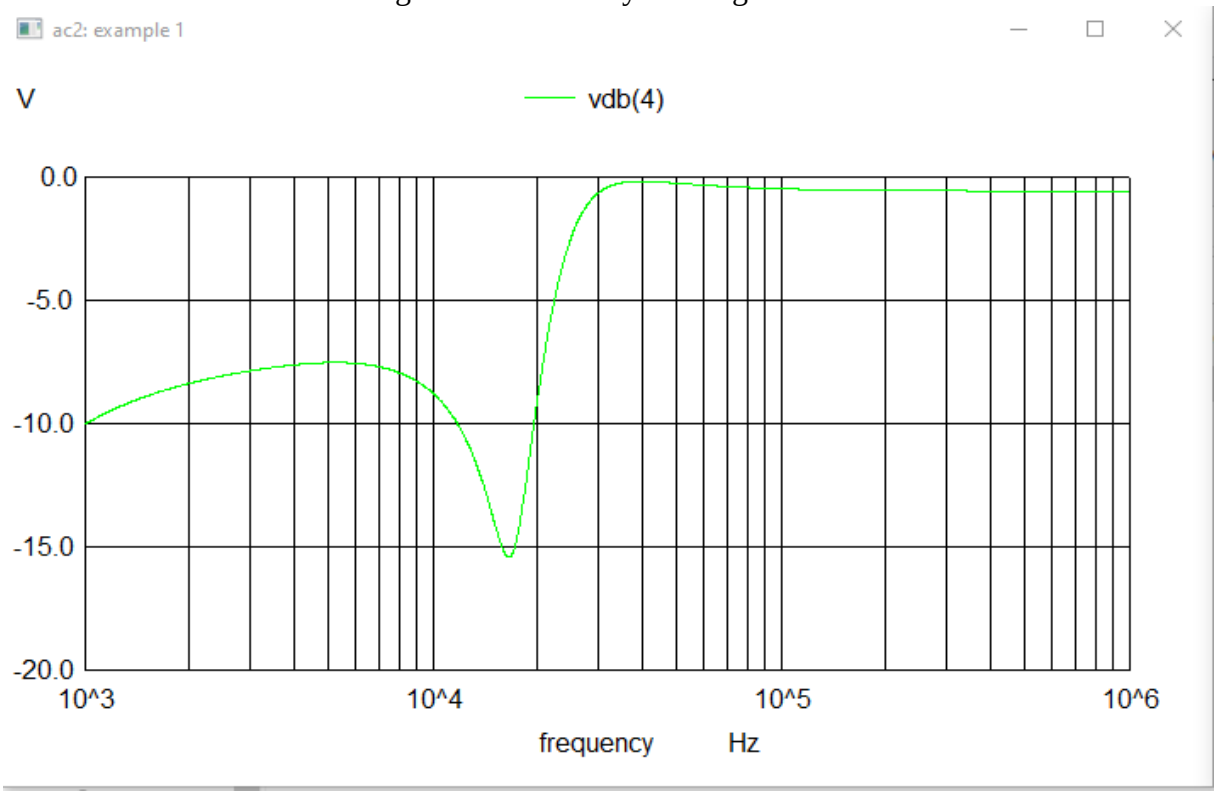


Figure 3.11. ac analysis using ngspice (gain)

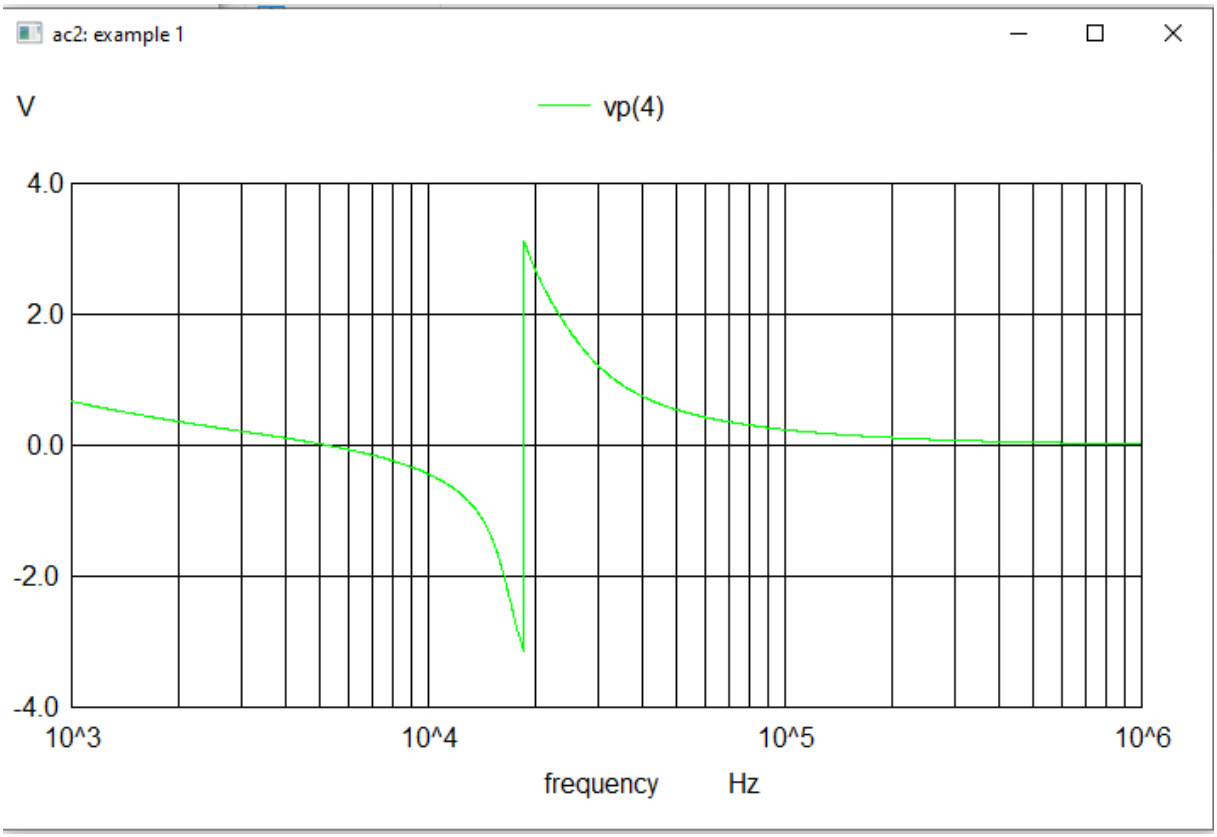


Figure 3.12. ac analysis using ngspice (phase in rad)

As it is shown in these figures, in frequency domain analysis, the results obtained by our simulator are rigorously the same to those obtained by the ngspice simulator. The same thing applies to the transient analysis except that for a certain time interval at the beginning of the simulation. This is due to the fact that ngspice requires a certain time for the solution to converge which is not the case in our simulator.

3.5 Conclusion

The results obtained from our MATLAB based simulator are very satisfactory comparing to those obtained from the ngspice simulator for both time and frequency analysis with a difference in the transition period before reaching the steady state, this difference is due to the fact that the ngspice (and all the SPICE based simulators) uses the differential algebraic

equations system (described in chapter 1) to solve all kind of circuits, that's why it takes this transition time to converge to the optimal solution, which is not the case in our implemented simulator , where we have used a suitable model of modified nodal analysis for our specific purposes.

Conclusion

Throughout this project, the main goal was to build a MATLAB based passive linear circuits simulator, with the case of harmonic sources.

Firstly, a general overview about the different topological network analysis techniques used to solve electrical network was given, with a special interest on the modified nodal analysis technique, used widely by electronic software simulators.

Secondly, a detailed description of the simulator is given, the netlist representation of circuits, the MNA model used, the algorithmic implementation steps, and the final MATLAB code are developed.

Finally, tests were performed to determine the reliability of our product and the results obtained were very satisfactory.

There is still much work to be done for future development that would enhance this work and increase its efficiency. For example expand the range of proposed tools, i.e.: our simulator deals only with a few number of electronic devices (capacitors, resistors and inductors) and supports only harmonic and sine wave sources, the general model of the modified nodal analysis (the differential algebraic equations system described in chapter one) can be used to deal with any type of sources.

APPENDIX

1-ECHELON Routine

```
% This function uses the echelon algorithm to
%
% detrmine a tree and its corresponding cotree of a
%
% graph from the node to branch incidence matrix.
%
% usage: [tree cotree]=echelon(A). where,
%
%        A is the node to branch incidence matrix (reduced form)
%        tree and cotree are arrays representing repectively the tree
%        and cotree sets of branches.
%
% For further details, please refer to the "Electrical Networks Theory and
Analysis"
% book, by John Choma, Jr., a Wiley-Interscience Publication, John Wiley &
Sons,
% (c) 1985, ISBN 0-471-08528-6.
%
%This function has been implemented by Mrs. A. HAMADI, Master Student at
IGEE/UMBB


function [tree, cotree]=echelon(A)
```

```
% Initialize the tree array.
tree=[];

% Initialize the cotree array.
[nr nc]=size(A);
cotree = 1:nc;

% Get a copy of the original matrix

copyA=A;

% Redo the same process until the matrix copy is empty.

while~isempty(copyA)

% Initialize the colunm position and the row position.
    colPos=1;
    col=copyA(:,colPos);
    rowPos=find(col);

% scan the columns of A from the left to the right until a column (colPos)
that has
% nonzero entries is found.
% scan the column (col) just found from the top until a nonzero element is
found.
```

```
% rowPos(1) is the index of the first nonzero element in column col.


        while length(rowPos)==0
         colPos=colPos+1;
         col=copyA(:,colPos);
         rowPos=find(col);
        end


% Update the tree array to include the index that corresponds to colPos.


        tree=[tree cotree(colPos)];


% Process the row at position rowPos(1) so that the first element is unity.


        row=copyA(rowPos(1),:);
        row=row(1)*row;


% If rowPos(1) is not unity then interchange the first row and row at
position rowPos(1).


        if rowPos(1)~=1
                row=copyA(rowPos(1),:);
         copyA(rowPos(1),:)=copyA(1,:);
        end
```

```
% Reduce the second non-zero element (if any) below row 1 in the column
upon which
% attention is focused to zero by elementary row operations.

        if length(rowPos)==2
         copyA(rowPos(2),:)=copyA(1,:)-
copyA(rowPos(2),colPos)*copyA(rowPos(2),:);
        end

% Delete row 1 and column colPos (which is a twig) from copyA.

    copyA(1,:)=[];
    copyA(:,colPos)=[];

% Delelte the branch index colPos from the cotree array

   cotree(colPos)=[];

end
```

2-MESH ANALYSIS

```
% Analysis by the Mesh method - Elliptic Network
%
% This Script implements in the time domain the response of an elliptical
filter
% at AC (sinusoidal) excitation.
%
% Inputs: - Source frequency (F)
% - The amplitude of the source (VM)
% - The load value (RL)
% - Number of cycles (periods) in the simulation interval (NC)
% - Number of simulation points per cycle (NS)
% Outputs: - Graphs of the excitation vs (t) is of the response vL (t) as a
function
%             time.


clear all;
clc; % Clear the screen


% Fixed circuit data



RS = 50; % Source resistance
rs = 1.0e-3;
C1 = 93.9e-9; % C1 = 93.9nF
```

```
C2 = 93.9e-9; % C2 = 93.9nF

C3 = 17.8e-9; % C3 = 17.8nF

L1 = 1.79; % L1 = 1.79H


% Get entries


F = input ('Enter the frequency value in Hz:');

T = 1 / F; % The source period

s = 1i * 2 * pi * F; % The la varieble complex

VM = input ('Enter the value of the source amplitude in V:');

RL = input ('Enter the value of the load in Ohm:');

NC = input ('Enter the number of cycles you want for the simulation:');

NS = input ('Enter the number of simulation points per cycle:');


% Determine the total number of simulation points


N = NC * NS;


% Create the simulation time interval


t = inspace (0, NC * T, N);


% Create the complex vector of the excitation source


VS = VM * exp (s * t);
```

```
% Build A

A = [1, 0, 0, 0, 0, 0, 1;
   -1, 0, 1, 0, 1, 1, 0;
    0, 1, 0, 1, -1, -1, 0];

% In order to give Y the diagonal form Block, o reorders the branches
%
% so that the branch (7) becomes a resistive branch.

order_branches = [1, 2, 7, 3, 4, 5, 6];

A = A (:, order_branches);

% Construct the main diagonal of Z

R = diag ([RS, RL, rs]);
D = diag ([1 / C1.1 / C2.1 / C3]);
L = diag ([L1]);

% Build Z, BL, IE and VE (initial)

Z = blkdiag (R, D / s, s * L);
```

```
% Determine a tree of the graph and proceed to the partitioning of A.
%
% A = [At | Ac].
%
% For this we call the echelon.m function which implements
% the step algorithm.

[tree, co-tree] = echelon (A);
At = A (:, tree);
Ac = A (:, co-tree);
[nrows, ncols] = size (Ac);
% Reorder Z
Z = Z (:, [tree, co-tree]);
% Build Bf = [Bt | U].
Bf = [(At \ Ac) ', eye (ncols)];
BL = Bf * Z * Bf ';
IE = zeros (7.1);
VE = zeros (7.1);

% Initializes the solution matrices

I = []; % Circuit currents
V = []; % Circuit voltages
```

```
for k = 1: N
    % update the 3rd element of VE which initially corresponds to the
    % branch containing the excitation VS

    VE (3) = - VS (k);

    % reorder IE and VE

    IE = IE ([tree, co-tree]);
    VE = VE ([tree, co-tree]);
VL = Bf * (VE-Z * IE); % build VL
    ic = BL \ VL;
    i = Bf '* ic;
    v = Z * (i + IE) -VE;

    % Update the solution matrices by adding a
    % new column (the solution at time t (k)).

    I = [I, i];
    V = [V, v];
end

% Obtain the instantaneous values vs (t) of VS (excitation) and vl (t) of V
(2)
% (reply). It suffices to extract the real parts of the vectors
```

```
% complex


% First, restore the original order


V = V (order branches, :);

I = I (order branches, :);


vl = imag (V (2, :));

vs = imag (VS);


% Draw the graphs




plot (t, vs, 'k -', t, vl, 'k--');

grid on

xlabel ('temp (sec)');

ylabel ('amplitude (v)')

title ('Response of the Elliptical filter to a sinusoidal excitation');

legend ('Excitation', 'Response');
```

3-CUT SET

```
% Analysis by the Cutoff method - Elliptic Network
%
% This Script implements in the time domain the response of an elliptical
filter
% at AC (sinusoidal) excitation.
%
% Inputs: - Source frequency (F)
% - The amplitude of the source (VM)
% - The load value (RL)
% - Number of cycles (periods) in the simulation interval (NC)
% - Number of simulation points per cycle (NS)
% Outputs: - Graphs of the excitation vs (t) is of the response vL (t) as a
function
%             time.


clear all;
clc; % Clear screen


% Fixed circuit data
RS = 50; % Source resistance
rs = 1.0e-3;
C1 = 93.9e-9; % C1 = 93.9nF
C2 = 93.9e-9; % C2 = 93.9nF
C3 = 17.8e-9; % C3 = 17.8nF
```

```
L1 = 1.79; % L1 = 1.79H


% Get entries


F = input ('Enter the frequency value in Hz:');

T = 1 / F; % The source period

s = 1i * 2 * pi * F; % The la varieble complex

VM = input ('Enter the value of the source amplitude in V:');

RL = input ('Enter the value of the load in Ohm:');

NC = input ('Enter the number of cycles you want for the simulation:');

NS = input ('Enter the number of simulation points per cycle:');


% Determine the total number of simulation points


N = NC * NS;


% Create the simulation time interval


t = inspace (0, NC * T, N);


% Create the complex vector of the excitation source


VS = VM * exp (s * t);


% Build A
```

```
A = [1, 0, 0, 0, 0, 0, 1;
   -1, 0, 1, 0, 1, 1, 0;
    0, 1, 0, 1, -1, -1, 0];

% In order to give Y the diagonal form Block, o reorders the branches
%
% so that the branch (7) becomes a resistive branch.

order_branches = [1, 2, 7, 3, 4, 5, 6];

A = A (:, order_branches);

% Construct the main diagonal of Y

G = diag ([1 / RS, 1 / RL, 1 / rs]);
C = diag ([C1, C2, C3]);
GAMMA = diag ([1 / L1]);

% Build Z, BL, IE and VE (initial)

Y = blkdiag (G, s * C, GAMMA / s);
```

```
% Determine a tree of the graph and proceed to the partitioning of A.
%
% A = [At | Ac].
%
% For this we call the echelon.m function which implements
% the step algorithm.

[tree, co-tree] = echelon (A);
At = A (:, tree);
Ac = A (:, co-tree);
[nrows, ncols] = size (At);
% Reorder Y
Y = Y (:, [tree, co-tree]);
% Build D = [U | Dc].
D = [eye (ncols), - At \ Ac];
Dcut = D * Y * D ';
IE = zeros (7.1);
VE = zeros (7.1);

% Initializes the solution matrices

I = []; % Circuit currents
V = []; % Circuit voltages

for k = 1: N
```

```
    % update the 3rd element of VE which initially corresponds to the
    % branch containing the excitation VS

    VE (3) = - VS (k);

    % reorder IE and VE
IE = IE ([tree, co-tree]);
    VE = VE ([tree, co-tree]);

    Icut = D * (IE-Y * VE); % build Icut
    vt = Dcut \ Icut;
    v = D '* vt;
    i = Y * (v + VE) -IE;

    % Update the solution matrices by adding a
    % new column (the solution at time t (k)).

    I = [I, i];
    V = [V, v];
end

% Obtain the instantaneous values vs (t) of VS (excitation) and vl (t) of V
(2)
% (reply). It suffices to extract the real parts of the vectors
% complex
```

```
% First, restore the original order

V = V (order branches, :);
I = I (order branches, :);

vl = imag (V (2, :));
vs = imag (VS);

% Draw the graphs



plot (t, vs, 'k -', t, vl, 'k--');
grid on
xlabel ('temp (sec)');
ylabel ('amplitude (v)')
title ('Response of the Elliptical filter to a sinusoidal excitation');
legend ('Excitation', 'Response');
```

References

(1)Design and Implementation of a MATLAB Based passive Network simulator

(Final year project report) _CHIBANE HOCINE, HAMADACHE WALID.

***(2)*Dr.K chennavenkatesh, MSRIT B'lore, "Network analysis", VTU e-learning center.**

(3)Michael Hanke,"An Introduction to the Modified Nodal Analysis" ,May 2006.

(4)Kenneth S .Kndert,"the designer's Guide to Spice and Spectre",springer US,1995.

***(5) http://en.wikibooks.org/wiki.org* /Ordinary Differential Equations.**

(6)web.mit.edu/viz/EM/visualizations/.../guide11.PDF Mutual Inductance: - MIT

(7) Web .*C. D.Hachtel and R.K. Brayton and F. G. Gustavson,***The sparse tableau approach**

To network analysis and design," IEEE Trans.

(8) *http://encyclopedia.thefreedictionary.com/computer+simulation* Computer simulation

(9) http://www.allaboutcircuits.com/textbook/direct-current/chpt-2/ computer-simulation-electric-circuits/