Fall 12-1-2015

Virtual Electronics Laboratory for Visualized Education and Training

Sreelatha Aihloor Subramanyam

VIRTUAL ELECTRONICS LABORATORY

FOR VISUALIZED EDUCATION AND TRAINING

by

SREELATHA AIHLOOR SUBRAMANYAM

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering.

Dr. David M. Beams, Committee Chair

College of Engineering

The University of Texas at Tyler
December 2015

The University of Texas at Tyler

Tyler, Texas.

This is to certify that the Master's Thesis of

SREELATHA AIHLOOR SUBRAMANYAM

has been approved for the thesis requirement on

29th October, 2015

For the Master of Science in Electrical Engineering

Approvals:

David M. Beams

Committee Chair: David M. Beams, Ph.D.

James K. Nelson Jr.

Co-chair: Dr. James K. Nelson, Jr., Ph.D., P.E.,

Melvin Robinson

Member: Melvin Robinson, Ph.D.

Chair and Graduate Coordinator: Hassan El-Kishky, Ph.D.

James K. Nelson Jr.

Dr. James K. Nelson, Jr., Ph.D., P.E.,

Dean, College of Engineering and Computer Science

Acknowledgement

I acknowledge, with gratitude, my debt of thanks to Professor Dr. David M. Beams for his advice and encouragement and to Dr. James K. Nelson for his contribution towards the animations of the project, aid and foresight.

I would like to thank Office of Sponsored Research of The University of Texas at Tyler, Tyler, Texas for its financial support.

Lastly I would like to thank my family and friends for their support.

Table of Contents

List of Figures

Abstract

VIRTUAL ELECTRONICS LABORATORY
FOR VISUALIZED EDUCATION AND TRAINING

Sreelatha Aihloor Subramanyam

Thesis Chair: Dr.David M.Beams, Ph.D

The University of Texas at Tyler
December, 2015

Virtual laboratories for electronics introductory courses have been developed previously with static web pages, LabVIEW (National Instruments) software and extra hardware such as video cameras and video servers. Most of these approaches to building virtual laboratories for introductory electronics courses were inefficient because of the unreliability of the service or expensive because third party software had to be adopted.

In the present research work, Microsoft Visual Studio, a sophisticated software application development environment, is used for the development of a virtual laboratory. This program has components that are lively therefore; it has the capability of excellent user interface. In addition to Microsoft Visual Studio C++, MFC (Microsoft Foundation Classes) were also used to implement the design.

The virtual laboratory uses nodal analysis to derive network equations from user-connected components. These network equations include Kirchhoff's current law equations and accommodate voltage sources and current sources—i.e., those in between nodes—using constitutive equations of the circuit. These network equations are grouped to form sub matrices, which are concatenated to form a large matrix of coefficients of KCL and constitutive relations. Simultaneously the initial voltage sources and current sources are formed as a matrix. Here, the fact that the voltage in the ground node is equal

to zero allows the program to delete the ground node related row and column of coefficients related to that node voltage, eventually the unknowns are calculated by using matrix operations.

The working of proposed algorithm is verified with to a circuit made of passive elements like resistances, capacitors, inductors and also circuits that contain controlled sources. And solution of differential equations uses a Runge-Kutta method. The resultant waveforms are compared with industry standard circuit simulators.

Chapter 1

1 Introduction

Technology has brought many changes in educational systems, among which is online education. Online education permits people to study at their convenience at any location offering internet access, including their homes. They are not required to travel. But one of the developments that are yet to be achieved in online education is introduction of realistic laboratories. While there exist some remote online laboratories such as iLab [1] (for microelectronics, control systems experiments), Lab Share [2] (for use in education to gain remote access to lab technologies), Go-Lab [3] (for high school science experiments), iSchool Remote Labs [4] and UMKC Remote Labs [5] (both are used to manipulate data remotely on school computers). But none of them are applicable to introductory electric circuit labs.

1.1 Capabilities and Limitations of Present Circuit Simulators

1.1.1 Capabilities of Present Circuit Simulators

At present, remote learning in electronics relies on circuit simulator programs. Active research on electronic circuit simulators was initiated by companies like IBM and universities like University of California, Berkeley with projects funded by US Defense department back in 1960's. IBM ECAP (Electronic Circuit Analysis Program) was introduced in 1960's [6].And it was followed by SPICE (Simulated Program with Integrated Circuit Emphasis), University of California, Berkeley in 1974.SPICE is basically a circuit simulator program to build a new circuit and simulate it to analyze its circuit parameters. Most of the current circuit simulators are built on SPICE.

And there exist multiple circuit simulation environments for digital, analog and mixed-signal circuitry. Some of them are Ngspice, GNU Cap, Circuit Logix, LT Spice, Top Spice, Circuit simulator1.5j, Mac Spice, 5Spice, Beige Bag Spice, Micro-Cap 10,

PECS, Proteus, QUCS, Solve Elec, Xspice, PSpice, Multisim, SiMetrix, TINA, and PowerSim [7].

Some of them are freely available, while others are licensed versions. Among these licensed circuit simulators are the widely used PSpice (Cadence Design Systems, San Jose, CA) and Multisim (National Instruments Austin, TX). PSpice is SPICE ported to the PC, introduced by MicroSim in 1984. MicroSim was later acquired by Cadence Design Systems [8] [9]. Multisim is derived from Electronic Workbench [10].

Capabilities of Multisim:

Multisim is widely used by designers, researchers, students, academicians and professionals. Its rich features are tailored according to the needs of different user requirements. Multisim is available in two versions, the designer version and the student version.

The features of the designer's version are [11]:

Designer's versions of Multisim have enhanced features that enable designers, to readily implement their ideas on circuits and analyze them with different variations in circuit parameters.

(a) *Industry standard component list*: This helps the design, prepared with these components to be similar to industry standard designs.
(b) *Customized symbols:* The designer is given the privilege to create his or her own symbols and make use of them in designs when standard symbols are not suitable.
(c) *Multiple designs:* Designers may create complex circuits in multiple schematics and still work on all of them together. They can compare their results.
(d) *Interface with other simulation environments* [12]: Multisim is capable of importing OrCAD designs and SPICE netlists. Netlists of SPICE programs may be imported into Multisim using arbitrary blocks or by saving the Netlist as a circuit (.cir extension).

(e) *Advanced analysis of the circuit*: Multisim permits users to examine minute details of the circuit.

(f) Easy debugging: National Instruments (NI) Multisim debugging tools help to trace errors such as 'time step too small to simulate' and disable simulation when circuits are not properly connected [13].

(g) Schematic Capture: From the given component list, users may easily draw the circuit. It is easy to locate different components that are readily available

(h) Circuit simulation: Circuit simulation is simple, and users may easily verify their expected results with circuit simulation results.

(i) Standard Component list: Student version of Multisim is provided with component lists and components lists that are manufacturer specific.

(j) 3D animations: 3D animations of prototype board, circuit components help students to easily visualize real components.

The student version of Multisim is made available at no charge but with limited access to components lists, circuit simulation tools and the like.

(a)Integration with NI ELVIS: Multisim may be integrated with NI ELVIS (National Instruments Educational Laboratory Virtual Instrumentation Suite).

Capabilities of Cadence PSpice [14] [15]:

Cadence PSpice is an electronic circuit simulation environment that simulates analog circuits, digital circuits, mixed signal circuits.

(a) PSpice is linked with Math Works MATLAB using Simulink to PSpice Interface (SLPS) link. The MATLAB Simulink interface is helpful in testing system level interfaces with electrical designs that are similar to real world applications.

(b) PSpice may encrypt circuit design models so that Intellectual Property (IP) is protected.

(c) Smoke Analysis is a feature in PSpice with which warns of electrically over stressed components.

(d) Component yield failures may be identified with Monte Carlo analysis.

1.1.2 Limitations of Present Circuit Simulation Environments

Besides the given capabilities of current circuit simulators there are certain limitations mentioned below. When any change is made in the schematic view of Multisim (any addition or deletion of a component) the change is not reflected in the animation view of Multisim, and vice versa.

(a) Some of the components and equipment in the animation view of the Multisim do not resemble those in the real laboratory.

(b) NI Multisim auto node naming conventions can unnecessary cause errors such as ‘circuit not grounded’ and deter simulation even though it is a valid circuit connection in a real laboratory [16].

(c) PSpice does smoke analysis, to analyze any electrical or thermal stress that may occur due to short circuit or open circuit. Failure modes are demonstrated, but smoke analysis is an option for which supplementary charges must be paid to have these add on.

(d) Learning all the commands for OrCAD PSpice applications is burdensome for introductory-level students.

(e) PSpice does not have any virtual test equipment such as an oscilloscope, digital multimeter or a function generator.

(f) The schematic view of Multisim includes failure modes; that are reflected in the circuit. But this is not reflected in the animation view of Multisim.

Self-contained laboratories:

National Instruments(NI) myDAQ is a portable data acquisition device that consists of virtual instrument software that includes an oscilloscope, a function generator and a digital multimeter. And it enables to measure and analyze circuits at anytime and anywhere. But NI myDAQ requires a hardware USB connection.

And other portable laboratory equipment available is Portable Electronics Experiment Kit (PEEK) (to virtual electronic instruments outside lab) [17], Lab–in-a-Box kit (to use virtual electronic instruments and components outside lab) [18].

1.2 Project VELVET

We propose a project named Virtual Electronics Laboratory with Visualized Education Training (VELVET) at University of Texas at Tyler to enable online students realistically simulate experiments like those in introductory electric circuit laboratory courses. In this project, we want to animate the process of constructing and analyzing electric or electronic circuits. Components may be selected from those available and connected to an animated prototyping board. Components may be added to the circuit or deleted from the circuit as may occur in construction process of a circuit.

Test equipment such as dual-channel oscilloscope, digital multimeter, waveform generator and 3-output DC power supply may be realized in the animation. Test equipment may be connected through animation to desired points in the circuit.

Transient analysis of the Electric Circuit Lab 1 in University of Texas at Tyler (EENG3104), resistors, capacitors, inductors, op amps (LM741), transistors, connecting wires, voltage regulators may be realized in animation.

When a component damages and produces smoke as in failure mode, it may be demonstrated through animation.

As experiments conducted in real labs are inherently transient in nature and so are the experiments in Virtual Laboratory.

1.3 Expected Benefits of Project VELVET

(a) The VELVET requires no physical lab, so there is no cost for the purchase of equipment and components. And as there are no physical equipment and components, the costs incurred due to maintenance will be avoided.

(b) There is no supplementary charge for features such as failure modes.

(c) VELVET real animation of the circuit-construction process and use of measurement equipment, whereas Multisim and PSpice begin with schematic capture, which is not a part of the laboratory experience.

(d) VELVET requires no additional hardware to connect to devices as that is required for portable kits such as NI myDAQ, PEEK and Lab-in-a-box. Hence students do not have to purchase a hardware kit that costs about hundreds of dollars. And these portable kits act as only supplement to regular face to face laboratory classes.

1.4 Overview of Project VELVET

Project VELVET, animations to build circuits may be implemented with Microsoft Visual Studio. Building the circuits is accomplished by picking components and connecting them to the prototyping board. Components may be added or deleted as required. The animation of test equipment will include an oscilloscope, a function generator and a digital multimeter.

It is intended that the final implementation of project VELVET include realistic component failure modes. This feature will enhance the real-time experience of the lab for online students. If a component experiences extreme electrical stress, it will sustain damage. This event will be captured in the animation.

In an ongoing circuit simulation, the components may be added or deleted while working on a circuit. It is very common for students to remove a component or add a component while a circuit is active.

Compared to a real laboratory, the virtual laboratory includes most real time activities such as constructing the circuit, connecting test equipment to the circuit, and manipulating the connections across different components to check readings or take measurements across different components. Animation of power supply connections and waveform generators is included. Circuits are re-configurable “on the fly”.

Chapter 2

2. Literature survey

A literature survey reveals that research in virtual laboratories has been active since the 1960's. Diverse fields analyzed the possibility of implementing online laboratories, due to the ample advantages presented in many research papers. For this reason, research work about the implementation of virtual laboratory has been carried out in many subjects including electrical and electronics teaching was studied. Different approaches to formulate network equations for building the basic anatomy of virtual laboratory were also studied.

Many papers on virtual laboratories in diverse areas such as process engineering, electrical machinery, physics, and chemistry have been published. For instance, Shaheen et al. [19] describe a remote laboratory for a Process Control unit in the process control and Automation Laboratory at Case Western Reserve University, accessed via internet. This system is designed to provide students with remote access to the process control unit. Therefore, this system uses a personal computer, employing both data acquisition card and an Ethernet card and connected to a process connected through a PLC (Programmable Logic Control) interface module and a LabVIEW.

Likewise, Tanyildizi and Orhan [20] discuss virtual instrument program. The authors specifically develop the virtual laboratory for synchronous machinery using Hyper Text Markup Language (HTML), Active Server Pages (ASP) and Borland C++ Builder. They explain how to perform the experiment on asynchronous starting of a synchronous mechanism and Voltage/frequency control in a synchronous motor in the virtual laboratory. The authors also explain how the use of a virtual laboratory avoids hazards and risks that can damage expensive machinery due to improper operation.

Xiaoyan et al. [21] describe the teaching electrical and electronics circuits in a virtual lab. The author uses a three layer Client/Server model consisting of a teaching

Environment Layer (Teaching Application Layer), a Network Interactive Layer and a Physical Layer. The Teaching Environment layer contains web pages with theorems, experimental content and wiring connections. The Network Layer acts as interface between the two other layers. The Physical Layer includes the data acquisition, as well as the network connection of client and server. Once the system has the wiring connections from the users they are recognized using LabVIEW, for simulation.

In a similarly, Basher et al. [22] outline a virtual laboratory for an introductory course in circuit analysis in an Electrical Engineering Technology program. It discusses the integration of LabVIEW with a computer laboratory at the school of Engineering Technology and Science at South Carolina State University.

Ben et al. [23] detail other virtual laboratory applications. The author describes a web based virtual oscilloscope laboratory experiment involving both hardware and software. The hardware used in this experiment consists of a PC (instrument controller), an oscilloscope and a signal generator. The PC or instrument controller has a GPIB (General Purpose Instrument Bus) interface card and Ethernet card. The software component is written in LabVIEW. The software includes Video Server and WWW Server. While Video Server provides visual feedback, WWW server hosts the virtual lab. The instruments are connected to the PC (Instrument Controller) through the GPIB card and the GPIB cable. LabVIEW is used for local instrument control. WWW server receives parameters from the client and then passes it to a CGI (Common Gateway Interface) program, which checks for the validity of parameters and establishes TCP (Transmission Control Protocol) connection with the instrument controller PC. This program invokes attached instruments such as an oscilloscope via GPIB.

2.1 Different approaches to network equation formulation:

The nodal analysis approach is the most common approach used to formulate network equations. But this nodal approach has some limitations like difficulty in the representation of voltage sources and current dependent elements such as inductors. Ho et al. [24] presents a modified nodal analysis (MNA) represented by

$$\begin{bmatrix} Y_R & B \\ C & D \end{bmatrix} \begin{bmatrix} V \\ I \end{bmatrix} = \begin{bmatrix} J \\ F \end{bmatrix} \quad (2.1)$$

In which Y_R is the reduced form of the nodal matrix excluding contributions due to voltage sources, current controlling elements, etc. B is the matrix with partial derivatives of KCL equations with respect to additional current variables. The matrices C and D represent constitutive relations with respect to unknown vector $\begin{bmatrix} V \\ I \end{bmatrix}$. V represents the node voltage matrix and I represent the branch current vector. Additionally, J and F are the matrices that include excitations of initial values of inductor currents and capacitor voltages. The matrix $\begin{bmatrix} Y_R & B \\ C & D \end{bmatrix}$ dimensions for a given circuit are the number of nodes minus the ground connection, plus branch currents. And it is a square matrix.

SPICE is a widely used simulator that uses the MNA for network equation formulation [25]. In SPICE DC analysis, capacitors and inductors are treated as short and open circuits respectively. For AC analysis, real conductances are replaced by complex admittances. For transient analysis components are represented in differential or integral form. For example inductor and capacitor equations are considered as

$$i_c = \frac{dq}{dt} = C\frac{dv_C}{dt} \quad (2.2)$$

$$v_L = \frac{d\emptyset}{dt} = L\frac{di_L}{dt} \quad (2.3)$$

where i_c and v_C are the current and voltage of capacitor C andi_L and v_L are the current and voltage of inductor L. The branch constituent equation for resistors, Ohm's law, is time-invariant. A numerical ordinary differential equation (ODE) solution is performed. Non-linear elements are solved by an iterative method at each time step.

Another widely known circuit analysis program is IBM ECAP (Electronic Circuit Analysis Program). It also uses the nodal analysis approach for the formulation of network equations. Branin et al. [26] demonstrate on ECAP, how the ohm’s law is depicted below.

$$\begin{bmatrix} I_G \\ V_R \end{bmatrix} = \begin{bmatrix} G & F \\ N & R \end{bmatrix} \begin{bmatrix} V_G \\ I_R \end{bmatrix} \quad (2.2)$$

The diagonal elements of G and R represent self-conductance and self-resistance. Voltage controlled current sources are represented as off diagonal elements of the G matrix, while the current controlled voltage sources are represented as off diagonal elements in the R matrix. The F matrix contains current controlled current sources and N matrix contains voltage controlled voltage sources. Where I_G, V_G correspond to current and voltage of matrices G and F. V_R, I_R correspond to current and voltage of matrices N and R.

Chapter 3

3. Development of the VELVET mathematical model

Figure 3.1 depicts a virtual breadboard with eight rows (or strips) of contacts and the connection of resistors. In the left-hand illustration, a resistor designated R5 has been connected between breadboard connection 1 and breadboard connection 5. The first lead of R5 is inserted into breadboard connection 1; since this is the first connection of the network; electrical node 1(represented in Fig. 3.1 by the circled number 1) is assigned to this connection. The second lead insertion is made to breadboard connection 5, which is then assigned to electrical node 2. Since R5 is the first component to be installed current in this component is designated i_1, flowing into R5 from the first lead (node 1) and leaving R5 at the second lead (node2). Similarly, the next component (R1) is connected between breadboard connections 5 and 8. The connection of R1 to breadboard connection 5 does not require designation of a new electrical node since connection 5 is already associated with electrical node2. However, the connection of the second lead of R1 to breadboard connection 8 causes connection 8 to be mapped to electrical node 3. The current i_2 in R1 flows into R1 from node2 to flows out node3.

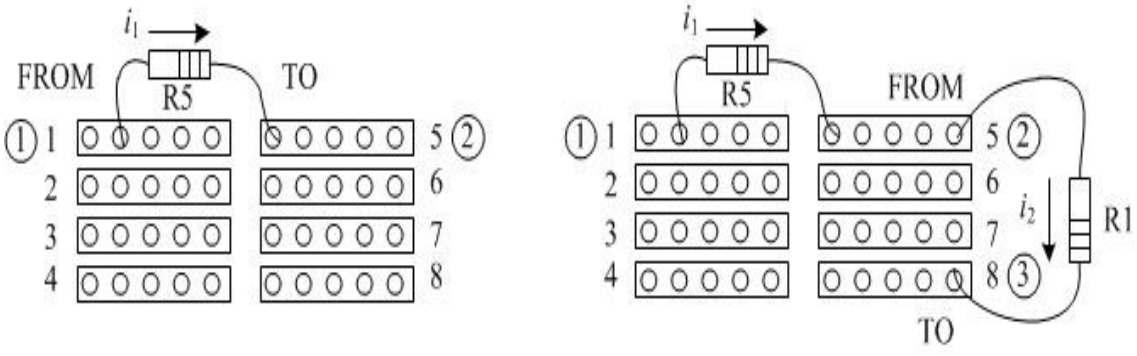


Fig. 3.1 Depiction of a simple virtual breadboard and connection of resistors

The algorithm is designed in such a way that is suitable for networks with reactive elements and controlled sources as well as resistors. This algorithm also takes care of

controlled sources. Networks with reactive elements, controlled sources, and ac voltage sources are investigated. In the pseudo code, matrices A_3, A_4 and Y represent KCL and matrices A_1, A_2 and S are derived from constitutive relations. Figure 3.2 is the flowchart of proposed algorithm.

Pseudo code for the proposed algorithm

(1) When a new node is added to the circuit add a column to the matrix A_1, add a new row and column to matrix A_2 and A_4.

(2) When a new branch is added to the circuit add a row to the matrix A_1.

(3) When a new component is added, then number of new branches is one and number of new nodes may be 0 or 1 or 2.

(4) Formation and dimensions of the matrix A_1.

A_1 rows =A_1rows+1

A_1 = A_1columns + number of new nodes

If the branch is resistor, then

A_1(new_row, from) = 1/R

A_1 (new_row, to) = -1/R

If the branch is voltage source or capacitor

A_1(new_row, from) = 1

A_1 (new_row, to) = -1

If the branch is current source or inductor

A_1(new_row, from) = 0

A_1 (new_row, to) = 0

(5) Formation and dimensions of matrix A_2

A_2 = (A_2rows +1, A_2 columns+1)

If component is resistor

A_2(new_node, new_branch)= -1

If component is voltage source or capacitor, then no changes are required

If component is an inductor or current source

A_2(new_node, new_branch)= -1

(6) Formation and dimensions of matrix A_4

A_4 (rows) = A_4(rows + number of new nodes)

A_4(columns) = A_4(columns +1)

If a component is added to the circuit

A_4 (from, new_branch) = 1

A_4 (to, new_branch) = -1

(7) Formation and dimensions of matrix S

S (rows) = S (rows+1)

If voltage source or capacitor, S (new_row) = voltage

If current source or inductor, S (new_row) = current

(8) Formation of zero matrix A3 of dimensions (nodes x *nodes), formation of column matrix Y of dimensions (nodes, 1).*

(9) Concatenate matrices A_1, A_2, A_3, A_4 *to form matrix A.*

(10) Delete column and row of ground node from matrix A to form A'. And also row corresponding to ground node in Y matrix to form Y'.

(11) Append matrix S and Y' to form S'.

(12) Solve for $[V'\ I]^T$ *from A' and S' using Guass elimination method.*

(13) If the circuit has reactive elements then Matrix B1 and B2 are formed to hold constitutive relations of reactive elements.

(14) Matrices B1 and B2 are appended to form B matrix. When the row corresponding to ground node is deleted then matrix B' is formed.

(15) A' matrix is also formed as stated in previous steps.

(16) Compute node voltage and branch currents $[V'\ I]^T$

(17) Compute xdot ($B' * [V'\ I]^T$ *)*

*(18) Using Runge- Kutta method K1=xdot * t*

(19) Add (initial values x0) x0+K1/2

(20) Substitute (x0 + K1/2) into S'(sources)

(21) Solve for node voltages and branch currents $[V'\ I]^T$

(22) Compute xdot ($B' * [V'\ I]^T$*)*

*(23) Add K2 =xdot * t*

(24) Add x0+K2/2

(25) Substitute (x0 + K2/2) into S'

(26) Compute $[V'\ I]^T$

(27) Compute xdot ($B' * [V'\ I]^T$*)*

*(28) Add K3 = xdot * t*

(29) Substitute (x0 +K3) into S'

(30) Compute node voltages and branch currents $[V'\ I]^T$

(31) Compute xdot ($B' * [V'\ I]^T$*)*

*(32) Add K4 = xdot * t*

(33) $[V'\ I]^T$ *= x0 +K1/6 +K2/3 +K3/3 + k4/6*

(34) Repeat the process for number of steps.

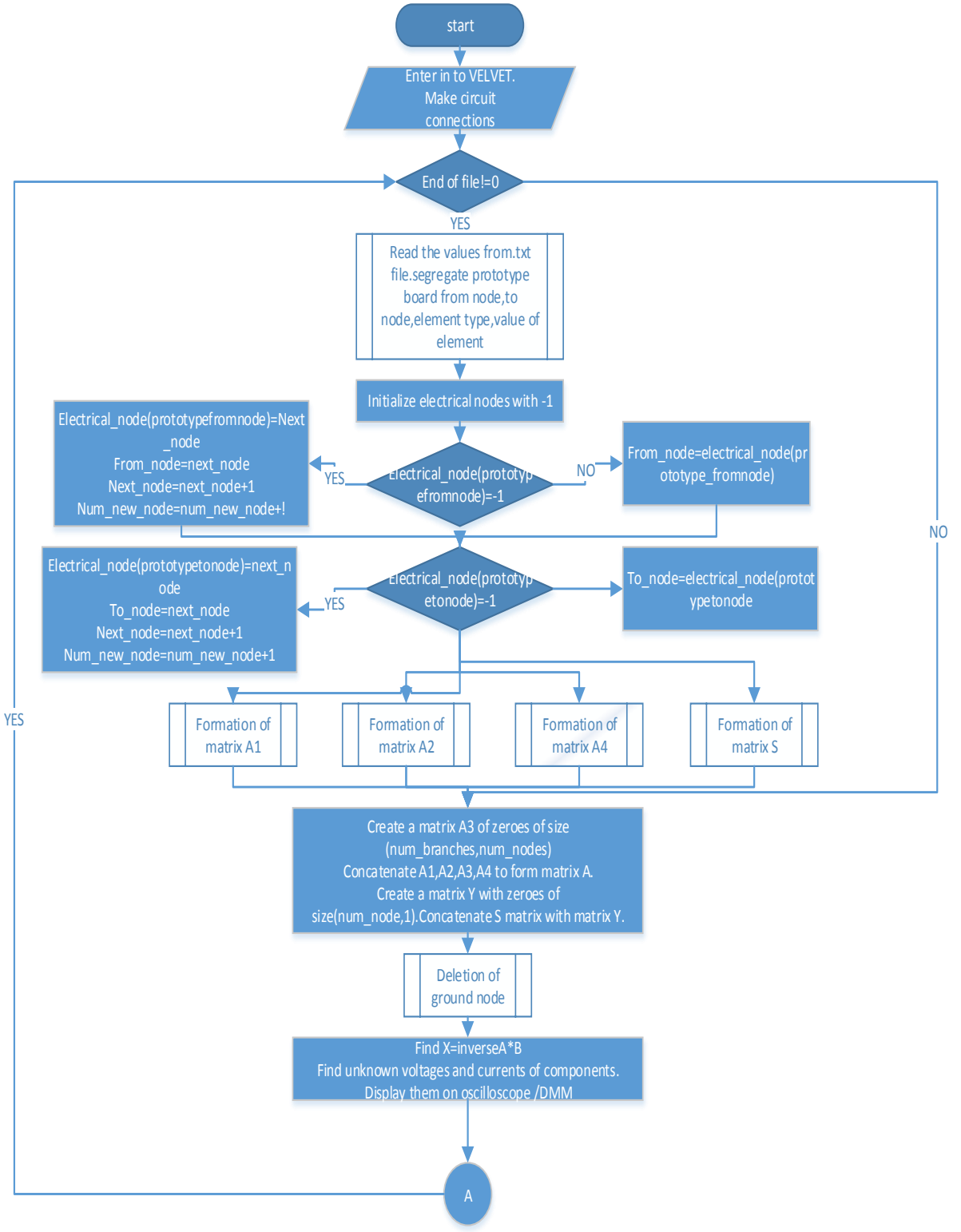


Fig. 3.2 Flowchart of proposed algorithm

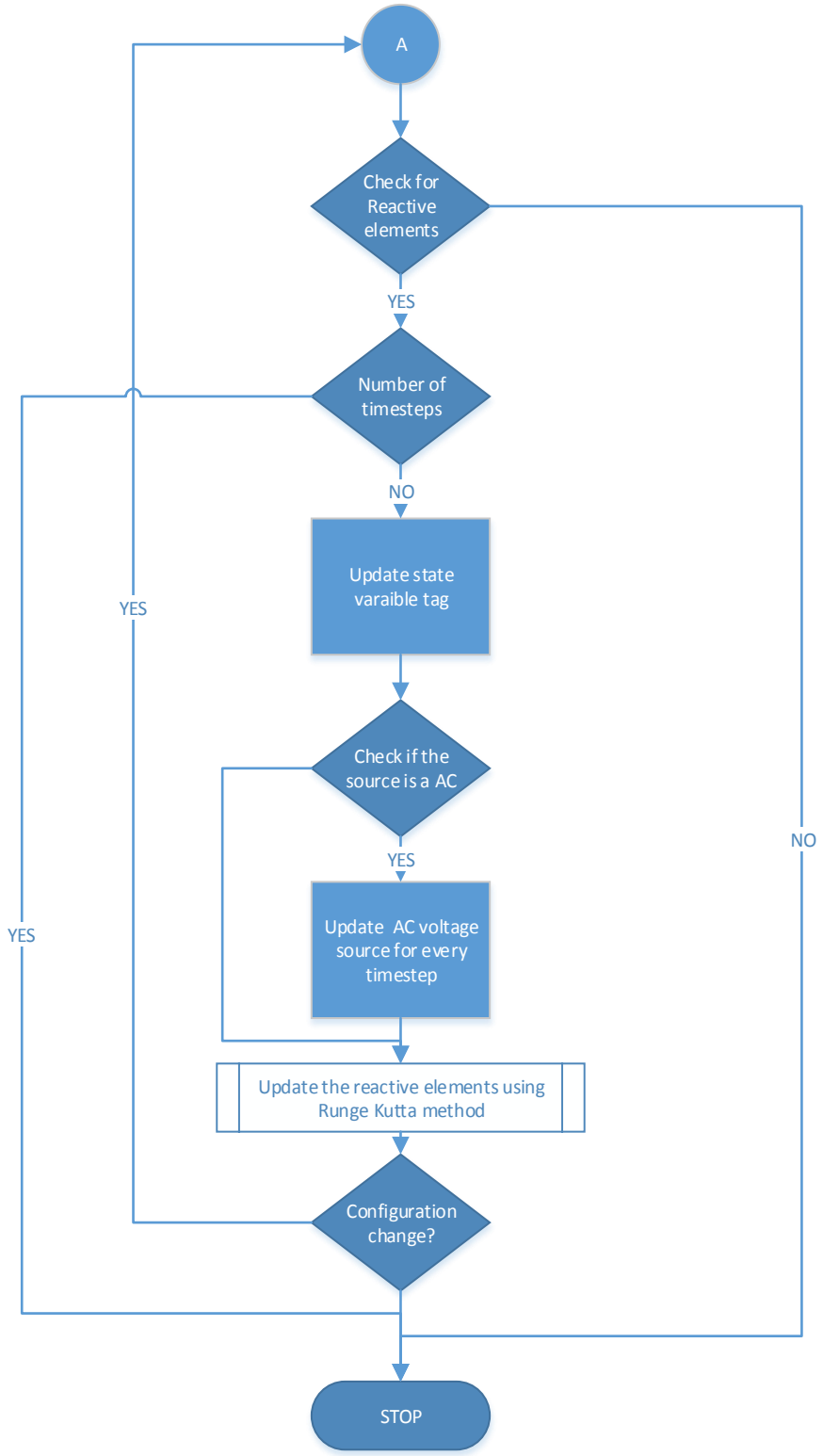


Fig. 3.3 Flowchart of proposed algorithm cont.

Network equations are assembled as the circuit is built. Figure 2 shows the flowchart of proposed algorithm. Matrices A1, A2, A4 and S are explained in detail in below given example.

3.1 Validation of the Algorithm

The above algorithm is demonstrated for the following circuits

(1) Resistive circuit.
(2) Circuits containing controlled sources.
(3) RLC circuit with sinusoidal excitation
(4) RLC circuit with triangle-wave excitation
(5) RLC circuit with square-wave excitation

The above algorithm is validated with the help of the following examples. Network equations are assembled from circuit topology, and the unknown values (node voltages and branch currents) are computed.

3.1.1 Resistive Circuit

Figure 3.1 is a simple example circuit consisting of three meshes involving six resistors and a DC voltage source. Numbers adjacent to the schematic indicate the assumed node numbers. There is no ground node shown in the circuit diagram; VELVET develops the network equations for all nodes and then modifies those equations after one of the nodes is selected as the reference (ground) node. Arrows indicate flow of currents in each branch.

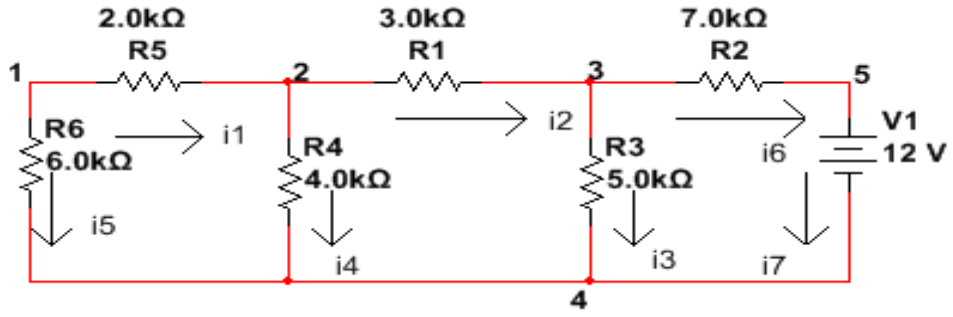


Fig. 3.4 Circuit diagram of example circuit

Application of Kirchhoff's Current Law (KCL), at each node gives the following relationships.

Node 1: $i_1 + i_5 = 0$ (3.1)

Node 2: $-i_1 + i_2 + i_4 = 0$ (3.2)

Node 3: $-i_2 + i_3 + i_6 = 0$ (3.3)

Node 4: $-i_5 - i_4 - i_3 - i_7 = 0$ (3.4)

Node 5: $-i_6 + i_7 = 0$ (3.5)

The constitutive relations for each branch give the following:

Branch with R5: $\frac{v_1 - v_2}{R5} - i_1 = 0$ (3.6)

Branch with R1: $\frac{v_2 - v_3}{R1} - i_2 = 0$ (3.7)

Branch with R3: $\frac{v_3 - v_4}{R3} - i_3 = 0$ (3.8)

Branch with R4: $\frac{v_2 - v_4}{R4} - i_4 = 0$ (3.9)

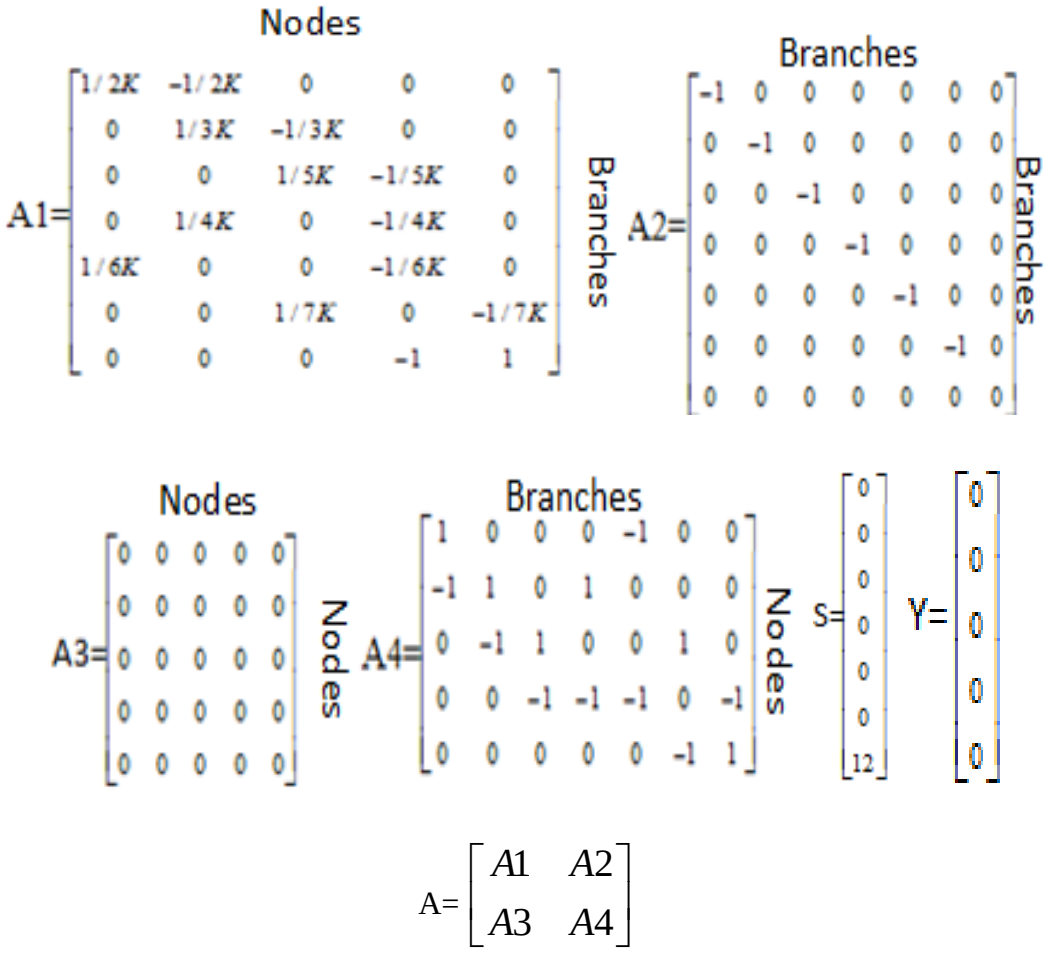
Branch with R6: $\frac{v_1 - v_4}{R6} - i_5 = 0$ (3.10)

Branch with R2: $\frac{v_3 - v_5}{R2} - i_6 = 0$ (3.11)

Branch V1: $v_5 - v_4 =$ potential of source V1 (12V in Fig.3.4) (3.12)

Using both Kirchhoff and constitutive equations we form both a square coefficient matrix *A* representing the left-hand sides of the equations and a column vector of known values representing the right-hand sides. The dimensions of matrix *A* are (N_{nodes} + $N_{branches}$) by (N_{nodes}+ $N_{branches}$) where N_{nodes} is the number of nodes and $N_{branches}$ is total number of branches. The matrix *A* is subdivided into four matrices (designated as A_1, A_2, A_3 and A_4 for the purpose of construction). A column vector of sources (*S*) and a column vector of zeroes (*Y*) are also formed and concatenated to form a

column vector of known values with ($N_{nodes}+N_{branches}$) elements. From the above equations, sub-matrices, A_1 , A_2 , A_3 and A_4 and vectors *S* and *Y* can be formed as:



$$A=\begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix}$$

Fig. 3.5 Matrices A_1 , A_2 , A_3 , A_4 and A ,vectors *S* and *Y* .

The combined matrix formulation can be represented as:

$$\begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix}\begin{bmatrix} S \\ Y \end{bmatrix}=\begin{bmatrix} V \\ I \end{bmatrix} \tag{3.13}$$

When these equations are solved, vector *V* contains node voltages and vector I contains branch currents. Matrices A_3, A_4 and *Y* represent KCL and matrices A_1, A_2 and *S* are derived from constitutive relations. Once the network equations are formulated, VELVET chooses a ground node based upon the circuit topology. The rules for selecting the ground node follow a hierarchy. If a DC power supply is connected to the circuit, the

node connected to the common terminal power supply is designated as ground. If no DC power supply is present, VELVET checks for the presence of the signal generator. If the signal generator is present, the node connected to generator reference terminal is designated as ground. If neither the power supply nor the signal generator is present, VELVET checks for the presence of the oscilloscope and/or the digital multimeter (DMM). If one of these test instruments is present, ground is assigned to the node connected to the reference lead of the instrument. (If both are present, ground is determined by the reference lead of the oscilloscope probe).

Once the ground node is determined, the column of coefficients in Matrix *A* corresponding to voltage at the ground node is deleted and the row representing the KCL equation at the ground node is also deleted. In the circuit of Fig 3.4, the common lead of the DC power supply is connected to node 4, and node 4 is thus designated as ground. This designation causes the modification of matrices *A* and *Y* to form new matrices A' and Y'. These new matrices are given below with deleted elements of the original matrices shown in light color.

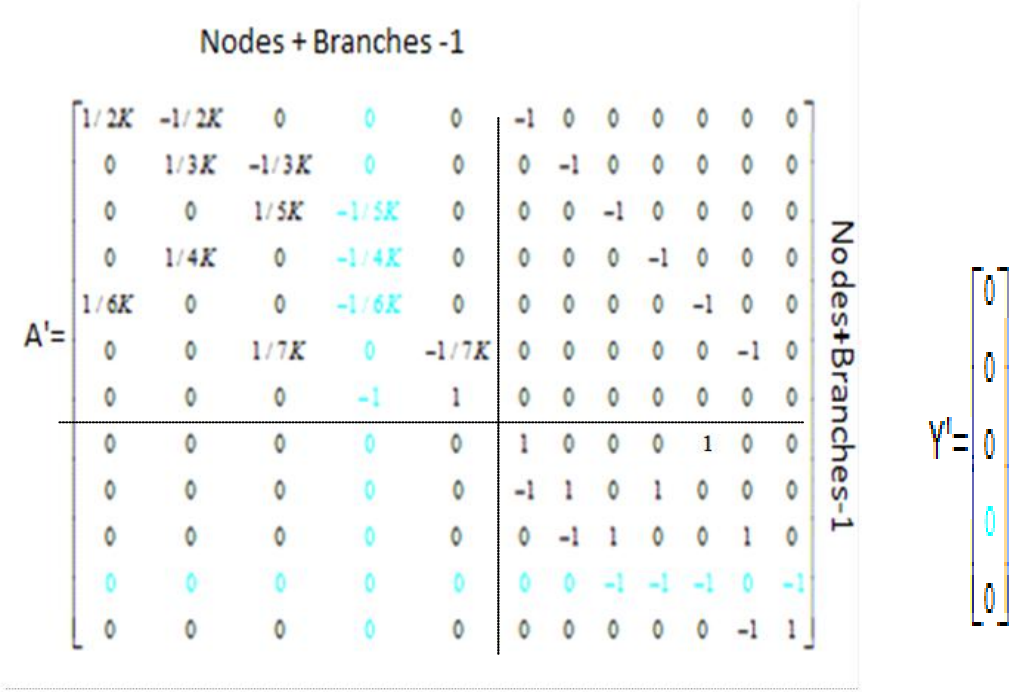


Fig. 3.6 Matrices A' and Y' for resistive network of Fig.3.4.

The upper left portion of A' is sub matrix A_1; the upper right portion is A_2. These represent constitutive relations. The lower-left portion is A_3 and the lower right portion is A_4. These are derived from KCL equations. The values of branch currents and non-ground node voltages are computed as:

$$\begin{bmatrix} V' \\ I \end{bmatrix} = A^{-1} \begin{bmatrix} S \\ Y' \end{bmatrix} \tag{3.14}$$

where V' represents node voltages (except the reference node) and I represent branch currents. For the circuit of Figure 3.1, the computed node voltages and branch currents are:

$$\begin{bmatrix} V' \\ I \end{bmatrix} = \begin{bmatrix} v1 \\ v2 \\ v3 \\ v5 \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \\ i6 \\ i7 \end{bmatrix} = \begin{bmatrix} 1.165 \\ 1.5534 \\ 3.30 \\ 12 \\ -0.00019 \\ -0.00058 \\ 0.00066 \\ 0.00038 \\ 0.00019 \\ -0.00124 \\ -0.00124 \end{bmatrix}$$

Fig. 3.7 Matrix $[V \quad I]^T$ for resistive network of Fig 3.4

The network equations are re-formulated whenever there is a change in the circuit topology. This allows students to re-configure their circuits "on the fly" as occurs when a circuit element is added to or removed in the middle of an experiment.

3.1.2 Circuit with a voltage controlled voltage source (VCVS)

In the circuit of Fig 3.8, the voltage source between nodes 6 and 2 is dependent on the node voltage that is between nodes 4 and 2. The constitutive relation for the branch containing the VCVS can be expressed as:

$$v_6 - v_2 - A_v(v_4 - v_2) = 0 \tag{3.15}$$

where A_v is voltage gain.

Two voltage sources V1 and V2 are included in the circuit of Fig 3.8. Ground is selected as the common terminal of V1 and V2. Final A' matrix for this circuit is shown below. The row and column in light type are the row and column that are deleted due to the designation of node 2 as ground.

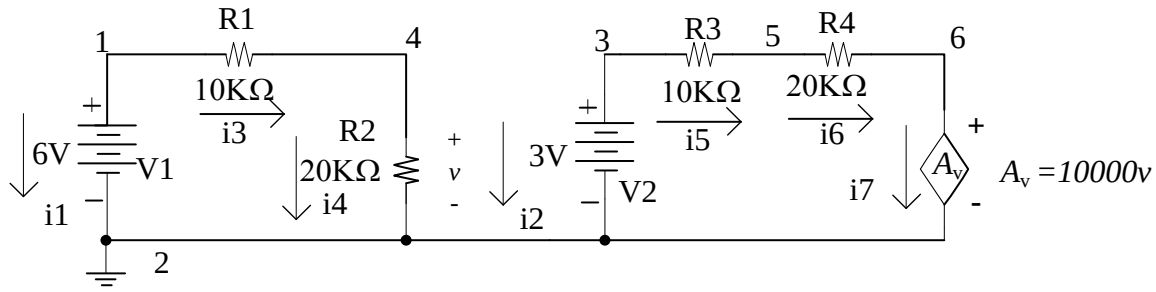


Fig. 3.8 Example circuit including voltage controlled voltage source

The column corresponding to the coefficients the voltage and the KCL equation for node 2 are marked in light color to indicate deleted.

$$A' = \left[\begin{array}{cccccc|ccccccc}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/10K & 0 & 0 & -1/10K & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1/20K & 0 & 1/20K & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 1/10K & 0 & -1/10K & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/20K & -1/20K & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 9999 & 0 & -10000 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{array}\right]$$

Fig. 3.9 Matrix A' for circuit of Fig. 3.8

Matrix $[S \quad Y']^T$ is the column vector of terms from the right side of both KCL equations and constitutive relations. Column matrix $[V' \quad I]^T$ contains the unknown node voltages and branch currents. These unknowns can be found by Gauss elimination.

Vectors $[S \quad Y]^T$ and $[V \quad I]^T$ for the circuit of Fig 3.8 are shown in Fig 3.10 below.

$$\begin{bmatrix} S \\ Y' \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} V' \\ I \end{bmatrix} = \begin{bmatrix} v1 \\ v3 \\ v4 \\ v5 \\ v6 \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \\ i6 \\ i7 \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ 4 \\ 3.9998 \\ 17.998 \\ -0.0002 \\ -0.000699 \\ 0.0002 \\ 0.0002 \\ -0.000699 \\ -0.000699 \\ -0.000699 \end{bmatrix}$$

Fig. 3.10 Matrices $\begin{bmatrix} V' \\ I \end{bmatrix}$ and $\begin{bmatrix} S \\ Y' \end{bmatrix}$ for circuit of Fig.3.8

3.1.3 Circuit with a voltage controlled current source circuit (VCCS)

An example of a circuit containing a VCCS is shown in Fig. 3.11.

The constitutive relationship for VCCS in Fig. 3.11 can be expressed as:

$$g_m v_3 - g_m v_4 - i_5 = 0 \tag{3.16}$$

where g_m is transconductance.

Matrix A' for the circuit of Fig 3.11 is shown in Fig. 3.12 below with deleted column and row designated as ground shown in light type. The constitutive relation for the VCCS is contained in the fifth row of A'.

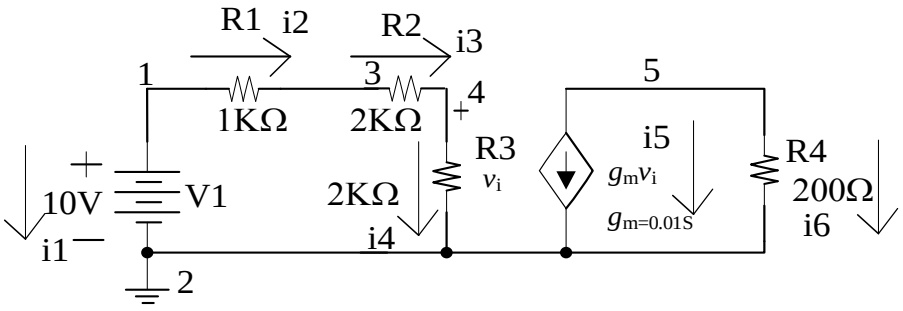


Fig. 3.11 Example circuit including voltage controlled current source

$$A' = \left[\begin{array}{ccccc|cccccc}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/1K & 0 & -1/1K & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2K & -1/2K & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & -1/2K & 0 & 1/2K & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0.01 & -0.01 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1/200 & 0 & 0 & 1/200 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{array}\right]$$

Fig. 3.12 Matrix A' for circuit of Fig. 3.11

$[S \quad Y']^T$ and $[V \quad I]^T$ for this circuit are shown in Fig. 3.13.

$$\begin{bmatrix} S \\ Y' \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} V' \\ I \end{bmatrix} = \begin{bmatrix} v1 \\ v3 \\ v4 \\ v5 \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \\ i6 \end{bmatrix} = \begin{bmatrix} 10 \\ 8 \\ 4 \\ -0.8 \\ -0.002 \\ 0.002 \\ 0.002 \\ 0.002 \\ 0.004 \\ -0.004 \end{bmatrix}$$

Fig. 3.13 Matrices $[S \quad Y']^T$ and $[V \quad I]^T$ for circuit of Fig.3.11

3.1.4 Circuit with a current controlled voltage source circuit (CCVS)

The example circuit of Fig. 3.14 below includes a current controlled voltage source, Here branch current i_1 controls the voltage source that is located between nodes 3 and 2. The expression for the constitutive relation of the CCVS can be expressed as:

$$v_3 - v_2 - R_m i_1 = 0 \tag{3.17}$$

where R_m is transresistance

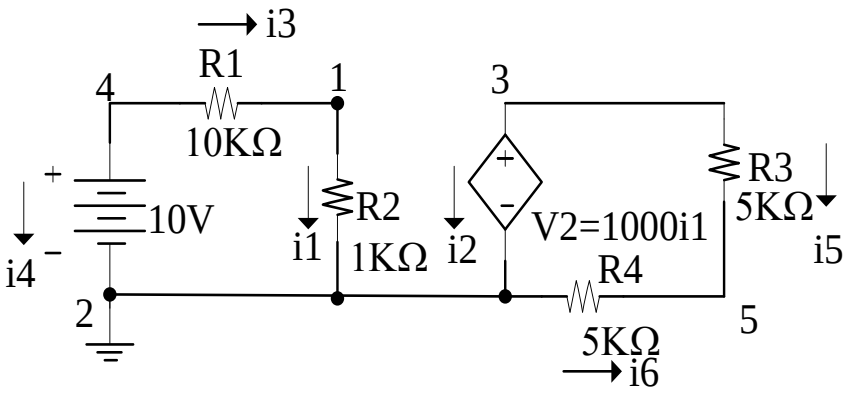


Fig. 3.14 Example circuit including current controlled voltage source

Matrices A', $[S \quad Y']^T$ and $[V' \quad I]^T$ are given in Fig. 3.15.The constitutive relation of the CCVS is given in the second row of A'.

$$A' = \left[\begin{array}{cccccc|cccccc}
10 & -10 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & -1000 & 0 & 0 & 0 & 0 & 0 \\
-1/10K & 0 & 0 & 1/10K & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/5K & 0 & -1/5K & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1/5K & 0 & 0 & -1/5K & 0 & 0 & 0 & 0 & 0 & -1 \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1
\end{array}\right]$$

$$\begin{bmatrix} S \\ Y' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} V' \\ I \end{bmatrix} = \begin{bmatrix} v1 \\ v3 \\ v4 \\ v5 \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \\ i6 \end{bmatrix} = \begin{bmatrix} 9.999e-5 \\ 0.9999 \\ 10 \\ 0.49995 \\ 0.000999 \\ -9.999e-5 \\ 0.000999 \\ -0.000999 \\ 9.995e-5 \\ -9.995e-5 \end{bmatrix}$$

Fig.3.15 Matrices A', $\begin{bmatrix} V' \\ I \end{bmatrix}$ and $\begin{bmatrix} S \\ Y' \end{bmatrix}$ for circuit of Fig. 3.14

3.1.5 Circuit with current controlled current source (CCCS)

Shown in Fig. 3.16 below is an example circuit including current controlled current source. Here a current source between nodes 3 and 2 is controlled by the branch current that flows from nodes 1 and 2.

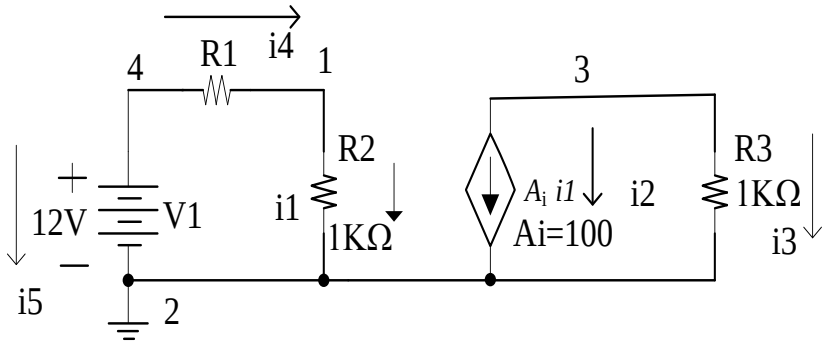


Fig. 3.16 Example circuit including current controlled current source

The constitutive relationship can be expressed as

$$A_i\, i_1 - i_2 = 0 \tag{3.18}$$

where A_i is current gain of CCCS.

Matrix A' for Fig. 3.16 is shown below in Fig. 3.17.The row and column deleted by the designation of node2 as ground are shown in light type. The constitutive relation for CCCS is found in second row of A'.

$$A' = \left[\begin{array}{cccc|ccccc} 1/0.1 & -1/0.1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & -1 & 0 & 0 & 0 \\ 0 & -1/5K & 1/5K & 0 & 0 & 0 & -1 & 0 & 0 \\ -1/10K & 0 & 0 & 1/10K & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Fig.3.17 Matrix A' for the circuit of Fig. 3.16

Vectors $[S \quad Y']^T$ and $[V' \quad I]^T$ for the circuit Fig. 3.16 is shown in Fig. 3.18.

$$\begin{bmatrix} S \\ Y' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} V' \\ I \end{bmatrix} = \begin{bmatrix} v1 \\ v3 \\ v4 \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \end{bmatrix} = \begin{bmatrix} 9.99e-5 \\ -499.995 \\ 10 \\ 0.000999 \\ 0.0999 \\ -0.0999 \\ 0.00099 \\ -0.00099 \end{bmatrix}$$

Fig.3.18 Matrices $\begin{bmatrix} V' \\ I \end{bmatrix}$ and $\begin{bmatrix} S \\ Y' \end{bmatrix}$ for circuit of Fig.3.16

3.1.6 *RLC* circuit with sinusoidal excitation

The circuit of Fig. 3.19 below shown is an *RLC* circuit with sinusoidal excitation. This is a practical network like one which may be encountered in electrical circuits laboratories.R3 represents a jumper wire in Fig. 3.19

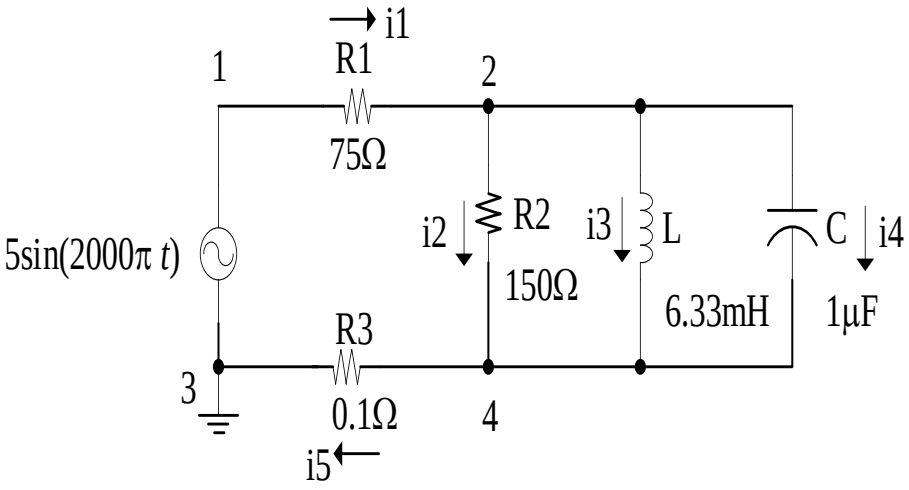


Fig. 3.19 Example circuit diagram of an *RLC* network with sinusoidal excitation.

The I V relationships of the reactive element of Fig. 3.19 are:

$$\frac{dI_L}{dt} = \frac{1}{L}(v_2 - v_4) \tag{3.19}$$

$$\frac{dV_C}{dt} = \frac{1}{C} I_4 \tag{3.20}$$

The constitutive relationships of L and C are:

$$i_5 = I_L \tag{3.21}$$

$$v_2 - v_4 = V_C \tag{3.22}$$

The present value of inductor current I_L and the present value of capacitor voltage V_C are included as the elements of *S'* matrix.

The differential equations (3.19) and (3.20) are implemented as given below:

$$\begin{bmatrix} \frac{dV_C}{dt} \\ \frac{dI_L}{dt} \end{bmatrix} = B' \begin{bmatrix} V' \\ I \end{bmatrix} \tag{3.23}$$

Matrix B' for Fig. 3.19 is shown below. The column in light type contains coefficients of the voltage at node 3; this column is removed due to the designation of node 3 as ground.

$$\text{B'}= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 \\ 0 & 157.97 & 0 & -157.97 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The first row of B' expresses (3.20); the second expresses (3.19).

$[V' \quad I]^T$ is found as solution to A' $[V' \quad I]^T = [S \quad Y']^T$ where S contains values of I_L and V_C at the time t.

Then equation (3.23) is integrated numerically using fourth order Runge-Kutta method to find the values of I_L and V_C at time $t + \Delta t$.The S vector is updated with the new values of $I_L(t + \Delta t)$ and $V_C(t + \Delta t)$, and $[V' \quad I]^T$is recomputed at this new point in time. If S contains time-varying independent sources, they are updated at each time step.

The matrices A' for Fig. 3.19 is shown in Fig. 3.20.

$$\text{A'}= \left[\begin{array}{cccc|cccccc} 1/75 & -1/75 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/0.1 & -1/0.1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1/150 & 0 & -1/150 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & -1 & 0 \end{array}\right]$$

Fig.3.20 Matrix A' for *RLC* circuit with sinusoidal excitation of Fig.3.19

The network of Fig. 3.19 was simulated with VELVET and Multisim. The output voltages at node 2 from both simulations are shown in Fig. 3.21 below. The agreement is evident. In both simulations the initial value of capacitor voltage was 2V and the initial inductor current was 0.

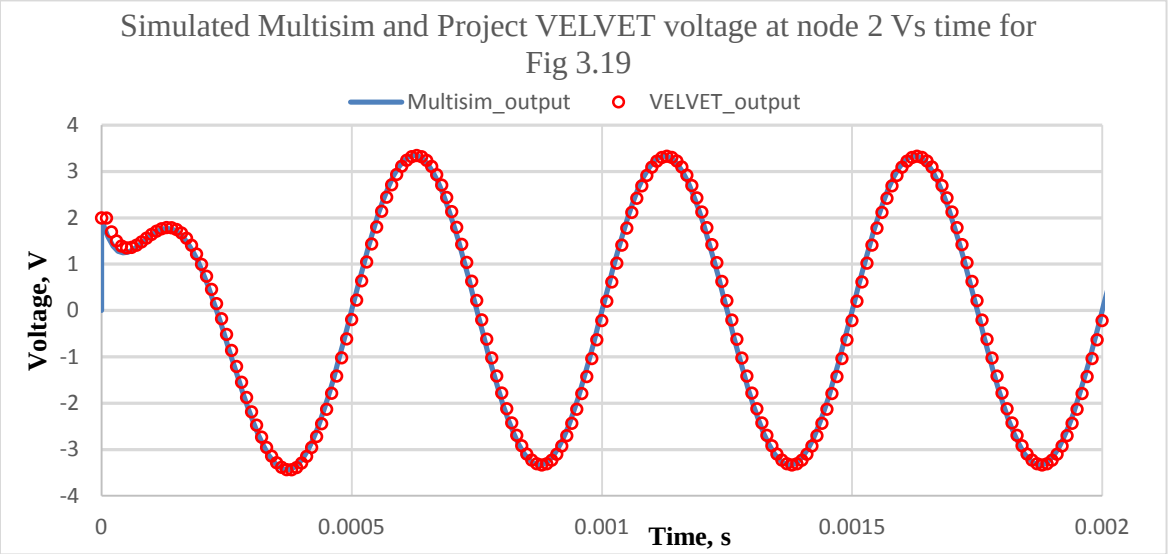


Fig. 3.21 Comparison of VELVET and Multisim simulations

of a *RLC* network excited by sinusoidal.

3.1.7 *RLC* circuit with square-wave excitation

The circuit of Fig.3.22 is the network of Fig. 3.19 driven with square wave excitation

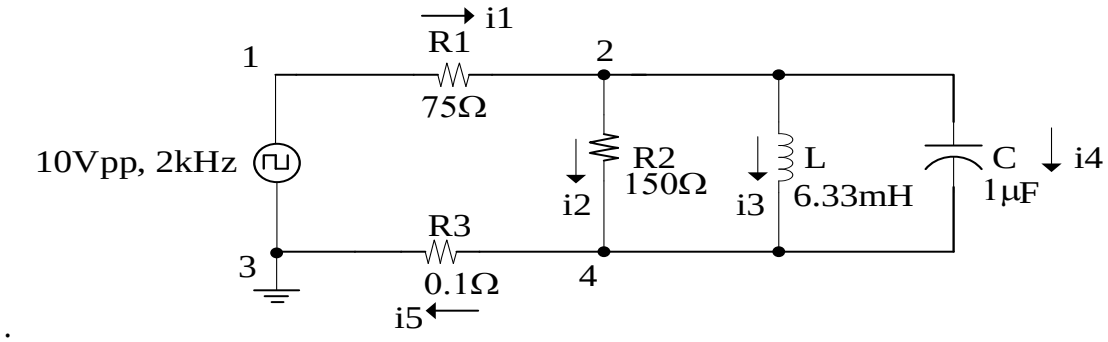


Fig. 3.22 Example circuit diagram of an *RLC* network with square-wave excitation.

The network of Fig. 3.22 was simulated with VELVET and Multisim. The output voltages at node 2 from both simulations are shown in Fig. 3.21 below. The agreement is evident. In both simulations the initial value of capacitor voltage was 2V and the initial inductor current was 0.

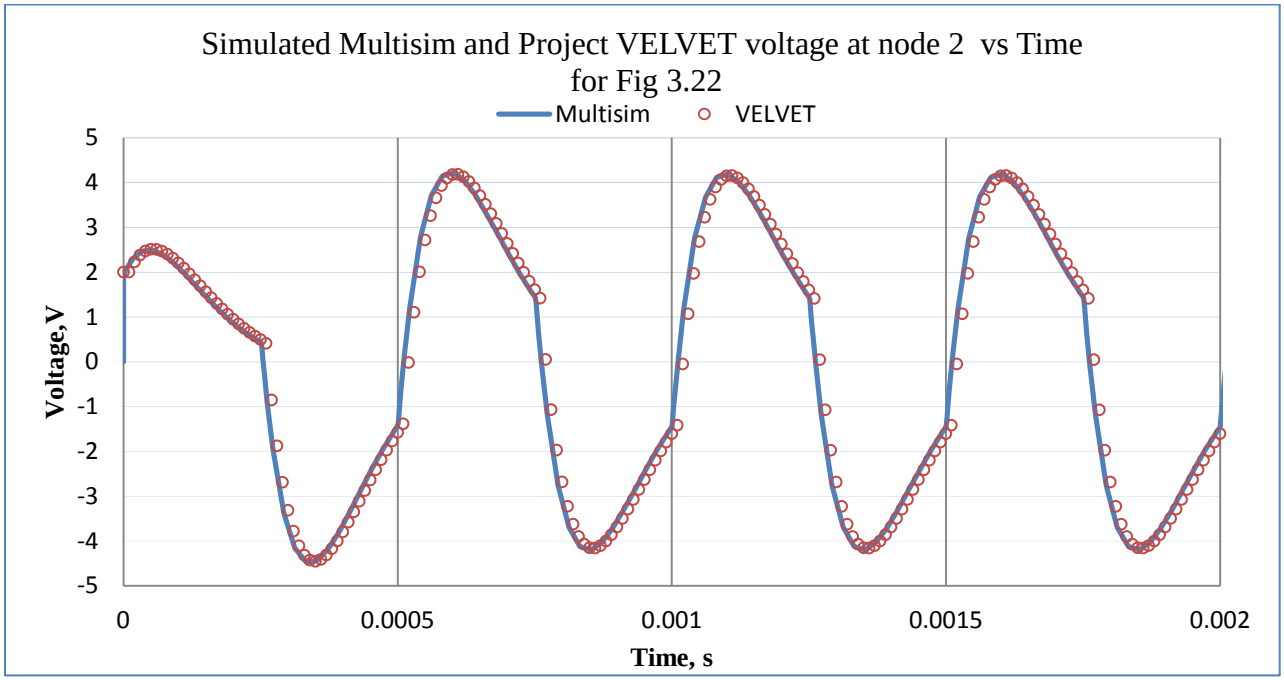


Fig. 3.23 Comparison of VELVET and Multisim simulations

of an *RLC* network excited by a square wave.

3.1.8 *RLC* circuit with triangle-wave excitation

Fig. 3.24 below is the network of Fig. 3.19 driven by triangle wave excitation.

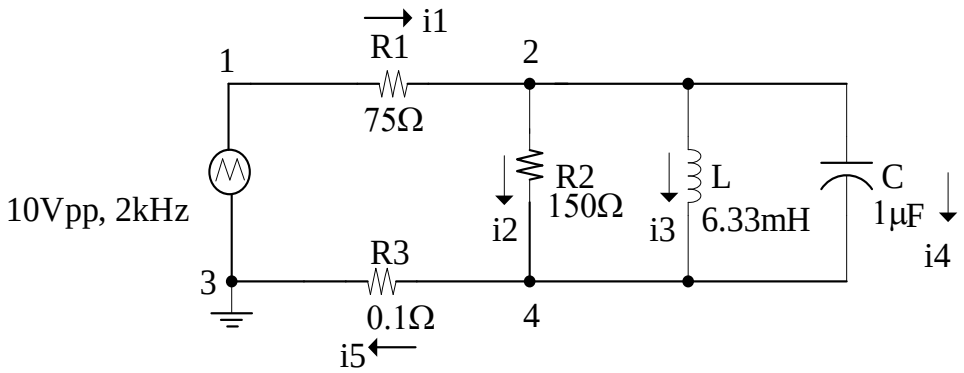


Fig. 3.24 Example circuit diagram of an *RLC* network with triangle-wave excitation.

The network of Fig. 3.24 was simulated with VELVET and Multisim. The output voltages at node 2 from both simulations are shown in Fig. 3.21 below. The agreement is evident. In both simulations the initial value of capacitor voltage was 2V and the initial inductor current was 0.

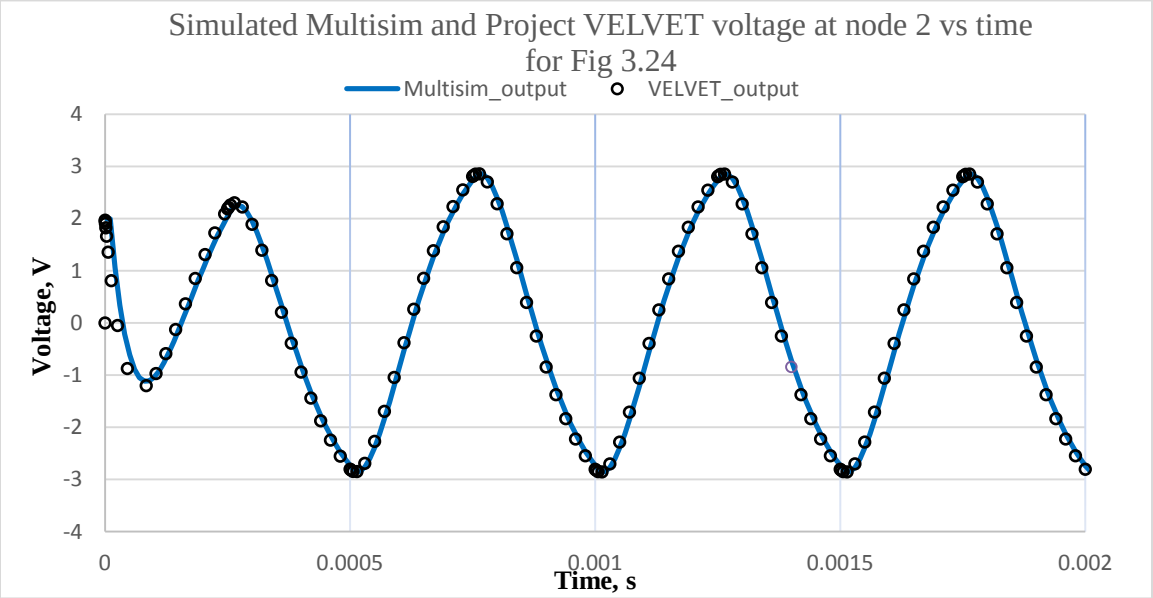


Fig. 3.25 Comparison of VELVET and Multisim simulations of a RLC network excited by triangle wave.

Chapter 4

4. Implementation of VELVET

The previous chapter has demonstrated the mathematical development of network equations and their solution for various electrical networks. The present chapter, will show how VELVET is developed and what are its different virtual components (breadboard, oscilloscope, multimeter, power supply, resistive elements). Additionally, this chapter will discuss how the mathematical solution is linked to the virtual instruments and how the data from the components connected to breadboard linked with algorithm.

Care is taken to make virtual components and instruments replicate physical components and instruments in both appearance and function.

4.1 VELVET components

4.1.1 Breadboard

The virtual breadboard of Fig. 4.1 in model works similarly to a physical breadboard. It has six columns of connection strips. Each strip has five connection points; connection to one connection point makes a connection to the whole strip.

Three power supply lines (i.e. positive power supply, negative power supply and ground) are also provided on the left side of the breadboard.

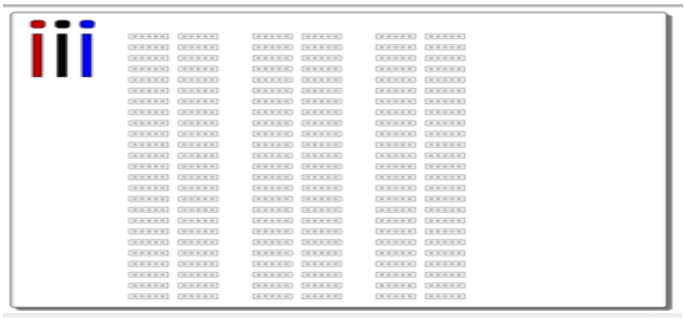


Fig. 4.1 Virtual Breadboard

4.1.2 Oscilloscope

The oscilloscope is a vital instrument in a circuits laboratory. Hence it is important that the virtual oscilloscope function like a physical oscilloscope. To keep its operation simple, only those basic functions that are used in a circuits laboratory are implemented in the virtual oscilloscope.

This oscilloscope displays waveforms versus time on its screen. The screen is divided into divisions. And each division has 10 units. Both vertical sensitivity and time base may be adjusted by the user. This oscilloscope presently has single channel.

The image of oscilloscope in VELVET is shown below in Fig. 4.2

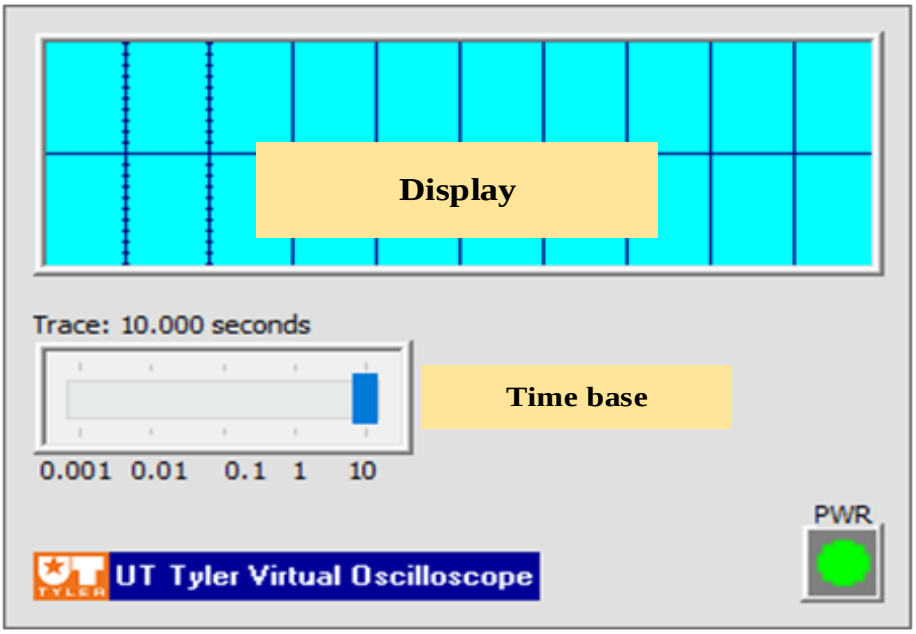


Fig. 4.2 Virtual Oscilloscope

Whenever the oscilloscope is connected to the circuit, its equivalent circuit (1MΩ paralleled by 20 pF) is added to the circuit.

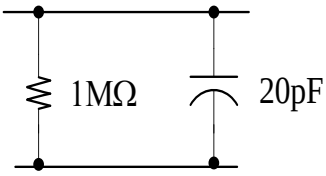


Fig.4.3 Equivalent circuit of oscilloscope input

4.1.3 Multimeter

Another important instrument for digital measurements is digital multimeter. The virtual digital multimeter has a screen to show the output quantity. The multimeter measures different parameters such as DC voltage, AC voltage, DC current, AC current and resistance. The virtual multimeter has two connection points to connect positive and negative probes.

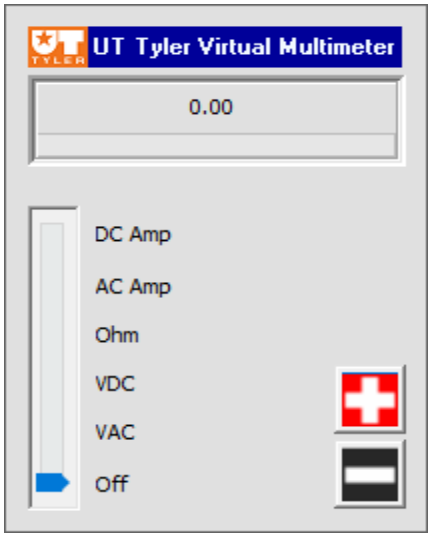


Fig. 4.4 Virtual Multimeter

When multimeter is connected as voltmeter, its equivalent circuit is a high resistance (as shown in Fig. 4.5) which will be connected to the circuit.

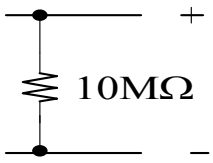


Fig. 4.5 Equivalent voltmeter circuit

Similarly when the multimeter is operated as an ohmmeter, its equivalent circuit (shown in Fig. 4.6), consists of a current source in parallel with a resistance of 10MΩ.

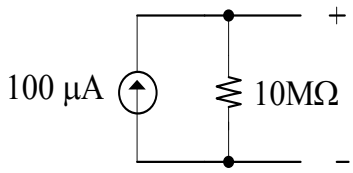


Fig. 4.6 Equivalent ohmmeter circuit of Agilent 34410A DMM on 1K-10K scale

Likewise when multimeter is operated as an ammeter then its equivalent circuit as referred in Fig. 4.7, a low resistance connected in series is connected into the circuit.

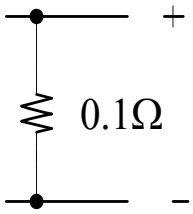


Fig. 4.7 Equivalent circuit of ammeter

4.1.4 Power supply

A virtual power supply as shown in Fig. 4.8 is also developed to be similar to a real power supply. It has positive, negative supply ranges and a ground. Positive and negative supply voltages are independently adjustable.

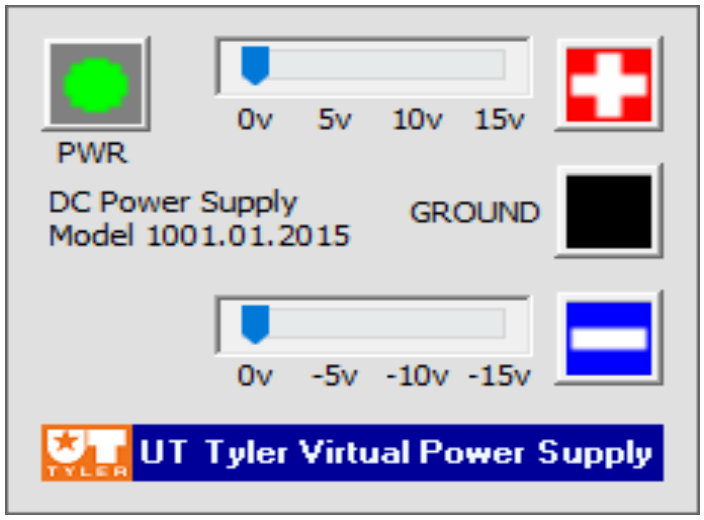


Fig. 4.8 Virtual Power supply

At present, the power supply can deliver unlimited currents. In future work, current limiting of the power supply will be included.

4.1.5 Resistive elements

Refer to Fig. 4.9; VELVET has virtual resistive elements in a separate tab under the general category of passive components. Different ranges of resistance are available in the tab.

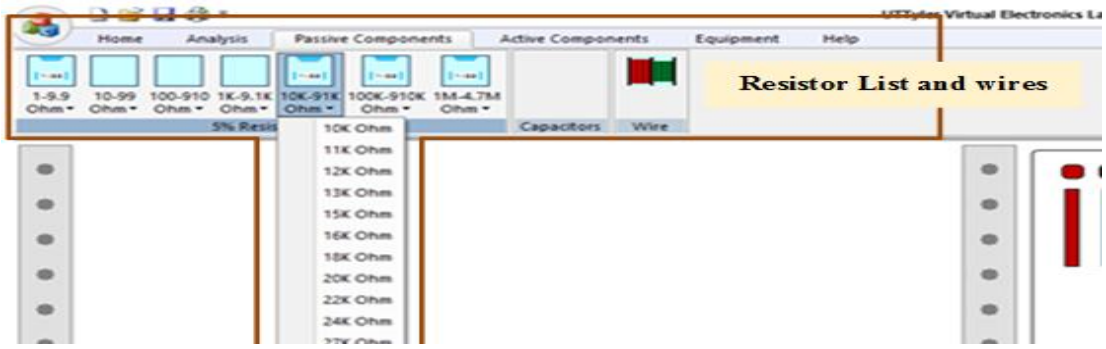


Fig. 4.9 Resistor drop-down list and connecting wires

When one clicks on the required component a message box pops up as shown in Fig 4.10 below.

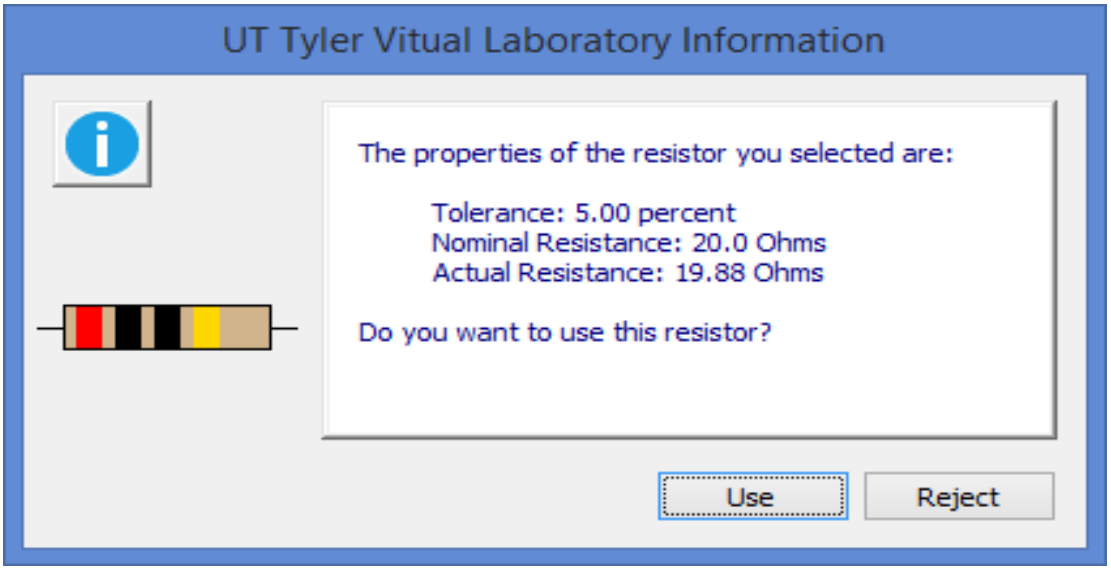


Fig. 4.10 Resistor with actual resistance value similar to realistic resistor.

This message box shows the properties of the selected resistors and asks the user to confirm the component selection. The properties that are shown here are nominal resistance, tolerance and actual resistance. The distribution of resistance values is done randomly using Gaussian probability.

4.1.6 Main screen of VELVET project

Different components are discussed. Figure 4.11 below shows the alignment of all the components. When one opens the VELVET project, the main screen is displayed. To the left are virtual instruments such as the oscilloscope, multimeter and power supply. To the right, the breadboard with power supply connection can be seen. Passive and active components are accommodated by different tabs at the top of the screen.

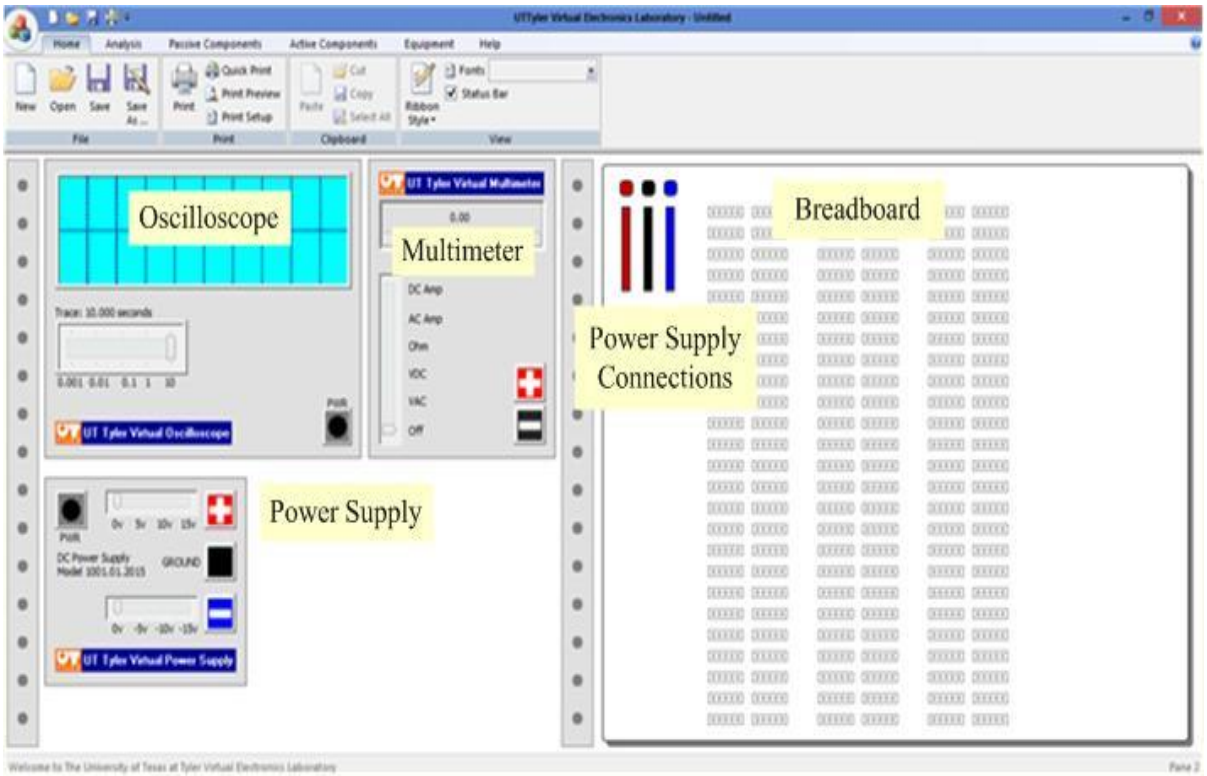


Fig. 4.11 Main screen of VELVET

4.2 Flow of data of animation and proposed algorithm

The flow of data from simulation to animation has yet to be developed.

4.3 Reconfiguration of circuit

In physical laboratories students often remove and add a component to the circuit that is already connected. This real scenario is implemented via reconfiguration of network equations whenever there is addition or deletion of a component. Network matrices are reconstructed whenever a component is added or deleted. This includes connection or disconnection of test equipment.

4.4 Proposed approach to handle failure modes

Failures of components occur commonly in physical laboratories. Some of the failures are due to manufacturing defects while some failures occur due to overstress on components. One such failure that belongs to latter case is electrical overstress of components. When a component is electrical over stressed for certain time component burns out.

These failures can also be imitated in the virtual lab to make the lab more realistic. An approach to handle failure modes is proposed here. Instantaneous power of each resistive element to be accounted and difference of dissipated power to rated power should be integrated over time to check if it falls over threshold power. When this quantity is above threshold power then condition to perform animation of failure is true. This allows users to observe the failure modes in VELVET.

Chapter 5

5.1 Conclusion

In previous chapters we have outlined the benefits and requirements of a virtual laboratory and described previous work on the virtual laboratory. In addition, we have outlined the mathematical development of network equations from the circuit topology. The proposed algorithm has been applied to various circuits, including ones containing controlled sources and reactive elements. The accuracy of the obtained results has been verified with industry standard circuit development software solutions.

The proposed algorithm has been tested with circuits that with sinusoidal excitation, square-wave excitation and triangle-wave excitation. The outputs waveforms are comparable with output waveforms generated by industry standard circuit development software. This is documented in Ch. 3.

The project has so far implemented the proposed algorithm in Visual Studio C++.NET: A virtual breadboard, virtual power supply and virtual instruments such as virtual oscilloscope, virtual multimeter are also implemented. A separate tab for active elements and passive elements has been designed. Resistive elements are designed with typical tolerance of 5% to replicate real components.

5.2 Future work

Even though virtual resistive elements and virtual instruments are developed the essential linkage between the animation and proposed algorithm is yet to be done. And other developments that are to be implemented are the development of animation of reactive elements, development of failure modes and development of necessary animation of failure modes.

Future work also includes development of mathematical models for nonlinear elements such as diodes and operational amplifiers (op-amps). Animation for nonlinear

elements, the linkage between the mathematical model and the animation will be developed.

An approach is proposed to handle diode. When diode is connected to the circuit its equivalent circuit as shown in Fig. 5.1 is added to the circuit. For a small signal silicon diode, typical values for resistance, r_D may be 10-15 Ω and typical values for voltage source V_D are 0.6-0.7V while the diode is conducting. In this process a new internal node which is not present in prototype board is created. In analysis, the diode will initially be assumed to be conducting; however, if the analysis gives a negative value of diode current I_D, r_D will be replaced with a very large value (10 MΩ) and the analysis will be replaced.

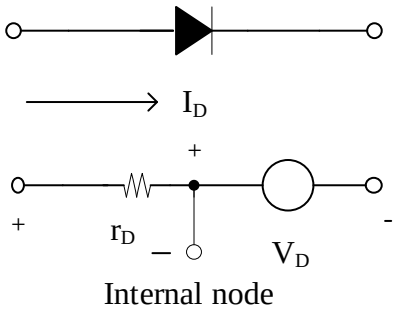


Fig. 5.1 Diode and its equivalent circuit.

As VELVET can handle controlled sources, an example of operational amplifier as shown in Fig 5.1 with controlled sources can be examined. The mathematical model for operational amplifiers can be developed as explained in the proposed algorithm and additionally three internal nodes which are not present in prototype board.

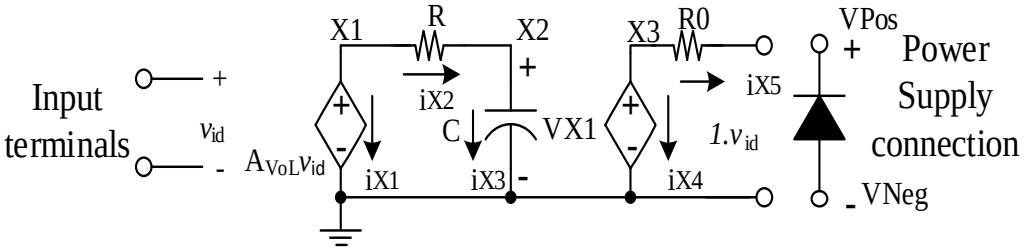


Fig 5.2 Example op-amp circuit with controlled sources

Application of Kirchhoff's Current Law (KCL), at each node gives the following relationships.

Node X_1: $i_{X1} + i_{X2} = 0$ (5.1)

Node X_2: $-i_{X2} + i_{X3} = 0$ (5.2)

Node X_3: $i_{X4} + i_{X5} = 0$ (5.3)

The expression for constitutive relations can be expressed as

$V_{X1} = \min (V_{Pos}, \max (A_{VoL.} V_{id}, V_{Neg}))$ (5.4)

where A_{VoL} is the dc open loop gain.

Break frequency of open loop gain = $\frac{1}{2\pi RC}$

V_{X1} is limited to the range of V_{Pos} to V_{Neg}. And this model does not draw dc operating current as the real op amp does. And here diode is used for reverse polarity of power supply.

References

[1] http://icampus.mit.edu/projects/ilabs/(Retrieved on January 16, 2015).

[2] A.Nafalski, Z. Nedic, J. Machotka, Ö. Göl, A. Scarino, J. Crichton, I. Gustavsson, J. M. Ferreira, D. Lowe, and S. Murray IEEE Transactions on Learning Technologies, 2009. "International Collaboration in Remote Engineering Laboratories: an Approach to Development.

[3] http://www.go-lab-project.eu/ (Retrieved on January 16, 2015).

[4] https://answers.syr.edu/display/ischool/iSchool+Remote+Lab (Retrieved on January 16, 2015).

[5] http://www.umkc.edu/is/remotelabs/ (Retrieved January 16, 2015).

[6] James Edward Fraser, Master Report on "A Guide for the classroom use of IBM ECAP/360".

[7] http://www.circuitstoday.com/circuit-design-and-simulation-softwares (Retrieved on January 16, 2015).

[8] http://circuit.groups.et.byu.net/Comer_solutions/Spice/history.html (Retrieved on January 21, 2015).

[9] http://www.orcad.com/about/orcad-history (Retrieved on January 21, 2015).

[10] http://en.wikipedia.org/wiki/National_Instruments_Electronics_Workbench_Group# History (Retrieved on January 21, 2015).

[11] http://www.ni.com/ (Retrieved on January 16, 2015).

[12] http://www.ni.com/tutorial/11710/en/ (Retrieved on January 30, 2015).

[13] http://digital.ni.com/public.nsf/allkb/4B99B2CD6C0C3B6A86257205005D58E0

[14] http://www.cadence.com/rl/Resources/datasheets/pcb_pspice_simulation_ds.pdf, (Retrieved on January 16, 2015).

[15] http://community.cadence.com/cadence_technology_forums/f/22/t/27484 (Retrieved on January 29, 2015).

[16] http://digital.ni.com/public.nsf/allkb/F7C4924C4E33D17A86257BF6006CCC32 (Retrieved on January 30, 2015).

[17] Ac 2011-2781: Using Portable Electronics Experiment Kits for Electronics Courses in a General Engineering Program.

[18] https://www.eng.vt.edu/news/virginia-tech-s-lab-box-showcased-national-department-heads-meeting (Retrieved on January 28, 2015).

[19] Mohamed Shaheen, Kenneth A. Loparo and Marcus R.Buchner "Remote Laboratory Experimentation" Proceedings of the American Control Conference, Philadelphia, Pennsylvania, June 1998.

[20] E.Tanyildizi and A.Orhan "A Virtual Electric Machine Laboratory for Synchronous Machine Application" Wiley online publications.

[21] Cui Xiaoyan, Zhang Xiaodong and Chen Xi "A Virtual Laboratory for Electrical and Electronics Teaching" 2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications Proceedings.

[22] Hasanul A.Basher, Saliman A.Isa, and M'Hamed A.Henini "Virtual Laboratory for Electrical Circuit Course" 2004 IEEE Southeast Con Proceedings.

[23] C.C.Ko, Ben M.Chen, S.H.Chen, V.Ramakrishnan, R.Chen, S.Y.Hu and Y.Zhuang "A large scale Web-based virtual oscilloscope laboratory experiment" IEEE Science and Education Journal Vol 9, No 2, pp 69-76, April 2000.

[24] Chung-Wen Ho, Albert E.Ruehli and Pierce A. Brennan "The Modified Nodal Approach to Network Analysis" IEEE Transactions on circuits and systems, Vol CAS-22, No 6, June 1975.

[25] Andrei Vladimirescu,"The Spice Book"Jhon Wiley &Sons, Inc.

[26] Franklin H.Branin, Jr, Gerald R.Hogsett, Richard L.Lunde and Lawrence E.Kugel "ECAP II A New Electronic Circuit Analysis Program" IEEE Journal of Solid State Circuits, Vol Sc-6, No4, August 1971.

Appendix-A

A.1 Software Documentation

```
/* Name: Sreelatha Aihloor Subramanyam
Project: Virtual Electronics Laboratory Visualized Education and Training(VELVET)
Thesis: December 2015
Date: 10/1/2015

Purpose of Program:This program takes the circuit parameters as a text file and
produces the node voltages and branch currents.

Format of Text file:
X X1 X2 X3 X4 X5 X6
X      - Prototype board from node
X1     - Prototype board to node
X2     - Symbol of the component
X3     - Controlled source from node
X4     - Controlled source to node
X5     -  Values of reactive elements.If the element is not reactive then 0 is
asserted
X6     - Values of the component

For Eg 4 2 L 0 0 0.1 1   here 4 is prototype board from node,2 is prototype board
to node,L is symbol for inductor ,0 is controlled source from node
 0 is controlled source to node ,0.1 is the value of the inductor,1 is the initial
branch current of inductor.

 Symbols used in this program
 R     - Resistor
 L     - Inductor
 C     - Capacitor
 P     - Positive Voltage
 N     - Negative Voltage
 Q     - Sinusoidal Excitation
 Y   - Square-wave Excitation
 Z   - Triangle-wave Excitation
 E     - Voltage controlled voltage source
 F     - Current controlled current source
 G     - Voltage controlled current source
 H     - Current controlled voltage source

*/
/*Preprocessor derivatives,Header files,file stream header files,string operation
header files */
//#include "stdafx.h"
#include<iostream>
#include<conio.h>
#include<fstream>
#include<stdlib.h>
#include<string.h>
```

```
#include<sstream>
#include<vector>
#include<string>
#include<math.h>
#include<cstring>
using namespace std;
/* Structure for a two dimensional matrix
  Data Type struct
  Elements pointer to two dimensional array */
struct TwoDMatrix
{
float **a;
int row;
int column;
};
/*   Class   matrix   contains   the   methods   for   create   matrix,initialize
matrix,horizontal contatenation,vertical concatenation,ground node deletion,
   guass elimination,s matrix update
    */
class Matrix
{
       public:

              struct TwoDMatrix createMatrix(int row, int coumn); //Create matrix
of desired row anf column method
              struct TwoDMatrix initialize_with(float val, struct TwoDMatrix x);
//Matrix initialize method
              struct   TwoDMatrix   hconcatenate(struct   TwoDMatrix   x,   struct
TwoDMatrix y); // horizontal concatenation method
              struct   TwoDMatrix   vconcatenate(struct   TwoDMatrix   x,   struct
TwoDMatrix y); // vertical  concatenation method
              struct TwoDMatrix Matrix :: mydelete(struct TwoDMatrix x, int r,int
c);//ground node deletion method
              vector  <double>  Matrix::  gauss(struct  TwoDMatrix  x,int  r);//gauss
elimination method
              struct   TwoDMatrix   Matrix   ::supdate(struct   TwoDMatrix   x,int
r,vector<double> ysvt,vector<double> K);//s matrix update method
};
/* Vertical Concatenate method
   This method is used to contatenate two rows one on above of other i.e. vertical
concatenation.
   parameters: x,y of data type struct  structure TwoDmatrix
   return data type struct a two dimensional array is returned
   */
struct TwoDMatrix Matrix :: vconcatenate(struct TwoDMatrix x, struct TwoDMatrix y)
{
              float **a;          //2D matrix a  pointer
              int i=0,j;        // indexing elements
       struct TwoDMatrix ret_mat; //return 2D matrix of struct data type

              ret_mat.row = x.row + y.row;   //rows of two matrices x and y are
added
              a=new float*[ret_mat.row];
              ret_mat.column = x.column;
              for(int i=0;i<ret_mat.row;i++)
              {
                     a[i]=new float[ret_mat.column];
              }
```

```
		while(i < x.row)
		{
			for(j=0;j<x.column;j++)
			{
				a[i][j]=x.a[i][j];
			}

			i++;
		}
		for(i=x.row;i < ret_mat.row;i++)
		{
			for(j=0;j<y.column;j++)
			{
				a[i][j]=y.a[i-x.row][j];
			}

		}
		ret_mat.a=a;
	return ret_mat;

}
/* Horizontal Concatenate method
   This method is used to contatenate two columns side by side i.e. Horizontal
concatenation.
   parameters: x,y of data type struct  structure TwoDmatrix
   return data type struct a two dimensional array is returned
*/

struct TwoDMatrix Matrix :: hconcatenate(struct TwoDMatrix x, struct TwoDMatrix y)
{
	float **a;
	int i=0,j;
	struct TwoDMatrix ret_mat;
	if(x.row == y.row)
	{
		ret_mat.row = x.row;
		a=new float*[x.row];

		for(i=0;i<ret_mat.row;i++)
		{
			a[i]=new float[x.column+y.column];
		}
		ret_mat.column=x.column+y.column;
		i=0;
		while(i < ret_mat.row)
		{
			for(j=0;j<x.column;j++)
			{
				a[i][j]=x.a[i][j];
			}
			for(j=x.column;j<ret_mat.column;j++)
			{
				a[i][j] = y.a[i][j-x.column];
			}
			i++;
		}
		ret_mat.a=a;
```

```
		}

	return ret_mat;
}
/*Create Matrix method:The purpose of this method is to create a two dimensional
matrix of given rows and columns.
	parameters: r(rows),c(columns) of data type int
	Return: A matrix of dimensions r x c is created.The return type is struct
TwoDMatrix
*/
struct TwoDMatrix Matrix :: createMatrix(int r, int c)
{
	float **a;
	a=new float*[r];
	for(int i=0;i<r;i++)
	{
		a[i]=new float[c];

	}
	struct TwoDMatrix ret_matrix;
	ret_matrix.a=a;
	ret_matrix.row=r;
	ret_matrix.column=c;
	return ret_matrix;
}
/*Initialize matrix method:The purpose of this method is to initialize the given
matrix with given value.This is essentially
  used to create zero matrix.This zero matrix is used in updating the size of the
given  matrix whenever a new component is
  added to the circuit.
  Parameters; val( value to be used to initialize) of float data type and x (given
two dimensional matrix to be updated)of
  data type struct TwoDMatrix.
  Return:Method returns a 2D matrix of data type struct TwoDMatrix
*/
struct TwoDMatrix Matrix :: initialize_with(float val, struct TwoDMatrix x)
{

	for(int i=0; i<x.row;i++)
	{
		for(int j=0; j<x.column;j++)
		{
			x.a[i][j]=val;
		}
	}
	return x;
}
/* supdate method: This method is exclusively used to update S matrix.The purpose
of this method is to update S matrix at the given index
 of whether the index may be of reactive elements or of sinusoidal /triangle-
wave/square -wave voltage  excitation.The respective elements
 of S are updated with given K matrix(where matrix has updated values).

 Parameters:x(Two dimensional matrix to be updated,in this case S matrix),r(rows
of data type int),vector K( of dta type double)(matrix with
 updated values)
 Return : two dimensional matrix of data type struct(here updated S matrix is
returned)
```

```
*/
struct TwoDMatrix Matrix ::supdate(struct TwoDMatrix x,int r,vector<double>
ysvt,vector<double> K)
{
	float **a=x.a;
	int i=0;
	int j;
	struct TwoDMatrix ret_mat;
	int k=0;

		for(int i=0;i<x.row;i++)
		{
			for(int j=0;j<x.column;j++)
			{
				while (k<r)
				 {
				 if(i== ysvt[k])
				 {
					a[i][j]=K[k]; k++;
				 }
				   break;
				}
			}
		}
		ret_mat.row=x.row;
		ret_mat.column=x.column;
		ret_mat.a=a;
		return ret_mat;
	}
/*mydelete: This method is exclusively used to delete rows and columns of the
given matrix.
 Here this method is used to delete column that belongs to ground node and KCL row
that belongs to ground node

Parameters: struct Two dimensional matrix x(matrix that is to be changed),r(row #
of data type int),c(column # of data type int)
Return : two dimenstional matrix of data type struct*/
struct TwoDMatrix Matrix :: mydelete(struct TwoDMatrix x, int r,int c)
{
	float **a=x.a;
	int i=0;
	int j;
	struct TwoDMatrix ret_mat;
	float **d ;
	d=new float*[x.row-1];
	for( i=0;i<x.row-1;i++)
	{
		d[i]=new float[x.column-1];

	}
	int DELETED_ROW=r;
	int DELETED_COLUMN=c;

	 for ( i = 0; i < x.row; i++)
	 if (i != DELETED_ROW)
	 {
	     for ( j = 0; j < x.column; j++)
```

```
                if (j != DELETED_COLUMN)
                {
                    if (i < DELETED_ROW && j < DELETED_COLUMN)
                        d[i][j] = x.a[i][j];
                    else if (i < DELETED_ROW && j > DELETED_COLUMN)
                        d[i][j-1] = x.a[i][j];
                    else if (i > DELETED_ROW && j < DELETED_COLUMN)
                        d[i-1][j]=x.a[i][j];
                    else
                        d[i-1][j-1]=x.a[i][j];
                }
        }

              ret_mat.row=x.row-1;
              ret_mat.column=x.column-1;
              ret_mat.a=d;
              return ret_mat;
}
/* guass: Guass method is used to perform guass elimination.This guass elimination
technique is ued to solve matrix unknowns
  X=A/B(AX=B).
  Parameters: x(the matrix A of data type struct TwoD matrix),r( number of rows in
the matrix)(data type int)
  Return: returns a vector of unknowns y(data type vector(double))

*/
vector <double> Matrix:: gauss(struct TwoDMatrix x,int r)
{
       int i=0;
       int j=0;
       int k=0;
       int n=r;

    for (int i=0; i<n; i++) {
        // Search for maximum in this column
        double maxEl = abs(x.a[i][i]);
        int maxRow = i;
        for (int k=i+1; k<n; k++) {
            if (abs(x.a[k][i]) > maxEl) {
                maxEl = abs(x.a[k][i]);
                maxRow = k;
            }
        }

        // Swap maximum row with current row (column by column)
        for (int k=i; k<n+1;k++) {
            double tmp = x.a[maxRow][k];
            x.a[maxRow][k] = x.a[i][k];
            x.a[i][k] = tmp;
        }

        // Make all rows below this one 0 in current column
        for (int k=i+1; k<n; k++) {
            double c = -x.a[k][i]/x.a[i][i];
            for (int j=i; j<n+1; j++) {
                if (i==j) {
                    x.a[k][j] = 0;
                } else {
```

```
                    x.a[k][j] += c * x.a[i][j];
                }
            }
        }
    }

    // Solve equation Ax=b for an upper triangular matrix A
    vector<double> y(n);
    for (int i=n-1; i>=0; i--) {
        y[i] = x.a[i][n]/x.a[i][i];
        for (int k=i-1;k>=0; k--) {
            x.a[k][n] -= x.a[k][i] * y[i];
        }
    }
    return y;
}
/* This function trim is used to trim the given string
 parameters: string S
 Return: The resultant string
*/
string trim( const string& s )
  {
  string result( s );
  result.erase( result.find_last_not_of( " " ) + 1 );
  result.erase( 0, result.find_first_not_of( " " ) );
  return result;
  }

int main()
{
       string list_of_words;   //no.of words in each line of text file
        double  w[1000],w1[1000];//array  of  double  to  store  the  values  of  x  at
various timesteps.to display them in a plot
        double t3=0;//cycle time for triangle wave
ifstream fin;  //file input
       fin.open("cccscircuit.txt"); // opens the text file

       int num_of_new_nodes = 0; // number of nodes in the circuit
       int next_node = 1;        // next node
       int prototype_nodes[30]; // array of proto type nodes.it is intialized to
30.i.e it can handle up to 30 nodes
       for(int i=0; i<30; i++)    //initialize array of proto type nodes with -1
.for identification that the node is
                                                        //not occupied
       {
              prototype_nodes[i] = -1;
       }

       int   pf,   pt,nodep,noden;   //proto   type   from   node(pf),proto   type   to
node(pt),controlled source dependent from node (nodep)
                                                // controlled source dependent to
node (noden)
       float       val,ampli,freq;                    //value(val)      of      the
component,ampli(amplitude),freq(frequency) of sine/triangle/square wave
       long double pi=3.14159265358979323846264338327950288 4;//PI value
       char type;    //type used for element type
       int  from_node,  to_node;//  electrical  from  node(from_node),electrical  to
node(to_node)
```

```
	string line;//line in the text file
	int j = 0; //index
	string	single_line,	spf,	spt,	sval,snodep,snoden,sval2,stype;//
line(single_line),string prototype from node(spf),string proto type

				// to node(spt)string value,nodep,noden,val2(value of
reactive element),element type
	const int n = 30; // no.of electrical nodes
	char ptype;	//element type

	 int	ground,ground1,ground2;//	reference	ground	node(ground),positive
voltage ground(ground1),neagative voltage ground(ground2)

	struct		TwoDMatrix		a,		a1,a2,a3,a4,s,		zeroes,
ones,y,s1,b,c,primea,primea1,primey,s2,u,B1,B2,B,primeB,svt,st;
	//define variables of data type struct
	Matrix m; // class declaration
	/* Initialze two dimensional matrices with size zero*/
	a1=m.createMatrix(0,0);
	a2=m.createMatrix(0,0);
	a4=m.createMatrix(0,0);
	s=m.createMatrix(0,0);
	y=m.createMatrix(0,0);
	s1=m.createMatrix(0,0);
	b=m.createMatrix(0,0);
	c=m.createMatrix(0,0);
	a=m.createMatrix(0,0);
	B=m.createMatrix(0,0);
	B1=m.createMatrix(0,0);
	B2=m.createMatrix(0,0);
	svt=m.createMatrix(0,0);
	st=m.createMatrix(0,0);
	primea=m.createMatrix(0,0);
	//xdot=m.createMatrix(0,0);
	float t=0; // time period
	int new_row = 0; // new row
	int new_branch = 0;// new branch
	float val2;// value of reactive elements
	double vg;//source voltage
	int	num_sv=0,num_sv1=0;//num_sv	used	to	index	the	number	of	reactive
elements for B1 ,num_sv1 is used to index number of reactive elements for B2
	string word;//each word
	string worda[7];//	7 strings in each line
  int	 cntr = 0,i=0; //counter(cntr),index(i)
	while(getline(fin, list_of_words))	//get line untill the end is reached
	{
		single_line = list_of_words;
		list_of_words += " ";
		istringstream iss( list_of_words );
		/* Each line is trimed into no.of words*/
		while (getline( iss, word, ' ' ))
		{
		worda[i] =trim(word);
			i++;
	    }
		i=0;
		/* Read each word into respective springs*/
```

```
        while(i<7)
       {
        // cout<<"the worda"<<worda[i]<<endl;
         spf=worda[0];
        spt= worda[1];
         stype=worda[2];
         snodep=worda[3];
         snoden=worda[4];
        sval2= worda[5];
         sval= worda[6];
        i++;
              }
        i=0;//initialize with 0
        /* convert strings into integers or floats*/

              pf = atoi(spf.c_str());
              pt = atoi(spt.c_str());
              ptype = stype[0];
              nodep= atoi(snodep.c_str());
              noden = atoi(snoden.c_str());

              val=atof(sval.c_str());
              val2=atof(sval2.c_str());

              //     cout<<ptype<<endl;
       /*     getchar();
              cout<<pf<<endl;  /*  These  statements  are  used  to  print  values  of
pf,pt,ptyoe,nodep,noden,val2,val//
              cout<<pt<<endl;
              cout<<ptype<<endl;
              cout<<nodep<<endl;
              cout<<noden<<endl;
              cout<<val2<<endl;
              cout<<val<<endl;
              getchar();

              */
              /*  Extraction  of  electrical  node  numbers,electrical  from  node,to
node and update number of electrical nodes*/
              if(prototype_nodes[pf] == -1)

              {
                     prototype_nodes[pf] = next_node;
                     from_node = next_node;
                     next_node = next_node+1;
                     num_of_new_nodes = num_of_new_nodes+1;

              }
              else
              {
                     from_node = prototype_nodes[pf];

              }

       //    cout<<"num_nodefrom"<<num_of_new_nodes<<endl;//    print    number    of
elelctrical node and from node
              //std::cout<<"from_node"<<from_node<<endl;
              if(prototype_nodes[pt] == -1)
```

```
	{
		prototype_nodes[pt] = next_node;
		to_node = next_node;
		next_node = next_node+1;
		num_of_new_nodes = num_of_new_nodes+1;
	}


	else
	{
		to_node = prototype_nodes[pt];
	}

	//std::cout<<"to"<<to_node<<endl;

	j++;//increment index

	/*for(int i=0;i<20;i++) // used to print array of electrical nodes
	{
	  cout<<prototype_nodes[i];
	}
	getchar();*/

	//############! A1 matrix
	// formation of  A1
from_node=from_node-1;
to_node=to_node-1;
 nodep=nodep-1;
 noden=noden-1;

	if(a1.row == 0)
	{
		a1=m.createMatrix(1,2);
		a1=m.initialize_with(0,a1);
	}
	else
	{     zeroes=m.createMatrix(1,a1.column);
			zeroes=m.initialize_with(0,zeroes);
			a1=m.vconcatenate(a1,zeroes);
		switch(num_of_new_nodes)
		{
		case 2:  zeroes=m.createMatrix(a1.row,2);
			 zeroes=m.initialize_with(0,zeroes);
				a1=m.hconcatenate(a1,zeroes);break;
		case 1: zeroes=m.createMatrix(a1.row,1);
			   zeroes=m.initialize_with(0,zeroes);
				a1=m.hconcatenate(a1,zeroes);break;
		default: break;
		}
	}

	 switch(ptype)
	 {
	 case 'R':    a1.a[a1.row-1][from_node]=(1/val);
				a1.a[a1.row-1][to_node]=-(1/val);break;
	 case 'N':  a1.a[a1.row-1][from_node]=1;
				a1.a[a1.row-1][to_node]=-1;break;
```

```
 case 'E':  a1.a[a1.row-1][nodep]=-val;
                    a1.a[a1.row-1][noden]=val;
                    a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
 case 'G':  a1.a[a1.row-1][nodep]=val;
                    a1.a[a1.row-1][noden]=-val;break;
 case 'P':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
case 'Q':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
case 'X':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
case 'Z':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
case 'Y': a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
 case 'C':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
 case 'H':  a1.a[a1.row-1][from_node]=1;
                    a1.a[a1.row-1][to_node]=-1;break;
 default: break;
 }

//################  A2 matrix formation

if(a2.row == 0)
{
       a2=m.createMatrix(1,1);
       a2=m.initialize_with(0,a2);
       new_branch=0;
       new_row=0;

}
else
{
       zeroes=m.createMatrix(1,a2.column);
       zeroes=m.initialize_with(0,zeroes);
       a2=m.vconcatenate(a2,zeroes);

       zeroes=m.createMatrix(a2.row,1);
       zeroes=m.initialize_with(0,zeroes);
       a2=m.hconcatenate(a2,zeroes);

       new_row=a2.row-1;
       new_branch=a2.column-1;
}

switch(ptype)
{
case 'R' : a2.a[new_row][new_branch] = -1;break;
case 'I': a2.a[new_row][new_branch] = 1;break;
case 'L': a2.a[new_row][new_branch] = 1;break;
case 'G' : a2.a[new_row][new_branch] = -1;break;
case 'F': a2.a[new_row][nodep]=val;
                a2.a[new_row][new_branch]=-1;break;
case'H': a2.a[new_row][nodep]=-val;break;
```

```
    default:a2.a[new_row][new_branch] = 0;break;
    }


    //############ !A4 matrix

           if(a4.row == 0)
    {
           a4=m.createMatrix(2,1);
           a4=m.initialize_with(0,a4);

           a4.a[from_node][a4.column-1]=1;
           a4.a[to_node][a4.column-1]=-1;

    }
    else
    {
           zeroes=m.createMatrix(a4.row,1);
                  zeroes=m.initialize_with(0,zeroes);
                  a4=m.hconcatenate(a4,zeroes);
           switch(num_of_new_nodes)
           {
           case 2:  zeroes=m.createMatrix(2,a4.column);
                    zeroes=m.initialize_with(0,zeroes);
                         a4=m.vconcatenate(a4,zeroes);break;
           case 1: zeroes=m.createMatrix(1,a4.column);
                      zeroes=m.initialize_with(0,zeroes);
                         a4=m.vconcatenate(a4,zeroes);break;
           default: break;
           }

           a4.a[from_node][a4.column-1]=1;
           a4.a[to_node][a4.column-1]=-1;
           }


    // ########## S matrix
if(s.row == 0)
{
       s=m.createMatrix(1,1);
       s=m.initialize_with(0,s);
}
else
{
       zeroes=m.createMatrix(1,1);
       zeroes=m.initialize_with(0,zeroes);
       s=m.vconcatenate(s,zeroes);
}
switch(ptype)
{
    case 'E':s.a[s.row-1][0]=0;break;
    case 'P':s.a[s.row-1][0]=val;break;
    case 'Q':s.a[s.row-1][0]=val;break;
    case 'Y':s.a[s.row-1][0]=val;break;
    case 'X':s.a[s.row-1][0]=val;break;
    case 'Z':s.a[s.row-1][0]=val;break;
     case 'N':s.a[s.row-1][0]=val;break;
     case 'L':s.a[s.row-1][0]=val;
```

```
                        break;
                case 'C':s.a[s.row-1][0]=val;
                        break;
        default:break;
        }
        /* if the source is sinusidal /triangle-wave / square-wave them
amplitude and frequency is derived*/
        if ((ptype =='Y')||(ptype == 'Q')||(ptype == 'Z'))
        {
               ampli=nodep+1;
                freq=noden+1;

        }
        //Indexing the source element in the S matrix (source tag)

         if((ptype =='X') || (ptype =='Y') || (ptype == 'Q') ||(ptype =='Z'))
             {
                        if(st.row == 0)
                         {
                            st=m.createMatrix(1,1);
                            st=m.initialize_with(0,st);
                            st.a[st.row-1][0]=s.row-1;
                        }

                        else{
                            zeroes=m.createMatrix(1,1);
                                   zeroes=m.initialize_with(0,zeroes);
                                   st=m.vconcatenate(svt,zeroes);

                               st.a[st.row-1][0]=s.row-1;
                        }

             }
             // Indexing for State Variable elements in the S matrix (state
variable Tag)

        if(ptype =='L')
             {
                        if(svt.row == 0)
                         {
                            svt=m.createMatrix(1,1);
                            svt=m.initialize_with(0,svt);
                            svt.a[svt.row-1][0]=s.row-1;
                        }
                        else{
                            zeroes=m.createMatrix(1,1);
                                   zeroes=m.initialize_with(0,zeroes);
                                   svt=m.vconcatenate(svt,zeroes);

                               svt.a[svt.row-1][0]=s.row-1;
                        }

        }
         if(ptype =='C')
             {
                        if(svt.row == 0)
                            {
                                   svt=m.createMatrix(1,1);
```

```
                                svt=m.initialize_with(0,svt);
                                svt.a[svt.row-1][0]=s.row-1;

                            }

                        else
                                {
                                zeroes=m.createMatrix(1,1);
                                zeroes=m.initialize_with(0,zeroes);
                                svt=m.vconcatenate(svt,zeroes);
                                svt.a[svt.row-1][0]=s.row-1;
                                }
                        }

       //Designate ground if voltage sources are either sinusoidal/triangle-
wave/square-wave.
              if((ptype=='Q') || (ptype=='X') ||(ptype=='Y') || (ptype =='Z'))
        {
                ground1 = to_node+1;  // have to tackle with situation of both dc
and ac supply is given in future

        }
       //to_node of  electrical node of element type of positive voltage source is
made reference node.
              //In previous execution statements to_node is decremented by 1 so 1
is added again to to_node.
        if(ptype=='P')
        {
                ground1 = to_node+1;

        }
        //if the voltage source is negative then from_node of the voltage source
is taken as ground
        if(ptype=='N')
        {
                ground2 = from_node+1;
        }
        //If both the grounds are equal then ground is ground corresponding to
positive voltage else ground is the ground
        //of positive voltage source.
        if(ground1 == ground2)
        {
               ground=ground1;
        }
        else
        {
               ground=ground1;
        }

         // B1 matrix formation
        /* B1 matrix is formed to consider reactive elements.It has dimensions of
nodes by branches*/

              if(B1.row == 0)
              {
                     B1=m.createMatrix(1,2);
                     B1=m.initialize_with(0,B1);
              }
```

```
                else
                {     zeroes=m.createMatrix(1,B1.column);
                              zeroes=m.initialize_with(0,zeroes);
                              if(ptype =='L')
                              {B1=m.vconcatenate(B1,zeroes);
                               num_sv++;
                              }
                              else if (ptype == 'C')
                                     {B1=m.vconcatenate(B1,zeroes);
                               num_sv++;
                              }

                       switch(num_of_new_nodes)
                       {
                       case 2:  zeroes=m.createMatrix(B1.row,2);
                                zeroes=m.initialize_with(0,zeroes);
                                     B1=m.hconcatenate(B1,zeroes);break;
                       case 1: zeroes=m.createMatrix(B1.row,1);
                                  zeroes=m.initialize_with(0,zeroes);
                                     B1=m.hconcatenate(B1,zeroes);break;
                       default: break;
                       }

                }

                switch(ptype)
                 {
                 case 'L':    B1.a[num_sv-1][from_node]=(1/val2);
                                     B1.a[num_sv-1][to_node]=-(1/val2);break;
                  default:break;
                }

                // Formationof B2 matrix
                /* B2 matrix is formed to take care of reactive elements.B1 and B2
are formed from constitutive relations*/
                if(B2.row == 0)
                {
                       B2=m.createMatrix(1,1);
                       B2=m.initialize_with(0,B2);
                       new_branch=0;


                }
                else
                {
                       zeroes=m.createMatrix(1,B2.column);
                       zeroes=m.initialize_with(0,zeroes);

                       if(ptype =='L')
                              {B2=m.vconcatenate(B2,zeroes);
                               num_sv1++;
                              }
                       else if (ptype == 'C')
                                     {B2=m.vconcatenate(B2,zeroes);
                               num_sv1++;
                              }
                       zeroes=m.createMatrix(B2.row,1);
                       zeroes=m.initialize_with(0,zeroes);
```

```
                        B2=m.hconcatenate(B2,zeroes);
                        new_branch=B2.column-1;
                }

                switch(ptype)
                {
                case 'C' : B2.a[num_sv1-1][new_branch] = 1/val2;break;
                default:break;
                }
                num_of_new_nodes=0;

        } // end of while loop

/*      for(int i=0;i<20;i++)
                {
                  cout<<prototype_nodes[i];
                }
                getchar();*/
        /*cout<<"svt tag"<<endl;
        for(int i=0;i<svt.row;i++)
        {
                for (int j=0;j<svt.column;j++)
                        cout<<svt.a[i][j]<<"\t";
        cout<<endl;

        } getchar();*/
        //getchar();

        /*cout<<"st tag"<<endl;
        cout<<"st row"<<st.row<<endl;
        cout<<"st column"<<st.column<<endl;

        for(int i=0;i<st.row;i++)
        {
                for (int j=0;j<st.column;j++){
                        cout<<st.a[i][j]<<"\t";
        cout<<endl;
                }
        }
        getchar();*/
        /*for(int i=0;i<B1.row-1;i++)
        {
                for(int j=0;j<B1.column;j++)
                {
                        cout<<B1.a[i][j]<<"\t";
                }
                cout<<endl;
        }*/
        //getchar();
        //cout<<"############"<<endl;
/*for(int i=0;i<B2.row-1;i++)
        {
                for(int j=0;j<B2.column;j++)
                {
                        cout<<B2.a[i][j]<<"\t";
                }
                cout<<endl;
        }
```

```
	*/
	// #########! B matrix Formation
	B=m.hconcatenate(B1,B2);   // concatenate B1 and B2
 /* for(int i=0;i< B.row;i++)
	{
		for(int j=0;j<B.column;j++)
		{
			cout<<B.a[i][j] <<"\t";
		}
		cout<<endl;
	}
  getchar();*/
   //cout<<"############################"<<endl;
   // delete ground column


	primeB = m.mydelete(B,B.row-1,ground-1); // delete the column corresponding
to ground
	/*for(int i=0;i< primeB.row;i++)
	{
		for(int j=0;j<primeB.column;j++)
		{
			cout<<primeB.a[i][j] <<"\t";
		}
		cout<<endl;
	}
	getchar();*///print B' matrix

	//######! A3 matrix Formation

	a3=m.createMatrix(a1.column,a1.column);
	a3=m.initialize_with(0,a3);

	//############! Y matrix  formation
	y=m.createMatrix(a1.column,1);
	y=m.initialize_with(0,y);

	//##############! Formation of S1 matrix
	s1=m.vconcatenate(s,y);

    //#################! After deletion of RHS element corresponding to ground node
	 primey=m.createMatrix(a1.column-1,1);
	primey=m.initialize_with(0,primey);

	// ##############! Formation of A matrix
	 b=m.hconcatenate(a1,a2);
	 c=m.hconcatenate(a3,a4);
	 a=m.vconcatenate(b,c);
	 /*
	 for(int i=0;i<a2.row;i++)
	 {
		for(int j=0;j<a2.column;j++)
		{
			cout <<a2.a[i][j]<<"\t";
	   }
		cout<<endl;
	 }*/ // print A2 matrix
```

```
	/*cout<<a4.row<<endl;
	cout <<a4.column<<endl;
	 for(int i=0;i<a4.row;i++)
	 {
		for(int j=0;j<a4.column;j++)
		{
			cout <<a4.a[i][j]<<"\t";
	    }
		cout<<endl;
	 }*/ // print A4 matrix

	/*cout<<a1.row<<endl;
	cout <<a1.column<<endl;
	 for(int i=0;i<a1.row;i++)
	 {
		for(int j=0;j<a1.column;j++)
		{
			cout <<a1.a[i][j]<<"\t";
	    }
		cout<<endl;
	 }*/ // Print A1 matrix

	//cout<<s.row<<endl;
	//cout<<s.column<<endl;
	/*for(int i =0;i<s.row;i++)
	{
	       for(int j=0;j<s.column;j++)
	       {
	       cout<<s.a[i][j]<<"\t";
	       }
	       cout<<endl;
	}getchar();*/ // Print S matrix
//cout<<"##############################"<<endl;

/*	cout<<a.row<<endl;
	cout<<a.column<<endl;
	for(int i =0;i<a.row;i++)
	{
	       for(int j=0;j<a.column;j++)
	       {
	       cout<<a.a[i][j]<<"\t";
	       }
	       cout<<endl;
	}
		getchar();*/ //Print A matrix

	//Delete row and column corresponding to ground node
	 primea=m.mydelete(a,a1.row+ground-1,ground-1);
	 /*for(int i=0;i<primea.row;i++)
	 {
		for(int j=0;j<primea.column;j++)
		{
			cout<<primea.a[i][j]<<"\t";
		}
		cout<<endl;
	 }
	getchar();*/ //Print A matrix after deletion of ground node
```

```
        //If there are no reactive elements in the circuit
        if (svt.row == 0)
        {
                s2=m.vconcatenate(s,primey); //concatenate S matrix and Y' matrix
and store in S2 matrix
                primea1=m.hconcatenate(primea,s2);//concatenate matrix A' and S2
matrix
               vector <double>x(primea.row); // Initialize a vector x (unknown node
voltages and branch current
                                              //vector of size primea.row
                x=m.gauss(primea1,primea.row);//Using guass method find unknown
values of vector x
       cout<<"##########################"<<endl;
        cout<<"The Results are:"<<endl;
        for(int i=0;i<primea.row;i++) //Print the output values of vector x
        {
                cout<<x[i]<<endl;
        }
       cout<<"##########################"<<endl;
        }
        else  //If there are reactive elements in the circuit
        {
        float t1=0; //timestep
       for(int i=0;i<1000;i++){ //This fir loop is used for 1000 timesteps

              if(st.row   >   0){   //If   the   voltage   source   doesnot   include
sinusoidal/triangle-wave/square-wave

                     if (ptype =='Q'){ //If voltage source is of sinusoidal type
then Vg=Asin(2*PI*f)

               vg = ampli*sin(freq*2*pi*t1);
                     }
                     if(ptype=='X'){//
                      vg=val;
                     }
                     if((ptype =='Y') ||(ptype == 'Z'))
                     {
                            double T=1/freq;

                            if (t3 <= T/2)
                            {
                                   if (ptype =='Y')
                                   vg= -ampli+(t3*((4*ampli)/T));
                                   else if (ptype =='Z')
                                          vg=ampli;
                            }
                            if( (t3 > T/2) && (t3 <= T))
                            {
                                   if(ptype =='Y')
                                          vg= ampli -((t3-T/2)*((4*ampli)/T));
                                   else if (ptype =='Z')
                                          vg= -ampli;
                            }

                            t3=t3+0.00001; //Increment timestep of triangle -wave
within
                            if(t3>T){
```

```
                                t3=t3-T;
                        }


                }

        t1=t1+0.00001; //Increment Timestep of the whole circuit

        vector <double>yst(st.row);  //st tag in vector form( st tag is used
to indexing source element in S matrix
        //cout<<"st.row"<<st.row<<endl;
        //cout<<"st.column"<<st.column<<endl;getchar();
    for(int i=0;i<st.row;i++) // convert two dimenstional  column matrix into a
vector for easy manipulation
    {
        for(int j=0;j<st.column;j++){
            yst[i]=st.a[i][j];

            }

    }
    /* Updating S matrix at the source element at every timestep*/
         for(int i=0;i<s.row;i++)
         {
            for(int j=0;j<s.column;j++)
            {

                 if(i== yst[0])
                {
                    s.a[i][j]=vg;

                }


            }
         }
         /* Print newly formed S matrix*/
    /*   for(int i=0;i<s.row;i++)
         {
            for(int j=0;j<s.column;j++)
            {
                cout<<s.a[i][j]<<"\t";
            }
            cout<<endl;
         }getchar();*/
        }

    s2=m.vconcatenate(s,primey); // Concatenate Y' matrix with new s matrix

    primea1=m.hconcatenate(primea,s2); // Concatenate s2 and A' matrices
    /* Print newly formed A' matrix*/
/*  for(int i=0;i<primea1.row;i++)
     {
         for(int j=0;j<primea1.column;j++)
         {
            cout<<primea1.a[i][j]<<"\t";
         }
         cout<<endl;
```

```
         }getchar();*/
       //cout<<"The above matrix is prime A with attached prime S(s2)"<<endl;
        vector <double>x(primea.row); //output of unknown values
       vector  <double>xdot(primeB.row);//xdot  contains  values  of  the  reactive
elements

        x=m.gauss(primea1,primea.row); //Find intermittent unknown values of node
voltages and Branch currents

       cout<<"#########################"<<endl;
        cout<<"The Results are:"<<endl;
        for(int i=0;i<primea.row;i++)
        {
               cout<<x[i]<<endl;
        }
               w[i]=x[0]; //This array is used to store values of input at various
timesteps
               w1[i]=x[1]; //This  array  is  used  to  store  values  of  output  at
various timesteps.In this particular case node 2 represents output

       cout<<"#########################"<<endl;

       for(int i=0;i<primeB.row;i++)
       {
              xdot[i]=0;
              for(int j=0;j<primeB.column;j++)
              {
                     xdot[i]+= primeB.a[i][j] *x[j]; // xdot= B'*x
              }
       }
       //cout<<"##############################"<<endl;
/*     for(int i=0;i<primeB.row;i++)
       {
                           cout<<xdot[i]<<endl;
       }
       */ //Print xdot
       vector <double>ysvt(svt.row);  //svt tag in vector form
       for(int i=0;i<svt.row;i++) //SVT tag that formed in two dimensional column
matrix is converted into vector form
       {                                                    //for easy manipulation
              for(int j=0;j<svt.column;j++){
                     ysvt[i]=svt.a[i][j];
                     //cout<<"ysvt"<<ysvt[i]<<endl;
                     }
              }

       //cout<<"########################"<<endl;
       vector<double>sv(s.row); //s column in vector form
       for(int i=0;i<s.row;i++) // Convert two dimensional S matrix into a vector
for easy manipulation
       {
              for(int j=0;j<s.column;j++)
              {
                     sv[i]=s.a[i][j];
              }
       }
       /* K1,K2,K3,K4 values are found for adopting Runge-Kutta method*/
```

```
	vector<double>ini(svt.row); //initial values of reactive elements that are
present in s matrix at the index of svt tag are

	for(int i=0;i<svt.row;i++) //stored vector ini of size (svt.row)
		{
			ini[i]=sv[ysvt[i]];

		}

	vector<double> K1(primeB.row);	//K1 = x'* timestep(t)

		vector<double> K12(primeB.row);
	for(int i=0;i<primeB.row;i++)
	{
	  K1[i]=xdot[i]*t;
	}
	/*cout<<"old K1=xdot*t"<<endl;
	for(int i=0;i<primeB.row;i++)
	{

				cout<<K1[i]<<endl;

	}
	*/
	  /*   for(int i=0;i<svt.row;i++)
		{
			xdot[i]=xdot[i]+ini[i];
			cout<<xdot[i]<<endl;
		}*/

	  for(int i=0;i<primeB.row;i++) //K1= initialvalues+K1/2
	  {
		 K12[i]=ini[i]+(K1[i]*0.5);
	  }
	/*
	  cout<<"new K1"<<endl;
		for(int i=0;i<primeB.row;i++)
	{

				cout<<K12[i]<<endl;

	}
	 */
	  // updating S with new state variables
	  s=m.supdate(s,primeB.row,ysvt,K12);
 /*  cout<<"S new matrix with newK1 updated"<<endl;
	  for(int i=0;i<s.row;i++)
		{
			for(int j=0;j<s.column;j++)
			{
				cout<<s.a[i][j]<<"\t";
			}
			cout<<endl;
		}
	  getchar();
		*/

	  s2=m.vconcatenate(s,primey);   // Concatenate s and Y'
```

```
          primea1=m.hconcatenate(primea,s2); // concatenate s2 and A'
           x=m.gauss(primea1,primea.row); // find intermitent unknowns with updated
s matrix

           for(int i=0;i<primeB.row;i++)
       {
              xdot[i]=0;
              for(int j=0;j<primeB.column;j++)  // xdot = B'*x
              {
                     xdot[i]+= primeB.a[i][j] *x[j];
                     //cout<<"xdot"<<xdot[i]<<endl;
              }
       }

       vector<double> K2(primeB.row);  //K2
              vector<double> K22(primeB.row);
       for(int i=0;i<primeB.row;i++) //K2 = xdot *t
       {
         K2[i]=xdot[i]*t;
       }

           for(int i=0;i<primeB.row;i++)
         {
                K22[i]=ini[i]+(K2[i]*0.5); //K2= initial values +K2/2
         }

           s=m.supdate(s,svt.row,ysvt,K22);  // Update  s   matrix  with  newreactive
elements
              //cout<<"new uopdated S with K2"<<endl;
       /*for(int i=0;i<s.row;i++)
              {
                     for(int j=0;j<s.column;j++)
                     {
                            cout<<s.a[i][j]<<"\t";
                     }
                     cout<<endl;
              }*/
              s2=m.vconcatenate(s,primey); // Concatenate s and Y'
               primea1=m.hconcatenate(primea,s2); //Concatenate s and A'
        x=m.gauss(primea1,primea.row); // find intermitent unknown values

           for(int i=0;i<primeB.row;i++)
       {
              xdot[i]=0;
              for(int j=0;j<primeB.column;j++)
              {
                     xdot[i]+= primeB.a[i][j] *x[j]; // xdot= B' * x
                     //cout<<"xdot"<<xdot[i]<<endl;
              }
       }
          vector<double> K3(primeB.row);
              vector<double> K31(primeB.row);
       for(int i=0;i<primeB.row;i++)
       {
         K3[i]=xdot[i]*t;  //K3=xdot*t
       }
           for(int i=0;i<primeB.row;i++)
         {
```

```
                K31[i]=ini[i]+(K3[i]);  //K3= initial values +K3
          }

           s=m.supdate(s,svt.row,ysvt,K31); //Update S matrix with new K3 values
          // cout<<"s updated with K3"<<endl;
          /* for(int i=0;i<s.row;i++)
                {
                      for(int j=0;j<s.column;j++)
                      {
                            cout<<s.a[i][j]<<"\t";
                      }
                      cout<<endl;
                }*/
           s2=m.vconcatenate(s,primey); //Concatenate s and Y'
           primea1=m.hconcatenate(primea,s2);//Concatenate A' and s2
         x=m.gauss(primea1,primea.row); //find intermitent unknown values

         for(int i=0;i<primeB.row;i++)
        {
              xdot[i]=0;
              for(int j=0;j<primeB.column;j++)
              {
                    xdot[i]+= primeB.a[i][j] *x[j]; //xdot=B'*x
                    //cout<<"xdot"<<xdot[i]<<endl;
              }
        }
           vector<double> K4(primeB.row);
         //  cout<<"value of K4:"<<endl;
           for(int i=0;i<primeB.row;i++)
        {
          K4[i]=xdot[i]*t; //K4=xdot*t
          //cout<<K4[i]<<endl;
        }

            vector<double> xn(primeB.row);
              //cout<<"value of xn:"<<endl;
              for(int i=0;i<primeB.row;i++)
              {
                    xn[i]=ini[i]+(K1[i]/6)+(K2[i]/3)+(K3[i]/3)+(K4[i]/6);       //
xn=initial values + K1/6 + K2/3 + K3/3 + K4/6
                    //cout<<xn[i]<<endl;


              }
              s=m.supdate(s,svt.row,ysvt,xn); //Update S matrix with new xn

               //cout<<"##########################"<<endl;
        //     cout<<"s updated with xn"<<endl;
        /*for(int i=0;i<s.row;i++)
               {
                      for(int j=0;j<s.column;j++)
                      {
                            cout<<s.a[i][j]<<"\t";
                      }
                      cout<<endl;
               }*/
              // Repeat the cycle
              if(t==0)
```

```
                {
                t=0.00001;
                }

        } //end of for loop
        }//end of else loop
         //This data.txt file is created to store values of input and output of
circuit with reactive elements and sinusoidal/
        //triangle- wave excitation or square-wave excitation.This file just
created to check results.this can be deleted in main program.
        ofstream outputFile;
        outputFile.open("data.txt");
        double t2=0;
        for(int i=0;i<1000;i++)
        {

                outputFile<<t2<<"\t";
                outputFile<<w[i]<<"\t";
                outputFile<<w1[i]<<"\n";
                t2+=0.00001;

        }
        outputFile.close();
        getchar();
        return 0;
}
```

A.2 Text Files for Example Circuits

Current controlled current sources (cccs.txt)

1 2 R 0 0 0 0.1

3 2 F 1 2 0 100

3 2 R 0 0 0 5000

4 1 R 0 0 0 10000

4 2 P 0 0 0 10

Current controlled voltage sources (ccvs.txt)

1 2 R 0 0 0 0.1

3 2 H 1 0 0 1000

4 1 R 0 0 0 10000

4 2 P 0 0 0 10

3 5 R 0 0 0 5000

2 5 R 0 0 0 5000

Voltage controlled voltage sources (vcvs.txt)

1 2 P 0 0 0 6

2 3 N 0 0 0 3

1 4 R 0 0 0 10000

4 2 R 0 0 0 20000

3 5 R 0 0 0 10000

5 6 R 0 0 0 20000

6 2 E 4 5 0 100000

Voltage controlled current sources (vccs.txt)

1 2 P 0 0 0 10

1 3 R 0 0 0 1000

3 4 R 0 0 0 2000

4 2 R 0 0 0 2000

5 2 G 3 4 0 0.001

5 2 R 0 0 0 200

Sinusoidal excitation

1 2 R 0 0 0 75

5 6 R 0 0 0 0.1

2 6 R 0 0 0 150

6 2 C 0 0 0.000001 -2

2 6 L 0 0 0.00633 0

1 5 Q 5 2000 0 10

Triangle-wave excitation

1 2 R 0 0 0 75

5 6 R 0 0 0 0.1

2 6 R 0 0 0 150

6 2 C 0 0 0.000001 -2

2 6 L 0 0 0.00633 0

1 5 Y 5 2000 0 10

Square-wave excitation

1 2 R 0 0 0 75

5 6 R 0 0 0 0.1

2 6 R 0 0 0 150

6 2 C 0 0 0.000001 -2

2 6 L 0 0 0.00633 0

1 5 Z 5 2000 0 10