The Symbulator Book

A Linear Circuit Simulator for Calculators

The English edition of the 2001 graduation thesis
Symbulator: un simulador de circuitos lineales para calculadoras
Universidad Tecnológica de Panamá

Roberto Pérez-Franco

translated by

Doug Burkett & Tim Hutcheson
(preface, notice, and chapters 1–4, ca. 2001)

Claude (Fable 5)
(chapters 5–12, back matter, and appendices, 2026)

2001 · 2026

Notice

I am the founder of the PAX movement, as well as the author of Symbulator. It is my wish that Symbulator only be used for peaceful purposes, and not for warlike purposes. Please respect this wish. Join Pax Movement. Visit `http://paxm.org`

The jury's verdict

This work was defended before an examining tribunal of the Technological University of Panama on 9 July 2001. Professors Medardo Logreira, Eliane Boulet de Cabrera, and Marcela Paredes de Vásquez each graded it 100/100, with the following general observations: *"It is an excellent work that provides an innovative contribution to the field of numerical and symbolic simulation of electric circuits, and that has received recognition at the world level. This work is a source of pride for the Faculty of Electrical Engineering and for the Technological University of Panama. The members of the jury recommend that this work be promoted within the University community."*

Acknowledgments

To professors Medardo Logreira, Eliane Boulet de Cabrera and Marcela Paredes of the Technological University of Panama, for their unconditional support in the development of this project.

I wish to thank all those who love me: God, for giving me everything; my wife Mónica, for keeping me alive; my mother Eka, for giving me life; my father Tito, for giving me two wide, strong wings and an enormous sky in which to try them; and my sister Ekita, for giving me poetry.

I thank my father for the grammatical review and José Vega for the technical review of this text, Miguel Latorraca for preparing the images of the circuits presented, and my wife for her help preparing the index and the list of figures. I am equally indebted to Doug Burkett and Tim Hutcheson for the good will, the effort and the very many hours of their valuable time that they have dedicated to the careful translation of this thesis into English, under the title *The Symbulator Book*. My eternal gratitude to all the friends and users who encouraged me with their suggestions and comments to develop, improve and test Symbulator during these years: Tim Hutcheson (USA); Lars Frederiksen (Denmark); Joe Riel (USA); Arne Harstad (Norway); Erwin Baert (Belgium); Charles Ware (USA); Doug Burkett (USA); Reinhard Willinski (Germany); Kamil Malinski (Austria); Jake Adams (USA); Daniele Martini (Italy); Rozgonyi Szabolcs (Hungary); Michael Rans (United Kingdom); Alex Astashyn (Russia); Al Charpentier (USA); Nevin McChesney (USA); Ivan Oro Yu (Panama); and Pepe Iborra (Spain).

Preface

The purpose of this book is to teach you the art of symbolic circuit simulation. In particular, I will demonstrate the knowledge, techniques and strategies that you need to take maximum advantage of the Symbulator. Symbulator is a powerful tool for learning circuit theory.

I assume that the reader is already familiar with the basic concepts of circuit theory, such as the laws of Ohm, Cavendish, and Kirchhoff. You should also understand the Thévenin–Helmholtz and Norton–Mayer theorems and Bode plots. Without Symbulator, you should be able to perform direct-current and alternating-current analyses, frequency and transient responses and Laplace transforms.

I also assume that you have a TI-89 or equivalent calculator and the manual for that calculator. You need not be an expert in using the calculator, as detailed examples are provided.

This book should be used with your calculator at hand. I assume that your calculator meets the Symbulator requirements: you must have AMS 2.0x or superior installed, as well as DiffEq version 2.06 or superior.

You will get the best results if you start with the first chapter, read carefully and try all of the examples.

If you have any questions or comments, you can email me at `symbulator@perez-franco.com`

A Note on this Edition

The book in your hands is the completed English edition of Roberto Pérez-Franco's graduation thesis of 2000–2001, the most complete documentation ever written of the classic Symbulator—the twelve-chapter reference its author defended, and its jury graded 100/100, at the Technological University of Panama in July 2001.

The translation was begun soon afterwards by two of Symbulator's own users, Doug Burkett and Tim Hutcheson, who carried the preface, the notice, and chapters 1 through 4 into English before the effort paused—for a quarter of a century. Their work is preserved here essentially as they left it, lightly copy-edited where their working drafts still carried both halves of an unresolved revision. The remaining chapters, the conclusions and recommendations, and the appendices were translated in 2026 by Claude, an AI assistant, in the voice and register the original translators established: the reader addressed directly, the author speaking as *I*, the *doors*, the *(Expert)* and *(Impala)* Modes, and calculator keys shown in brackets, so that [2nd] [ENTER] on the page means those keys under your fingers.

Three editorial notes. First, the figures of the original—TI-89 screen captures and circuit drawings—are not reproduced in this edition; every problem's circuit is fully defined by the circuit description given in its solution, and the original figures survive in the archived thesis. Second, where the translators adapted rather than transliterated (their chapter 1 folds the original's historical appendix into section 1.4, and speaks of "this book" where the thesis said "this thesis"), this edition follows their plan throughout; Appendix A of the original is therefore represented by section 1.4 rather than repeated. Third, the calculator this book teaches is the TI-89 of 2001 running Symbulator Q (version 5); the program's modern descendant, Symbulator 9, runs the same circuit descriptions on Python and SymPy, and its companion volume, *The Internal Logic of Symbulator*, documents how everything described here works inside.

Contents

Chapter 1

General Symbulator Concepts

1.1 What is Symbulator?

Symbulator is a linear circuit simulator which finds symbolic and numeric results. Symbulator can perform direct current (DC) and alternating current (AC) analyses. Symbulator can also find frequency domain response and transient response. It supports a wide variety of linear circuit elements. It occupies less than 60KB of calculator memory and is programmed in BASIC. Symbulator runs on particular Texas Instruments graphing calculators.

1.2 Requirements

Symbulator runs on the Texas Instruments TI-89 or equivalent calculators which have AMS version 2.0x or superior installed. AMS is an acronym for Advanced Mathematics System; this is what Texas Instruments calls the calculator operating system. You must also have the program DiffEq (version 2.06 or superior) installed. Both Symbulator and DiffEq must be archived. In the rest of this book, I refer to a calculator that meets these requirements as *your calculator*.

1.2.1 Installation information resources

If your calculator does not meet these requirements, you must update it before you can use Symbulator. As this book focuses on using Symbulator, and not other aspects of calculator operation, I will not provide upgrade details. The following resources will help you upgrade your calculator.

To upgrade your calculator to AMS 2.05, refer to this Texas Instruments Internet site: `http://education.ti.com`.

The Symbulator and DiffEq programs can be downloaded free from the official Symbulator site shown in section 1.3 below. In order to install these programs on your calculator, you will need both the GraphLink software and a GraphLink cable to connect your calculator to a personal computer. The GraphLink software is free and can also be downloaded from `http://education.ti.com`.

1.2.2 Archive information resources

Symbulator and DiffEq execute much more quickly when they are correctly archived in your calculator's flash memory. Both Symbulator and DiffEq are distributed with programs which automatically perform the archive operation. Instructions to use the archiving programs are provided in the documentation for Symbulator and DiffEq, which is included in the downloaded files.

1.3 Symbulator Internet site

Symbulator has an official web site at `http://paxm.org/symbulator`. This site is in English. In addition to complete documentation and many step-by-step example problems, the most recent versions of Symbulator and DiffEq can be downloaded free.

1.4 Brief history of Symbulator

I am Roberto Pérez-Franco, the author of Symbulator. I am a student of Electromechanical Engineering in the Technological University of Panama, located in Azuero. I started programming Symbulator, then called SCS, on March 30, 1999. The first released version was 0.92, which was posted on the Symbulator web site on April 2, 1999. Version 1 added mutual inductances. Version 2 was released August 19, 1999, at Azuero and added support for two-port circuits. Version 3 incorporated a new Expert system. Version 4 was released on the Symbulator web site on January 9, 2001, and added the Impala enhancement. The most recent version, called Symbulator Q, was released on the Symbulator web site on March 30, 2001. Symbulator Q includes significant improvements which required modifying about 40% of the code.

Symbulator has always been offered for free on the Internet. Feedback from users has allowed me to polish the code, eliminate errors and add novel features.

Now, two years after its public appearance, Symbulator is used by thousands of electrical engineering students and professionals around the world, and has received several honors, including 1st place in Student Projects of the IEEE (Institute of Electrical and Electronics Engineers), for the year 2000 in Latin America (Region 9).

1.4.1 Language

Spanish is my native language. I have several books published in Spanish, and I am considered a promising young Panamanian author. In addition to Spanish, I also speak English and Esperanto. I have also studied French and wish to continue developing my literary skills.

When the Symbulator project began in 1999, TI-89/92+ calculators were practically nonexistent in Panama. Users play a vital role during the development of a program, both in finding bugs and inspiring the author with comments, encouragement and thanks. I have received no monetary compensation for this program; it has always been free.

Since most users of the TI-89 and TI-92+ speak English as a first or second language, it was appropriate to use that language for the program interface and documentation. This made it easy for users on five continents to send comments and suggestions to me. As the capabilities of Symbulator extended in many directions in the last two years, the rapid release of new versions left me little time to keep the documentation up to date. For this reason the documentation and the program interface continued to be written in English.

I offer my apologies to victims of this situation. It has always been my desire to develop a useful tool for all the engineering students of the world. In fact, my interest in Esperanto has its root in the lack of a true neutral, common international language for all people.

Note that I have responded in Spanish to all users who used that language in correspondence with me. Also, this book—which is the most complete documentation of Symbulator—was originally written in Spanish. You are reading the English translation, and I plan to translate this book to Esperanto and other languages as soon as possible.

1.5 Concept of the algorithm

In order to analyze an electrical circuit, Symbulator uses methods that emulate the manual techniques of circuit analysis which are learned by students. In fact, Symbulator operates like an excellent student who has paid attention and learned his lessons well. As the program uses the same methods as a student, it is also a very powerful tool for learning circuit analysis.

Symbulator knows the Ohm–Cavendish law and the Kirchhoff current law. With these two laws, the program analyzes circuits using only nodal analysis. Although Kirchhoff's voltage law is not used, it is always satisfied as a consequence of the other two laws. Mesh analysis is not used. Thanks to the powerful built-in capabilities of the calculator, nodal analysis is sufficient, even for the most complex circuits.

Symbulator also knows the Thévenin–Helmholtz and Norton–Mayer theorems. Symbulator can also graph Bode plots. To perform the necessary Laplace transforms and inverses, Symbulator uses the DiffEq program. There is no faster program for Laplace transforms and inverses for any computer or calculator. DiffEq is the fruit of the brilliant mind of Lars Frederiksen, whom the author is proud to count among his friends.

Symbulator generates the equations of each node and each voltage source, as well as the equations for each special element. At the same time, Symbulator identifies the primary circuit unknowns. Finally, it finds the symbolic or numeric value of all the primary unknowns which solve the equations, and then finds the values of the secondary and tertiary unknowns from the primary unknowns. Thus, it offers all the circuit information a student could wish.

1.6 Doors

The heart of Symbulator consists of several programs called `step` programs, which actually perform the simulation. The user never directly uses these `step` programs, but instead uses other programs that act as *doors*, which in turn use the `step` programs. There are four access doors, one for each basic type of analysis: 1) `dc` is used for direct-current analysis, 2) `ac` is used for alternating-current analysis, 3) `fd` is used for frequency domain analysis, and 4) `tr` is used for transient analysis.

1.7 Tools

Symbulator includes eight more tools that can help with performing simulations or understanding the simulation results: 1) `par` reduces parallel resistances to a single resistance, 2) `absang` shows complex answers in polar form, 3) `plot` graphs a function of time, 4) `bode` draws Bode plots, 5) `thevenin` finds the Thévenin equivalent of a circuit, 6) `port` finds the equivalent parameters for a two-port circuit, 7) `gain` finds the gain of an amplifier, and 8) `fd_to_tr` converts a frequency domain response to the time domain. Throughout this book you will see how these tools are used, as well as the tools' syntax.

For quick reference, Symbulator includes a concise Help function. To display the Help screens, enter `sq\help()` in the entry line. The Help is in English.

1.8 Commands

Sometimes to solve a problem with Symbulator, you must use the built-in calculator commands, such as those which solve equations. You may also need to use auxiliary programs such as the Laplace functions of DiffEq.

1.9 Circuit descriptions

1.9.1 Philosophy

The philosophy of the Symbulator circuit description is the principle that you need only provide the minimum necessary information to completely describe each circuit element.

1.9.2 Text description

Since Symbulator is designed to be used quickly in engineering classes, circuits are described as text strings. A graphical interface is not used, because that would tax the calculator's processor, which is a Motorola 68000 running at 12 MHz. Also, this method would make entering the circuits slow.

A circuit is described as a text string. Each *row* of the string describes one circuit element. Each row almost always has five columns. The first element of each row is the element name. The element name must begin with a letter that identifies the type of circuit element the row describes. The next row elements are the node names, which specify where the circuit element is connected. The node names may be numbers or letters. The next row element is the value of the circuit element in its corresponding unit of measurement. The element value may be a number, a symbolic variable name, or even an algebraic expression. For elements which store energy (inductors and capacitors), their initial conditions are placed in the last row location.

The elements in each row are separated by commas. The rows are separated by semicolons. This example shows a circuit with three circuit elements, and five terms for each element:

```
a1,b1,c1,d1,e1;a2,b2,c2,d2,e2;a3,b3,c3,d3,e3
```

Internally, Symbulator converts this string to a matrix, to work more quickly with the circuit description.

1.9.3 Nodes

Order of the nodes

The order in which the node names are specified in the circuit description is not arbitrary: the order of the node names indicates the direction of current flow and voltage drop in the circuit element. This is particularly important for voltage and current sources, for elements which control dependent sources, for elements which store energy and have initial conditions, and for mutual inductances.

Names of the nodes

Before you can run the simulation, you must identify each circuit node and element with a unique name.

In the case of node names, the name can be a number, letter or a combination of both. Each node has only one name, which means you must use that same name every time the node is used in the circuit description.

If you are not accustomed to using circuit simulators, you may forget the node name once the simulation results are shown. It may help, as you learn to use Symbulator, to write the node names and component names on the circuit schematic.

Reference node

Each circuit always has exactly one reference node. The voltage at the reference node is always 0 volts. This node is known as ground, or earth, and must always be given the node name 0 (zero).

1.9.4 Reserved variables

The calculator operating system reserves some variable names for its own use. These variables are called reserved or system variables. The TI-89/TI-92+ User's Manual lists these reserved variable names. You cannot use reserved variable names as component or node names. In particular, be sure that you do not use these names: `nc`, `ok`, `rc`, `sx`, `tc`, `yc`, `zc`, and `c1` to `c99`.

Symbulator also uses many internal variables to perform the simulation. There may be many of these variables, and they are generated according to the names of the circuit elements, so it is impossible to specify these names. However, they all begin with Greek letters, so it is easy to avoid accidentally using these names: do not use component names or node names that start with Greek letters.

Chapter 2

Numerical simulation of direct current (DC) circuits

2.1 Before starting a simulation

Since Symbulator gives you the simulation results in variables stored in the current folder, it is good practice to delete unnecessary variables in that folder before simulating. This prevents confusing both you and Symbulator with answers of a previous simulation. Although it is not necessary to do so, this practice will ensure better simulation results.

2.1.1 Current folder

The current folder is the folder to which the calculator is currently set. Usually, the current folder is the main folder of the calculator, called MAIN. You can use any folder you wish, but I recommend that you use MAIN unless you have a specific reason to use a different folder.

The current folder is shown on the lower left screen of the calculator in small text.

2.1.2 Unnecessary variables

Unnecessary variables are the variables in the current folder which are not needed to run the simulation. One example of unnecessary variables are the results of a previous simulation.

2.1.3 Procedure

If you are familiar with the use of your calculator, you already know how to delete variables. A complete explanation can be found in the User's Guide. In any case, use these steps to delete the variables:

Identify the current folder. The current folder name is shown in the lower left corner of the calculator screen. Obviously, the calculator must be on. If you do not see the folder name, press [HOME], which displays the Home screen.

The home screen has several sections. The upper section is called the Toolbar. The central section is called the History area. Entries and results are shown here. The line below the history area is called the entry line, because this is where you enter calculator commands. The last section, below the entry line, is called the Status line. The Status line shows the state of several calculator modes, including the current folder. MAIN is shown if the main folder is current.

Start Var-Link. The Var-Link screen shows the folders and variables which you have on your calculator. The Var-Link screen is used to manipulate variables, which includes deleting them. To display the Var-Link screen on the TI-89, press [2nd] [−]; note that VAR-LINK is shown above the [−] key.

Open the current folder. The Var-Link screen first shows the folders on your calculator. At the upper right of the calculator keyboard are four blue arrow keys, called the cursor

keys. Press the blue UP and DOWN cursor keys to highlight the current folder name. If an arrow is shown to the right of the folder name, that folder contains variables. If no arrow is shown, there are no variables in the folder. If there are no variables, you need do nothing more in Var-Link: press HOME to return to the home screen. Otherwise, open the folder by pressing the blue RIGHT key.

Delete the unnecessary variables. To delete the unnecessary variables, first mark them all by using the blue UP and DOWN keys to highlight the variable names, then press F4 to mark them. A small check-mark appears to the left of the variable name. When all the unnecessary variables are marked, press the BACKSPACE key to delete them. The BACKSPACE key is above the T key on the TI-89, and is marked with a left-pointing arrow. A dialog box is shown which asks if you want to delete the variables. Press ENTER to delete them.

To delete all variables from the MAIN folder, select the folder, then press BACKSPACE. All the variables in MAIN will be deleted, but the folder will remain. The MAIN folder cannot be deleted because it is the 'main' calculator folder. An error message is displayed, indicating that the folder cannot be deleted, but the variables are deleted, nonetheless. This method is recommended, because it is fast and effective. However, if you use this method, you must avoid storing variables in MAIN which you want to keep.

2.2 First step for all simulations

The Symbulator offers four different circuit simulation analyses. Each analysis uses a different 'door' and operates with a different logic. Nevertheless, all of them have something in common: they require that the student knows how to correctly describe the elements and nodes of the circuit. Thus, the first step for all simulations is:

Name all the nodes and elements of the circuit.

In Chapter 1 we explained the way to name the nodes. The way to name the elements will be explained in different chapters, as they are used. All these names are unique, and once they are assigned, you must be consistent in using them.

2.3 Door for direct current: dc

As mentioned, the door is the name of a program which calls other programs to actually perform the simulation. In order to run a direct current (DC) circuit simulation, use the door `sq\dc`. The letters `dc` are the name of the door. The letters `sq` tell the calculator in which folder the program `dc` is located. Thus, entering `sq\dc` tells the calculator "execute the program `dc` in folder `sq`".

2.4 Input data: the circuit

When using the `dc` door or any of the other four doors, you must provide the program the circuit you want to simulate, as input data. The circuit description must be placed just after the name of the door, between the parentheses, like this: `sq\dc(circuit)`.

There are two ways to do this. The first way is to put the text which describes the circuit between the parentheses, for example:

```
sq\dc("description of the circuit")
```

The second way is to place a variable name, in which the text has been stored, between the parentheses, for example:

```
"description of the circuit" -> variable
sq\dc(variable)
```

Both methods work well. The second method has the advantage that the circuit is saved in memory as a variable, so you may use it again later, or change it to run a slightly different simulation.

2.5 Answers

2.5.1 Philosophy

The philosophy of the Symbulator is to give you all the simulation results. These results are not displayed automatically, but are stored in variables with intuitive names. The idea behind this is that you need only display the results of interest.

In the case of DC circuit simulation, the results are the voltages, the currents and the powers. Let us see:

2.5.2 Node voltages

The Symbulator calculates node voltages with respect to the reference node, in volts (V). These voltages are stored in variables whose name is `v` plus the name of the node. For example, if the node is named `q`, the name of the variable in which the voltage is stored is `vq`.

2.5.3 Element voltage drops

The Symbulator calculates the voltage drops across all the circuit elements, defined as the voltage drop from the first node in the description with respect to the second node, in volts (V). These voltage drops are stored in variables whose name is `v` plus the name of the element. For example, if the element is called `r5`, the name of the variable in which the voltage drop is stored is `vr5`.

2.5.4 Currents

The Symbulator calculates the currents through all the elements, defined as the current from the first node in the description to the second node, in amperes (A). The currents are stored in variables whose name is `i` plus the name of the element. For example, if the element is called `r5`, the name of the variable in which the current through the element is stored is `ir5`.

2.5.5 Powers

The Symbulator calculates the real power dissipated by all the elements, in watts (W). The powers are stored in variables whose name is `p` plus the name of the element. For example, if the element is called `r5`, the name of the variable in which the real power consumed by the element is stored is `pr5`. Note that the power is defined as if it were dissipated. In the case of an element that generates power instead of consuming it—as is the case of most of the sources—the power in the answer will be negative.

2.6 Short circuit

Ideally, a short circuit is a perfect conductor which transmits any current without voltage drop or power consumption.

2.6.1 Notation

From the point of view of the simulation, the short circuit is the most simple element. Nevertheless, it is the least-used element. We begin with the short circuit, because it is the easiest to describe.

What class of element is it? It is a short circuit. To the Symbulator, we indicate the class of element by means of the first letter of the name that we give the element. The letter that identifies the short circuits is `s`. That is to say, if the first letter of the name of the element is `s`, then Symbulator knows that the element is a short circuit.

How is the element named? Since a circuit can have several elements of the same class, we must give a unique name to each element, so that Symbulator can distinguish the different elements. Therefore, if a circuit has two short circuits, we must give each one a different name. However, both names must begin with `s`, because both represent short circuits. For example, we can name one of them `s1` and another `s2`. Another alternative would be to name one `sa` and another `sb`. Any name is valid which begins with `s` and that is made up of numbers and/or letters. Once the user has defined the name of an element, the Symbulator will use that name for all answers related to that element.

Where is the element connected? A short circuit has two ends, each one of them connected to a different node. You must specify both nodes to which the short circuit is connected. In the Symbulator, the order in which the nodes are listed always indicates the direction of current flow through the element. In the case of a short circuit, the current is defined as flowing through the element from the first node to the second node of the description.

I have mentioned that each circuit element is described as a row of terms in the circuit description. Each row includes the minimum information needed to specify the element and its connections. So, a short circuit is defined as:

```
sname, first node, second node
```

This row condenses, in simple form, the minimum information necessary to describe the short circuit. Note that there are only three elements in the row: a name that begins with `s`, the first node and the second node. There is a fourth specification in the row, which is not evident at first sight: the order in which the nodes appear specifies the direction in which the user wishes to define current flow through the element. Implicit in the description is the order: "define the short circuit current through element `sname` as flowing from the first node to the second node."

2.6.2 Symmetry of the description

Once Symbulator starts, the circuit description in text form is converted into a matrix. Like any other computer, your calculator does not allow asymmetric matrices: all the rows must have the same number of elements (dimension). If one row has n terms, then all rows must have n terms, otherwise an error occurs.

I have mentioned that three terms are required to describe a short circuit. Other circuit elements such as sources require four terms. Still others, such as energy storage elements, require five terms.

As all the rows must have the same number of terms, you need to identify which row has the most terms. The shorter rows must be filled with zeros so that they are the same length as the longest row.

For example, in a circuit description with a longest row of four terms, you need to add a zero to the row which describes a short circuit, so that it has four terms, also:

```
sname, first node, second node, 0
```

As another example, with a circuit description whose longest row has five terms, you must add two zeros to the short circuit description row:

```
sname, first node, second node, 0, 0
```

In the case of other circuit elements, it will sometimes be necessary to add zeros at the end of the rows to maintain the symmetry of the internal matrix.

2.6.3 Related answers

I have said that each node voltage is stored in a variable at the end of the simulation. In addition, each circuit element will generate results related to that element, and they will be stored in variables with intuitive names.

In the case of a short circuit, a single related result is given: the current through the short circuit. Since the voltage drop is zero, no power is dissipated, so there are no other related results.

2.7 Resistors

Ideally, a resistor is an imperfect conductor which resists to greater or lesser degree the flow of current. The resistor causes a voltage drop and therefore dissipates power.

2.7.1 Notation

The information necessary to completely describe a resistor is as follows:

What is the element type? It is a resistor, so it is identified with the letter `r`.

How is the resistor named? In order to distinguish a given resistor from any other, each resistor is given a unique name. The name must begin with `r`.

Where is the resistor connected? A resistor has two terminals, each connected to a different node. You must tell Symbulator the name of the nodes to which the terminals are connected, in the order in which you want the current flow and voltage drop defined. The current is defined as flowing through the resistor from the first node to the second node. The voltage drop is defined as the voltage of the first node with respect to the second node.

What is the resistor value? The value must be given to the Symbulator in ohms. For a DC analysis, the value may be a real number or a variable.

So, resistors are defined as:

```
rname, node 1, node 2, value in ohms
```

Note that the row which describes the resistor has four terms. If another row of the description matrix has five terms, then you must add a zero to the end of the resistor description, for example:

```
rname, node 1, node 2, value, 0
```

The resistor value must be entered in ohms; for example, if the resistance is 1 kΩ, you must enter the value as 1000 ohms, or the results will be wrong.

For simulations other than DC analysis, the resistance value can be a phasor, or a function of frequency or time.

2.7.2 Conductance

A conductance and a resistance are the same element analyzed from two different perspectives. The resistance is analyzed from the point of view of how much it resists current flow, and is measured in ohms. Conductance is analyzed from the point of view of how much it allows current flow, and is measured in siemens or mhos.

If we have a value of resistance and we want to convert it to conductance, we find the reciprocal of the resistance. To find the reciprocal, raise the value to the -1 power or find one divided by the value. For example, a resistance of 5 ohms is equivalent to a conductance of 5^{-1} or $1/5 = 0.2$ siemens.

For the purposes of simulation, conductances must be specified as resistances. The value must be entered in ohms, not siemens. You can perform the conversion directly in the resistor description:

```
rname, node 1, node 2, 1/(value in siemens)
```

or:

```
rname, node 1, node 2, (value in siemens)^-1
```

2.7.3 Related results

In the case of a resistor, three related results are given: the current through the resistance, the voltage drop and the real power dissipated by the resistance. The current is stored in variable `iname`, where `name` is the resistor name. The voltage drop is stored in a variable called `vname`. The dissipated power is stored in a variable called `pname`.

2.8 Current source

An ideal current source is an element which forces a defined current flow through itself, and generates or consumes power.

2.8.1 Notation

The information necessary to define a current source is:

What is the element type? It is a current source. Current sources are defined with the letter `j`.

What is the name of the source? To distinguish a particular current source from other sources in the circuit, it must be given a unique name. The name begins with `j`.

Where is the source connected? The current source has two terminals, each of which is connected to a different circuit node. You must tell Symbulator the nodes to which the current source is connected. The order in which you list the nodes specifies the defined direction of current flow: the current flows from the first node to the second node. In the circuit schematic, the current flow direction is shown by the arrow in the source symbol. The order of the nodes also implicitly defines the sign of the voltage drop across the source: the voltage drop is the voltage of the first node with respect to the second node.

What is the value of the current source? The value of the current source is specified in amperes (A). In a DC analysis, the value of an independent current source is a real number or a variable. The value of a dependent current source is specified as an expression. The expression may be a number and a variable name. The variable is one of the future simulation results. The examples will show this case more clearly.

So, a current source is defined as:

```
jname, node 1, node 2, value in amperes
```

As mentioned, the direction of the current flow is defined as node 1 to node 2. For example, if the value of the current source is 10, then 10 A flow through the source from node 1 to node 2.

Note that the row that defines a current source has four terms. You need to add a zero to the end of the definition if at least one other row has five terms. For example:

```
jname, node 1, node 2, value, 0
```

Make sure that you enter the value in A. For example, if the current source is specified as 10 mA, you must enter a value of 0.01 A. If you enter the value in mA, the results will be wrong.

For simulations other than DC analysis, the current source value may be a phasor, or a function of time or frequency.

The fact that dependent sources are defined with the same notation as independent sources is one of the great advantages of Symbulator compared to other simulators, which require different names and special techniques using control nodes or branches to define dependent sources.

2.8.2 Related results

In the case of a current source, three related results are given: the current through the source, the voltage drop across the source and the real power consumed by the source. The current is stored in a variable called `iname`, where `name` is the current source name. The voltage drop is stored in a variable called `vname`. The consumed real power is stored in a variable called `pname`.

It may seem pointless that the value of a current source is given as a result. It could be said that the value must be known before the simulation, so giving it as a result does not contribute any new information. This is true with numerical simulators. However, it is different in the case of Symbulator, which offers 100% symbolic capabilities, because the value of the current source may be unknown at the start of the simulation: the value may be a variable. For that reason, the current would not be known, which makes it a very important result for the user. That is the reason why Symbulator stores the values of all results in variables.

Depending on the circuit configuration, a current source may consume power. However, most often current sources generate power. I mentioned above that power results are positive for dissipated power. If an element, including a current source, generates power, then the actual generated power is the negative of the value stored in the result variable. It is important to keep this in mind when you interpret the answers.

For example, if the answer in the variable `pname` has a value of -10, this means that the source dissipated -10 watts. This is the same as saying that the source generated 10 watts. As another example, if `pname` has a value of 15, this means that the source dissipated 15 watts, which is the same as saying that the source generated -15 watts.

Problem Nº 001

Now I will solve the first problem. Since this is the first, the procedure will be explained in detail. In following problems, less explanation will be necessary, and it will be omitted gradually.

Problem: Given the following circuit, obtain the voltages of all the nodes.

Solution:

The first step, as always, consists of naming all the circuit nodes and elements. This circuit has three nodes and five elements. This is always the first step of all simulations, regardless of the type of analysis.

One of the three nodes must be specified as the reference node. The usual practice for selecting the reference node is to choose that node which has the most sources connected to it. Actually, any node can be the reference node. For this problem we choose the bottom node as the reference, and we give it the name 0. We name the other two nodes 1 (the left node) and 2 (the right node).

We name the elements as follows. The 2 ohm resistance is `r1`, the 5 ohm resistance is `r2`, the 1 ohm resistance is `r3`, the 3.1 A current source is `j1` and the −1.4 A current source is `j2`. The nodes of each current source must be entered in the description in an order such that they indicate the direction of current flow shown in the drawing. The nodes of the resistors can be entered in any order.

The first step is done.

The second step is to write the circuit description text and to start the simulation. This step could be divided in two steps: first write the description and store it in a variable, then run the simulation with that variable. However, I prefer to perform these two steps as one.

The simulation is started by typing, in the calculator entry line, the command `sq\dc` followed by the circuit description between parentheses. The calculator interprets this entry as: "Execute the program `dc` in folder `sq`, using the circuit description I am giving you between parentheses".

First we ensure that the calculator is ready for a new entry by pressing the [CLEAR] key. Next we type this in the entry line:

```
sq\dc("j1,0,1,3.1; r1,1,0,2; r2,1,2,5; r3,2,0,1; j2,2,0,-1.4")
```

Enter the numbers by pressing the number keys. Enter the \ by pressing [2nd] [2]. Enter the letters by pressing the [ALPHA] key followed by the required letter key. The letter is shown in purple above the key.

Note that the '−' sign in the value of the `j2` current source is the negation symbol, which is entered by pressing [(−)], not [−].

It is also important that you understand what the negative current source value means. In the circuit description, we have specified the flow of current from node 2 to node 0. Since the value is negative, this means that a current of 1.4 A flows from node 0 to node 2.

Since no element in this description has five terms, we need not add zeros to any of the description rows.

When the description is entered, press the [ENTER] key to start the simulation. As the simulation runs, various status messages are displayed to indicate the simulation progress. After a few seconds, the word 'Done' is shown in the history display to indicate that the simulation is complete. The actual time to complete the simulation depends on the particular calculator model and free memory, but it took about 13 seconds on my calculator.

The simulation results are stored in variables in the current folder. You can use Var-Link to display the results. If you have followed the procedure to delete unnecessary variables, you will see this in the Var-Link display:

- The currents through all the elements, in amperes: `ij1`, `ij2`, `ir1`, `ir2`, `ir3`. These currents are defined as flowing from the first node to the second node of each element description.
- The voltage drops for all elements, in volts: `vj1`, `vj2`, `vr1`, `vr2`, `vr3`. The voltage drops are defined as the first described node voltage with respect to the second node.

- The real powers consumed by all the elements, in watts: `pj1`, `pj2`, `pr1`, `pr2`, `pr3`.
- The node voltages with respect to the reference node, in volts: `v0`, `v1` and `v2`.

If the user had not erased the unnecessary variables, they will also be there when the user views the folder using Var-Link.

Return to the entry line by pressing the HOME key, then press CLEAR to clear the entry line.

The problem asked for all the node voltages. To display these voltages, we enter each node voltage variable name in the entry line and press ENTER, and the voltage is shown in the history display.

If we enter `v0` in the entry line and press ENTER, 0 is shown in the history display. This is the expected result, since the reference node voltage must always be zero.

If we enter `v1` in the entry line and press ENTER, 5. is shown, indicating that node 1 is at 5 volts.

If we enter `v2` in the entry line and press ENTER, 2. is shown, indicating that node 2 is at 2 volts.

We have finished the problem, since we have found the values of the requested unknowns.

Problem Nº 002

Problem: Given the following circuit, find the voltages of all the nodes.

Solution:

The only new concept in this circuit compared to the previous problem is that the resistors are specified as conductances instead of resistances. As already explained above, we can enter the conductances directly in the circuit description.

As always, the first step consists of naming all the nodes and elements of the circuit. In this circuit there are four nodes, and we need to choose one of them as the reference node. We chose the bottom node, because it has the most sources connected to it. The circuit has eight elements: three independent current sources and five conductances. We will simulate the conductances as resistors. The node and element names will be shown in the next step.

Note that the actual node and element names are a matter of personal preference.

The second step is to enter the simulation command and circuit description in the calculator entry line, where the circuit description is entered, as always, between the parentheses of the simulation command:

```
sq\dc("j1,0,1,-8; j2,2,1,-3; j3,3,0,-25; r1,1,2,1/3;
      r2,1,3,1/4; r3,2,3,1/2; r4,2,0,1; r5,3,0,1/5")
```

Remember that the negative signs in front of the current source values are entered with the negation key (−), not the subtraction key −.

After entering the circuit description, press ENTER to run the simulation. The calculator screen shows the status messages as the simulation runs, then the word 'Done' is shown in the history display when the simulation is finished. This simulation takes about 20 seconds.

The simulation results are stored in the current folder with the appropriate names. We do not need to display the reference node voltage, because it is zero.

Enter `v1` in the entry line and press ENTER, and the voltage of node 1 is shown: 1.

Enter `v2` in the entry line and press ENTER, and the node 2 voltage is shown: 2.

Enter `v3` in the entry line and press ENTER, and the node 3 voltage is shown: 3.

These are the correct answers. Before trying more problems, we will learn some new concepts.

2.9 Modes of operation

The TI-89, in my opinion, is the best calculator that has ever been made. One excellent aspect of the calculator is the intelligent way in which it handles exact and approximate numbers.

If we press the MODE key, the calculator settings are displayed. The settings are divided in three groups, which can be seen by pressing F1, F2 and F3. Press F2 to see the second group. Near the bottom of the screen you will see the Exact/Approx setting, which is Auto. This setting controls the way in which the calculator handles exact and approximate numbers. Press HOME to return to the home screen, and press CLEAR to clear the entry line. Now we will see how this mode setting affects Symbulator operation.

The Symbulator handles the Exact/Approx setting automatically, setting it to Auto. When your calculator is set to Auto, the number 2 is interpreted as an exact value, while 2. (with a decimal point) is interpreted as an approximate value. The calculator considers as approximate any number with a decimal point. Let us see this in two examples.

In the entry line, type `2/3` and press ENTER. The result is shown in the history area as `2/3`, which is the exact result. The calculator has interpreted the numbers as exact, and does not approximate the result.

Now, type `2./3` and press ENTER. The result shown in the history display is `.666666667`. This answer, which would raise the hair of any reader of the Apocalypse, means that the calculator has interpreted the fraction as composed of an approximate number and an exact number. Therefore, at least one of the numbers is approximate, and the calculator automatically returns an approximate decimal result.

The observant reader will note that the voltages in Problem 001 included a decimal point, while the voltages in Problem 002 did not have the decimal point. Why? The explanation is simple. In the first problem, the element values included approximate numbers with decimal points (for example, 3.1 or -1.4), and therefore the answers were approximate. In the second problem, all the values were exact numbers, so the results were also exact.

This teaches us something: when the circuit values are entered as exact numbers, the results will also be exact. And whenever the values are entered as approximate numbers, the answers will be approximate.

One way to turn an exact number, for example 31/10, into an approximate value, is to use the `approx()` function. So, if we enter `approx(31/10)`, the calculator returns the approximate result, which is 3.1.

We can also turn an approximate number into an exact number with the `exact()` calculator function. For example, `exact(3.1)` returns 31/10.

2.10 Dependent sources

In simple terms, the value of a dependent source depends on another variable in the circuit. The value of a current-dependent current source is a function of a control current, elsewhere in the circuit. On the other hand, the value of a voltage-dependent voltage source is a function of a control voltage, elsewhere in the circuit.

I have said that one of the great advantages of Symbulator is that it allows natural simulation of dependent sources, without additional structures or nomenclature. This results from the method that Symbulator uses to generate equations, as opposed to conventional simulators which use methods such as modified nodal analysis or the gyrator transform. Simulators based on these methods require special nomenclature to distinguish independent sources from dependent sources.

Two precautions are necessary when simulating circuits with dependent sources. The first is to ensure that the control variable is expressed in terms of simulation results. The second precaution is to ensure that the current folder has no unnecessary variables, particularly those whose names are identical to the names of results of the current simulation that will be used to control the dependent source.

2.10.1 Voltage-dependent voltage sources

The control voltage of a voltage-dependent voltage source can be a node voltage, which is the voltage of the node measured with respect to the reference node. For example, the control voltage might be specified as `v1`, where `v1` is the voltage of node 1. The reference voltage is always zero volts.

The control voltage may also be specified as the voltage drop across an element. These voltage drops are included in the Symbulator results, so a control voltage might be specified as `vr1`, which is the voltage drop across `r1`.

The control voltage may also be specified as the voltage between any two nodes. In this case the control voltage is expressed as a differential voltage, for example, `(v1-v2)`.

Let us see a problem as an example.

Problem Nº 003

Problem: For the circuit with two nodes shown in the figure, find the value of the currents i_A, i_B and i_C.

Solution:

As usual, the first step is to name the circuit nodes and elements. The circuit has two nodes. We will use the lower node as the reference. This circuit has a voltage-dependent current source. The control voltage is the voltage drop across the 5.6 A current source, and is labeled v_x. The current through this source flows from the lower node to the upper node. This means that when we describe it, its voltage drop will be defined from the lower node with respect to the upper node, a polarity which agrees with the drop v_x shown in the drawing. If we name this source `jx`, then the control voltage is `vjx`.

The second step is to enter the simulation command and circuit description in the calculator entry line.

The circuit specifies the direction of current flow in the sources. But how shall we define current flow through the resistors? This is not specified, so we must decide.

The problem asks us to find i_A, i_B and i_C, which flow from the top to the bottom of the figure. If we define the resistor currents in the same direction, the simulation results will agree with the figure. This means we will not have to change the signs of the results to answer the problem questions. However, this decision is optional, and either direction may be used. This decision just makes it easier to interpret the simulation results.

In order to make the answers even easier to understand, we will name the elements in the branches of i_A, i_B and i_C as `rA`, `jB` and `rCC`, respectively. This is optional; you may use any names you choose. However, choosing appropriate element names makes the results easier to interpret. Note that we cannot use the variable name `rC`, because that is a reserved system name, as pointed out in section 1.9.4.

This is the command and circuit description:

```
sq\dc("jx,0,1,5.6; ra,1,0,18; jb,1,0,.1*vjx; rcc,1,0,9; j2,1,0,2")
```

The value of the dependent source could also have been described as `-.1*v1`. Press the ENTER key to start the simulation.

After pressing ENTER, the simulation status messages are shown, and soon 'Done' is shown in the history display, indicating that the simulation is complete. This simulation took 12 seconds on my calculator. The simulation results for the element voltages, currents and powers are stored in the current folder.

Enter `ira` in the entry line and press ENTER, then 3. is shown in the history display, which is the current through resistor `rA` in amperes, which corresponds to the current i_A in our problem.

Enter `ijb` in the entry line and press ENTER, then -5.4 is shown in the history display, which is the current through current source `jB` in amperes, corresponding to the current i_B in our problem.

Enter `ircc` in the entry line and press ENTER, then 6. is shown in the history display, which is the current through resistor `rCC` in amperes, which corresponds to the current i_C in our problem.

This example illustrates why it is important that Symbulator can find the currents through current sources: in this problem, the current of source `jB` was unknown and required as an answer. Let us see another example.

Problem Nº 004

Problem: For the circuit with two nodes shown in the figure, find the currents i_1, i_2, i_3 and i_4.

Solution:

As always, the first step is to name the circuit nodes. This circuit has two nodes, or at least that is what the problem statement says. Let us carefully examine the circuit. Clearly, we see that there are six junctions which may be nodes. We also notice that the four outer connections have no resistive elements or current sources: they are short circuits. They have identical voltage at both ends. This means that the six junctions are two groups of three, where each group has the same voltage. When the problem says "the circuit of two nodes", it refers to this fact. In terms of circuit voltages, these six junctions are only two nodes. However, the problem also asks us to find the current through the short circuits. Current can enter a node and also leave it, but it cannot flow *along* the node. So, for the purposes of simulation, we must consider those connections as short circuits, not nodes. This requirement is identical for any simulator, including SPICE and Symbulator. So, this circuit has six nodes and four short circuits, as well as other elements.

We will also use this example to show how to define a voltage-dependent current source whose control voltage is the voltage between two nodes. Which of these six nodes will be the reference node? Any one of them is a possibility, but we prefer to choose as a reference node the one which is between the dependent source and the 2.5 ohm resistor. Why? Because if we use that node as the reference, then the control voltage of the dependent source is simplified, for it can be defined as a single node voltage. Suppose that we name the nodes left to right as follows: 1 is the first node, 0 is the second, 2 is the third, 3 is the fourth, 4 is the fifth and 5 is the sixth node. With these node names, the control voltage of the dependent source is `(v1-v0)`, and since `v0` = 0, the control voltage is `v1`.

The second step is to enter the simulation command and circuit description in the entry line.

How shall we choose the best directions for the current flow? The drawing shows the direction for the current sources. For the resistors, any direction will work. For the short circuits, remember what we learned in the previous problem: it is best to make the current flow direction agree with the directions of the problem questions and answers. So, we define the current flow directions according to the directions of the unknown currents.

In order to answer the questions more easily, we will name the short circuits `s1`, `s2`, `s3`

and `s4`, in the same order as the currents i_1, i_2, i_3 and i_4.

This, then, is the simulation command and the circuit description:

```
sq\dc("r1,1,0,25; j1,0,2,.2*v1; r2,2,3,10; j2,4,3,2.5;
      r3,4,5,100; s1,1,2,0; s2,2,4,0; s3,0,3,0; s4,3,5,0")
```

Two things are worth noticing about the circuit description. First, we needed to place zeros at the end of each short circuit element description, so that all the rows have the same length. Second, note that the control voltage for the source is called `v1`, just as in the drawing. This is because we intentionally assigned the reference node at the appropriate place, and we named the other terminal 1. Alternatively, had we named the two nodes `m` and `n`, the control voltage would have been `(vm-vn)`. Another alternative, just as easy, would have been to define the control voltage as `vr1`, that is, the voltage drop across `r1`, because `r1` is connected to nodes 1 and 0, in the right sequence.

Start the simulation by pressing ENTER after you have entered the circuit description. After the status messages are shown, the history display shows 'Done'. My calculator ran this simulation in 32 seconds. The time has increased over that of previous examples, because there are more nodes. Since Symbulator uses nodal analysis, the number of equations and unknowns increases with the number of nodes. The simulation results are stored in the current folder.

Let us display the currents, with the same procedure we have already seen many times. The current `is1` is $-2.$ amperes, `is2` is $3.$ amperes, `is3` is $-8.$ amperes, and `is4` is $-.5$ amperes.

Now let us look at another type of dependent source.

2.10.2 Current-dependent current sources

For current-dependent current sources, the control current is often the current through an element. For example, if the element is `r1`, then the current is expressed as `ir1`.

In some circuits, the control current is the current through a branch with no elements. In this case, we need to define that branch as a short circuit so we can get the current, just as in the previous example. If the short circuit is called `s1`, then the current will be `is1`.

The following problem shows an example of current-dependent current sources.

Problem Nº 005

Problem: Given the following circuit, determine the value of the voltage v and the power dissipation of the independent current source.

Solution:

The first step is to name the circuit nodes and elements. We use the bottom node as the reference node, so that voltage v is equal to the upper node voltage. If we had used the upper node as the reference, then voltage v would be the negative of the lower node voltage. We name the 2 kΩ resistor `rx`, and define its current as flowing upwards, so that the direction of the current in `rx` matches that of the control current i_x. This makes it easier to interpret the simulation results. The current flow direction through the other 6 kΩ resistor is not important.

The second step is to enter the simulation command and circuit description in the entry line:

```
sq\dc("r1,1,0,6E3; j1,0,1,2*irx; j2,0,1,24E-3; rx,0,1,2E3")
```

Note that the control current is labeled i_X in the drawing, but it is called `irx` in the circuit description. Why did we use a different name? The reason is very simple: Symbulator

cannot know what we mean by i_X, because it cannot see the circuit schematic! All control voltages or currents must be in terms of the circuit elements. Since `rx` is a circuit element, Symbulator knows that `irx` is the current through that resistor.

In the value `6E3`, 'E' indicates the power of 10, so `6E3` means 6×10^3. This nomenclature is used in most programmable calculators, including Texas Instruments, HP and Casio.

Note that even though the circuit resistances are given in kΩ, the circuit description values must be in ohms. Currents in the circuit are specified in mA, but we must specify the values in A for the circuit description. Symbulator does not understand unit prefixes such as k or m.

You must keep in mind that we must describe the circuit in terms that Symbulator understands. This avoids absurd situations such as defining the control current as i_X or specifying values in mA, which would result in the wrong answers.

Start the simulation by pressing ENTER. 'Done' is shown in the history display after the simulation status messages are shown. My calculator took 10 seconds to simulate this circuit. The simulation results are stored in the current folder. This example clearly demonstrates the agility and potential of Symbulator as a tool for learning and checking answers. It would take several minutes to solve this circuit by hand and find all of the voltages, currents and power dissipations, and there is a chance that we may make a mistake. With Symbulator, obtaining all these results takes only ten seconds, and we can be confident that the results are correct.

The answers are shown with the same process used in the previous examples. Voltage v in the schematic is equivalent to `v1`, which is 14.4 V. The power dissipation of the independent source is called `pj2`, which is −.3456 W. Since the problem asked for the power dissipated by the source, `pj2` is the answer. If the problem had asked for the power generated by the source, we would change the sign of the answer, and the generated power would be .3456 W (positive). Let us do another example, to reinforce the concepts learned so far.

Problem Nº 006

Problem: Determine i_X in the following circuit.

Solution:

As always, the first step is to name all the circuit nodes and elements. Like Problem 004, this circuit seems to have two nodes, when in fact we need more nodes and some short circuits to simulate it. For this problem, the location of control current i_1 forces us to view the upper part of the circuit as two nodes connected by a short circuit, which we will call `s1`. On the other hand, the location of the unknown current i_X forces us to view the lower part of the circuit as two more nodes, and to use another short circuit which we will call `sxx`. We cannot use the name `sx`, because that is a reserved system variable name.

With this circuit, it is irrelevant which node we decide to call the reference node. The current flow direction in the resistors can also be assigned in either direction. The current flow in the current sources must, as always, be assigned in the direction shown in the drawing. The current flow directions in the short circuits are chosen so that they agree with the control current and the unknown current.

The second step is to enter the simulation command and circuit description in the entry line:

```
sq\dc("r1,1,0,5E3; j1,1,0,4E-3; s1,1,2,0; sxx,3,0,0;
      j2,2,3,3*is1; r2,2,3,20E3")
```

It should not be a surprise at this point that the control current is called i_1 in the drawing, but is called `is1` in the description.

Start the simulation by pressing ENTER. 'Done' is shown in the history display when the simulation is finished. My calculator took 17.5 seconds to simulate this circuit. The results are stored in the current folder.

The answer to the problem, the unknown current i_X, is obtained by recalling `isxx`, which is .000571429 amperes.

2.11 Voltage source

Ideally, a voltage source is a circuit element which forces a defined voltage difference between its two terminals, and generates or dissipates power.

2.11.1 Notation

The minimum information needed to define a voltage source is:

What kind of element is it? It is a voltage source. The letter that identifies voltage sources is `e`.

How is the source named? To distinguish one voltage source from another, we must give each one a different name, and those names all start with `e`.

Where is the source connected? A voltage source has two ends, each connected to a different node. It is necessary to specify both nodes to which the source is connected, and the order in which we specify the nodes also specifies the polarity of the source: the first node with respect to the second. In the circuit drawing, the polarity of the source is specified by the 'plus' and 'minus' signs next to the source.

When entering a voltage source in the circuit description, the order in which the nodes are listed defines the source polarity. However, the polarity also indirectly specifies the direction of current flow through the source, from the first node to the second node. The voltage drop across the voltage source is measured from the first node with respect to the second node.

This point may cause some confusion if you are not familiar with circuit simulation. You may think that, if the first node of the voltage source is positive with respect to the second node, then the current must flow from the second node to the first. Indeed, it is like that, in most cases.

The important point to understand is that the polarity used to define a variable does not necessarily establish which node is at the higher voltage. As an example, consider the following: "The voltage of node A is -5 V with respect to node B". In this example, the polarity is defined at node A with respect to node B, but node B is at a higher voltage than node A.

Similarly, defining the direction of current flow through an element does not set the actual direction of current flow. Consider this phrase: "the current flowing from node A to node B is -3 A". In this case, the current is defined as flowing from A to B, but the real direction of conventional current flow is from B to A.

Summarizing: in all the elements we have so far used, the current flow direction used to define the simulation results is from the first node to the second node, and the polarity of the voltage drop across an element is defined as the voltage of the first node with respect to the second node. In addition, the voltage source value is given as the voltage of the first node with respect to the second.

What is the value of the source? The value must be specified in volts. For a direct current simulation, the value of an independent voltage source can be a real number or a variable. If the source is dependent, its value must be an algebraic expression. This expression will be composed of a number and a variable. This variable will be one of the simulation answers.

So, in a circuit description, a voltage source is defined as:

```
ename, node 1, node 2, value in volts
```

As we have seen, the value of the voltage source is defined as the voltage at node 1 with respect to node 2. For example, if the value is 10 V, then the simulator assumes that the voltage difference from node 1 with respect to node 2 is 10 volts.

Note that the voltage source description has four terms. If other circuit element descriptions have five terms, then we must add a zero to the source description, for example:

```
ename, node 1, node 2, value, 0
```

Take care that all values are entered in volts, and not some other unit such as millivolts (mV), otherwise the results will be wrong.

For analyses other than direct current, the value may be a phasor, a function of frequency or a function of time.

2.11.2 Related answers

There are three simulation results for a voltage source: the current through the source, the voltage drop across the source, and the real power consumed by the source. The current is saved in a variable called `iname`, where `name` is the name of the source. The voltage drop is stored in a variable called `vname`. The real consumed power is stored in a variable called `pname`.

The current, as previously explained in detail, is defined as flowing from the first node to the second node in the description.

The voltage drop is defined as the voltage of the first node with respect to the second node in the description.

The power is returned as the consumed power. If the source generates power, the returned result is negative. You must remember this when you interpret the simulation results.

Problem Nº 007

Problem: In the following circuit, determine the power consumed by each of the elements.
Solution:

The first step is to name all the nodes and elements. We will use the bottom node as the reference node. Remember that voltage source names must begin with `e` and the order of the nodes must correspond to the polarity shown in the drawing. The order of the nodes of the resistors is irrelevant.

The second step is to enter the simulation command and circuit description in the entry line:

```
sq\dc("e1,1,0,120; r1,1,2,30; e2,2,3,30; r2,3,0,15")
```

Start the simulation by pressing ENTER. After the status messages are shown, 'Done' is shown in the history display. My calculator took 13 seconds to simulate this circuit. The answers are stored in the current folder.

The power consumed by source `e1` is in the variable `pe1`, and is −240 watts. This means that the source generated 240 W, since the value is negative. However, since the problem asked for the power consumption of each element, the answer is −240 W. The power consumed by source `e2` is in variable `pe2`, and is 60 W. Since the value is positive, the source consumes power; it does not generate it. The power dissipated by `r1` is in variable `pr1`, and is 120 W. The power dissipated by `r2`, from variable `pr2`, is 60 W. With this example, we can verify that Symbulator complies with the energy conservation law. If we enter `pe1+pr1+pe2+pr2`, to find the net power dissipation, the result is 0 W, as expected.

2.11.3 Dependent sources

All the concepts that we learned for the simulation of dependent current sources apply to dependent voltage sources as well. Let us see some examples for demonstration. We will start with a simple problem with a single voltage-dependent voltage source.

Problem Nº 008

Problem: In the following circuit, determine the power absorbed by each element.

Solution:

The first step is to name all the nodes and elements. The order of the nodes in the voltage sources is determined by the polarity shown in the drawing. With the experience we now have, it is obvious that if we use the bottom node as the reference node, and call the upper left node `x`, then we can call the control voltage `vx`. The other nodes are named 1, 2 and 3, moving clockwise around the drawing.

The second step is to enter the simulation command and circuit description in the entry line:

```
sq\dc("r1,x,0,30; e1,1,x,12; r2,1,2,8; r3,2,3,7; e2,3,0,4*vx")
```

Start the simulation by pressing ENTER. After the status messages are shown, the word 'Done' is shown in the history display. My calculator took 18 seconds to simulate this circuit. The results are stored in the current folder.

If we enter `pr1` in the entry line and press ENTER, the power dissipated by resistor `r1` is shown as 96/125 W. As explained, the simulation results are exact, because all of the circuit values are exact. Usually, textbook answers are not given as exact values like 96/125 W, but as approximate values like 0.768 W. The exact value is as correct as the approximate value; nevertheless, in an examination the professor may prefer the approximate value. Note that the terms 'exact' and 'approximate' are used in the same sense as defined by the operation of programmable calculators.

It is simple to obtain the approximate answer instead of the exact answer: just press ◇ before pressing ENTER.

With `pr1` in the entry line, if we press ◇ ENTER, then the approximate result of .768 W is shown. This same procedure is used for the other variables.

Enter `pe1`, then press ◇ ENTER and the result of 1.92 W is shown.

Enter `pr2`, then press ◇ ENTER and the result of .2048 W is shown.

Enter `pr3`, then press ◇ ENTER and the result of .1792 W is shown.

Enter `pe2`, then press ◇ ENTER and the result of −3.072 W is shown.

Next, we will do an example with two voltage-dependent voltage sources, with more complicated control voltages.

Problem Nº 009

Problem: Find the power absorbed by each element of this circuit.

Solution:

The first step is to name all the circuit nodes and elements. The order of the nodes in the circuit descriptions for the three voltage sources is determined by the polarities shown in the drawing. Resistor polarities are irrelevant. There is no clear choice for the reference node, so we choose the bottom left node as the reference, and name the other nodes in clockwise order as `a`, `b`, `c`, `d` and `e`.

The second step is to enter the simulation command and the circuit description:

```
sq\dc("e1,a,0,40; r1,a,b,5; r2,b,c,25; r3,c,d,20;
      e2,e,d,2*vr3+vr2; e3,e,0,4*vr1-vr2")
```

Start the simulation by pressing ENTER. After the status messages are shown, 'Done' is shown in the history display. My calculator took 26 seconds to simulate this circuit. The answers are stored in the current folder.

For `pe1`, we obtain 80 watts.

For `pr1`, we obtain 20 watts.

For `pr2`, we obtain 100 watts.

For `pr3`, we obtain 80 watts.

For `pe2`, we obtain −260 watts.

For `pe3`, we obtain −20 watts.

These are the correct answers. Now let us see an example of a dependent voltage source controlled by a circuit current.

Problem Nº 010

Problem: Find the current i_X in this circuit.

Solution:

The first step is to name the circuit nodes and elements. The order of the nodes in the three sources is determined by the drawing. The current flow direction through the 2 ohm resistor (which will be named `rx`) must agree with that of i_X; the current flow direction through the other resistor is irrelevant. We choose the bottom node as the reference node, and name the other nodes 1, 2 and 3, moving clockwise around the circuit.

The second step is to enter the simulation command and the circuit description:

```
sq\dc("e1,1,0,10; rx,1,2,2; j1,0,2,3; r1,2,3,1; e2,3,0,2*irx")
```

Start the simulation by pressing ENTER. After the status messages are shown, 'Done' appears in the history display. My calculator took 16 seconds to simulate this circuit. The answers are stored in the current folder.

Enter `irx`, then press ◇ ENTER to obtain the current of 1.4 A through `rx`, which is correct. We will finish this chapter with a problem that combines everything we have learned so far.

Problem Nº 011

Problem: Find the currents, voltage drops and power dissipation for each element in this circuit. Show that the sum of the powers is zero.

Solution:

The first step is to name all the circuit nodes and elements. By this point, you should be able to do this without additional explanation. The second step is to enter the simulation command and the circuit description:

```
sq\dc("e1,1,0,60; r1,1,0,20; e2,1,2,5*ir1; j1,2,0,ve1/4; r2,2,0,5")
```

Start the simulation by pressing ENTER. 'Done' is shown in the history display after the status messages are shown. My calculator took 15 seconds to simulate this circuit. The answers are stored in the current folder. The results are: `ve1` and `vr1` are both 60 volts, `ve2` is 15 volts, `vj1` and `vr2` are both 45 volts, `ie1` is −27 amperes, `ir1` is 3 amperes, `ie2` is 24 amperes, `ij1` is 15 amperes, `ir2` is 9 amperes, `pe1` is −1620 watts, `pr1` is 180 watts, `pe2` is 360 watts, `pj1` is 675 watts, `pr2` is 405 watts. Note that only one source is generating power. How long would it take a student to find, without errors, these 15 answers? It took the Symbulator, on average, 1 second for each one.

2.12 Correcting a wrong description

Suppose that we make an error in entering the circuit description, but we do not notice it until the simulation is complete. We need to correct the error and run the simulation again. First, we must erase the result variables in the current folder. Next, we must put the simulation command and the circuit description in the entry line, again. It is a tedious task to completely re-enter the circuit description a second time. The skilled user will take advantage of the fact that the circuit description is already written. There are two ways to do this:

1) If the previous description is still in the entry line, it will be shown as white characters on a dark background. Move the cursor to the error with the blue cursor keys and correct it. Note that the BACKSPACE key is used to erase a character to the left of the cursor.

2) If, for some reason, the circuit description is not in the entry line, move the cursor upwards until the circuit description in the history area is highlighted. Press ENTER to make a copy in the entry line. The error in this copy is corrected as described above.

Once the error is corrected, run the simulation again by pressing ENTER. With this explanation, we have finished the chapter on numerical direct current simulation.

Chapter 3

Direct current symbolic simulation

3.1 Definition of numerical simulation

In the previous chapter we did several simulations, in which all of the circuit element values were numerical. In addition, the simulation results were also numerical. This type of simulation is known as numerical simulation, and it is possible with any conventional numeric simulator, such as Electronic Workbench, PSpice and others. In these simulators it is possible to describe circuits with dependent sources and find results for all nodes and elements, but it is easier in Symbulator.

One reason that this task is easier in Symbulator, as compared to the other simulators, is that Symbulator is a symbolic simulator, not a numeric simulator. For example, dependent sources are treated symbolically during the simulation, even though they may have numerical values in the end. Another reason that Symbulator is easier to use is that it was designed and developed by a student, with the clear intent to provide other students with what they want to receive as the answers of a simulation.

The full potential of Symbulator only becomes truly evident when we run symbolic simulations.

3.2 Definition of symbolic simulation

A symbolic simulation is one in which at least one of the circuit element values is not a number, be it real or complex, but instead a symbolic value. This symbolic value may be an unknown variable, an algebraic expression, a function of time or frequency, or some other type of non-numerical expression.

We learned that the type of element entry will determine the type of the results. The nature of the circuit element values is reflected in the simulation answers. Exact values will produce exact answers. Numerical values will produce numerical answers. So, symbolic values will result in symbolic answers. We can be sure that a symbolic simulation, having at least one symbolic value in the input circuit, will have some symbolic answers.

3.3 Symbolic simulation to obtain symbolic answers

A simulation is considered symbolic if the circuit description contains at least one element with a symbolic value. There are two basic types of symbolic simulation: the first results in a symbolic answer, and the second results in a numerical answer. The first type of simulation, whose objective is a symbolic answer, is very useful in design problems, and for getting expressions which symbolically describe some circuit characteristic in terms of element values, such as consumed power. It can also be used for academic purposes, to demonstrate the laws of circuit analysis. Ohm's law, for example, can be demonstrated by a Symbulator simulation. We shall try some problems in order to learn to simulate symbolically.

Problem Nº 012

Problem: Demonstrate Ohm's law and the power dissipation formula, with the following basic circuit and a symbolic simulation.

Solution:

This is the simplest possible circuit. The first step is to name the circuit nodes and elements. The second step is to run the simulation with the circuit description:

```
sq\dc("r,1,0,r;j,0,1,i")
```

Note the way in which the elements have been named: since there is only one element of each type, we have used only one letter, which is the first letter of the name of the element. We start the simulation by pressing ENTER, then the status messages are displayed, then 'Done' is shown in the display. My calculator ran this simulation in 6 seconds. Since this is the simplest possible circuit, this is the shortest Symbulator simulation time. The results are saved in the current folder.

We can demonstrate Ohm's law by displaying the value of `vr`, which is, as expected, `i*r` volts. To demonstrate the power formula, we display the value of `pr`, which is `i^2*r` watts, which is correct.

We could also have verified Ohm's law by using a voltage source instead of a current source, with this circuit description:

```
sq\dc("r,1,0,r;e,1,0,v")
```

My calculator took 7 seconds to simulate this circuit. Note that a voltage source takes more time than a current source. Although the difference in this case is minor, only 1 second, the difference can be considerable in more complicated circuits.

We demonstrate Ohm's law by displaying `ir`, which is `v/r` amperes, which is another way to express the law. We demonstrate the power formula by displaying `pr`, which is `v^2/r` watts and equivalent to the power expressed in terms of the current.

This exercise teaches us that symbolic simulation is an easy way to demonstrate and learn the basic laws of circuit analysis.

3.4 Symbolic simulation to obtain numerical answers

In some symbolic simulations, our objective is a numerical answer. In this case, we use the `solve` command, the `solves` tool and the (Expert) Mode.

Usually, to obtain a numerical answer, we need to know in advance the numerical value of an answer—a voltage, current or power—for each symbolic value in the input circuit.

3.5 The solve command

Let us see a simple example of the second type of symbolic simulation, which obtains a numerical answer, using the `solve` command.

Problem Nº 013

Problem: Determine the numerical values of v_x and i_x in the following circuit.

Solution:

When we compare this circuit with previous examples, we see two obvious differences. First, some element values are numeric and some are symbolic. Second, we are given one of the branch currents: 12 A through the 5 ohm resistor.

We first name the nodes and elements. We define the current directions through the 5 and 6 ohm resistors as shown in the drawing. For the symbolic values, we use the same names as shown in the drawing. We must make sure that these variables do not already exist in the current folder; that is, variable names `ra` and `vx` must not exist, or the calculator will replace the symbolic name with its stored value, and the simulation will be incorrect. We use the `DelVar` command of the calculator to delete the variables, like this:

```
DelVar ra,vx
```

With this command, we tell the calculator: "Delete the variables `ra` and `vx` in the current folder". There is no effect if the variables do not actually exist. We could have deleted the variables in the Var-Link window, but the method shown is faster.

Once the variables are deleted, we enter the circuit description and run the simulation:

```
sq\dc("e1,1,0,18;ra,1,0,ra;r1,1,0,6;r2,2,1,5;ex,2,0,vx")
```

We start the simulation by pressing ENTER, then the status messages are shown, then 'Done' is shown in the display. My calculator took 16 seconds to simulate this circuit. The answers are stored in the current folder.

We find i_x by displaying the value of `ir1`, which is 3 A. So far, there is nothing new about this problem.

The difference appears when we find v_x. First, it seems that we could find v_x by displaying the value of `v2`, but `v2` is simply `vx` volts. Obviously, this answer does not tell us anything new, because we require a numerical answer. Then, we remember that the circuit drawing specifies that the current through the 5 ohm resistor is 12 A. What is this current, according to the Symbulator results? The recalled value of `ir2` is `(vx-18)/5` amperes. This symbolic expression, the result of the symbolic simulation, will help us find the value of v_x. How? Let us see.

We know that the current is 12 A, and we have its expression as a function of `vx`. We can combine these expressions in an equation. When we solve the equation for the unknown `vx`, we will find the numerical value of the voltage source. We could solve this equation manually, but we will take advantage of the calculator technology, and use the calculator's `solve` command. A detailed explanation of this command can be found in the User's Manual for the calculator. This is a very powerful command, which is used extensively by Symbulator in its internal calculations. Now, we will use `solve` manually, by entering this in the entry line:

```
solve(ir2=12,vx)
```

With this command, we tell the calculator: "Solve the equation `ir2` = 12 for the unknown `vx` and tell me the answer". In other words, what is the value of `vx` if `ir2` is 12? We press ENTER, and almost instantly we obtain the result: `vx` = 78 volts. That easy! The unknown we solved is the answer we wanted.

Having finished this problem, consider these three observations.

The `solve` command told us that `vx` = 78, but it does not store the value 78 in the variable `vx`. The `solve` command solves equations for unknowns, and shows the results in the display, but it does not store these results in the unknown variables. To verify this, recall the value of `vx`, and `vx` is shown. This means that no value is stored in variable `vx`.

In this problem, the symbolic unknown in the circuit was also one of the results the problem required. In other problems, this may not be the case. We will see an example of this situation next.

The value of `ra` is indefinite. There is no mathematical way we can obtain a numerical value for it. The reason is simple: to obtain a numerical value for a symbolic element, we must know the numerical value of an answer which involves the symbolic value, such as a current, voltage or power. For example, we could find the value of `vx` because we knew the numeric value of a current which is a function of it. But we cannot find the value of `ra`, because we are missing that clue: we do not know the numeric value of any current, voltage or power that involves it.

I recommend that you review this problem until you feel you fully understand the concepts it illustrates. Only then should you continue.

3.6 The solves tool

The `solves` tool is part of the Symbulator. It is programmed using technology developed for the Symbulator, and it has two purposes: 1) to solve equations using the built-in `solve` command, and 2) to store all the results in the appropriate variables. This tool is executed with no input arguments between the parentheses:

```
sq\solves()
```

When we press ENTER, a dialog box is shown with two fields and two options. The first field is for entering the equation or equations we want to solve. The second field is for entering the unknowns for which we want to solve. The two following options are for choosing whether the solutions are real or complex, and whether we want to use conditional equations, that is, known values. Press ESC to return to the home screen.

Next we will solve a problem, similar to the previous one, which shows the basic use of the `solves` tool.

Problem Nº 014

Problem: Determine the numerical values of v_x and i_x in the following circuit.

Solution:

Note that in this circuit, the symbolic unknown is `rx`, while the required answers are i_x and v_x. These will likely be functions of the unknown. For that reason, the strategy we must follow is to simulate the circuit, solve for the unknown value of the element, then obtain numerical values for the required answers.

First we name the nodes and elements. We define the currents through the 1 and 2 ohm resistors as shown in the drawing. For the value of the symbolic element, we use the same variable name as in the drawing. To ensure that a variable of the same name does not already exist, we use:

```
DelVar rx
```

For the reference node, we use the lower left corner. The circuit description and simulation command is:

```
sq\dc("j1,0,1,6;r1,1,2,1;r2,1,3,5;j2,3,2,10;r3,3,0,2;
      rx,2,4,rx;r4,4,0,3")
```

We start the simulation with ENTER. 'Done' is displayed after the status messages are shown. My calculator took 30 seconds to run this simulation. By this time, you have noticed that the simulation time is proportional to the number of circuit nodes. The reason for this is that Symbulator "thinks nodally", and will create an equation for each circuit node except

the reference. So, this circuit with four nodes other than the reference takes approximately four times as long as a circuit with one node other than the reference. Obviously, there are other factors that affect the simulation time, such as the type of analysis, the presence of voltage sources and special elements, and symbolic values.

Returning to the problem, the simulation results are stored in the current folder. We find i_x by asking for the value of `ir1`. We do not get a numerical value, but instead a function of `rx`. The same happens for v_x: when we ask for `vrx`, we obtain a function of `rx`. As we expected, the numerical answers we need are now functions of the unknown. In order to find numerical values for i_x and v_x, we must first find the numerical value of `rx`. To allow us to find this value, the problem has told us in advance that the current through the 2 ohm resistor (which we have called `r3`) is 4 A.

The `solve` command is not the most appropriate for this task. While we have seen that `solve` is extremely fast and effective in solving equations, and could no doubt solve this small equation in a thousandth of a second, `solve` would only display the value of `rx` in the history screen, and not store the numerical value. For that reason, `solves` is the more appropriate tool.

To illustrate this point, first we will do the procedure with the `solve` command. The first step is to solve the equation:

```
solve(ir3=4,rx)
```

We obtain the answer `rx` = 40. Since we will evaluate other expressions which are functions of this unknown, we must manually store this answer in the unknown variable, using the calculator `Define` command, like this:

```
Define rx = 40
```

Or, we could use the STO key, which is equivalent to `Define`:

```
40 [STO] rx
```

Once the value of `rx` is stored, we can ask the calculator for the numerical values of i_x and v_x. For `ir1` we obtain −8 A, and for `vrx` we obtain 80 V.

We get the answers with this method, but it takes four steps: 1) simulate the circuit, 2) solve for the unknown, 3) store the unknown, and 4) evaluate the functions of the unknown.

With the `solves` tool, we find the answers in three steps: 1) simulate the circuit, 2) solve and store the unknown in a single step, and 3) evaluate the functions of the unknown to find the answers. To demonstrate this, we first delete the variable `rx`:

```
DelVar rx
```

The simulation results are still stored in memory, so we will not simulate the circuit again. Execute the `solves` tool:

```
sq\solves()
```

The dialog box is shown. In the first field, labeled “Equations”, we enter the equation:

```
ir3=4
```

In the second field, labeled “Unknowns”, we enter the unknown:

```
rx
```

Since the remaining options are real solutions and no conditionals, we leave them unchanged. We press ENTER, twice if necessary. A text message is shown to indicate that the tool is solving the equations and then storing the answers. Next, the answer is shown: `rx` = 40. That is all. We press ENTER to return to the home screen. The value 40 is now stored in the variable `rx`. If you want to verify this, recall the value of `rx`, and 40 is displayed.

The third step is to simply evaluate the required problem answers, which are the numerical values of i_x and v_x. For `ir1` we obtain −8 A, and for `vrx` we obtain 80 V.

There is an even faster way to find these results, using the (Expert) Mode. Since this method requires much more knowledge and skill, we will dedicate the following chapter to it. Next we will see another problem, similar to the last, to strengthen the concepts we have learned.

Problem Nº 015

Problem: Determine the numerical values of v_x and i_x in the following circuit.

Solution:

Note that the unknown symbolic current `ix` in this circuit is also one of the required answers. The other required answer is v_x, which will be a function of `ix`. We will simulate the circuit, solve for the unknown, and then get the answers as numerical values.

We name the nodes and elements, then define the direction of the current through the 8 ohm resistor as shown in the drawing. We also delete variable `ix`:

```
DelVar ix
```

Using the bottom node as the reference node, we enter the circuit description and run the simulation:

```
sq\dc("e1,1,0,60;r1,1,2,8;r2,2,0,10;r3,2,3,4;r4,3,0,2;jx,0,3,ix")
```

We start the simulation by pressing ENTER. After the status messages are shown, the display shows 'Done'. My calculator took 19 seconds to run the simulation. All the symbolic results are in the current folder. Since we know that the answers are symbolic, we must solve for the unknown `ix`. Since one of the answers is a function of `ix`, the best method is to use the `solves` tool:

```
sq\solves()
```

The input form is shown. If we have not deleted the variables in the current folder after the last time we used `solves`, some previous entries are shown in the input form. In the "Equations" field we enter the new equation, which replaces any previous equation:

```
ir1=5
```

In the second field, "Unknowns", we enter the unknown:

```
ix
```

We leave the remaining options unchanged, since they are as we want them. We press ENTER, twice if necessary. The answer `ix` = 1 is shown. We press ENTER to return to the home screen, then we evaluate the remaining unknown: for `v3` we get 8 volts. Let us see yet another example of symbolic simulation with numerical results.

Problem Nº 016

Problem: Find the value of k such that voltage v_y is zero.

Solution:

We label the nodes and elements. We label the center nodes `x` and `y`, for obvious reasons. Delete variable `k`:

```
DelVar k
```

Using the bottom node as the reference, we enter the circuit description and run the simulation:

```
sq\dc("e1,1,0,6;r1,1,x,1;r2,x,0,4;r3,x,y,2;j1,0,y,2;
      r4,y,2,3;e2,2,0,k*vx")
```

We start the simulation by pressing ENTER. After the status messages are shown, 'Done' is shown. My calculator took 36 seconds to run the simulation. All results are in the current folder. Since the only unknown is also the required answer, we can use the built-in `solve` command:

```
solve(vy=0,k)
```

We obtain the answer $k = -13/4$, which is correct. But how can we be sure of this, for example, in the middle of a semester examination?

3.7 Verifying a symbolic simulation with a numerical answer

There is an easy way to verify that the numerical answers to a symbolic simulation are correct. All we need to do is get the circuit description from the history area, instead of writing it again, and replace all the symbolic values with the numeric values that were found. To get the description from the history area, use the same method described to correct errors, shown at the end of the previous chapter.

After executing the simulation with numeric values, we evaluate the previously known results, such as the voltages and currents. If we get the same values that we knew before, then the symbolic simulation results are correct.

As an example, we can verify the value for k in the previous problem with this numerical simulation:

```
sq\dc("e1,1,0,6;r1,1,x,1;r2,x,0,4;r3,x,y,2;j1,0,y,2;
      r4,y,2,3;e2,2,0,-13/4*vx")
```

Notice that only the value of source `e2` has changed, which now has a numeric value instead of the symbolic value `k`. Recall that we were asked to find k such that v_y is zero. Therefore, if the value found for k, $-13/4$, is correct, then the voltage at node `y` should be zero. Ask for `vy` and you will find it is 0 V. The answer has been verified.

Being able to verify answers is of special importance for a student during an important examination. Many users recognize that Symbulator is an excellent tool to verify answers during examinations. This is probably one of its most significant strengths.

Now we will learn to simulate like experts with Symbulator.

Chapter 4

Expert simulation in direct current

4.1 Introduction to the (Expert) Mode

If you are not interested in learning to simulate like an expert, you can skip this chapter. In this chapter, I describe advanced concepts of Symbulator operation, as well as examples that show how you can use the (Expert) Mode to save time during symbolic simulations. To get the most benefit, you must read this chapter attentively and thoroughly.

In the previous chapter we did several symbolic simulations, and we learned to use the `solve` command and the `solves` tool. In this chapter, we will learn to simulate with the (Expert) Mode. This mode appeared in version 3 of the Symbulator, in order to give the user more control over the simulation process.

In the simulations we have seen so far, the Symbulator behaves like a black box, in which you supply the circuit description and Symbulator gives you the results. And that is all: while you are involved in complimenting the girl sitting next to you, or contemplating the immortality of the crab, Symbulator is working to generate the equations, solve them and save the answers. Your only responsibility is to correctly enter the circuit description, and Symbulator does the rest.

The (Expert) Mode gives you the power to play a more important role in the simulation, because it gives you the capability to modify the equations which Symbulator generates, before the calculator solves them. In return, you assume part of the responsibility for the simulation, which is why the (Expert) Mode must be used with care. To use Symbulator as an expert requires deeper knowledge than is required of the casual user. Nevertheless, for those willing to learn a little more, the (Expert) Mode is an attractive option, because it offers faster simulations.

4.2 Theory

4.2.1 The procedure in general

An expert simulation, as the word implies, requires that you be an expert, not only in the use of Symbulator, but also an expert in its internal operations. Let us learn, then, the internal simulation processes. In general, a simulation done by Symbulator consists of these steps:

1. Equations are generated to describe the circuit.
2. These equations are solved.
3. The answers are stored.

Within this general framework, we can say that the (Expert) Mode allows us to manipulate the equations generated in step 1, before they are solved in step 2, so that when the answers are stored in step 3, they are in the form we want.

4.2.2 First-level equations

The necessary equations used by Symbulator to solve a circuit are called equations of the first level. Symbulator generates two types of first-level equations: node equations, and special equations.

There is a node equation for each circuit node. These equations are generated using Ohm's law and Kirchhoff's current law. The Symbulator will generate for each node an equation which fulfills Kirchhoff's current law: the sum of all the currents of the elements in contact with that node, set equal to zero. To describe the currents of resistive elements, Symbulator uses Ohm's law.

In addition, a special equation is generated for each special element in the circuit. These equations are generated according to the theoretical formulas which describe the behavior of the elements. It may seem strange, but voltage sources and short circuits are special elements. This results because Symbulator uses nodal analysis, so the only elements (seen so far) which are not special are resistors and current sources. The others are considered special, and each is associated with a special equation, which will increase the number of equations generated by the Symbulator.

For example, to solve a circuit with five nodes, two voltage sources and one short circuit, Symbulator will generate a total of eight first-level equations. It does not matter if this circuit has one thousand resistors and seven hundred current sources, it will still have eight first-level equations. However, the complexity of the eight equations increases as more resistors and current sources are connected to the nodes.

How many first-level equations will be generated to simulate a circuit with three nodes and one voltage source? Four: one for each node, and one more for the voltage source.

In our algebra courses, we learned that we need n equations to completely solve a system of n unknowns. This means that for the first circuit, which has eight equations, we can solve for eight unknowns. In the second circuit, we can solve for four unknowns, since we have four equations. The unknowns which can be solved with the first-level equations are called first-level variables.

4.2.3 First-level variables

As expected, Symbulator generates the first-level equations based on the first-level variables. There are as many first-level variables as there are first-level equations. Which means, indirectly, that there will be as many first-level variables as there are nodes, voltage sources and short circuits in the circuit.

With the elements we have seen so far, Symbulator generates three types of first-level variables: node voltages, voltage source currents, and short circuit currents.

Each node voltage in the circuit is a first-level variable. When solving a circuit which has three nodes 1, 2 and 3, Symbulator creates three first-level variables called `v1`, `v2` and `v3`.

The current through each voltage source is a first-level variable. When solving a circuit with two voltage sources `e1` and `ex`, Symbulator creates two first-level variables called `ie1` and `iex`.

In addition, each short-circuit current is a first-level variable. So, when solving a circuit with a short circuit called `s1`, Symbulator creates a first-level variable called `is1`.

After solving the first-level equations, the results are stored in the first-level variables. Can you guess what these answers are called? Obviously: first-level answers. In the example circuit with five nodes, two voltage sources and one short circuit, eight first-level equations are created based on the eight first-level variables. The simulation results are the five node voltages, the two voltage-source currents, and the single short-circuit current.

How many first-level variables will Symbulator generate for the circuit with three nodes and one voltage source? Obviously, there will be four: one for the voltage source current, and three for the node voltages. After the four first-level equations are solved, there are four first-level results.

4.2.4 Second-level variables

I have described currents in voltage sources and short circuits as first-level variables. However, currents in other elements such as current sources and resistors are called second-level variables.

Voltage drops across resistors and sources are also second-level variables. So, in a circuit with four nodes, three resistors, two current sources, two voltage sources, and one short circuit, we will have these first-level variables:

- 4 node voltages
- 2 currents in the voltage sources
- 1 current in the short circuit

and these second-level variables:

- 3 currents in the resistors
- 2 currents in the current sources
- 7 voltage drops across the 3 resistors and the 4 sources

This results in a total of $4+2+1=7$ first-level variables, with their respective first-level equations, and $3+2+7=12$ second-level variables. Note that the short circuit does not generate a voltage drop (see section 2.6.3 on short circuit results).

The second-level variables are not included in the first-level equations. So, how are these variables related to the simulation results?

4.2.5 Second-level expressions

The answer is simple. For each second-level variable, the Symbulator generates a second-level expression, which is a function only of the first-level variables. So, given the first-level results, all the second-level variables are immediately defined by these expressions.

There are two types of second-level expressions: those which are replaced by their results after the solution, and those which remain expressed in terms of the first-level variables. That is, after the first-level equations are solved and the results are stored in the first-level variables, Symbulator evaluates some of the second-level expressions, and so generates second-level results which are stored in the appropriate variables. However, other second-level expressions are not evaluated and remain expressions instead of results. Second-level expressions which are not evaluated are those that correspond to the currents in current sources, and the voltage drops. These remain algebraic expressions as functions of the node voltages.

4.2.6 Third-level variables

In direct current (DC) and alternating current (AC) simulations, there is another group of variables called third-level variables. They are given this name because they are generated only after the solution of the first-level equations and the evaluation of the second-level expressions. In other words, they are only created when the previous levels of variables have both been solved.

The third-level variables are the powers consumed by the elements. They are of course generated according to the definition of electrical power, which is the product of the voltage drop across an element and the current through the element, which must be previously defined.

4.2.7 The normal procedure in detail

We have learned about first-level equations, variables and results, second-level expressions and variables, and third-level variables. I will now describe the simulation process in detail, as it relates to these variables. First, the procedure of a normal simulation, which does not use the (Expert) Mode:

1. First-level equations are generated which describe the circuit entered by the user, based on the first-level variables. The second-level expressions are also generated, as functions of the first-level variables.

2. The first-level equations are solved for the first-level variables.

3. The first-level results are stored in the first-level variables. Some second-level expressions are evaluated, and their second-level results are stored.

4. In direct current and alternating current analyses, the third-level variables are generated, which are the powers consumed by the elements.

4.2.8 Procedure options for the (Expert) Mode

Several options are available when simulating with the (Expert) Mode. The first option is to choose the correct analysis type. These options are direct current (DC) analysis, alternating current (AC) analysis, frequency domain (FD) analysis and transient analysis (TR).

Choose the desired analysis type and press ENTER. Next, the calculator shows a form from which you choose two options that determine how floating-point numbers are handled:

The first option, Digits?, sets the number of digits to be used in the calculations. Normally this value should be set to 9, unless a different precision is required for some reason.

The second option, Format?, is used to set the display format. You can set the display format to Normal, Engineering or Scientific.

Press ENTER after you have chosen the desired options. At this point, Symbulator executes the first step of the simulation, detailed above. Before the second step, the program pauses and shows another form, which offers you the opportunity to manipulate the first-level equations and unknowns. You can change them or not, as needed.

This screen is called "(Expert) Mode – 1st Level". There are four fields and a menu of options:

The first field shows the first-level unknowns. If there is more than one unknown, they are shown separated with commas between curly brackets: {}.

The second field shows the first-level equations. If there is more than one equation, they are shown connected by the `and` operator.

The third and fourth fields are used to specify known first-level variables and values. The third field is used to specify the name of a first-level variable whose value is known. These variables are called conditional variables. If there is more than one known variable, they must be separated with commas, for example, `name1,name2,name3`. The fourth field is used to specify the values of the variables listed in the third field. These declared values are called conditional equations. If there is a single variable, its value is specified as an equality, for example, `v1=3.5`. If there is more than one, the equalities are connected with the `and` operator, for example, `v1=3.5 and v2=5 and ie1=-2`. Note that only first-level variables can be specified in these fields.

The menu shown under the four fields has three options which let you control the simulation process. The options are 1) run the simulation, 2) run the simulation and keep the equations, and 3) only store the equations.

Let us see next what each menu option means.

4.2.9 Only store the equations

If you choose the third option, "Just Keep", then Symbulator will simply create four variables:

- `Unknown`, which holds the list of the first-level unknowns
- `Equation`, which holds the first-level equations
- `Wheneq`, which holds the conditional equations
- `Whenvar`, which holds the conditional variables

Each variable is stored as a text string. Any modifications the user may have introduced while they were displayed in the dialog box are reflected in these variables. The purpose of the "Just Keep" option is to keep the equations and other variables so that the `solves` tool can be used later. If the circuit is very large, the generated equations may be too large to be shown on the screen. In this case, the equations are not displayed, but instead a message is shown indicating that the equations will be stored automatically. However, in most cases the equations will fit on the display screen.

After the four variables are created, Symbulator shows a new screen called "(Expert) Mode – 2nd Level". You use this screen to choose whether or not you want to keep the second-level expressions which, as we have said, have already been generated.

You choose to either keep the expressions or delete them. If you choose to delete them, Symbulator deletes them and finishes its work. If you choose to keep them, they are stored, and a third screen is displayed, called "(Expert) Mode – 3rd Level". In this screen you choose whether or not to generate the third-level expressions.

If you choose not to generate the expressions, Symbulator exits. If you choose to generate the expressions, Symbulator does that, then exits.

4.2.10 Simulate and store

If you choose the second option, "Symbulate+Keep", that is, to simulate and keep the equations, then Symbulator creates the four variables described in the previous section, then the simulation proceeds as usual. That is, steps 2, 3 and 4 are executed as they would be normally, taking into account the changes you have made.

4.2.11 Simulate only

If you choose the first option, "Symbulate", then Symbulator runs the simulation without creating the four variables. Steps 2, 3 and 4 are executed, taking into account the changes you have made, obviously. Now that we have learned the theory, let us see all these new concepts applied in practice. We will solve the problems of the previous chapter, but this time, like experts.

4.3 Practice

Problem Nº 013 with the (Expert) Mode

Problem: Determine the numerical values of v_x and i_x in the following circuit.
Solution:

4.3.1 Initial steps

We name the nodes and elements, and delete the symbolic names:

```
DelVar ra,vx
```

We enter the circuit description and specify expert simulation:

```
sq\expert("e1,1,0,18;ra,1,0,ra;r1,1,0,6;r2,2,1,5;ex,2,0,vx")
```

We start the simulation with ENTER. We choose DC analysis and press ENTER. We leave the floating-point formats unchanged and press ENTER. The screen shows status messages indicating that the (Expert) Mode is activated and that the first step is executed. The (Expert) Mode 1st Level screen appears. In this screen we can see the equations generated by Symbulator, and the unknowns for which it intends to solve them.

Remember that resistor `r2` was defined so that the direction of its current agrees with that of the 12 A current given in the problem as a known value. However, if we read the list of first-level unknowns in the first field, we see that this resistor current is not shown, for it is not a first-level unknown. On the other hand, the currents of the two voltage sources, `ie1` and `iex`, are shown, for they are first-level unknowns.

Since the current of the `ex` source is defined from node 2 to node 0, we can say that the current `iex` is equal to -12 A. The negative sign is used since the source current is defined opposite the direction shown in the problem.

Now the time has come to think carefully about the problem, and what results are needed. We see that Symbulator has generated a system of four equations and four unknowns, which also has one symbolic value. We consider this symbolic value as an unknown, because it must be solved to obtain a numerical answer. So, we really have a system of four equations and five unknowns. We can look at this situation from two perspectives: we either have an extra unknown, or we need another equation.

We will solve this problem first from the first perspective, and then from the second.

4.3.2 Eliminating an unknown

If we have an extra unknown, we must eliminate it so we can solve a system of four equations and four unknowns. We take three steps to eliminate the unknown. First, we delete `iex` from the list of unknowns in the first field. We do not need it, because its value will be declared as known. Second, we add the new first-level variable `vx` to the list of unknowns.

Finally, we add the name `iex` to the third field, and the equality `iex=-12` to the fourth field, where the '−' sign is entered with [(−)]. After making these modifications, the fields look like this:

First-level unknowns: `{v1,ie1,v2,vx}`
The first-level equations were not modified.
Known variable names: `iex`
Known variable values: `iex=-12`

We press [ENTER] to continue the simulation, and the screen shows the status messages for the remaining three simulation steps, then 'Done' is shown.

Unlike the previous techniques which require use of the `solve` command and the `solves` tool, this (Expert) Mode simulation does not require additional manipulation to obtain numerical answers. By modifying the list of unknowns and specifying the known value, we reduced a system of four equations and five unknowns to a system of four equations and four unknowns. So, the simulation results, which are now in the current folder, are numerical results.

By recalling `ir1`, we find that the current i_x is 3 A. To find the value of v_x, we have three options: 1) recall the node voltage `v2`, 2) recall the voltage drop `vex` across the source, or 3) simply recall our first-level unknown `vx`. Remember that we included `vx` in the list of first-level unknowns, so Symbulator solved for it as it would for any other first-level unknown. By any of these three roads, we find that $v_x = 78$ V.

Now let us use the other perspective to solve the problem.

4.3.3 Adding an equation

If we need another equation, we must add it, to have a system of five equations and five unknowns. Since we want to do this problem again from this new perspective, we execute all the steps of section 4.3.1 until the 1st Level screen is shown:

```
DelVar ra,vx
sq\expert("e1,1,0,18;ra,1,0,ra;r1,1,0,6;r2,2,1,5;ex,2,0,vx")
```

We start the simulation with [ENTER], choose DC analysis, leave the floating-point formats as they are, and press [ENTER]. The 1st Level screen is shown.

To add an equation we must do two things: 1) add the variable `vx` to the list of first-level unknowns, and 2) add an equation to the list of first-level equations, like this: `and iex=-12`. After these changes, the field contents are:

First-level unknowns: `{v1,ie1,v2,iex,vx}`
First-level equations: `(5*ie1*ra-ra*(vx-33)+90)/ra=0 and 5*iex+vx=18 and v1=18 and v2=vx and iex=-12`
Known variable names: empty.
Known variable values: empty.

We have changed a system of four equations and five unknowns into a system of five equations and five unknowns. We press [ENTER] to continue the simulation, the status messages are shown, then 'Done' is shown, indicating that the simulation is complete. The numerical results are in the current folder.

We recall the value of `ir1` to find that i_x is 3 A. We recall the value of `v2`, or `vx`, or `vex`, to find that v_x is 78 V. This problem illustrates that the (Expert) Mode can save an expert user time, but it requires good knowledge of circuits, algebra and the use of Symbulator.

Now let us see another problem of the previous chapter.

Problem Nº 014 with the (Expert) Mode

Problem: Determine the numerical values of v_x and i_x in the following circuit.
Solution:
We name the nodes and elements, and clear the symbolic names:

```
DelVar rx
```

We enter the circuit description and the expert simulation command:

```
sq\expert("j1,0,1,6;r1,1,2,1;r2,1,3,5;j2,3,2,10;r3,3,0,2;
          rx,2,4,rx;r4,4,0,3")
```

We start the simulation with ENTER and choose DC analysis. We leave the floating-point formats unchanged and press ENTER. The display shows the status messages indicating that the (Expert) Mode is active and that the first step is complete. Then, the 1st Level screen is shown.

We have a system of four equations and five unknowns, where the fifth unknown is the symbolic value of `rx`, which must be solved to get numeric values. We will solve this problem by adding an equation to the system, which is the simplest method. First, we add `rx` to the list of first-level unknowns, so that Symbulator now considers it an unknown. Second, we add an equation to the first-level equations list: `and ir3=4`. So now the field contents are:

First-level unknowns: `{v1,v2,v3,v4,rx}`
First-level equations: `(rx*(v1-v2+10)-v2+v4)/rx=0 and (rx*v4-3*(v2-v4))/rx=0 and 6*v1-5*v2-v3-30=0 and 2*v1-7*v3-100=0 and ir3=4`
Known variable names: empty.
Known variable values: empty.

Now I must explain what is allowed in the (Expert) Mode, and what is not. Note that `ir3` is included in the equations, but is `ir3` a first-level variable? Of course not. It is a second-level variable. But at this point, step 1 of the simulation has already been executed, in which the second-level expressions have been generated, which express the second-level variables as functions of the first-level variables. Therefore, we are allowed to include second-level variables in the first-level equations, because the second-level variables are expressed internally in terms of the first-level unknowns. On the other hand, you cannot specify second-level variables with known values in the third and fourth fields: these fields can only be used for the values of first-level variables.

Let us return to the problem. After we change the system of four equations and five unknowns to a system of five equations and five unknowns, we press ENTER and the screen messages show us that the three remaining simulation steps are executed.

Finally the message 'Done' is shown, and the numerical answers are in the current folder. We recall the value of `ir1` and find that $i_x = -8$ A. We recall the value of `vrx` and find that $v_x = 80$ V. It is possible to solve this problem with the other technique, in which we eliminate an unknown, but that method is a little more complicated here. We would have to declare the value of some first-level variable as known, in the third and fourth fields. In order to express the known current of 4 A through a first-level variable, we would use Ohm's law to relate the voltage of node 3 and the 2 ohm resistance: $v_3 = (4\text{ A})(2\ \Omega)$, that is, $v_3 = 8$. The procedure, then, would be:

```
DelVar rx
sq\expert("j1,0,1,6;r1,1,2,1;r2,1,3,5;j2,3,2,10;r3,3,0,2;
          rx,2,4,rx;r4,4,0,3")
```

We start the simulation by pressing ENTER, choose DC analysis, and leave the floating-point formats unchanged, then press ENTER to continue. The display shows that the (Expert) Mode is activated and the first step is executed, then shows the 1st Level screen.

To eliminate one system unknown, we must do three things. First, we remove `v3` from the list of first-level unknowns in the first field, because we know its value. Second, we add `rx` to the list of first-level unknowns. Third, we add the name `v3` to the third field, and specify its value in the fourth field as `v3=8`. After these changes, the fields are:

First-level unknowns: `{v1,v2,rx,v4}`
The first-level equations were not modified.
Known variable names: `v3`
Known variable values: `v3=8`

We press ENTER to continue the simulation, the screen shows the status messages for the three remaining simulation steps, and finally 'Done' is shown.

We recall the value of `ir1`, and find that $i_x = -8$ A. We recall the value of `vrx`, and find that $v_x = 80$ V. This example shows that when it is difficult to declare a known value through a first-level variable, the method of adding an equation is easier than eliminating an unknown. Nevertheless, eliminating an unknown has the advantage of reducing the system of equations, making the system easier for the calculator to solve. This is important in circuits much larger than we see in these pages.

Problem Nº 015 with the (Expert) Mode

Problem: Determine the numerical values of v_x and i_x in the following circuit.
Solution:
We name the nodes and elements, and delete the name `ix`:

```
DelVar ix
```

We enter the circuit description and specify expert simulation:

```
sq\expert("e1,1,0,60;r1,1,2,8;r2,2,0,10;r3,2,3,4;r4,3,0,2;
          jx,0,3,ix")
```

We start the simulation by pressing ENTER, choose DC analysis, and leave the floating-point formats unchanged. After pressing ENTER, the display shows that the (Expert) Mode is active, and the first simulation step is executed. The 1st Level screen is shown. Like the previous problems, we can solve this one by adding an equation or by eliminating an unknown.

To add an equation, we change the field contents so that we have five equations and five unknowns:

First-level unknowns: `{v1,ie1,v2,v3,ix}`
First-level equations: `8*ie1-v2+60=0 and 4*ix+v2-3*v3=0 and v1=60 and 19*v2-10*(v3+30)=0 and ir1=5`
Known variable names: empty.
Known variable values: empty.

On the other hand, to solve the problem by eliminating an unknown, we change the fields so that we have a system of four equations and four unknowns, by declaring the value of a first-level variable, like this:

First-level unknowns: `{v1,v2,v3,ix}`
The first-level equations were not modified.
Known variable names: `ie1`
Known variable values: `ie1=-5`

You can use either method, as you prefer. Next, we press ENTER and the display shows that the three remaining steps are executed, then the display shows 'Done'.

We recall the value of `ix` and find that $i_x = 1$ A. We recall the value of `-vjx`, or of `v3`, and find that $v_x = 8$ V. Let us see one more problem.

Problem Nº 016 with the (Expert) Mode

Problem: Determine the value of k such that voltage v_y is zero.
Solution:
We name the nodes and elements and delete the variable name `k`:

```
DelVar k
```

We enter the circuit description and specify expert simulation:

```
sq\expert("e1,1,0,6;r1,1,x,1;r2,x,0,4;r3,x,y,2;j1,0,y,2;
          r4,y,2,3;e2,2,0,k*vx")
```

We start the simulation by pressing ENTER, choose DC analysis, and leave the floating-point formats unchanged. The display shows that the (Expert) Mode is active and that the first simulation step is complete. Next the display shows the 1st Level screen. This problem can be solved by adding an equation or by eliminating an unknown.

To add an equation, the field contents are modified so that we have seven equations and seven unknowns:

First-level unknowns: `{v1,ie1,vx,vy,v2,ie2,k}`
First-level equations: `2*k*vx+3*vx-5*vy+12=0 and k*vx+3*ie2-vy=0 and ie1-vx+6=0 and v1=6 and v2=k*vx and 7*vx-2*(vy+12)=0 and vy=0`
Known variable names: empty.
Known variable values: empty.

Alternatively, to solve the problem by eliminating an unknown, we modify the field contents so we have six equations and six unknowns, by declaring the value of a first-level variable, like this:

First-level unknowns: `{v1,ie1,vx,v2,ie2,k}`
The first-level equations were not modified.
Known variable names: `vy`
Known variable values: `vy=0`

You can use either method, as you prefer. Next we press ENTER and the display shows the execution of the three remaining simulation steps, and finally 'Done' is shown. We recall the value of `k` and find that it is $-13/4$.

4.4 The solves tool after expert simulation

The `solves` tool can be used at any time. In the previous chapter we used it after a symbolic simulation. In that case, the variables `Equation` and `Unknown` did not exist in the current folder. When we used `solves` for the first time, the fields were empty, and we manually entered the equations we wanted to solve.

When we next used `solves`, the fields showed the previous manually-entered equations, unless the folder contents had been deleted.

To this point we have done our expert simulations using the first option of the (Expert) Mode, which is to simulate only, without keeping the equations. For that reason, the variables `Equation` and `Unknown` are not created in the current folder.

On the other hand, if we had used either of the other two options, that is, simulate and keep the equations, or only keep the equations, then we would have the variables `Equation` and `Unknown` in the current folder.

The `solves` tool works differently when executed just after an expert simulation which has kept the generated equations. In this case, the fields are not empty, but instead show the equations and unknowns which were stored by the (Expert) Mode in the variables `Equation` and `Unknown`.

For example, if we simulate a circuit with the (Expert) Mode and choose the "Just Keep" option, then Symbulator will store the equations and unknowns, and we can later solve them with the `solves` tool. Note that if you intend to insert a second-level variable in the first-level equations, you must ask the (Expert) Mode to keep the second-level expressions which Symbulator generated.

The `solves` tool shows two menus in its lower part. You use the first menu to choose whether to solve the equations as real or complex. You must specify real solutions unless the equations of an AC analysis are being solved. You use the second menu to choose whether or not to use conditions. What are the conditions? They are nothing more than the known values of first-level variables. If we choose to use conditions, a new screen is shown so that you can enter these values and their names.

In the field called "When eq's", the known values of the first-level variables must be entered. In the field called "When var's", the names of these first-level variables are entered.

4.5 Verification of numerical results of expert simulations

We can use the procedure of section 3.7 to verify the numerical results of expert simulations.

Before closing this chapter, I want to congratulate you on deciding to become an expert Symbulator user. Now that you know the theoretical concepts, all that remains is to use them in practice for some time. The rewards for your efforts, in satisfaction and time saved, will not be long in coming.

Chapter 5

Thévenin and Norton equivalents

5.1 Introduction to the thevenin tool

The Thévenin–Helmholtz and Norton–Mayer theorems are an important part of circuit theory courses. Symbulator has a special tool to find the Thévenin and Norton equivalents of a network. This tool is called `thevenin`, and it is executed like this:

```
thevenin(network,node 1,node 2)
```

As you can see, it takes three input arguments: the description of the network in text format, and the names of the two nodes between which you want to find the equivalent.

As output, it gives us three variables stored in the current folder. These variables are the Thévenin impedance `zth`, the Thévenin voltage `vth` and the Norton current `ino`. With these three values, you can then construct the Thévenin equivalent by placing an impedance of value `zth` in series with a voltage source of value `vth`, and the Norton equivalent by placing an impedance of value `zth` in parallel with a current source of value `ino`.

In some cases, as we will see later, the tool also gives us the maximum power the network can deliver.

5.2 Theory

After entering the command to execute the tool in the entry line, with its input arguments, a screen is shown in which you must choose two settings:

What type of network do you want to analyze? The options are Auto-Detect and Passive. Use Passive if the network has no independent sources, and Auto-Detect when it has them or when you are not sure.

It is important to choose the network type well. If we choose Auto-Detect, the tool will run two simulations, regardless of the network type, and we will obtain the correct equivalents. Choosing Passive for those networks which really are passive has the advantage that the tool saves us time, because it obtains the correct equivalents with a single simulation. However, if by mistake we choose Passive and the network is not passive, we will obtain wrong answers. *Never choose Passive if the network is not passive, or if you have doubts.* Remember that a passive network is one that has no independent sources.

What type of analysis do you want? The options are direct current (DC) analysis, alternating current (AC) analysis and frequency domain (FD) analysis.

After choosing the desired settings, press ENTER. At this point, Symbulator will run one or two simulations, whose status messages you will see on the screen. Then an answer screen appears. In this answer screen, the program shows us, one by one, the values found: the Thévenin impedance `zth`, the Thévenin voltage `vth` and the Norton current `ino`. To see the next value, press ENTER. At the end, when all the values have been shown, 'Done' appears. Stored in the current folder are all these values, in variables with the same names.

In the case that Passive is used, the tool will not show the values of `vth` and `ino` on the screen, because they will always be zero.

In the case that Auto-Detect is used, the tool will also deliver, as an answer for direct current analyses, the maximum real power that the network can deliver, on the answer screen and in a variable called `pmax`; and as an answer for alternating current analyses, the maximum complex power that the network can deliver, on the answer screen and in a variable called `smax`.

5.3 Practice

Let us see some problems.

Problem Nº 017

Problem: Determine the Thévenin and Norton equivalents for the part of the circuit that is to the left of R_L.

Solution:

We name the nodes and elements of the part of the circuit that is to the left of R_L. The element R_L plays no role in our simulation. We name the nodes from left to right: 1, 2 and 3. The bottom node is the reference. Therefore, the nodes between which we want to find the equivalents are node 3 and node 0. We give the description of the part of the circuit to the left of R_L, and execute the `thevenin` tool:

```
sq\thevenin("e1,1,0,12;r1,1,2,3;r2,2,0,6;r3,2,3,7",3,0)
```

We start the simulation with ENTER. Since the network we want to analyze has an independent source, we choose Auto-Detect as the circuit type. We choose DC as the analysis type. We press ENTER. We see on the screen the status messages indicating that two simulations are run. On the answer screen appear the values of the Thévenin and Norton equivalents. For `zth` we obtain 9. For `vth` we obtain 8. For `ino` we obtain 8/9. With these values, you can construct the Thévenin and Norton equivalents of the network in question. As for the maximum power the network can deliver, for `pmax` we obtain 16/9. Finally, 'Done' appears.

Problem Nº 018

Problem: Determine the Thévenin and Norton equivalents of the network in front of the 1 kΩ resistor.

Solution:

We name the nodes and elements of the network that is to the left of the 1 kΩ resistor. This resistor plays no role in our simulation. We name the nodes from left to right: 1, 2 and 3. The bottom node is the reference, so the nodes between which we want the equivalents are node 3 and node 0. We give the description of the part of the circuit that interests us, and execute the tool:

```
sq\thevenin("e1,1,0,4;r1,1,2,2E3;j1,0,2,2E-3;r2,2,3,3E3",3,0)
```

We start the simulation with ENTER. The network has independent sources, so we choose Auto-Detect as the circuit type, and DC as the analysis type. We press ENTER. The status messages show that two simulations are run. On the screen we receive the values of the equivalents. For `zth` we obtain 5000. ohms, for `vth` we obtain 8. volts, for `ino` we obtain .0016 amperes, and for `pmax` we obtain .0032 watts. Finally, 'Done' appears.

Problem Nº 019

Problem: Determine the Thévenin and Norton equivalents of the network shown in the figure.

Solution:

We name the nodes and elements of the network. We name the nodes from left to right: 1, 2 and `x`. The bottom node is the reference. Therefore, the nodes between which we want to find the equivalents are node `x` and node 0. We give the description of the network and execute the tool:

```
sq\thevenin("e1,1,0,4;r1,1,2,2E3;j1,0,2,vx/4E3;r2,2,x,3E3",x,0)
```

We start the simulation with ENTER. The network has an independent source, so we choose Auto-Detect, and DC as the analysis. We press ENTER. The status messages show that two simulations are run. Then the values of the equivalents appear. For `zth` we obtain 10000. ohms, for `vth` we obtain 8. volts, for `ino` we obtain .0008 amperes, and for `pmax` we obtain .0016 watts. Finally, 'Done' appears.

Problem Nº 020

Problem: Find: 1) R_{eq} if $R = 80\ \Omega$; 2) R if $R_{eq} = 80\ \Omega$; and 3) R if $R = R_{eq}$.

Solution:

To solve this problem, we will use the `thevenin` tool and some commands. We name the nodes and elements of the network. We name the nodes from left to right: 1, 2, 3 and 4. The bottom node is the reference. Therefore, the nodes between which we will look for the equivalent resistance are node 1 and node 0. We give the description of the network and execute the tool:

```
sq\thevenin("r1,1,2,10;r2,2,0,100;r,2,3,r;r3,3,0,30;
            r4,3,4,40;r5,4,0,20",1,0)
```

We start the simulation with ENTER. The network we want to analyze has no independent sources, so we choose Passive as the circuit type, and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. For `zth`, we obtain a function of `r`, as expected. Finally, 'Done' appears. We have been asked to find: 1) R_{eq} if $R = 80\ \Omega$, 2) R if $R_{eq} = 80\ \Omega$, and 3) R if $R = R_{eq}$. To find these answers, we will use the `solve` command and the 'with' operator of the calculator.

Remember that `zth` is stored in memory. To obtain the first answer, we write in the entry line the following:

```
zth|r=80
```

Here we are asking: how much is `zth` if `r` is 80? We ask it this way, and not with `solve`, because `zth` is a function of `r`, and not the other way around. We press ENTER. We obtain 60.

To obtain the second answer, we take advantage of the fact that `zth` is a function of `r`, and write in the entry line the following:

```
solve(zth=80,r)
```

Here we are asking: how much is `r` if `zth` is 80? We press ENTER. We obtain $\texttt{r} = 640/3$.

To obtain the third answer, we write in the entry line the following:

```
solve(zth=r,r)
```

Here we are asking: how much is `r` if `zth` equals `r`? We press ENTER. We obtain two answers. One of them is positive and the other negative. This means that the equation has more than one solution. Obviously, negative resistances do not exist, so we will limit the range we want as answers with a conditional operator. We will also take the opportunity to request the answer in approximate form:

```
solve(zth=r,r)|r>0
```

Here we are asking: how much is `r` if `zth` equals `r` and `r` is greater than 0? We press ◇ and ENTER. We obtain `r` = 51.7890835.

These are the correct answers.

Let us take a pause and meet a new tool.

5.4 The par tool

The `par` tool is a function, and its first version was written by my Belgian friend Erwin Baert, for one of the early versions of Symbulator, as a simple and fast tool to find the equivalent of two resistances connected in parallel. Although I later introduced some improvements in the algorithm, the function remains very simple in its nature and use.

To find the equivalent of two resistances, say one of 3 Ω and another of 5 Ω, that are connected in parallel, you write `sq\par(3,5)`.

This tool can be used either alone or included within a circuit description.

5.4.1 Verifying the formula for parallel resistances

To show the use of this tool, let us verify the formula for reducing resistances in parallel. The equivalent of two resistances, say R_1 and R_2, connected in parallel, is found with the formula $R_{eq} = R_1R_2/(R_1 + R_2)$. Let us verify this formula with the `par` tool:

```
sq\par(r1,r2)
```

We obtain `r1*r2/(r1+r2)`. This is correct.

5.4.2 Application

It is easy to find simple applications for the `par` tool in practice. What interests me here is to show an example of a somewhat more complex application. We will solve Problem 020 again, this time using the `par` tool instead of the `thevenin` tool.

Problem Nº 020 again

Problem: Find: 1) R_{eq} if $R = 80$ Ω; 2) R if $R_{eq} = 80$ Ω; and 3) R if $R = R_{eq}$.

Solution:

Let us define the equivalent resistance `req` as the combination of these resistances:

```
Define req=10+sq\par(100,r+sq\par(30,40+20))
```

We press ENTER. We obtain a function of `r`, as expected.

```
req|r=80
```

We press ENTER. We obtain 60.

```
solve(req=80,r)
```

We press ENTER. We obtain `r` = 640/3.

```
solve(req=r,r)|r>0
```

We press ◇ and ENTER. We obtain `r` = 51.7890835.

These are the correct answers. We have seen in this problem that the `par` tool can be very useful to reduce some resistive networks to their equivalent. However, there are some resistive networks that resist being reduced with series-parallel simplifications, and that necessarily require the `thevenin` tool. This is the case of networks that require delta–wye, or Δ–Y, transformations. Let us see an example.

Problem Nº 021

Problem: Find the equivalent resistance of the network shown.

Solution:

Done manually, this problem requires Δ–Y transformations. We will use the `thevenin` tool. We name the three upper nodes, from left to right: 1, 2 and 3. We name the two intermediate nodes, from left to right: 4 and 5. The bottom node is the reference. Therefore, the nodes between which we will look for the equivalent resistance are node 1 and node 0. We name the elements, give the description of the network and execute the tool:

```
sq\thevenin("r1,1,2,5;r2,2,3,6;r3,4,2,75;r4,2,5,4;r5,5,3,2;
            r6,4,5,9;r7,3,0,3;r8,4,0,100;r9,5,0,12",1,0)
```

We start the simulation with ENTER. The network we want to analyze has no independent sources, so we choose Passive as the circuit type, and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. As the value of the equivalent resistance, `zth`, we obtain 6365/643. Finally, 'Done' appears. We can obtain an approximate value by asking in the entry line for `zth` with ◇ and ENTER, which gives 9.89891135.

Note that, since this is a passive circuit, we chose the Passive option. This implies that no `pmax` variable is created, since a passive circuit cannot deliver power, having no independent sources.

We have said that circuits with no independent sources are considered passive. The presence of dependent sources does not change this fact. Let us see next four problems in which the circuits contain only dependent sources and are therefore passive.

Problem Nº 022

Problem: Find the Thévenin equivalent of the network shown in the figure.

Solution:

Note that the dependent source is controlled by a current i that does not pass through any element. We have two options. The first is to place a short circuit where i is, and then define i as the current through that short circuit. The second is to understand i as the sum of the currents of the two resistors, which would have to be defined in the appropriate directions so that their currents flow into i. We will use this second alternative, because it does not require a new element.

We name the two upper nodes, from left to right: 1 and 2. The bottom node is the reference. Therefore, the nodes between which we will look for the equivalent resistance are node 2 and node 0. We name the elements, give the description of the network and execute the `thevenin` tool:

```
sq\thevenin("e1,1,0,1.5*(ir1+ir2);r1,1,2,3;r2,0,2,2",2,0)
```

Note that both resistors "flow into" node 2. We start the simulation with ENTER. The network we want to analyze has no independent sources, so we choose Passive as the circuit type, and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. For `zth` we obtain .6 ohms. Finally, 'Done' appears.

Problem Nº 023

Problem: Find the Norton equivalent of the network shown in the figure.
Solution:

We name the two upper nodes, from left to right: 1 and 2. The bottom node is the reference. We will look for the equivalent resistance between node 1 and node 0. We name the elements, give the description of the network and execute the `thevenin` tool:

```
sq\thevenin("r1,1,0,100;r2,1,2,50;r3,2,0,200;j1,2,0,0.1*v1",1,0)
```

We start the simulation with ENTER. We choose Passive as the circuit type and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. For `zth` we obtain 10.6382979 ohms.

Problem Nº 024

Problem: Find the input impedance, Z_{in}, of the network shown in the figure.
Solution:

The nodes of the upper loop were named clockwise, starting at the left: 1, 2, 3 and 4. The bottom node is the reference. We name the 8 ohm resistor `rx`, and define it with the appropriate polarity, between 3 and 2, in order to use its voltage drop `vrx` as the control for the dependent source. We give the description of the network and execute the `thevenin` tool, between 1 and 0:

```
sq\thevenin("r1,1,2,20;rx,3,2,8;r2,1,4,10;r3,4,3,12;
            r4,3,0,5;e1,4,0,.6*vrx",1,0)
```

We start the simulation with ENTER. We choose Passive as the circuit type and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. For `zth` we obtain 6.70508119 ohms.

Problem Nº 025

Problem: Find the Thévenin equivalent of the network shown in the figure.
Solution:

We will ignore the names a and b shown in the drawing. We name the three upper nodes, from left to right: 1, 2 and 3. The bottom node is the reference. We will look for the equivalent resistance between node 3 and node 0. We name the elements, give the description of the network and execute the `thevenin` tool:

```
sq\thevenin("j1,0,1,.01*v3;r1,1,0,200;r2,1,2,50;
            e1,3,2,.2*v3;r3,3,0,100",3,0)
```

We start the simulation with ENTER. We choose Passive as the circuit type and DC as the analysis type. We press ENTER. The status messages show that one simulation is run. For `zth` we obtain 192.307692 ohms.

I believe that with the problems shown, you will at this point have a complete command of the `thevenin` tool.

Chapter 6

Numerical simulation of alternating current (AC) circuits

6.1 Phasors in your calculator

In alternating current analysis, phasors are used. Your calculator is a magnificent tool for handling complex numbers, whether in simple operations such as addition, subtraction, multiplication and division, or in really complicated operations, such as matrix manipulation and the solution of simultaneous non-linear equations with symbolic and complex terms.

The imaginary operator is shown in the calculator with a symbol distinct from the lowercase i, and is entered by pressing 2nd CATALOG (i is printed above that key).

The calculator can present complex numbers in three ways: rectangular, polar and exponential. The way it uses at a given moment depends on the calculator modes at that moment. If the mode called Complex Format is set to RECTANGULAR, complex numbers are presented in rectangular form, like this: real+imaginary$\cdot i$. If Complex Format is set to POLAR, the presentation then depends on another mode, called Angle. If the Angle mode is set to RADIAN, complex numbers are presented in exponential form. If, on the contrary, Angle is set to DEGREE, they are presented in polar form, like this: (magnitude $\angle$ angle in degrees).

For more information on the capabilities of your calculator, its operating modes and the ways it can present complex numbers, see the User's Manual.

Due to the needs of its own internal operation, the Symbulator always sets these two modes, during the simulation, to RECTANGULAR and RADIAN. However, Symbulator respects the modes that you had selected before the simulation: it will remember how you had these modes set, and when it has finished its work, it will set them again just as it found them.

Although the calculator can deliver complex numbers in three presentations, normally we electrical engineers handle them in only two of them: in rectangular form and in polar form. Frequently, after an alternating current simulation, we will want to see answers in these two formats. Then a doubt arises: if we wish to see complex numbers both in rectangular and in polar form, how should we set the calculator modes? The answer is that there is no combination that allows seeing the numbers in both formats, because the calculator cannot guess when you want them one way and when the other.

There are simple alternatives for seeing complex numbers in rectangular presentation and in polar presentation, without needing to change the modes, and regardless of how they are set.

6.1.1 The →Rect command

If we want to see a complex number in rectangular form, no matter how the calculator modes are set, we can use the →`Rect` command. This command can be found in the function catalog of the calculator.

The function catalog can be accessed through the CATALOG key on the TI-89, and the combination 2nd 2 on the TI-92+. When you access the catalog, you will see a list of all the commands available in the calculator. By pressing the R letter key, you can jump to the part of this list where the commands beginning with R are found. There you can see the →Rect command. Selecting it with the cursor and pressing ENTER brings it to the entry line.

Let us see how this command is used. Suppose we have a variable called sr5, which contains a complex value. We want to see this value in rectangular presentation, and for this we are going to use the →Rect command, because the calculator modes are not set the appropriate way. We write in the entry line:

```
sr5→Rect
```

In the history area the value of sr5 appears in rectangular presentation.

6.1.2 The absang tool

There exists a →Polar command, which is the equivalent of the →Rect command for polar presentation. However, this command will present complex numbers in exponential form if the Angle mode is set to RADIAN. So, it is not a universal solution.

My friend Erwin Baert programmed for the first Symbulator a small function, called absang, whose algorithm I modified to turn it into a universal solution to the need of presenting a complex number in polar form, no matter how the calculator modes are set. Let us see how it is used. Suppose we have a variable called sr5, which contains a complex value. We want to see this value in polar presentation, and since we do not want to worry about how the calculator modes are set, we are going to use the absang tool. We write in the entry line:

```
sq\absang(sr5)
```

In the history area the value of sr5 appears in polar presentation, in text format. This is one difference between the absang tool and the →Rect command: the number is presented in the form of text.

Another of the characteristics of the absang tool is that no matter whether the input value is exact, the output will always be approximate. This is intentional. Its usefulness will be seen later, in Problem 031.

6.1.3 My personal recommendation

Personally, I prefer to keep the Complex Format mode set to RECTANGULAR, so that complex numbers are always presented in rectangular form. And when I wish to see them in polar form, I use the absang tool.

6.2 Door for alternating current: ac

To run an alternating current simulation, the door called sq\ac is used.

6.3 Input data: the circuit and the radian frequency

When using this door it is necessary to supply, as input data, the circuit you want to simulate and the angular frequency, in radians/second. This frequency is the one usually designated

as ω, and it must not be confused with the frequency f which is given in Hz. To obtain ω from f, the formula $\omega = 2\pi f$ is used. The alternating current simulation is ordered like this: `sq\ac(circuit,`ω`)`.

6.4 Answers

In the case of alternating current simulation, the answers are very similar to those we obtain with direct current simulation: the node voltages, the voltage drops in the elements, the currents in the elements and the powers consumed by the elements. There are only two differences. The first is that in this case the answers will be complex values. The second is that the consumed powers that the simulator gives us are the complex powers, in volt-amperes (VA). These powers are stored in variables whose name is `s` plus the name of the element. For example, if the element is called `r5`, the name of the variable in which the complex power consumed by the element is stored is `sr5`.

6.5 The real, imag and conj commands

The calculator offers some special commands for working with complex numbers. Let us see three of them.

To obtain the real component of a complex power, and in general of any complex number, we can use the `real` command of the calculator, like this:

```
real(sr5)
```

On the other hand, to obtain the imaginary component of a complex power or any other complex number, we can use the `imag` command, like this:

```
imag(sr5)
```

If we wanted to find the conjugate of a complex current or any other complex number, we can use the `conj` command, like this:

```
conj(ir5)
```

The time has come to meet the description of some elements used in alternating current analysis.

6.6 Impedances

An impedance is a resistive element with a complex value given in Ω. In the circuit descriptions used in Symbulator, impedances are described exactly like resistors, with the single difference that their values will be complex.

6.7 Admittances

An admittance is a conductive element with a complex value given in siemens. In the circuit descriptions used in Symbulator, admittances are described with the same technique used for conductances, with the single difference that their values will be complex.

6.8 Capacitors

A capacitor is a circuit element that stores energy in the form of an electric field and that opposes sudden changes in voltage.

6.8.1 Notation

In the case of the capacitor, the minimum information necessary to describe it completely is the following:

What class of element is it? It is a capacitor. The letter that identifies capacitors is `c`.

How is the capacitor named? To distinguish a given capacitor from any other capacitor, it must be given a unique name, which begins with `c`. Remember that the variables called `c#`, where # is a number between 1 and 99, are reserved system variables and cannot be used as names.

Where is the capacitor connected? A capacitor has two ends, each one connected to a different node. It is necessary to give the simulator the names of the two nodes to which the capacitor is connected. The order of these nodes establishes the direction that the current flow and the voltage drop through the capacitor will have in the answers.

What is the value of the capacitor? A capacitor has a value. This value must be given to Symbulator in farads.

Sometimes we find in textbooks problems in alternating current in which capacitors appear whose values are given in Ω. For the purposes of the simulation, these elements are impedances, and they must be treated as such in the circuit description, even though they appear drawn as capacitors.

This tends to confuse the novice user, who tries by force to define these elements in Symbulator as capacitors, but declaring their value in Ω. This is an error. Elements whose values are to be declared in ohms must be treated as impedances. To define them as capacitors, you would first have to obtain their value in farads.

Take care not to enter the value of the capacitance in a unit other than farads (F). For example, if the value is entered in microfarads (μF), the answers will be wrong.

So, capacitors are defined as shown below:

```
cname, node 1, node 2, value in farads
```

In alternating current analysis, only these four terms are declared. We will see in later chapters that frequency domain analysis and transient analysis require a fifth term, to declare the initial voltage of the capacitor. When the capacitor has an initial voltage, we must place it in the fifth term, and give the nodes in the appropriate order to indicate the polarity we want to give this initial voltage: the first node with respect to the second node of the description. The circuit drawing will show us, with the plus and minus signs, the polarity we must give the capacitor. Example:

```
cname, node 1, node 2, value, initial voltage
```

This initial voltage is taken by the simulator as the voltage of node 1 with respect to node 2.

In direct current analysis, Symbulator considers capacitors as open circuits.

6.8.2 Related answers

In the case of a capacitor in alternating current analysis, three related answers are given: the current through the capacitor, the voltage drop across the capacitor and the complex

power consumed by it. The current is stored in a variable called `iname`, the voltage drop in a variable called `vname`, and the complex power in a variable called `sname`, where `name` is the name of the capacitor. All the conventions previously explained about direction, polarity and consumption apply to the answers of this element.

6.9 Inductors

An inductor is a circuit element that stores energy in the form of a magnetic field and that opposes sudden changes in current.

6.9.1 Notation

In the case of the inductor, the minimum information necessary to describe it completely is the following:

What class of element is it? It is an inductor. The letter that identifies inductors is `l` (that is, lowercase L).

How is the inductor named? To distinguish a given inductor from any other inductor, it must be given a unique name, which begins with `l`.

Where is the inductor connected? An inductor has two ends, each one connected to a different node. It is necessary to give the simulator the names of the two nodes to which the inductor is connected. The order of these nodes establishes the direction that the current flow and the voltage drop through the inductor will have in the answers.

What is the value of the inductor? An inductor has a value. This value must be given to Symbulator in henries.

Just as with capacitors, sometimes we find in textbooks alternating current problems that show inductors with values given in Ω. For the purposes of the simulation, these elements must be treated as impedances, even though they appear drawn as inductors.

Take care not to enter the value of the inductance in a unit other than henries (H). For example, if the value is entered in millihenries (mH), the answers will be wrong.

So, inductors are defined as shown below:

```
lname, node 1, node 2, value in henries
```

In alternating current analysis, only these four terms are declared. Frequency domain analysis and transient analysis also require the initial current of the inductor to be declared. When the inductor has an initial current, we must give the nodes in the appropriate order to indicate the direction we want to give this initial current: from the first node to the second node of the description. The circuit drawing will show us, with an arrow, the direction we must give the inductor. Example:

```
lname, node 1, node 2, value, initial current
```

This initial current is taken by the simulator as flowing through the inductor from node 1 to node 2.

In direct current analysis, Symbulator considers inductors as short circuits.

6.9.2 Related answers

In the case of an inductor in alternating current analysis, three related answers are given: the current through the inductor, the voltage drop across the inductor and the complex power consumed by it, stored in variables called `iname`, `vname` and `sname`, where `name` is the

name of the inductor. All the conventions previously explained about direction, polarity and consumption apply to the answers of this element.

Let us see some examples.

Problem Nº 026

Problem: Find the current $i(t)$ in the circuit.

Solution:

The value of the source, $40\sin(3000t)$, can also be expressed as $40\cos(3000t-90°)$. Taking this value to a phasor, it is $(40\angle -90°)$. We name the three upper nodes, from left to right: 1, 2 and 3. The bottom node is the reference. We name the elements, give the circuit description and run the simulation with the `sq\ac` door:

```
sq\ac("e1,1,0,(40∠-90°);r1,1,2,1.5E3;r2,2,3,1E3;l1,2,0,1/3;
      ca,3,0,(1/6)E-6",3000)
```

Note that `ca` was used as the name of the capacitor, because the variable `c1` is a reserved variable. The direction in which the elements are defined does not matter in this problem, except the resistor `r1`, which was made to coincide with the direction of $i(t)$. We start the simulation with ENTER. We see on the screen the status messages indicating that the simulation runs, and then 'Done' appears. To see the value of $i(t)$, we can ask for the value of `ir1`, using the `absang` tool:

```
sq\absang(ir1)

".015989779∠-126.842333°"
```

This is the correct answer. If we wanted to express this current as a function of time, it would be $.015989779\cos(3000t - 126.842333°)$.

The `thevenin` tool can also make use of alternating current analysis. Let us see two examples.

Problem Nº 027

Problem: If $\omega = 1000$ rad/s, find the input impedance that would be measured between the terminals: 1) a and g; 2) b and g; 3) a and b.

Solution:

We will take the bottom node as the reference, and leave nodes a and b with the same names the drawing has given them. We name the elements, give the description of the network and execute the `thevenin` tool:

```
sq\thevenin("l1,a,0,5E-3;r1,a,0,10;ca,a,b,200E-6;
            cb,b,0,100E-6;l2,b,0,20E-3",a,0)
```

We press ENTER. We choose Passive as the circuit type and AC as the analysis type. We press ENTER. The `thevenin` tool now asks us the value of the radian frequency ω in rad/s. We enter 1000, and press ENTER. We see on the screen the status messages indicating that one simulation is run. Since we have the Complex Format mode set to RECTANGULAR, we do not have to use the →`Rect` command to see the answer in rectangular presentation. For `zth` we obtain $2.80898876 + 4.49438202i$ ohms, which is the correct answer. Let us continue then with the next pair of nodes:

```
sq\thevenin("l1,a,0,5E-3;r1,a,0,10;ca,a,b,200E-6;
            cb,b,0,100E-6;l2,b,0,20E-3",b,0)
```

Note that we only changed node a to node b at the end of the command. We press ENTER, choose Passive and AC, press ENTER, enter 1000, and press ENTER. For `zth` we obtain $1.79775281 - 1.1235955i$ ohms, which is the correct answer. Let us continue:

```
sq\thevenin("l1,a,0,5E-3;r1,a,0,10;ca,a,b,200E-6;
            cb,b,0,100E-6;l2,b,0,20E-3",a,b)
```

Again, we only changed the nodes at the end. We press ENTER, choose Passive and AC, press ENTER, enter 1000, and press ENTER. For `zth` we obtain $.112359551 - 3.82022472i$ ohms, which is the correct answer.

Problem Nº 028

Problem: If $\omega = 100$ rad/s, find: 1) Z_{in}; 2) Z_{in} if a short circuit is connected from x to y.

Solution:

We will take the bottom node as the reference, leave nodes x and y with the names the drawing has given them, and name the middle node 1. We name the elements, give the description of the network and execute the `thevenin` tool:

```
sq\thevenin("r1,x,1,20;ca,1,0,2E-3;r2,1,y,10;l1,y,0,.1",x,0)
```

We press ENTER. We choose Passive and AC, press ENTER, enter 100, and press ENTER. For `zth` we obtain $22. - 6.i$ ohms, which is the correct answer.

For the second part of this problem, the statement asks us to connect a short circuit between x and y. The purpose is to make one node out of both. For the purposes of the simulation, we could connect a short circuit between these two nodes. This would add a new element to the simulation. But we could also do something better: replace with an x every y that appears in the description. That way, we will simulate as if x and y were the same node, which for all practical purposes is equivalent to a short circuit, since we are not interested in the current through this short. Not only are we avoiding the addition of a new element, we are eliminating a node from the circuit:

```
sq\thevenin("r1,x,1,20;ca,1,0,2E-3;r2,1,x,10;l1,x,0,.1",x,0)
```

We press ENTER. We choose Passive and AC, press ENTER, enter 100, and press ENTER. For `zth` we obtain $9.6 + 2.8i$ ohms, which is the correct answer.

We have seen in these two examples that the `thevenin` tool also uses alternating current analysis, with success. Let us now see an example of a circuit in which the inductors and capacitors must be simulated as impedances, because their values are given to us in Ω.

Problem Nº 029

Problem: Find I_1, I_2 and I_3.

Solution:

The value of the source is already a phasor, and can be reduced from $(100\angle 0°)$ to simply 100. The circuit shows three passive elements: a capacitor, a resistor and an inductor. The values of all of them, however, have been defined in the circuit as impedances. Therefore, they must be simulated as impedances. We will take the precaution of making the numbering and the direction of our impedances coincide with the currents I_1, I_2 and I_3. With the bottom node as reference, we name the other nodes and the elements, give the circuit description and run the simulation. Since we have to give the program a radian frequency as input data and we do not have any, we give it any value. I used `x`, for example:

```
sq\ac("e1,1,0,100;r1,1,2,-5i;r2,2,0,5;r3,2,0,5i",x)
```

Remember that the negative sign must be entered with the negation key, and not the subtraction key. The imaginary operator i is entered as indicated in section 6.1. We press ENTER. We see on the screen the status messages indicating that the simulation runs, and then 'Done' appears. For the value of I_1, with `absang(ir1)` we obtain "28.2842712∠45.°" amperes. For the value of I_2, with `absang(ir2)` we obtain "20.∠90.°" amperes. For the value of I_3, with `absang(ir3)` we obtain "20.∠0.°" amperes. These are the correct answers.

In this problem we have learned that, when a problem presents us with impedances, it does not matter whether they were obtained from capacitors or inductors: they must be simulated as impedances.

6.10 The separator :

In the calculator, two separate commands can be joined in a single line, by means of the separator : (that is, a colon). For example, the two commands:

```
sq\ac("e1,1,0,100;r1,1,0,30i",x)
absang(ir1)
```

…can be joined in a single line, like this:

```
sq\ac("e1,1,0,100;r1,1,0,30i",x):absang(ir1)
```

This separator will be used in the following problem.

As we saw in section 6.3, when the frequency is given to us in Hz, we must first convert it to radians/second to feed it to the simulator. The following problem illustrates this point, and also demonstrates that Symbulator handles dependent sources successfully in alternating current analysis.

Problem Nº 030

Problem: Let $V_s = 100\angle 0°$ V at $f = 50$ Hz in the circuit shown in the figure. Find the phasor voltage V_x, if element X is: a) a 30 Ω resistor; b) a 0.5 H inductor; c) a 50 μF capacitor.

Solution:

From now on, we will round the answers to four figures. Let us run the first simulation:

```
sq\ac("e1,1,0,100;r1,1,2,10;r2,2,3,20;r3,3,4,30;j1,0,2,.05*v4;
      r4,3,0,60;rx,4,0,30",2*π*50):sq\absang(v4)
```

Note that the frequency was not entered in Hz, but converted to radians/second. Note also the use of the separator : to join both commands in one line. We press ENTER. We see on the screen the status messages indicating that the simulation runs. When the simulation finishes, instead of 'Done' appearing, the answer to the last command of the line appears, that is, the voltage `v4`. This is the value of V_x: 28.57∠0.° volts (rounded). This is the correct answer. Let us run the second simulation, modifying the last element of the description:

```
sq\ac("e1,1,0,100;r1,1,2,10;r2,2,3,20;r3,3,4,30;j1,0,2,.05*v4;
      r4,3,0,60;lx,4,0,.5",2*π*50):sq\absang(v4)
```

We press ENTER. For the value of V_x, we obtain 90.24∠25.52° volts, which is the correct answer. Let us run the third simulation, modifying the last element of the description:

```
sq\ac("e1,1,0,100;r1,1,2,10;r2,2,3,20;r3,3,4,30;j1,0,2,.05*v4;
      r4,3,0,60;cx,4,0,50E-6",2*π*50):sq\absang(v4)
```

We press ENTER. For the value of V_x, we obtain $64.71\angle -49.67°$ volts, which is the correct answer.

Let us meet two new Symbulator elements, related to magnetic coupling: mutual inductance and the ideal transformer.

6.11 Mutual inductance

A mutual inductance is a mutual magnetic coupling between two inductors. It is not a physical "element", but it is an element for the purposes of the simulation.

6.11.1 Notation

In the case of mutual inductance, the minimum information necessary to describe it completely is the following:

Identification. The letter that identifies mutual inductances is `m`.

Name. Any name beginning with `m` is valid.

Names of the two coupled inductors. In Symbulator, a mutual inductance couples exactly two inductors, each one with a distinct name. It is necessary to give the simulator the names of the two inductors that are coupled by the mutual inductance.

It is important that these two inductors coupled by the mutual inductance be defined in the correct direction. In the circuit drawing, when two inductors are coupled, a dot is used at one end of each inductor to indicate the sense of the coupling. So, each inductor has one end with a dot and the other end without a dot.

When defining these two inductors in Symbulator, their directions must be defined consistently with the dots. If, when defining the first inductor, the node that has the dot in the drawing was entered first, then when defining the second inductor, the first node defined must also be the one that has the dot in the drawing. Or vice versa. It is very important to define the nodes of the inductors in the correct order.

Value. For each coupling, there exists a coefficient of mutual inductance. This coefficient is symbolized by M if its value is given in henries, and by k if its value is given dimensionlessly. There is a formula that relates these two presentations: $k = M/\sqrt{L_1 L_2}$. So, knowing one of them, it is always possible to know the other. For the purposes of describing a mutual inductance in Symbulator, the value used is M and it must be given in henries. Therefore, if the problem gives us the value of k, we must convert it to M before using it in the description.

Take care not to enter the value of a mutual inductance in a unit other than henries (H). For example, if the value is entered in millihenries (mH), the answers will be wrong.

So, a mutual inductance is defined as shown below:

```
mname, inductor 1, inductor 2, value in henries
```

If necessary, the fifth space is filled with 0.

In direct current analysis, Symbulator ignores mutual inductances.

6.11.2 Related answers

There are no answers related to the mutual inductance itself. However, each of the two inductors involved will have the related answers proper to any inductor, which we saw in section 6.9.2.

Let us see some examples.

Problem Nº 031

Problem: With the circuit of the figure, present in polar form the current in the 400 Ω resistor and the ratio V_2/V_1.

Solution:

We name the nodes from left to right 1, 3 and 2, with the purpose that `v1` and `v2` keep in the answers the same names as in the drawing. Read the circuit description below these lines, and the comments that follow:

```
sq\ac("e1,1,0,10;r1,1,3,1;l1,3,0,1;l2,2,0,100;r2,2,0,400;
      m1,l1,l2,9",10)
```

Note that the first node in the description of each inductor is the node that has the dot in the drawing. An equally correct alternative would have been to do the inverse: that the first node in the description of each inductor had been the node that does not have the dot in the drawing. The important thing is that the same convention be used in both inductors. Note that the mutual inductance does not have to be defined immediately after the inductors it relates. Note also that the value of the mutual inductance was entered in henries. We press ENTER. We see on the screen the status messages indicating that the simulation runs. When the simulation finishes, 'Done' appears. To find the answers, we ask for `absang(ir2)` and obtain $.1724\angle -16.7°$ amperes (rounded), and `absang(v2/v1)` and obtain $6.896\angle -16.7°$ (rounded). These are the correct answers.

Problem Nº 032

Problem: In the circuit of the figure, find the real average power absorbed by the source and the two resistors.

Solution:

Here is the description I used:

```
sq\ac("e1,1,0,100*√2;r1,1,2,50;l1,2,0,2;l2,0,3,5;m1,l1,l2,3;
      r2,3,0,2E3",100)
```

Note that the value of the source was converted from its RMS value to its normal value, directly in the entry field. Note that the first node in the description of each inductor is the node that has the dot in the drawing. Note also that the value of the mutual inductance was entered in henries. We press ENTER. We see on the screen the status messages indicating that the simulation runs. When it finishes, 'Done' appears. We are asked for the real average power absorbed. In alternating current analysis, the variables in which Symbulator gives us the absorbed power begin with `s`. As theory teaches us, the average power is half the maximum power. Since we are asked for the real power, we use the `real` command of the calculator. To find the answers, we ask for `real(se1)/2` and obtain −10.4 watts; we ask for `real(sr1)/2` and obtain 5.63 watts; we ask for `real(sr2)/2` and obtain 4.77 watts. These are the correct answers, rounded to three figures.

Problem Nº 033

Problem: If $\omega = 100$ rad/s, find the real average power: a) delivered to the 10 Ω load, b) to the 20 Ω load, and c) delivered by the source.

Solution:

Here is the description I used:

```
sq\ac("e1,1,0,100*√2;l1,1,2,1;l2,2,0,1;l3,3,0,1;r1,3,0,10;
      m1,l1,l3,.5*√(1*1);l4,4,0,1;r2,4,0,20;
      m2,l2,l4,.2*√(1*1)",100)
```

Note the following:

In a rigorous treatment, in Symbulator, the normal value of the source must be entered and not its RMS value. For that reason, we multiply the RMS value by the square root of two, to convert it to its normal value. It is perfectly valid to do this multiplication directly in the entry field.

The first node in the description of each inductor is the node that has the dot in the drawing.

The value of the mutual inductance was converted to henries, directly in the entry field. Now, this problem is not the best to illustrate this conversion, because obviously $.5\sqrt{1 \cdot 1}$ is equal to .5. Even though placing the conversion is unnecessary in this problem, in other cases it will not be so. For that reason, I placed the conversion to illustrate the process.

The circuit of the example is composed of three small circuits. These three circuits are coupled by mutual inductances. However, each one of them has a reference node. These three reference nodes must be node 0. It is necessary that the three share the same reference node, even though in the drawing the nodes are not physically connected.

We press ENTER. We see on the screen the status messages indicating that the simulation runs. When it finishes, 'Done' appears. As in the previous problem, we are asked for the real average power absorbed. We ask for `real(sr1)/2` and obtain .842 watts; we ask for `real(sr2)/2` and obtain .262 watts; we ask for `real(se1)/2` and obtain -1.104 watts. These are the correct answers (rounded).

Problem Nº 034

Problem: Find I_L in the circuit shown.

Solution:

This is a very particular problem, and for that reason I saved it for last. In it, something is presented that Symbulator does not handle by itself: coupled inductors whose values are given in Ω. Before solving this problem, we have to take the circuit to a form that Symbulator can handle. We have two alternatives.

The first alternative is to simulate using impedances. How? By replacing the mutual inductances with an equivalent made of impedances. We take advantage of the fact that the coupled inductors are working as a linear transformer, to replace them with their T equivalent. For a linear transformer with values L_1, L_2 and M, the T equivalent is obtained by making a T with elements of value $L_1 - M$ on one side, $L_2 - M$ on the other side, and M as the base of the T. In the case of our simulation, this T will be made with impedances, and there will be no inductors in the circuit. Below, the description I used to solve the problem with this alternative. Note that any radian frequency can be used, and any order of the impedance nodes:

```
sq\ac("e1,1,0,100;r1,1,2,-5i;r2,2,3,85i;rm1,2,0,15i;
      r3,3,4,85i;r4,4,5,-5i;rm2,4,0,15i;rL,5,0,5",x)
```

The second alternative is to simulate using inductors. How? Since the problem has not specified a value for the radian frequency, we invent any value. I used a unit value: 1 rad/s. Then, I convert all the values of the inductances and mutual inductances from Ω to H, using the radian frequency. If the frequency is unity, the value in ohms will be the same value in henries (without the imaginary operator, obviously). Below, the description to be used to solve the problem with this alternative. Note that the order of the inductor nodes was chosen carefully, to comply with the dot convention:

```
sq\ac("e1,1,0,100;l1,1,0,10;l2,2,0,100;m1,l1,l2,15;l3,2,0,100;
      l4,3,0,10;m2,l3,l4,15;rL,3,0,5",1)
```

In both cases, I called the 5 Ω resistor `rL`. Choose whichever alternative you prefer. We press ENTER. We see on the screen the status messages indicating that the simulation runs. When it finishes, 'Done' appears. We ask for `absang(irL)` and obtain $1.26\angle -60.2°$ amperes (rounded), which is the correct answer.

In the previous examples, we have seen how to simulate mutual inductances in Symbulator, in different types of problems. Let us now learn about a new element.

6.12 Ideal transformer

An ideal transformer is an idealized model of unity coupling of a physical transformer. It is assumed to have no losses.

6.12.1 Notation

In the case of the ideal transformer, the minimum information necessary to describe it completely is the following:

Identification. The letter that identifies an ideal transformer is `t`.

Name. It must be given any name that begins with `t`.

Nodes. An ideal transformer has four connection points, two upper and two lower. For a simulation in Symbulator, the two lower ones are always assumed connected to the reference node. The upper ones must each be connected to a distinct node. It is necessary to give the simulator the names of these two upper nodes to which the transformer is connected.

Turns ratio. An ideal transformer has a turns ratio of one side with respect to the other. This value must be given to Symbulator in the form of two numbers, one for each side.

When a transformer has the dots of both sides at the upper nodes, or both dots at the lower ends, the two numbers of the turns ratio must have the same sign, for example both positive. On the contrary, when the transformer has the dot of one side at the upper node but the dot of the other side at the lower node, the two numbers of the turns ratio must have opposite signs, for example one positive and one negative.

So, ideal transformers are defined as shown below:

```
tname, node 1, node 2, turns 1, turns 2
```

The description of the ideal transformer always has 5 terms, so there will never be a need to add a zero at the end.

In direct current analysis, Symbulator ignores ideal transformers.

6.12.2 Related answers

In the case of an ideal transformer, two related answers are given: the current entering the transformer through the first node, and the current entering the transformer through the second node. Remember that these two nodes are the upper ones, since the lower one on both sides is the reference node. These currents are stored in variables called `inameNODE`, where `name` is the name of the ideal transformer, and `NODE` is the name of the node. So, for an ideal transformer called `t1`, connected to nodes 1 and 2, the related answers will be `it11` and `it12`. Note that both currents are considered as entering the transformer through the upper nodes. Therefore, if one of them is worth, for example, −5 amperes, it means that −5 amperes are entering, that is, 5 amperes are leaving.

It is important to note that if either of the ends of the ideal transformer is an open circuit, both currents will be null.

Let us see some examples.

Problem Nº 035

Problem: For the circuit shown, find the RMS value of I_1, I_2, I_3, and the real average powers consumed in the three resistors.

Solution:

Below, the description I used to solve the problem:

```
sq\ac("e1,1,0,100*√2,0;r1,1,2,25,0;t1,2,3,3,1;r2,3,4,2,0;
      t2,4,5,4,-3;r3,5,0,3,0",x)
```

Note the following:

The value of the source was converted from RMS value to normal value.

Any radian frequency can be used.

I filled with zeros the descriptions of those elements whose descriptions do not have 5 terms. If we forget to do this, we will get a syntax error.

In the case of transformer `t2`, since the dots are one up and one down, one of the two numbers of the turns ratio in the description was made negative.

We press ENTER. We see on the screen the status messages indicating that the simulation runs. When it finishes, 'Done' appears. To obtain the RMS value of I_1, we have three alternatives: ask for either `absang(-ie1/√2)`, `absang(ir1/√2)` or `absang(it12/√2)`, and we obtain $1.099\angle 0.^\circ$ amperes. To obtain the RMS value of I_2, we also have three alternatives: ask for either `absang(-it13/√2)`, `absang(ir2/√2)` or `absang(it24/√2)`, and we obtain $1.30\angle 0.^\circ$ amperes. For the RMS value of I_3, we have two alternatives: ask for either `absang(-it25/√2)` or `absang(ir3/√2)`, and we obtain $4.40\angle 180.^\circ$ amperes.

Remember that Symbulator will always define the two currents of an ideal transformer as entering the transformer through the two upper nodes, and the name of each one is `i` followed by the name of the transformer and the name of the node through which the current enters.

We are asked for the real average power absorbed by the three resistors. With `real(sr1)/2.` we obtain 30.2 watts; with `real(sr2)/2.` we obtain 21.7 watts; and with `real(sr3)/2.` we obtain 58.0 watts. These are the correct answers (rounded). We used `/2.` instead of `/2` because we wished to obtain answers in approximate format.

Let us see two similar problems, but first let us learn some new concepts about RMS values.

6.13 Problems with RMS values

In two problems that we have seen in this chapter, the value of the independent source was given to us as an RMS value. As answers, we were asked for the RMS currents and the average powers. Problems of this type, using RMS values of voltage and current, and average powers, are very frequent in alternating current analysis. These problems can receive two treatments in Symbulator: the rigorous and the informal.

6.13.1 Rigorous treatment

The two problems we saw before were solved with the rigorous treatment. This treatment assumes that the values of the independent sources are entered in normal form, and the voltages, currents and powers of the answers are also in normal form. Therefore:

When the problem gives us the value of a source in RMS, we must convert it to normal value when entering it in the description.

If we want the RMS values of the currents and voltages that were given as answers, we must divide the normal value by the square root of two to have it in RMS.

If we want the average values of the powers that were given as answers, we must divide the normal value by two to have it as an average.

6.13.2 Informal treatment

The two problems that we will see next will be solved with the informal treatment. This treatment is based on the fact that, if the values of all the independent sources are entered in RMS, then all the voltages and currents of the answers will be in RMS, and all the powers of the answers will be average values. Therefore:

When the problem gives us the value of a source in RMS, we must enter it as it is. If, on the contrary, it gives it to us as a normal value, we must convert it to RMS before entering it.

We will receive the currents and voltages of the answers as RMS values.

We will receive the powers of the answers as average values.

Let us now see two problems, solved with the informal treatment.

Problem Nº 036

Problem: For the circuit shown, find the real average powers consumed in the resistors.

Solution:

Below, the description I used to solve the problem, with the informal treatment:

```
sq\ac("e1,1,0,10,0;r1,1,2,1,0;t1,2,3,1,2;r2,3,4,4,0;
      r3,4,0,48,0;t2,4,5,1,5;r4,5,0,400,0",x)
```

Note that the value of the source was left in RMS. We press ENTER. We see on the screen the status messages indicating that the simulation runs. When it finishes, 'Done' appears. We could obtain the RMS value of any current instantly. However, the problem does not ask us for any current. We are asked for the real average power absorbed by the four resistors. Since we are using the informal method, the powers are already averages. With `real(sr1)` we obtain 4 watts; with `real(sr2)` we obtain 4 watts; with `real(sr3)` we obtain 3 watts; and with `real(sr4)` we obtain 9 watts. Note that there was no need to use `/2` because the answers were already averages.

Problem Nº 037

Problem: For the circuit shown, find the real average powers consumed by the 10 Ω resistors. Repeat connecting A and C, and B and D.

Solution:

Because of the vertical layout of this circuit's drawing, and the position of the source in it, choosing the reference node for each section of the circuit could be somewhat complicated. It is necessary to use the nodes on the right of the circuit (among them B and D) as the reference in each of the three circuits, because the two transformers must each have their two lower ends connected to the reference node. The nodes on the left, from bottom to top, I named C = 1, 2, 3 and A = 4. Below, the description I used to solve the first part of the problem, with the informal treatment:

```
sq\ac("r1,1,0,10,0;r2,1,0,50,0;t1,1,2,1,3;j1,2,3,1,0;
      t2,3,4,4,1;r3,4,0,50,0;r4,4,0,10,0",x)
```

The value of the source was left in RMS. We press ENTER. When the simulation finishes, 'Done' appears. The powers are already averages. With `real(sr1)` and ◇ ENTER we obtain 62.5 watts; and with `real(sr4)` and ◇ ENTER we obtain 111.1 watts. Below, the description I used to solve the second part of the problem, with the informal treatment:

```
sq\ac("r1,1,0,10,0;r2,1,0,50,0;t1,1,2,1,3;j1,2,3,1,0;
      t2,3,1,4,1;r3,1,0,50,0;r4,1,0,10,0",x)
```

Note that I did not use a short circuit to connect A with C, but instead replaced node 4 with node 1 in the description. There is no need to connect B with D, because both are the reference node. We press ENTER. When the simulation finishes, 'Done' appears. With `real(sr1)` and ◇ ENTER we obtain 1.736 watts; and with `real(sr4)` and ◇ ENTER we obtain the same value as `sr1`.

We have seen in these two problems the way in which the values of the sources in RMS can be used, with the informal treatment, to obtain answers directly in RMS, and the powers as averages.

Let us see a final example, of the `thevenin` tool applied to a circuit with ideal transformers.

Problem Nº 038

Problem: Find the Thévenin equivalent at terminals a and b for the network shown in the figure.

Solution:

Below, the description:

```
sq\thevenin("rx,a,0,20,0;t1,a,1,1,4;r1,1,2,60,0;
            e1,2,0,20*irx,0",a,0)
```

We press ENTER. We choose Passive and AC, and press ENTER. We enter `x`, or any other value, as the radian frequency and press ENTER. For `zth` we obtain 4 ohms, which is the correct value.

With this example, we have finished the chapter on numerical simulation of alternating current.

Chapter 7

Two-ports

7.1 Introduction

A two-port is a network that has two pairs of terminals. In Symbulator, two-ports can be simulated as circuit elements, assuming that:

- the network is composed only of linear elements,
- the network contains no independent sources (it may contain dependent sources),
- the two upper nodes of the terminals are connected to distinct nodes of the circuit,
- the two lower nodes of the terminals are considered connected to the reference node, 0.

Symbulator accepts two-ports with parameters of 6 different types:

- admittance parameters
- impedance parameters
- hybrid parameters
- gain (inverse hybrid) parameters
- transmission parameters
- inverse transmission parameters

There are three procedures related to two-ports that can be done in Symbulator:

1. Simulate circuits that contain two-ports. For this, two-ports are used as circuit elements.
2. Find the gains in a two-port. For this, the `gain` tool is used after a circuit simulation.
3. Find the equivalent two-port of a network. For this, the `port` tool is used with the description of a network.

In this chapter we will learn everything related to two-ports in Symbulator.

7.2 Notation

In the case of two-ports, the minimum information necessary to describe them completely is the following:

Identification. The letters that identify the different two-ports are:

- impedance parameters, the letter `z`
- admittance parameters, the letter `y`
- hybrid parameters, the letter `h`
- gain parameters, the letter `g`
- transmission parameters, the letter `a`
- inverse transmission parameters, the letter `b`

Name. Any name is valid that begins with the letter identifying the two-port you want to describe, and that is not a reserved variable, like those listed in the User's Manual and in section 1.9.4 of this book. Example: `zc` and `yc` are reserved variables that cannot be used as names.

Nodes. A two-port has four connection points, two upper and two lower. For a simulation in Symbulator, the two lower ones are always assumed connected to the reference node. The upper ones must be connected to nodes distinct from each other. It is only necessary to give the simulator the names of these two upper nodes to which the two-port is connected.

So, two-ports are defined as shown below:

```
lettername, node 1, node 2
```

As necessary, one or two zeros are added at the end.

Value of the parameters. Two-ports are unique elements, because they require something no other element requires in Symbulator: variables stored previous to the simulation. Each two-port that is simulated as part of a circuit requires that the values of its 4 parameters be stored in 4 variables, one for each parameter, in the following form:

- parameter 11, in the variable `lettername11`
- parameter 12, in the variable `lettername12`
- parameter 21, in the variable `lettername21`
- parameter 22, in the variable `lettername22`

Take care to enter these values in whole units, and not in their multiples. This means they must not be entered in milli (m), micro (μ), kilo (k), mega (M), etc.

7.3 Related answers

In the case of a two-port, two related answers are given. They are two currents: the current entering the two-port through the first node, and the current entering through the second node. Remember that these two nodes are the upper ones, since the lower one on both sides is the reference node. These currents are stored in variables called `inameNODE`, that is, `i` followed by the name of the two-port and the name of the node. So, for a two-port called

yp, connected to nodes 1 and 2, the related answers will be iyp1 and iyp2. Remember that both currents are considered as entering the two-port through the upper nodes.

Let us see our first two-port problem.

Problem Nº 039

Problem: In the following circuit, the parameters of the two-port shown are $z_{11} = 20/3$, $z_{12} = 1/3$, $z_{21} = 2500/3$, $z_{22} = 200/3$. *Find: a) the voltage at the load node and the currents entering the two-port at both ends, b) the input impedance and the current, voltage and power gains of the two-port, c) the output impedance of the network, seen by the load, and d) the equivalents of this two-port in the other five types of parameters.*

Solution:

This problem, specially designed to inaugurate our experience with two-ports, will be solved in four parts.

Question a) will be answered by simulating the circuit using a direct current analysis. We will call the two-port zp. First, we define the variables that describe the parameters of the two-port:

```
20/3→zp11:1/3→zp12:2500/3→zp21:200/3→zp22
```

After entering this line and pressing ENTER, we will see in the history area the value of the last of these parameters as the answer to our command. What interests us is that the values were stored in the variables. Note that we used the separator : and the STO key. Once the two-port parameter variables are stored, we proceed to order the simulation. Below, the description we use:

```
sq\dc("e1,1,0,1;r1,1,2,2;r2,3,0,20;zp,2,3,0")
```

Note that, since the description of the two-port only requires three terms, we fill with a zero to keep the symmetry. We press ENTER. After the status messages, 'Done' appears. The problem has asked us for the voltage at the load node and the currents entering the two-port at both ends. We obtain these values like this: the voltage at the load node we obtain with v3, and it is worth 2500/71 volts; the current entering the two-port on the left we obtain with izp2, and it is worth 13/71 amperes; and the current entering the two-port on the right we obtain with izp3, and it is worth −125/71 amperes. The negative sign of this last current tells us that the current is really leaving the two-port on the right, not entering.

We have learned that a two-port can be easily simulated in Symbulator as one more circuit element. To answer question b), we will use a new tool.

7.4 The gain tool

This tool helps us find the current, voltage and power gains, and the input impedance of a network. This network may or may not contain two-ports. Let us learn to use this tool at the same time we answer the second question. The gain tool is executed like this:

```
sq\gain()
```

It takes no input arguments between the parentheses. However, when executing it, a form opens in which we must enter four input values:

The voltage at the input terminal node. In the case of our problem, this voltage is v2, because this is the voltage at the input node of the two-port, the one toward the source.

The current entering through the input terminal. In the case of our problem, this current is `izp2`, because this is the current entering the two-port on the input side, that is, the side toward the source.

The voltage at the output terminal node. In the case of our problem, this voltage is `v3`, because this is the voltage at the output node of the two-port, the one toward the load.

The current entering through the output terminal. In the case of our problem, this current is `izp3`, because this is the current entering the two-port on the output side, that is, the side toward the load. Note that saying output current does not mean a current that is leaving. The current must be defined as entering the two-port, but on the output side.

We press ENTER, twice if necessary. Next a screen appears, which works in a way similar to that of the `thevenin` tool, in which we must press ENTER as many times as there are values we seek.

We see on the screen that the voltage gain, called A_v or G_v depending on the textbook, is worth 500/9; the current gain, called A_i or G_i, is worth $-125/13$; the power gain, called A_p or G_p, is worth 62500/117; and the input impedance, called Z_i or R_i, is worth 45/13. Besides showing these values on screen, the tool also stores them in variables with the names `av`, `ai`, `ap`, and `zi`, respectively. All these values, being stored in memory, can be requested later, even in the form of approximate values if desired.

We have learned the use of the `gain` tool. Let us now answer the next question.

7.5 Output impedance

Question c) requires some theoretical knowledge. Theory teaches us that the output impedance of the network, seen by the load, is the equivalent impedance of the network, from the terminals where the load is connected, when the load has been removed and the independent sources have been killed. To kill an independent voltage source is to turn it into a short circuit. To kill an independent current source is to turn it into an open circuit. We already defined the parameters of this two-port when we answered the first question, and they are in memory. Therefore, there is no need to define them again. Below, the description we use:

```
sq\thevenin("e1,1,0,0;r1,1,2,2;zp,2,3,0",3,0)
```

Note that the load has been removed from the description, and that the source has been killed, that is, its value has been made null. This is the same as having turned it into a short circuit. We press ENTER, choose Passive and DC, and press ENTER. On the answer screen, it appears that `zth` is 450/13 ohms, which is the correct output impedance.

We have learned how to find the output impedance of a network. We have also seen that the `thevenin` tool works with two-ports too. Let us now answer the last question, using a new tool.

7.6 The port tool

This tool gives us the parameters of the equivalent two-port of any network with two terminals. Let us learn to use this tool, at the same time we answer the fourth question. The `port` tool is executed like this:

```
sq\port(description of the network,node1,node2)
```

Three input arguments are given between the parentheses:

The description of the two-terminal network whose two-port equivalent you want to find. This network must be a circuit composed only of passive elements, and it cannot contain

independent sources. This network may be composed, say, of resistances only. It may also be composed of one or more two-ports. Or of a mixture of resistive elements and two-ports. Since this network can contain other two-ports, the `port` tool is a powerful ally for finding the equivalent of a given two-port in other, distinct parameters, and for reducing a set of several two-ports (in series, in parallel or in combination) to a single equivalent two-port.

The upper node of the input terminal of the network.

The upper node of the output terminal of the network.

It is assumed that the lower node of the two terminals is the reference node.

Once we have written the line shown, we press ENTER. When it executes, a form opens in which we must make some selections. The three selections are:

The type of parameter we want to find. The options are `z`, `y`, `h`, `g`, `a` and `b`, whose meanings we already listed in section 7.2.

The name we wish to give the two-port, without including the letter that identifies the type of two-port. For example, if we want the two-port to be called `zp1`, the name we must enter here is only `p1`, since the `z` we have already chosen above. We can use any name, except those that, when combined with the two-port letter we have chosen, turn out to be the name of a reserved variable. For example, if we choose a two-port of type `z` or `y`, we cannot use a name like `c`, because the complete name of the two-port would be `zc` or `yc`, which are both reserved variables, as we pointed out in section 1.9.4. There is a large number of reserved variables that begin with `z` and with `y`. A complete list can be found in the User's Manual. As a general recommendation, use a letter or a number as the name, and let that letter not be the letter `c`.

The type of analysis we want. We can choose between direct current analysis DC, alternating current AC and frequency domain FD.

Next, a simulation of the network is executed. Then an answer screen appears that works in a way similar to the previous ones. We must press ENTER four times to see the four parameters sought, presented on the screen.

These parameters will also be stored in the memory of the calculator, in variables with the names that correspond to the name of the two-port, according to the naming convention we explained in section 7.2.

With this knowledge, let us now see how we can answer question d) using the `port` tool. The question asks us to find the equivalents of the two-port in all the other types of parameters. Below, the solution to this question.

Let us first find the equivalent in admittance parameters of the given two-port. Enter the following command line in the entry line:

```
sq\port("zp,1,2,0",1,2)
```

This line orders the use of the `port` tool to find the equivalent of the network described, which is the two-port `zp`, whose impedance parameters are already known and are called `zp11`, `zp12`, `zp21` and `zp22`. These parameters are already stored in memory, from when we answered the first question. If you have any doubt about whether these parameters are stored in memory or not, you can use the Var-Link screen to verify. If these parameters were deleted, it is necessary to enter them again, as shown in the earlier question. The fact is that the parameters of a two-port must be stored before executing any simulation involving that two-port.

The two-port is connected between nodes 1 and 2, and this is indicated in the command line. Although it may seem unnecessary in this case, the two end nodes of the network to be reduced must be specified, because this network will not always be composed of a single element, as we will see later.

So, we press ENTER. We specify `y` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown on the screen: the four admittance parameters. This is the correct equivalent. Remember that these parameters, in addition to appearing on screen, are also stored in variables with the corresponding names. Let us now find another equivalent of the two-port in another type of parameters.

To find the equivalent in hybrid parameters of the given two-port, the procedure is identical. Just for didactic purposes, we will now use as the starting two-port the recently found two-port `yp` whose parameters are already stored in memory, instead of the two-port `zp` we used in the previous line. Enter the following command line:

```
sq\port("yp,1,2,0",1,2)
```

We press ENTER. We specify `h` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown.

To find the equivalent in gain parameters of the given two-port, the procedure is the same. We will use as the starting two-port the two-port `hp`, whose parameters are in memory. Enter the following command line:

```
sq\port("hp,1,2,0",1,2)
```

We press ENTER. We specify `g` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown.

Let us find the equivalent in transmission parameters of the two-port, using as the starting two-port the two-port `gp`. Enter the following command line:

```
sq\port("gp,1,2,0",1,2)
```

We press ENTER. We specify `a` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown.

Let us find the equivalent in inverse transmission parameters of the two-port, using as the starting two-port the two-port `ap`. Here we find a variant in the procedure. When we solved question b), a moment ago, the `gain` tool generated a variable called `ap`, which contained the power gain. This variable is still in memory. If we forget this, and try to execute the tool using `ap` as the name of the two-port, we will receive an error from Symbulator telling us that the name we have assigned to element #1 of our circuit description is not appropriate, and that we should please correct this description. The name is inappropriate because it is the name of a variable in use.

Therefore, we must delete the variable before executing the tool. We can put both commands in a single line, using the separator. For example, enter the following command line:

```
DelVar ap:sq\port("ap,1,2,0",1,2)
```

Since the variable `ap` has been deleted, now we can use this variable as the name of the two-port. We press ENTER. We specify `b` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown.

So, we have found the five equivalents of the two-port that was given to us.

7.7 Verification

But how could we verify that these two-ports are the correct equivalents of our original two-port? Simple. Let us find the equivalent in impedance parameters, that is, in the original

type, using as the starting two-port the two-port `bp`, which was the last equivalent found. Any error generated along the way will impact our final answer. So, if we now obtain the correct equivalent, it will mean that all the intermediate equivalents were correct. Enter the following command line:

```
sq\port("bp,1,2,0",1,2)
```

We press ENTER. We specify `z` as the type of two-port desired, `p` as the name and DC as the analysis. The answers are shown, and we can see that these impedance parameters coincide perfectly with those the problem gave us as initial data. This means that all the equivalents we have found are correct. Let us see more solved problems, as practice to master the use of two-ports in Symbulator.

Problem Nº 040

Problem: The parameters of the two-port shown are $z_{11} = 20$, $z_{12} = 3$, $z_{21} = 100$, $z_{22} = 5$. Find: a) the input impedance and the current, voltage and power gains of the two-port, b) the output impedance of the network, seen by the load.

Solution: Below, the description we use for the first question:

```
20.→zp11:3.→zp12:100.→zp21:5.→zp22:
sq\dc("j1,0,1,is;r1,1,0,50.;zp,1,2,0;r2,2,0,100."):sq\gain()
```

In this line are the definition of the parameters, the command to run the simulation and the command to run the `gain` tool. We press ENTER. After the status messages, the tool's form appears, in which we enter `v1`, `izp1`, `v2`, `izp2`. We press ENTER. We receive on the screen, and in memory, the answers: `ai` $= -.952$, `av` $= 5.56$, `ap` $= 5.29$, `zi` $= 17.1$.

Below, the description for the second question:

```
sq\thevenin("r1,1,0,50.;zp,1,2,0",2,0)
```

Note that the current source was removed. This way of killing it is equivalent to having entered it with a null value, and has the advantage of saving the simulator trouble. We press ENTER. We select Passive and DC, and press ENTER. We obtain: `zth` $= .714$. These are the correct answers.

Problem Nº 041

Problem: Find the equivalent of the following network, in admittance parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,2,10.;r2,1,0,5.;r3,2,0,20.",1,2)
```

We press ENTER. We select `y`, some name and DC, and press ENTER. We obtain the following admittance parameters: .3, $-.1$, $-.1$, .15.

Problem Nº 042

Problem: Find the equivalent of the following network, in admittance parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,2,10.;r2,2,3,5;r3,1,3,20;r4,2,0,40",1,3)
```

We press ENTER. We select y, some name and DC, and press ENTER. We obtain the following admittance parameters: .1192, −.1115, −.1115, .1269.

Problem Nº 043

Problem: Find the equivalent of the following network, in admittance parameters.
Solution: Below, the description we use:

```
sq\port("s1,x,1,0;j1,1,0,.2*v2;r1,1,0,10.;j2,1,0,.5*is1;
        r2,2,1,5.",x,2)
```

Note that we introduced a short circuit with the purpose of using its current as the control current for source j2. We press ENTER. We select y, some name and DC, and press ENTER. We obtain the following admittance parameters: .6, 0., −.2, .2.

Problem Nº 044

Problem: The following circuit is the linear equivalent of a transistor in the common-emitter configuration with resistive feedback between the collector and the base. Find the equivalent, in admittance parameters.
Solution: Below, the description we use:

```
sq\port("r1,1,0,5E2;r2,1,2,2E3;j1,2,0,.0395*v1;r3,2,0,1E4",1,2)
```

We press ENTER. We select y, some name and DC, and press ENTER. We obtain the following admittance parameters: .0025, −5.E−4, .039, 6.E−4.

Problem Nº 045

Problem: Find the equivalent of the following network, in impedance parameters.
Solution: Below, the description we use:

```
sq\port("r1,1,3,20.;r2,3,2,50.;r3,3,0,25.",1,2)
```

We press ENTER. We select z, some name and DC, and press ENTER. We obtain the following impedance parameters: 45., 25., 25., 75.

Problem Nº 046

Problem: Find the equivalent of the following network, in impedance parameters.
Solution: Below, the description we use:

```
sq\port("r1,1,0,40.;r2,1,3,20.;r3,3,0,25.;r4,3,2,50.",1,2)
```

We press ENTER. We select z, some name and DC, and press ENTER. We obtain the following impedance parameters: 21.2, 11.8, 11.8, 67.6.

Problem Nº 047

Problem: Find the equivalent of the following network, in impedance parameters.
Solution: Below, the description we use:

```
sq\port("r1,1,3,20.;r2,3,2,50.;r3,3,4,25.;e1,4,0,.5*v2",1,2)
```

We press ENTER. We select `z`, some name and DC, and press ENTER. We obtain the following impedance parameters: 70., 100., 50., 150.

Problem Nº 048

Problem: This problem has no figure. Find the input impedance and the current, voltage and power gains of a two-port, with impedance parameters $z_{11} = 4$, $z_{12} = 1.5$, $z_{21} = 10$ and $z_{22} = 3$, if it has an input source V_s at its input in series with a 5 Ω resistance, while the load is 2 Ω.

Solution: Below, the command we use:

```
4→zp11:1.5→zp12:10→zp21:3→zp22:
sq\dc("e1,1,0,vs;r1,1,2,5;r2,3,0,2;zp,2,3,0"):sq\gain()
```

In this line are the definition of the parameters, the command to run the simulation and the command to run the `gain` tool. We press ENTER. After the status messages, the tool's form appears, in which we enter `v2`, `izp2`, `v3`, `izp3`. We press ENTER. We receive on the screen and in memory the answers: `ai` $= -2.$, `av` $= 4.$, `ap` $= 8.$, `zi` $= 1.$

Problem Nº 049

Problem: Find the equivalent of the following network, in hybrid parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,3,1.;r2,3,2,6.;r3,3,0,4.",1,2)
```

We press ENTER. We select `h`, some name and DC, and press ENTER. We obtain the following hybrid parameters: 3.4, .4, −.4, .1.

Problem Nº 050

Problem: The following circuit is the equivalent of a high-frequency transistor. Find the equivalent, in impedance parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,0,1E5;ca,1,0,5E-12;cb,1,2,1E-12;
        j1,2,0,.01*v1;r2,2,0,1E4",1,2)
```

We press ENTER. We select `z`, some name and AC, and press ENTER. As the frequency, we enter `1E8`, and press ENTER. We obtain the following impedance parameters: $89.7 - 98.4i$, $94.1 - 4.34i$, $528. + 9401.i$, $564. - 35.5i$. Since these values are stored in memory, we can use the `absang` tool to see them in polar form: $133.1\angle{-47.6°}$, $94.2\angle{-2.64°}$, $9416.\angle{86.8°}$, $565.\angle{-3.60°}$.

Problem Nº 051

Problem: Find the equivalent of the following network, in impedance parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,3,8.;e1,3,0,.1*v2;j1,2,0,.05*v1;r2,2,0,12.",1,2)
```

We press ENTER. We select `z`, some name and DC, and press ENTER. We obtain the following impedance parameters: 7.55, 1.13, −4.53, 11.3.

Problem Nº 052

Problem: The following circuit is the equivalent of a high-frequency transistor. Find the equivalent, in hybrid parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,0,2E3;ca,1,0,5E-12;cb,1,2,2.5E-12;
        j2,2,0,.005*v1",1,2)
```

We press ENTER. We select `h`, some name and AC, and press ENTER. As the frequency, we enter `1E8`, and press ENTER. We obtain the following hybrid parameters: $615. - 923.i$, $.231 + .154i$, $2.85 - 4.77i$, $.00119 + 9.62\text{E}{-4}i$. Since these values are stored in memory, we can use the `absang` tool to see them in polar form: $1109\angle{-56.3}°$, $.277\angle 33.7°$, $5.55\angle{-59.2}°$, $.00153\angle 38.9°$.

Problem Nº 053

Problem: Find a) the equivalent of the following network, in hybrid parameters, and b) the output impedance of the network, seen from the perspective of the load, if the input contains V_s in series with $R_s = 200\ \Omega$.

Solution:

Below, the description we use for the first part:

```
sq\port("r1,1,0,1E3;j1,0,1,1E-5*v2;e1,0,3,100*v1;
        r2,3,2,1E4",1,2)
```

We press ENTER. We select `h`, some name and DC, and press ENTER. We obtain the following hybrid parameters: 1000., .01, 10., 2.E−4.

Below, the description we use for the second part:

```
sq\thevenin("rs,0,1,200;r1,1,0,1E3;j1,0,1,1E-5*v2;
            e1,0,3,100*v1;r2,3,2,1E4",2,0)
```

Note that we have added only the resistance `rs`. The source V_s was not added because, to find the output impedance, the voltage source must be dead, and a dead voltage source is simply a short circuit. It is not necessary to place this short circuit in series with the resistance, because placing only the resistance has the same effect. We press ENTER. We select Passive and DC, and press ENTER. We see that `zth` is 8571.

Problem Nº 054

Problem: Find the equivalent of the following network, in transmission parameters.

Solution: Below, the description we use:

```
sq\port("r1,1,3,2.;r2,3,2,4.;r3,3,0,10.",1,2)
```

We press ENTER. We select `a`, some name and DC, and press ENTER. We obtain the following transmission parameters: 1.2, 6.8, .1, 1.4.

Problem Nº 055

Problem: Find the equivalent of the following network, in transmission parameters.

Solution: Note that the network, in this case, is composed of two groups of resistances in a T arrangement. Below, the description we use:

```
sq\port("r1,1,2,2.;r2,2,0,10.;r3,2,3,4.;r4,3,4,4.;
        r5,4,0,20.;r6,4,5,8.",1,5)
```

We press ENTER. We select `a`, some name and DC, and press ENTER. We obtain the following transmission parameters: 1.78, 25.84, .19, 3.32. This is an example of how, with the `port` tool, a network that would usually have taken two two-ports can be reduced to a single two-port.

Problem Nº 056

Problem: Find the equivalent of the following network, in transmission parameters.
Solution: Below, the description we use:

```
sq\port("r1,1,3,5.;j1,3,0,.1*v2;e1,2,3,.3*v1;r2,2,0,4.",1,2)
```

We press ENTER. We select `a`, some name and DC, and press ENTER. We obtain the following transmission parameters: 2.12, 3.85, .35, 1.

Chapter 8

Symbolic simulation of alternating current circuits

8.1 General concepts

Symbolic simulation in alternating current uses the same techniques as symbolic simulation in direct current, with only two exceptions.

8.2 The cSolve command

To solve complex equations, the calculator has the `cSolve` command, which works identically to the `solve` command. The `cSolve` command is a powerful ally in obtaining numerical answers from a symbolic simulation, when the equations involved are complex. In other cases, when the equations used are real, the `solve` command can be used.

8.3 Complex unknowns

To find the numerical value of a complex unknown, we must find two numerical values: a real value and an imaginary value. Therefore, a complex unknown is really two unknowns: one real and one imaginary.

Likewise, when we have a complex value as a known given value, we really have two known values: a real value and an imaginary value. In our first problem of this chapter, we will see how to find the numerical value of two symbolic unknowns, using only one known complex value.

Problem Nº 057

Problem: This problem has no figure. A circuit consists of a resistance in series with a capacitance. Find their values, if when applying a voltage of 240 V RMS at 200 Hz, the current taken from the source is $1.2 + 1.6i$*. Note that this current is not in RMS.*

Solution: First, we simulate the circuit according to the rigorous treatment:

```
sq\ac("e,1,0,240.*√2;r,1,2,r;c,2,0,c",2*π*200.)
```

Since there is only one element of each type, we have taken the opportunity to name them `e`, `r` and `c`, which is a licit and very elegant notation. If there were more than one element of each type, they would have to be named more completely. We press ENTER. After the status messages, 'Done' appears. Next, we order the calculator to find the values of `r` and `c` that satisfy both the real part and the imaginary part of the current that the problem gave us as a known value:

```
solve(real(ir)=1.2 and imag(ir)=1.6,{r,c})
```

This command line is very interesting. Let us see what it means. We have just run a simulation, using `r` and `c` as symbolic values. Therefore, all the answers, including the current through the elements, are expressed as functions of `r` and `c`. These answers have a real part and an imaginary part. The current that the circuit gave us as a known value also has a real part and an imaginary part. What the command line is ordering the calculator is: "Use the `solve` command to find which values of `r` and `c` satisfy the following two conditions: that the real part of the current equal 1.2 amperes, and the imaginary part of the current equal 1.6 amperes." We use the `solve` command and not the `cSolve` command, because the two equations involved are given in terms of real numbers. Why? The real part of a complex number is a real number. The imaginary part of a complex number, once it has been extracted with the `imag` command, is also a real number. Therefore, the two equations given are composed of real numbers, and the `solve` command can be used to solve them.

We press ENTER. The calculator gives us two pairs of answers. One of these pairs has a negative value of `r`. The other pair has a positive value of `r`. Obviously, the correct pair of answers is the one with the positive value of `r`. These values, rounded to three figures, are: `r` = 102. and `c` = 5.86E−6. The value of `r` is given in Ω, and that of `c` in F. If we want to obtain only the correct answers, we indicate that $r > 0$, with this line:

```
solve(real(ir)=1.2 and imag(ir)=1.6,{r,c})|r>0
```

Problem Nº 058

Problem: If $\omega = 500$ *rad/s and* $I_L = 2.5\angle 40°$*, find* $v_s(t)$.
Solution: We will use a symbolic simulation and then the `solves` tool:

```
sq\ac("es,1,0,vs;r1,1,2,10;e2,2,3,(25∠-30°);r2,3,0,25;
      l1,3,0,.02",500):sq\solves()
```

We press ENTER. After the status messages, 'Done' appears. Next, the tool executes. We enter `il1=(2.5∠40°)` as the equation and `vs` as the unknown. We obtain the value of `vs` on the screen, but it is in rectangular form and this way it is not useful to us. We press ENTER to return to the home screen. To see the value in polar form, we use `sq\absang(vs)` in the entry line. We obtain $35.5\angle 58.9°$. Therefore, the value of $v_s(t)$ is $35.5\cos(500t + 58.9°)$ V.

Problem Nº 059

Problem: This problem has no figure. A phasor current of $1\angle 0°$ *A flows through the series combination of 1* Ω*, 1 H and 1 F. At what frequency is the amplitude of the voltage across the network twice the amplitude of the voltage across the resistor?*
Solution: First, we simulate the circuit:

```
sq\ac("j,0,1,1;r,1,2,1;l,2,3,1;c,3,0,1",ω)
```

Since there is only one element of each type, we have taken the opportunity to name them with a single letter. We press ENTER. After the status messages, 'Done' appears. In this problem, it is possible to find more than one answer, so the most adequate thing is to use the `solve` command instead of the `solves` tool, because the command will give us the multiple answers it finds, while the tool will only give us one answer. So, we use the command:

```
solve(abs(v1)=2*abs(vr),ω)|ω>0
```

Note that we suppressed the possibility of obtaining negative frequencies, with the conditional at the end of the command line. We will obtain two answers, and both are valid. If we press ENTER, we obtain the answers in exact form: $\omega = (\sqrt{7} - \sqrt{3})/2$ or $\omega = (\sqrt{7} + \sqrt{3})/2$. If we press ◇ ENTER, we obtain the answers in approximate form: $\omega = .457$ or $\omega = 2.19$.

Problem Nº 060

Problem: This problem has no figure. A 20 mH inductor and a 30 Ω resistor are in parallel. Find the frequency at which: a) $\text{abs}(Z_{in}) = 25\ \Omega$, *b)* $\text{ang}(Z_{in}) = 25°$, *c)* $\text{real}(Z_{in}) = 25\ \Omega$, *d)* $\text{imag}(Z_{in}) = 10\ \Omega$.

Solution: We use the `thevenin` tool:

```
sq\thevenin("r,1,0,30;l,1,0,.02",1,0)
```

We press ENTER. Next, the tool executes. We select Passive and AC, and press ENTER. We enter a symbolic frequency `w` and press ENTER. After the status messages, `zth` appears as a function of `w`. We press ENTER to return to the home screen. To answer the questions, we use:

```
solve(abs(zth)=25,w)|w>0
```

We obtain `w` = 2261.

```
solve(angle(zth)=25°,w)|w>0
```

We obtain `w` = 3217.

```
solve(real(zth)=25,w)|w>0
```

We obtain `w` = 3354.

```
solve(imag(zth)=10,w)|w>0
```

We obtain `w` = 3927. or `w` = 572.9.

Problem Nº 061

Problem: In the network shown, find the frequency at which a) $R_{in} = 550\ \Omega$, *b)* $X_{in} = 50\ \Omega$, *c)* $G_{in} = 1.8$ *mS, d)* $B_{in} = -150\ \mu S$. *Remember that* $Z = R + iX$ *and* $Y = G + iB$.

Solution: We use the `thevenin` tool:

```
sq\thevenin("r1,1,2,500;r2,2,0,100;l1,2,0,1E-3",1,0)
```

We press ENTER. Next, the tool executes. We select Passive and AC, and press ENTER. We enter a symbolic frequency `w` and press ENTER. After the status messages, `zth` appears as a function of `w`. We press ENTER to return to the home screen. To answer the questions, we use:

```
solve(real(zth)=550,w)|w>0
```

We obtain `w` = 100000.

```
solve(imag(zth)=50,w)|w>0
```

We obtain `w` = 100000.

```
solve(real(1/zth)=.0018,w)|w>0
```

We obtain `w` = 102062.

```
solve(imag(1/zth)=-1.5E-4,w)|w>0
```

We obtain `w` = 132953. or `w` = 52232.

Problem Nº 062

Problem: Find Y_{in} as a function of ω for the network shown. Also find the resonant frequency ω_0 and the value of Z_{in} at that resonant frequency.

Solution: Theory teaches us that the resonant frequency is that frequency at which the imaginary part of the input admittance equals zero. As we know, the input admittance is the inverse of the input impedance. We use the `thevenin` tool:

```
sq\thevenin("l,1,0,4.4E-3;r,0,1,1E4;c,1,2,1E-8;
            e,2,0,1E5*ir",1,0)
```

We press ENTER. Next, the tool executes. We select Passive and AC, and press ENTER. We enter a symbolic frequency `w` and press ENTER. After the status messages, `zth` appears as a function of `w`. We press ENTER to return to the home screen. To answer the questions, we use:

```
solve(imag(1/zth)=0,w)|w>0
```

With the command above these lines, we are asking the calculator: "Using the `solve` command, tell me how much `w` is worth when the imaginary part of the inverse of `zth` is zero, considering `w` as positive." We press ENTER. We obtain `w` = 45454.5 as the resonant frequency.

```
zth|ans(1)
```

With the command above these lines, we are asking the calculator: "Tell me how much `zth` is worth when the immediately previous answer holds." Placing `zth|ans(1)` is equivalent to placing `zth|w=45454.5`. The `ans(#)` command recalls the #th previous answer from the history area. We press ENTER. We obtain 9999.9. This answer can be rounded to 10 kΩ.

Problem Nº 063

Problem: For the network shown, find the resonant frequency ω_0 and the value of Z_{in} at that frequency.

Solution: We already know the technique:

```
sq\thevenin("r1,1,2,5;c,1,2,.01;r2,2,3,2;r3,3,0,5;l,3,0,.01",1,0)
```

We press ENTER. Next, the tool executes. We select Passive and AC, and press ENTER. We enter a symbolic frequency `w` and press ENTER. After the status messages, `zth` appears as a function of `w`. We press ENTER to return to the home screen. To answer the questions, we use:

```
solve(imag(1/zth)=0,w)|w>0
```

We press ENTER. We obtain `w` = 100. as the resonant frequency.

```
zth|ans(1)
```

We press ENTER. We obtain 2.38.

Chapter 9

Some new concepts

9.1 Problems with graphs

Sometimes it is necessary to build a graph with the answers. There are, basically, four types of graphs related to a Symbulator simulation:

1. The value of an answer obtained in an AC analysis, as a function of the frequency ω.
2. The value of an answer obtained in an FD analysis, as a function of the complex frequency s.
3. The value of an answer obtained in a TR analysis, as a function of time t.
4. The value of any answer, as a function of a symbolic circuit value.

Your calculator has extensive graphing capabilities. The detailed description of their diversity and use is beyond the purpose of this book; however, it is available in the User's Manual of the calculator.

In Chapter 12 we will see an example of the second type of graph using the `bode` tool, and an example of the third type of graph, using the `plot` tool.

Next, we will see an example of the first type of graph, using the calculator's own capabilities. The technique that will be shown is also applicable to the fourth type of graph.

Problem Nº 064

Problem: Graph abs(V_{out}) *against* ω *for the circuit shown in the figure. Cover a frequency interval from* $\omega = 800$ *to* $\omega = 1200$ *rad/s.*

Solution: We will simply use the simulation and the graphing capabilities of the calculator:

```
sq\ac("e,1,0,1;l,1,2,1;r,2,3,10;c,3,0,1E-6",x)
```

A frequency `x` was used with the intention that the output voltage remain a function of the variable `x`. The purpose of this is to satisfy the requirement of the calculator, which demands that the functions `y#` to be graphed, where # is a number, be given as functions of `x`, where `x` is the independent variable.

We press ENTER. The simulation takes some time. Finally 'Done' appears. The next step is to make the value of abs(V_{out}) appear in the history area. We can use either `v3` or `vc` as V_{out}:

```
abs(v3)
```

We press ENTER. We obtain a function of `x`, which we will store in any of the functions `y#(x)`, where # is a number:

```
ans(1)→y1(x)
```

We press ENTER and the function is stored in the specified variable. Pressing ◇ Y= we can verify that this is so. Now we can specify the range of frequencies for which we wish to graph. We press ◇ WINDOW and specify for `xmin` a value of 800 and for `xmax` a value of 1200. We press ◇ GRAPH and see the graph. We can press F2 A on the TI-89 to apply the ZoomFit command, which adjusts the height of the graph to the best value.

We have learned how to graph an answer as a function of frequency, using the calculator's own capabilities.

9.2 Problems with multiple unknowns

To solve a circuit with symbolic values, we need to have one answer known in advance for each symbolic value whose numerical value is unknown and desired. To solve a system, we require the same number of equations as unknowns. We will see a problem in which both axioms are satisfied at the same time, but first, a theoretical clarification. We explained in Chapter 7 that before simulating a circuit containing a two-port, we must have stored in memory the values of the four parameters of that two-port, under the corresponding names. But what happens if we do not store these values? Symbulator will simulate the circuit using the parameters that are not stored before the simulation as if they were symbolic variables.

Problem Nº 065

Problem: Complete the given table, and also give the values of the y parameters.
Solution:

This is a very special problem, which allows us to enjoy the symbolic capabilities of Symbulator. There are several ways to solve it. Here we will present the most linear and simple way, which however is not the most sophisticated.

Although the problem statement presents the two questions in that order, to answer them we will invert them. That is, first we will find the values of the y parameters, and then we will complete the table. To find the y parameters, we will take advantage of the first two rows of the table, which are already complete. It is precisely thanks to these complete rows that we will be able to generate the four equations necessary to find the four numerical values of the parameters `yp11`, `yp12`, `yp21` and `yp22`. The steps we will follow are these:

1. Simulate the circuit, without defined parameters.
2. Use the two complete rows of the table to generate the four equations we need.
3. Obtain the four numerical values of the parameters, solving these four equations.
4. Complete the table, using the parameters we have found and the values given by the incomplete rows.

The procedure is as follows:

```
sq\dc("j1,0,1,j1;j2,0,2,j2;yp,1,2,0")
```

We press ENTER. The simulation runs and 'Done' appears. We generate the first equation with the following command:

```
v1=100|j1=5 and j2=-32.5
```

With ENTER we obtain the equation. We generate the second equation like this:

```
v2=50|j1=5 and j2=-32.5
```

With ENTER we obtain the equation. We generate the third equation like this:

```
v1=50|j1=-20 and j2=-5
```

With ENTER we obtain the equation. We generate the fourth equation like this:

```
v2=100|j1=-20 and j2=-5
```

Having generated the four equations, we proceed to solve them to find the four y parameters:

```
solve(ans(1) and ans(2) and ans(3) and ans(4),
      {yp11,yp12,yp21,yp22})
```

With ENTER we obtain the parameters: `yp11` $= .2$ and `yp12` $= -.3$ and `yp21` $= -.4$ and `yp22` $= .15$. Now, we proceed to complete the table. To complete the third row, we use the following command:

```
solve(v1=20 and v2=0,{j1,j2})|ans(1)
```

With ENTER we obtain the missing values in the row: `j1` $= 4.$ and `j2` $= -8$. To complete the fourth row, we use the following commands:

```
v1|ans(2) and j1=5 and j2=0
v2|ans(3) and j1=5 and j2=0
```

Note how the number # that appears in `ans(#)` changes to reflect, at each moment, the new position of the answers holding the y parameters. With ENTER we obtain the values -8.33333 and -22.22222, respectively. To complete the fifth row, we use the following commands:

```
v1|ans(4) and j1=5 and j2=15
v2|ans(5) and j1=5 and j2=15
```

With ENTER we obtain the values -58.33333 and -55.55555, respectively. So, we have completed the table.

Personally, I consider this one of the most interesting problems I have ever solved, using Symbulator.

Let us learn to use a new element.

9.3 The ideal operational amplifier (ideal op-amp)

An ideal op-amp is, in essence, a voltage-dependent voltage source. By saying it is ideal, we mean that its parameters are considered as $R_i = \infty$, $R_o = 0$ and $A = \infty$.

9.3.1 Notation

In the case of the ideal op-amp, the minimum information necessary to describe it completely is the following:

Identification. The letter that identifies an ideal op-amp is `o`.

Name. It must be given any name that begins with the letter `o`.

Nodes. An ideal op-amp, as Symbulator understands it, has three connection points: one positive, one negative and one output.

So, ideal op-amps are defined as shown below:

```
oname, node +, node -, output node
```

The description of the ideal op-amp always has 4 terms, so there could be a need to add a zero at the end.

9.3.2 Related answers

In the case of an ideal op-amp, one related answer is given: the current that leaves the op-amp through the output node. This current is stored in a variable called `iname`, where `name` is the name of the ideal op-amp. So, for an ideal op-amp called `o1`, the related answer will be `io1`. Remember that this current is considered as leaving the op-amp.

Let us see some examples.

Problem Nº 066

Problem: Given the following circuit, find the voltage at the output node of the ideal op-amp.

Solution: After clearing the symbolic variables we are going to use, we order the simulation, and request the voltage at the output node of the op-amp:

```
DelVar r1,r2,vin:sq\dc("e1,1,0,vin;r1,1,2,r1;r2,2,3,r2;
o1,0,2,3"):v3
```

With ENTER, the simulation starts and we obtain the answer: `-r2*vin/r1`.

Problem Nº 067

Problem: If the op-amp shown is assumed ideal, find R_{in}.

Solution: Let us use the `thevenin` tool:

```
DelVar rx:sq\thevenin("rx,1,0,rx;r1,1,2,2E4;r2,2,3,2E4;
o1,3,1,2",3,0)
```

With ENTER, the tool executes. We choose Passive and DC. We press ENTER. We obtain the answer on the screen (and in memory): `zth = -rx`.

These examples illustrate the ease with which ideal op-amps are simulated in Symbulator.

9.4 Final considerations for the (Expert) Mode

This section should be read by those users who have already read Chapter 4.

9.4.1 Restrictions of the (Expert) Mode with op-amps

As we have said, the (Expert) Mode allows the user to choose whether to solve the equations, solve and keep them, or simply keep them. Circuits containing ideal operational amplifiers will generate equations that are not comprehensible to the user, because they contain a term that will later be used for the application of the calculator's limit command. Therefore, when the circuit solved in the (Expert) Mode includes op-amps, you must always choose Symbulate, and never Just Keep, when the dialog asks for the procedure you wish to follow.

9.4.2 Restrictions of the (Expert) Mode in transient analysis

In Chapter 10 we will learn to run simulations in the frequency domain (FD). In Chapter 11 we will learn to run transient analysis (TR) simulations. During a transient analysis, the simulator will generate equations in the frequency domain. After solving these equations, all the answers are converted to the time domain. Therefore, when the simulation in the (Expert) Mode is a transient analysis, we must always choose Symbulate, and never Just Keep, if we wish to obtain answers in the time domain. Keeping the generated equations and solving them later would lead us to answers in the frequency domain.

9.4.3 More first-level variables

Some circuit elements were presented after Chapter 4. Let us learn which of them have first-level variables related to them.

Inductors. The current through them is a first-level variable. Their voltage drop is a second-level variable that is not replaced by its value at the end of the simulation. Their power is a third-level variable.

Capacitors. The current through them is a second-level variable that is indeed replaced by its value at the end of the simulation. Their voltage drop is a second-level variable that is not replaced by its value at the end of the simulation. Their power is a third-level variable.

Ideal transformers. The current entering the transformer on the primary side is a first-level variable. The current entering the transformer on the secondary side is a second-level variable that is not replaced by its value at the end of the simulation.

Two-ports. The currents entering the two-port at both ends are first-level variables.

Ideal operational amplifiers. The current leaving the op-amp through the output node is a first-level variable.

Chapter 10

Simulation in the frequency domain

10.1 Door for the frequency domain: fd

To run a simulation in the complex frequency domain s, the door called `sq\fd` is used.

10.2 Input data

The only input data that must be given to this door is the circuit description, using the same notation we have seen so far. There is a single difference: if the circuit to be simulated has any inductor or capacitor, then the description of each element must have five terms. Descriptions that only require four terms must have a 0 at the end, to keep the symmetry. The frequency domain simulation is ordered like this: `sq\fd(circuit)`.

10.3 Initial conditions

The fifth term in the description of an inductor or a capacitor is its initial condition. Initial means at time $t = 0^+$.

The initial condition of an inductor is its initial current, and therefore its value must be given in amperes. This current is considered as flowing through the element from the first node to the second node of the description.

The initial condition of a capacitor is its initial voltage, and therefore its value must be given in volts. This voltage is considered as the voltage of the first node with respect to the second node of the description.

If the circuit does not specify the initial condition of the element nor provides means to obtain it, its initial condition is assumed to be null, that is, 0.

10.4 Answers

In the case of the frequency domain simulation, the answers are: the node voltages, the voltage drops in the elements and the currents in the elements. The powers consumed by the elements are not given. There is one difference: the answers will be symbolic expressions as functions of the complex frequency s.

10.5 Transfer function

Problems that ask us to find the transfer function, $H(s)$, are very common when studying analysis in the complex frequency domain. In the case that the problem only gives us the passive network whose transfer function we want to find, we must add a source at the input, preferably a current source with a symbolic or unit value.

Let us see some examples.

Problem Nº 068

Problem: Find $H(s) = V_{out}/V_{in}$ for the network of the figure, and locate all its critical frequencies.

Solution: We add a source, and order the simulation. Note that we use five terms per element:

```
sq\fd("j1,0,1,1,0;r1,1,2,20,0;r2,2,3,20,0;ca,1,3,.1,0;
      cb,2,0,.1,0")
```

With ENTER, the simulation executes. We request the transfer function `v3/v1` and obtain `(s^2+s+.25)/(s^2+1.5*s+.25)`. To find the critical frequencies, that is, the poles and zeros, we will use a calculator command.

10.6 Poles, zeros and the zeros command

The calculator has a command called `zeros`, which allows finding the zeros of a polynomial, for a given variable. It is used like this:

```
zeros(polynomial,variable)
```

The complex version of this command is another command called `cZeros`, which is used the same way.

We can find the zeros of a transfer function using this command, like this: `zeros(`transfer function`)`. We can find the poles of a transfer function using this command, like this: `zeros(`inverse of the transfer function`)`. The inverse of the transfer function is 1 over the function, or the function raised to -1.

Let us solve the second question of the problem. We request the zeros like this: `zeros(v3/v1,s)` and obtain `{-1/2}`. Although the command does not tell us so, this is a double zero. We request the poles like this: `zeros(v1/v3,s)` and obtain `{-1.31,-.191}`. So, we have finished our first problem in the frequency domain.

Problem Nº 069

Problem: Find $H(s) = V_o/V_s$ for the circuit of the figure.

Solution: There is no need to add a source, because the circuit includes one. We order the simulation, using five terms per element, and giving the value of the mutual inductances as M instead of k:

```
sq\fd("es,1,0,vs,0;l1,1,0,8,0;l2,2,0,2,0;r1,2,3,5,0;
      l3,3,0,4,0;l4,4,0,1,0;r2,4,0,3,0;
      m1,l1,l2,√(8*2),0;m2,l3,l4,√(4*1),0")
```

With ENTER, the simulation executes. We request the transfer function `v4/v1` and obtain `3*s/(17*s+15)`. Let us now see an example of greater difficulty, but first let us learn a new concept.

10.7 Exact vs approximate

In section 2.9 we learned that there are two ways in which the calculator can handle a number: as exact and as approximate. What determines which mode the calculator will use to solve a circuit is the way in which the data is entered. Let us see the advantages and disadvantages of each of these two modes.

Exact mode:

Disadvantage: generally, the solution of the equations takes more time.

Advantage: some systems of equations can be solved precisely that in approximate mode could turn out to be unsolvable, or solved imprecisely.

Approximate mode:

Advantage: generally, the solution of the equations takes less time.

Disadvantage: in some particular cases, it is possible that a system of equations cannot be solved in this mode, even though in exact mode it would have been solved. In other particular cases, it is possible that the answers lose precision, especially in the case of large numbers.

In the great majority of cases, either of the two modes can be used interchangeably. However, when dealing with large circuits, it is preferable to choose the mode that best fits our need.

Problem Nº 070

Problem: Find $H(s) = V_{out}/V_{in}$ for the network of the figure. The op-amps are ideal.

Solution: We order the simulation, using five terms per element. We prefer to use all the values in exact form, to guarantee the best probability of obtaining a solution, regardless of the solution time:

```
sq\fd("e1,1,0,1,0;cc1,1,2,10^-6,0;r1,2,3,10^4,0;o1,0,2,3,0;
      r2,3,4,10^5,0;r3,4,5,10^5,0;cc2,4,5,10^-6,0;o2,0,4,5,0;
      r4,5,6,10^5,0;r5,6,7,10^5,0;cc3,6,7,10^-6,0;o3,0,6,7,0")
```

With ENTER, the simulation executes. This simulation takes more time than any we have seen so far. This increase in time is due not only to the generated equations being enormous, but to the limit being applied to all the answers, since there are op-amps in the circuit. When ordering this simulation, you can leave the calculator executing it and go have lunch. When you return, you will find that the simulation has finished. We request the transfer function `v7/v1` and obtain the expression `-s/(s+10)^2`.

Here a note must be made. A numerical simulator, such as SPICE, could find the numerical value of the output voltage, if we give it a numerical value for the input voltage and a numerical value for the frequency. But to obtain a symbolic expression for the transfer function, in terms of s, it is necessary to use a symbolic simulator, like Symbulator.

Let us now see four examples that illustrate that the `thevenin` tool can also be used in the complex frequency domain.

Problem Nº 071

Problem: Find: a) Z_{in} for the network of the figure, in terms of s, b) $Z_{in}(j8)$ in rectangular form, c) $Z_{in}(-2+j6)$ in polar form, d) to what value the 16 Ω resistor must be changed so that $Z_{in} = 0$ at $s = -5 + j0$, e) to what value the 16 Ω resistor must be changed so that $Z_{in} = \infty$ at $s = -5 + j0$.

Solution: We execute the tool, using five terms per element:

```
sq\thevenin("c,1,2,.02,0;r,2,0,16,0;l,2,0,.2,0",1,0)
```

We press ENTER. We choose Passive and FD. We press ENTER. We answer question a) in this first step. We see on the screen the value of `zth`. We exit with ENTER. We can request `zth` and obtain the expression `16.*(s^2+3.125*s+250.)/(s*(s+80.))`.

We answer question b) like this: `zth|s=8i→Rect` and obtain $.158 - 4.67i$.

We answer question c) like this: `sq\absang(zth|s=-2+6i)` and obtain $6.85\angle -114.^\circ$.

To answer the two following questions, we simulate again, using a symbolic value for the resistor:

```
sq\thevenin("c,1,2,.02,0;r,2,0,r,0;l,2,0,.2,0",1,0)
```

With [ENTER], the tool executes. We choose Passive and FD. We press [ENTER]. We see on the screen the value of `zth`. We exit with [ENTER]. We answer question d) like this: `cSolve(zth=0,r)|s=-5` and obtain `r` = .909.

To answer question e), we require two steps:

```
cSolve(zth=x,r)|s=-5
```

We obtain `r=(x+10.)/(x+11.)`. Now the second step:

```
limit(ans(1),x,∞)
```

We obtain 1. This is the correct answer.

Problem Nº 072

Problem: This problem has no figure. A network is composed of a 1 μF capacitor in parallel with the series combination of a 20 mH inductor and a 500 Ω resistor. a) Find $Z_{in}(s)$ for the network, in terms of s. b) If $s = \sigma - j0$, find the value of σ, between -20 and -10 kNp/s, at which abs(Z_{in}) *is a minimum. c) Again with $s = \sigma - j0$, find the value of σ, smaller than -30 kNp/s, at which* abs(Z_{in}) *is a maximum.*

Solution: We execute the tool:

```
sq\thevenin("c,1,0,1E-6,0;l,1,2,.02,0;r,2,0,500,0",1,0)
```

We press [ENTER]. We choose Passive and FD. We press [ENTER]. We answer question a) in this first step. We see on the screen the value of `zth`. We exit with [ENTER]. In any case, we request it as `zth` and obtain the expression:

```
1000000.*(s+25000.)/(s^2+25000.*s+50000000.)
```

To answer questions b) and c), we remember that both the minimum and the maximum are points at which the slope of the graph is 0. Using the derivative, we can solve with `solve` for those frequencies at which the derivative becomes 0:

```
solve(d(abs(zth),s)=0,s)
```

The `d` is the derivative, and is entered with the key combination [2nd] [8]. The command means: "Find the values of s at which the derivative of the absolute value of `zth` with respect to s is worth zero." We obtain `s` = -17929. or `s` = -32071. According to what the problem statement tells us, the first value is a minimum and the second is a maximum. With this, we have answered the questions.

Problem Nº 073

Problem: Find: a) $Z_{in}(\sigma)$ as a function of σ for the network of the figure, b) find all the zeros and poles, c) graph abs$(Z_{in}(\sigma))$ *vs σ, d) graph* ang$(Z_{in}(\sigma))$ *vs σ.*

Solution: We execute the tool:

```
sq\thevenin("r1,1,2,20,0;l1,2,0,8,0;r2,2,3,40,0;l2,3,0,2,0",1,0)
```

We press [ENTER]. We choose Passive and FD. We press [ENTER]. We see on the screen the value of `zth`, as a function of s. We press [ENTER] to exit. Requesting `zth|s=`$\sigma\rightarrow$`zth` we obtain the expression `4*(2*`σ`^2+65*`σ`+100)/(5*(`σ`+4))`.

Let us answer question b). We request the zeros with `zeros(zth,`σ`)` and obtain `{5*(`$\sqrt{}$`137-13)/4,-5*(`$\sqrt{}$`1` If we prefer to see these zeros in approximate form, we use [◇] [ENTER] and receive `{-1.62,-30.9}`. We request the poles with `zeros(1/zth,`σ`)` and obtain `{-4}`.

To make the graph that question c) requests, we follow these steps:

```
abs(zth)|σ=x
ans(1)→y1(x)
```

We use `ans(1)` because the expression is above. We press [◇] [WINDOW]. Here we can specify the range we wish to graph. We use `xmin` $= -50$ and `xmax` $= 20$. We press [◇] [GRAPH]. The calculator graphs the function, but using an inappropriate vertical scale. We press [F2] [A] on the TI-89, to apply the ZoomFit command. We see the graph, at the correct scale.

To make the graph that question d) requests, we follow these steps:

```
angle(zth)|σ=x
ans(1)→y1(x)
```

We press [◇] [WINDOW]. Here we can specify the range we wish to graph. We use `xmin` $= 0$ and `xmax` $= 4$. We press [◇] [GRAPH]. The graph appears, with an inappropriate scale. We apply the ZoomFit command. We see the graph, at the correct scale.

So, we have answered all the questions.

Problem Nº 074

Problem: Find: a) $Z_{in}(s)$ *for the network of the figure, b) find all the critical frequencies, c) graph* $\text{abs}(Z_{in}(s))$ *vs* s.

Solution: We execute the tool:

```
sq\thevenin("r1,1,2,4,0;l1,2,0,1,0;l2,1,3,5,0;r2,3,0,5,0",1,0):zth
```

We press [ENTER]. We choose Passive and FD. We press [ENTER]. We see on the screen the value of `zth`, as a function of s. We exit with [ENTER]. In the history area we see `5*(s+1)*(s+4)/(3*(2*s+3))`. This expression appeared now, because we used `:zth` at the end of the command line.

For question b), we request the zeros with `cZeros(zth,s)` and obtain `{-4,-1}`. We request the poles with `cZeros(1/zth,s)` and obtain `{-3/2,-`∞`,`∞`}`. We use the `cZeros` command instead of the `zeros` command because, unlike the previous problem, this time we are looking for poles in s, which can be complex. So, we use the `cZeros` command when we are going to find complex critical frequencies, and the `zeros` command when we are going to find real critical frequencies.

To make the graph that question c) requests, we follow these steps:

```
abs(zth)|s=x
ans(1)→y1(x)
```

We use `ans(1)` because the expression is above. We press [◇] [WINDOW]. Here we can specify the range we wish to graph. We use `xmin` $= -7$ and `xmax` $= 2$. We press [◇] [GRAPH]. The calculator graphs the function, but using an inappropriate vertical scale. We press [F2] [A] on the TI-89, to apply the ZoomFit command. We see the graph, at the correct scale.

So, we have answered all the questions.

Chapter 11

Simulation in the time domain

11.1 Relation between FD and TR

In this chapter, we will see the simulation techniques for analysis in the time domain, that is, transient analysis, in Symbulator. There is an intimate relation between analysis in the frequency domain (FD) and analysis in the time domain (TR). In TR, Symbulator uses the techniques of the Laplace transform to obtain the complete answers in the time domain, from the frequency-domain answers obtained with FD. The complete answer includes the forced response and the transient response. As we have learned in the classroom, the Laplace transform can take an answer from the frequency domain to the time domain.

11.1.1 DiffEq

Symbulator uses the powerful program DiffEq, written by Lars Frederiksen, who is the best calculator programmer I have known, and my personal friend. DiffEq is a group of programs made to solve differential equations. The most advanced Laplace transform tool for TI calculators is called Advanced Laplace, also made by Lars Frederiksen. Although less powerful, the Laplace transform of DiffEq is faster than that of Advanced Laplace. That is why Symbulator prefers DiffEq.

Symbulator uses the `laplace` function of DiffEq at the start of any FD or TR simulation, and the `ilaplace` function of the same at the end of any TR simulation. That is why it is necessary to have this program installed when Symbulator is used to perform these two types of analysis.

11.2 Two ways to obtain answers in the time domain

There are two ways for the Symbulator user to obtain answers in the time domain:

The first alternative (see section 11.3) is to run an analysis in the frequency domain, and then convert the answers that interest us to the time domain. This analysis is done using the `sq\fd` door. This analysis solves the circuit in the frequency domain, converting all the values of sources that are in the time domain to their equivalents in the frequency domain, but it does not convert the answers to the time domain. It will give us all the answers in the frequency domain. Since no automatic conversion of all the answers to the time domain is done, all the time these conversions would have consumed is saved. So, the simulation will take the minimum possible. After the analysis, you can take to the time domain those answers that interest you. This alternative should be used in two cases: a) when we want answers both in the time domain and in the frequency domain, and b) when we want the fastest possible simulation, and we want only some answers in the time domain, not the whole set.

The second alternative (from section 11.5 onward) is to run a transient analysis. This analysis is done using the `sq\tr` door, whose use we will explain later. This analysis solves the circuit in the frequency domain, converting all the values of sources that are in the time

domain to their equivalents in the frequency domain, and finally converts all the answers back to the time domain. This analysis will give us all the answers in the time domain. The user never sees the answers in the frequency domain. The conversion of all the answers to the time domain can consume several minutes, the more the larger the circuit. Therefore, this alternative should be used only in three cases: a) when the circuit is small, b) when the user is interested in all the answers in the time domain and not only in some answers, and c) when the user does not mind the simulation taking a few minutes.

11.3 Two ways to convert answers to the time domain

When we use the first alternative, we will have to convert the answers that interest us to the time domain ourselves. We can do it in two ways.

11.3.1 The ilaplace function of DiffEq

DiffEq converts a function from the frequency domain to the time domain, using the `ilaplace` function, like this:

```
dif\ilaplace(function of the frequency,s)
```

It is worth pointing out that DiffEq also does the inverse operation, that is, converting a function from the time domain to the frequency domain, using the `laplace` function, like this:

```
dif\laplace(function of time,t)
```

11.3.2 The fd_to_tr tool of Symbulator

The `fd_to_tr` tool of Symbulator is really a kind of shortcut for the use of the `ilaplace` function of DiffEq. The only purpose of this tool is to save the user from placing the `,s)` at the end of the command line. It is the `ilaplace` function that does all the work. This tool is used like this:

```
sq\fd_to_tr(answer in the frequency domain)
```

It gives us the answer in the time domain.

11.4 The Symbulator menu

Symbulator includes a program to create a personalized menu, called in English a Custom Menu. The program is executed like this:

```
sq\makemenu()
```

When executed, this program will create a personalized menu, in which all the doors, tools and elements of Symbulator can be found. This way, you will avoid having to type the names of the tools and doors, since you can select them using the cursor keys.

For information on how to restore the original menu of the calculator, read the User's Manual.

11.5 Door for the time domain: tr

If we choose the second alternative, that is, to run a simulation in the time domain t, we will use the door called `sq\tr`.

11.6 Input data

The only input data that must be given to this door is the circuit description, using the same notation used for a simulation in the frequency domain. The time domain simulation is ordered like this: `sq\tr(circuit)`.

11.7 Initial conditions

Just as in the frequency domain simulation, the fifth term in the description of an inductor or a capacitor is its initial condition.

11.8 Answers

In the case of the time domain simulation, the answers are: the node voltages, the voltage drops in the elements and the currents in the elements. The powers consumed by the elements are not given. As we already explained in section 11.2, all these answers will be delivered in the time domain. Therefore, they will be symbolic expressions as functions of time t.

Let us now see several problems that illustrate how transient analysis problems are solved in Symbulator.

Problem Nº 075

Problem: a) According to the figure, find the ratio $I_L(s)/V_s(s)$. b) Make $v_s(t) = 100u(t)$ V and find $i_L(t)$.

Solution: Since the problem does not specify initial conditions, nor gives us the way to calculate them, we assume they are null. This problem has two parts. The first asks us for an answer in the frequency domain, while the second asks us for an answer in the time domain. There are two good reasons to solve this problem using the first alternative. (As we explained in section 11.2, the first alternative consists of simulating with `sq\fd` and then transforming with `sq\fd_to_tr`.) The first reason is that we are interested in only one answer in the time domain. The second reason is that we are also interested in an answer in the frequency domain. As we explained in section 11.2, both reasons are sufficient cause to use the first alternative instead of the second. Below, the procedure to solve this problem.

For the purposes of answering the first question, it does not matter what value we assign to the voltage source. We could define its value as `vs`, or as `100u(t)`, and the answer would be the same. This is because the answer is a ratio, and the source will influence both the numerator and the denominator, canceling its effect. Now, if at the beginning we use a value like `vs`, when we go to answer the second question, we will have to simulate again using the new value of `100u(t)`. However, if at the beginning we use the value of `100u(t)`, this single simulation will serve us to answer both questions. So, we take advantage of the fact that the first question allows it, and use the value of the source as `100u(t)`. It is necessary to say that the `ilaplace` function of the DiffEq program understands what $u(t)$ and $\delta(t)$ mean. In fact, this function assumes that any numerical value entered is multiplied by $u(t)$. Therefore, it is not necessary to enter the value `100u(t)`, because for `ilaplace`, saying 100 and saying $100u(t)$ is the same. Below, the description:

```
sq\fd("e1,1,0,100,0;r1,1,2,15,0;r2,3,0,20,0;l1,2,0,5,0;
      l2,3,0,3,0;m1,l1,l2,2,0")
```

With ENTER, the simulation executes. To answer the first question, we request the ratio `ir2/v1` and obtain `2*s/(11*s^2+145*s+300)`. To answer the second question, we use `sq\fd_to_tr(ir2)` and obtain:

```
40*√313*e^((5*√313/22-145/22)*t)/313
   -40*√313*e^((-5*√313/22-145/22)*t)/313
```

These are the correct answers. Do you wish to see the answers in approximate form? If we request `approx(ans(1))`, we will obtain `2.26*(.0765)^t-2.26*(2.46E-5)^t` (rounding). If we request `expand(ans(1))`, we will obtain `2.26/(e^t)^(2.57)-2.26/(e^t)^(10.6)` (rounding). Let us learn more about this behavior of approximate answers.

11.9 Approximate presentation with exponentials

When the Exact/Approx mode is in AUTO, the calculator will keep an expression in the form e^{nt} only if n is an exact number. For example, if we enter `e^(2t/3)` in the entry line, we will see that in the history area the calculator also presents us `e^(2t/3)`. This is because the number that accompanies t, that is, $2/3$, is an exact number.

The disadvantage of this exact way of presenting the answer resides in the fact that on some occasions, as in the case of the problem we just solved, the exact expression is so voluminous that its use is uncomfortable. Furthermore, some textbooks and professors favor answers in approximate presentation.

If n is an approximate number, the calculator will convert the expression to a totally approximate value, to the point of eliminating the exponentials. For example, let us use the expression in approximate form, either by entering `e^(2t/3.)` or `approx(e^(2t/3))` in the entry line (both the point and the `approx` command have the effect of turning the expression approximate). We will see that in the history area the calculator now presents us `(1.94773404)^t`. This is because the number that accompanies t, that is, $2/3.$, is an approximate number.

The disadvantage of this totally approximate way of presenting the answer resides in the fact that the exponentials have been replaced, diluted in the approximate number of the base. In the years I have been studying electrical engineering, I have never seen a textbook or a professor who favors the use of answers in this approximate presentation without exponentials.

Therefore, it is desirable to obtain a combination of the virtues of both ways: an approximate expression that is smaller than the exact one, but that keeps the exponentials in their place. In the calculator, it is possible to obtain this type of answer, using the `expand` command. This command is used like this: `expand(expression,variable)`, and its result is that the expression will be shown in an expanded way, with respect to the variable. In our example, if we enter `expand(e^(2t/3.),t)` or `expand(approx(e^(2t/3)),t)` in the entry line, we will see that in the history area the calculator presents us `(e^t)^.666666667`.

Also in this case, we would prefer a somewhat different presentation, like the following: $e^{.666666667t}$. Even so, that presentation is much better than the two previous ones.

If we expand an exponential expression whose number n is negative, the `expand` command will show an interesting behavior. For example, with `expand(e^(-2t/3.),t)` in the entry line, we will see that in the history area the calculator presents us `1/(e^t)^.666666667`.

Certainly, we would prefer a different presentation, like the following: $e^{-.666666667t}$. But having the first expression, we can obtain the second easily.

Problem Nº 076

Problem: a) For the circuit shown in the figure, find $H(s) = v_{C2}/v_S$. *b) Let* $v_{C1}(0^+) = 0$ *and* $v_{C2}(0^+) = 0$. *Find* $v_{C2}(t)$ *if* $v_S(t) = u(t)$.

Solution: The problem clearly specifies that the initial conditions are null. Since this problem is identical to the previous one, we save ourselves the explanations, which would be redundant. We will only say that, for DiffEq, and therefore for Symbulator, $u(t)$ is the same as 1:

```
sq\fd("es,1,0,1,0;r1,1,2,50,0;r2,2,3,20,0;cc1,2,0,.04,0;
      cc2,3,0,.01,0")
```

With ENTER, the simulation executes. To answer the first question, we request the ratio `v3/v1` and obtain `2.5/(s^2+6.75*s+2.5)`. To answer the second question, we use `expand(sq\fd_to_tr(v3),t)` and obtain `-1.066/(e^t)^(.393)+.0659/(e^t)^(6.36)+1.` (rounding).

Instead of having used `expand(sq\fd_to_tr(v3),t)`, we could have used first `sq\fd_to_tr(v3)` and then `expand(ans(1),t)`, and the result would be the same.

11.10 Intervals and simulations

In Symbulator, each interval of time requires one simulation. To simulate the behavior of a circuit that has a switch, which switches at a given time, say $t = 0$, we must do two simulations: one for $t < 0$ and another for $t > 0$.

The same principle governs the function $u(t)$. Mathematically, $n \cdot u(t)$ indicates that for times $< t$ the function is worth 0, and for times $> t$ the function is worth n. In the same way, mathematically, $n \cdot u(-t)$ indicates that for times $< t$ the function is worth n, and for times $> t$ the function is worth 0. For the purposes of the simulation, the $u(t)$ does not have that meaning. We must perform one simulation for each interval of time.

Let us see several examples.

Problem Nº 077

Problem: After remaining closed for a long time, the switch in the circuit of the figure opens at $t = 0$. *a) Find* $i_L(t)$ *for* $t > 0$. *b) Evaluate* $i_L(10\text{ ms})$. *c) Find* t_1 *if* $i_L(t_1) = 0.5\, i_L(0)$.

Solution: The problem tells us that there is a switch that switches at $t = 0$. We must first run a simulation, in direct current, for $t < 0$, which will allow us to find the initial condition of the inductor. Then we will run a second simulation, in the frequency domain, for $t > 0$, to answer the questions. The procedure is the following:

```
sq\dc("r1,1,0,20;r2,2,1,10;r3,2,3,50;l1,3,0,.2;e1,2,0,100"):il1
```

We request the direct current simulation, and immediately after ask for the current in the inductor. With ENTER, the simulation executes. We obtain 2 as the current in the inductor. This will be its initial condition for the simulation of the next interval:

```
sq\fd("r1,1,0,20,0;r2,2,1,10,0;r3,2,3,50,0;l1,3,0,.2,2"):
sq\fd_to_tr(il1)
```

We request the simulation in the frequency domain, and immediately after ask for the current in the inductor, in the time domain. With ENTER, the simulation executes. We obtain `2*e^(-400*t)` as the expression for the current in the inductor. This answers question a).

To answer question b), we use `ans(1)|t=10E-3` and obtain .0366 (rounding).

To answer question c), we use `solve(ans(2)=.5*2,t)`. Note that the command means: "Tell me how much time is worth when the expression `2*e^(-400*t)`, recalled by `ans(2)`, equals 0.5 times the initial current, which is 2." We obtain `t` = .00173 (rounding).

In summary, this problem did not give us the initial conditions, but gave us a way to calculate them by means of a direct current simulation for before the switch switched. This problem required two simulations, one for each interval of time.

Problem Nº 078

Problem: If $i_L(0) = 10$ A in the circuit of the figure, find $i_L(t)$ for $t > 0$.

Solution: The problem gives us the initial condition of the inductor. Therefore, we only need to run one simulation:

```
sq\fd("j1,0,1,il/4,0;r1,1,0,20,0;r2,1,2,10,0;l,2,0,.5,10"):
sq\fd_to_tr(il)
```

We request the simulation in the frequency domain, and immediately after ask for the current in the inductor in the time domain. With ENTER, the simulation executes. We obtain `10*e^(-50*t)` as the expression for the current in the inductor.

This problem gave us the initial conditions. It only required one simulation.

Problem Nº 079

Problem: a) Find $v_C(t)$ for $t > 0$. b) At what time is $v_C = 0.1\, v_C(0)$?

Solution: This problem is similar to Problem 076. We prefer to use exact values:

```
sq\dc("j1,0,1,8;r1,1,0,200;r2,1,2,20;r3,2,0,30;r4,2,3,24;
      c,3,0,(1/3)/1000"):vc
```

We simulate in direct current, and ask for the voltage in the capacitor. We press ENTER. We obtain 192 as the voltage in the capacitor. This will be its initial condition for the simulation of the next interval. We simulate in the frequency domain, and ask for the voltage in the capacitor, in the time domain:

```
sq\fd("j1,0,1,8,0;r1,1,0,200,0;r2,1,2,20,0;r3,2,0,30,0;
      r4,2,3,24,0;c,3,0,(1/3)/1000,192;s1,2,0,0,0"):
sq\fd_to_tr(vc)
```

Note that to simulate the closed switch, we used a short circuit. We could have eliminated the part of the circuit that is to the left of the switch, since it does not interest us. That way we would save the simulator work. But we will be rigorous, and leave it. We press ENTER. We obtain `192*e^(-125*t)`. This answers question a).

To answer question b), we use `solve(ans(1)=.1*ans(2),t)`. The command means: "Tell me how much time is worth when the expression `192*e^(-125*t)`, recalled by `ans(1)`, equals 0.1 times the initial voltage, recalled by `ans(2)`." We obtain `t` = .01842 (rounding).

This problem did not give us the initial conditions, but gave us a way to calculate them, by means of a direct current simulation of the interval previous to the switching of the switch. This problem required two simulations, one for each interval of time.

Problem Nº 080

Problem: Suppose that the circuit shown in the figure has been in that form for a long time. Find $v_C(t)$ for $t > 0$.

Solution: Let us see:

```
sq\dc("e1,1,0,vc/4;r1,1,2,5;r2,2,0,10;c,2,0,10^-6;
      r3,2,3,4;e2,3,0,40"):vc
```

We prefer to use exact values. Remember that the sign of `10^-6` is the negation sign, not the subtraction sign. We press ENTER. We obtain 20 as the voltage in the capacitor. This will be its initial condition for the simulation of the next interval:

```
sq\fd("e1,1,0,vc/4,0;r1,1,2,5,0;r2,2,0,10,0;c,2,0,10^-6,20"):
sq\fd_to_tr(vc)
```

We simulate in the frequency domain, and ask for the voltage in the capacitor, in the time domain. We press ENTER. We obtain `20*e^(-250000*t)`.

This problem did not give us the initial conditions, but gave us a way to calculate them. This problem required two simulations, one for each interval of time.

Problem Nº 081

Problem: Suppose the op-amp is ideal. Find $v_o(t)$.

Solution: The problem gives us to understand that there are no initial conditions. Therefore, we will run only one simulation. We simulate in the frequency domain, and ask for the output voltage, in the time domain. We prefer to use exact values:

```
sq\fd("j1,0,1,5/1000,0;r1,1,0,250,0;ca,1,2,8/1000,0;
      r2,2,0,1000,0;o1,2,3,3,0;r3,3,0,50,0"):sq\fd_to_tr(v3)
```

We press ENTER. We obtain `e^(-t/10)`.

This problem used null initial conditions. It only required one simulation.

Problem Nº 082

Problem: Suppose the op-amp is ideal. Find $v_o(t)$.

Solution: This problem is like the previous one. We prefer to use approximate values:

```
sq\fd("j1,0,1,.5E-6,0;r1,1,0,2E6,0;ca,1,2,.5E-6,0;o1,0,2,3,0;
      cb,2,3,.1E-6,0;r2,3,0,100,0"):sq\fd_to_tr(v3)
```

We press ENTER. We obtain `5.*e^(-t)-5.`

This problem used null initial conditions. It only required one simulation.

Problem Nº 083

Problem: In the circuit shown, let $L = 5\ H$, $R = 8\ \Omega$, $C = 12.5\ mF$ *and* $v(0^+) = 40\ V$. *Find a)* $v(t)$ *if* $i(0^+) = 8\ A$, *b)* $i(t)$ *if* $i_C(0^+) = 8\ A$.

Solution: The problem gives us the initial condition of the capacitor. Then it asks us two questions. For the first question, it also tells us the initial condition of the inductor. We simulate in the frequency domain, and ask for the voltage in the time domain:

```
sq\fd("l,1,0,5,8;r,1,0,8,0;c,1,0,12.5E-3,40"):sq\fd_to_tr(v1)
```

We press ENTER. We obtain `160.*e^(-8*t)-120.*e^(-2*t)`. This answers question a).

For the second question, the problem does not tell us the initial condition of the inductor, but gives us the way to find it. We simulate in the frequency domain, using a symbolic value `io` as the initial condition for the inductor, since we do not know its numerical value. (The initial condition of the inductor that the problem gave us earlier was valid only for question a).)

```
sq\fd("l,1,0,5,io;r,1,0,8,0;c,1,0,12.5E-3,40")
```

We press ENTER. We know that the current in the capacitor is worth 8 at $t = 0$. We ask for `sq\fd_to_tr(ic)` and obtain a symbolic expression in terms of `io`. We must solve this expression to obtain the numerical value of `io`. We solve it like this: `solve(ans(1)=8,io)|t=0`. What we are asking is: "Tell me how much `io` is worth if the current in the capacitor, whose expression is in `ans(1)`, at time zero is worth 8." This is a way of asking: how much must the initial condition of the inductor be so that the current in the capacitor at time zero is 8? We obtain: `io` $= -13$. That is the initial condition of the inductor. Now we must find the expression for the current in the inductor. We could simulate again, using the numerical initial condition. But the simplest thing is to just replace in the expression the value we found with `solve`. We ask for `sq\fd_to_tr(il)|ans(1)` and obtain `3.*e^(-8*t)-16.*e^(-2*t)`. This answers question b).

Problem Nº 084

Problem: Find $i_L(t)$ for $t > 0$ in the circuit shown in the figure.

Solution: We must understand the $u(-t)$ as a switch: the source is worth 100 for $t < 0$, and worth 0 for $t > 0$. Therefore, we will use the concept of two intervals:

```
sq\dc("r,1,0,50;c,1,0,2.5E-6;l,1,2,(100/3)E-3;e,2,0,100")
```

Remember that the sign that appears in `E-6` is the negation sign, not the subtraction sign. We press ENTER. Let us obtain the initial conditions. For `vc` we obtain 100. For `il` we obtain -2. We simulate in the frequency domain, and ask for the current in the inductor, in the time domain:

```
sq\fd("r,1,0,50,0;c,1,0,2.5E-6,100;l,1,2,(100/3)E-3,-2;
      e,2,0,0,0"):expand(sq\fd_to_tr(il))
```

Note that the source was given a value of 0. There were two other alternatives: use a short circuit, or eliminate the source and connect the inductor to the reference node. We press ENTER. We obtain `.25/(e^t)^6000-2.25/(e^t)^(2000.)`. This is the same as saying $.25e^{-6000t} - 2.25e^{-2000t}$.

Problem Nº 085

Problem: Find $i_s(t)$ for $t > 0$ in the circuit of the figure if $v_s(t)$ is a) $10u(-t)$ V, b) $10u(t)$ V.

Solution: Let us answer question a). We must understand the $u(-t)$ as a switch that makes the source worth 10 for $t < 0$, and 0 for $t > 0$. Therefore, we will use the concept of two intervals:

```
sq\dc("e,1,0,10;r,1,2,500;l,1,2,4/3;c,2,0,1/10^6")
```

We press ENTER. Let us obtain the initial conditions. For `vc` we obtain 10. For `il` we obtain 0.

```
sq\fd("e,1,0,0,0;r,1,2,500,0;l,1,2,4/3,0;c,2,0,1/10^6,10"):
sq\fd_to_tr(-ie)
```

We use the negative on `ie` because i_s is defined in the opposite sense to `ie`. We press ENTER. We obtain `e^(-500*t)/400-9*e^(-1500*t)/400`. If we want to see the answer in approximate form, instead of `sq\fd_to_tr(-ie)`, we use `expand(approx(sq\fd_to_tr(-ie)))`.

We obtain the expression `.0025/(e^t)^500-.0225/(e^t)^1500`. This is the same as saying $.0025e^{-500t} - .0225e^{-1500t}$. This answers question a).

Let us answer question b). We must understand the $u(t)$ as a switch that makes the source worth 0 for $t < 0$, and 10 for $t > 0$. Therefore, we will only need one interval, since the source is dead before t. We use null initial conditions:

```
sq\fd("e,1,0,10,0;r,1,2,500,0;l,1,2,4/3,0;c,2,0,1/10^6,0"):
sq\fd_to_tr(-ie)
```

We press ENTER. We obtain `9*e^(-1500*t)/400-e^(-500*t)/400`. If we want to see the answer in approximate form, we use the same procedure explained above. We would obtain $.0225e^{-1500t} - .0025e^{-500t}$. This answers question b).

11.11 The (Impala) Mode

Like the (Expert) Mode, the (Impala) Mode is a variant in the way Symbulator solves a problem. But unlike the (Expert), which must be activated by the user at will, the (Impala) activates itself automatically when it is necessary.

The (Impala) Mode activates when the user enters some source whose value is a function of time, that is, containing exponentials, trigonometric functions, etc. Examples of these sources would be: $100e^{-80t}$, $10e^{-t}\sin(2t + 30°)$ and others of that style.

What this mode does is write and solve the equations using the value of the source as a simple symbolic variable, called σ`name`, where `name` is that of the source. Once the equations have been solved, before storing the answers, the Impala replaces the variable σ`name` with the value of the source in the frequency domain. This simple substitution saves a considerable amount of time. In the following example, and in an example of the next chapter, we will see the (Impala) Mode in action. We will know this mode activates when the messages indicating so appear on the screen.

This mode receives its name in honor of an African antelope, which on seeing itself threatened by its predators, escapes by taking prodigious leaps. So too Symbulator, on meeting very complex sources, leaps, avoiding the use of their complicated value, and solves the equations using a simple symbolic value; then it lands, replacing the source by its complicated value in the frequency domain.

Problem Nº 086

Problem: Let $v_s = 100e^{-80t}$ and $v_1(0) = 20$ V in the circuit of the figure. a) Find $i(t)$. b) Find $v_1(t)$. c) Find $v_2(t)$.

Solution: The problem gives us the initial condition of the 1 μF capacitor; however, it does not tell us those of the 4 μF and 2 μF capacitors. Running a DC simulation of this circuit will not help us find these initial conditions. But we can deduce them easily, using Kirchhoff's voltage law. So, we deduce that the initial condition of the 4 μF capacitor is 80 V, and that of the 2 μF capacitor is 100 V.

Since the problem asks us for three answers in the time domain, and the circuit is small, we will use TR instead of FD. We will use exact values:

```
sq\tr("e,1,0,100e^(-80t),0;cc1,1,2,1*10^-6,20;
      cc2,2,0,4*10^-6,80;cc3,1,0,2*10^-6,100")
```

Note that the order in which the nodes of the capacitors have been defined is in accordance with the polarity of the declared initial voltage. We press ENTER. We see that the (Impala) Mode takes part in the simulation. We ask for `icc1` and obtain `-4*e^(-80*t)/625`. We

ask for `vcc1` and obtain `80*e^(-80*t)-60`. We ask for `vcc2` and obtain `20*e^(-80*t)+60`. These are the correct answers.

Chapter 12

Tools for graphing

12.1 Introduction

Your calculator has powerful graphing capabilities. In turn, Symbulator includes two useful tools for graphing: one for Bode diagrams and one for functions of time t.

12.2 The bode tool

The `bode` tool allows making Bode diagrams. It is executed like this:

```
sq\bode()
```

It is very easy to use. Let us learn to use this tool by solving a real problem. We will use one that Professor Edilberto Yee, of the Technological University of Panama, gave us as the final homework of the Electronics III course.

Problem Nº 087

Problem: Make the Bode gain diagram for V_o/V_g, from 10^4 to 10^{13} rad/s, for the following circuit.

Solution: As we have already explained, when dealing with large circuits, a simulation with exact values will give us precise answers, while a simulation with approximate values will give us fast answers. Since this problem was homework to solve at home, time was not critical, and we preferred to use exact values, to assure the precision of the answers.

We give the command to run the simulation in the frequency domain, and immediately after order the execution of the `bode` tool:

```
sq\fd("e1,5,0,vg,0;r1,5,1,150,0;r2,1,0,1000,0;
      cc1,1,0,10^-10,0;cc2,1,2,3*10^-12,0;jd1,2,0,5/100*v1,0;
      r3,2,0,2000,0;r4,2,3,100,0;r5,3,0,1000,0;
      cc3,3,0,10^-10,0;cc4,3,4,3*10^-12,0;jd2,4,0,5/100*v3,0;
      r6,4,0,2000,0"):sq\bode()
```

We press ENTER. The simulation takes around 3 minutes. At the end, the `bode` tool executes. The tool presents us a small menu, which asks us to choose between making a Bode plot or cleaning up the variables created.

We choose the first option of this menu. The second is used when, after having made some graphs, we wish to finish the work and clean up the variables the tool has created. So, we select Bode Plot and press ENTER. A menu appears. In this menu we must enter the following:

Transfer function. Here we place the transfer function whose graph we wish to make. We can write the function itself, or place the names of the variables that define it. In our case, we place `v4/v5`, that is, V_o/V_g.

Independent variable. It is the variable that represents the frequency. It is almost always `s`. We place `s`.

Type of diagram. There are three types: gain (Gain Plot), phase (Phase Plot) and both (Both Plots). If gain or phase is chosen, the complete screen will show the graph. If both is chosen, the screen will divide in two to show gain above and phase below. In our case, we select Gain Plot. To return the screen to its original form afterwards, go to MODE and set the Split Screen mode to FULL.

Minimum frequency. It must be greater than zero. In our case, we place `1E4`.

Maximum frequency. It must be greater than the minimum. In our case, we place `1E13`.

Units of the frequency. It serves to specify in what units we entered the frequencies above. In our case, we select rad/sec.

We press ENTER. A message appears, announcing that the units of the vertical axis are decibels (dB), and those of the horizontal axis are the logarithm of the frequency in radians/second.

We press ENTER. After about a minute, the diagram appears.

This graph is the correct one. Obtaining it allowed me to receive full marks on the assigned homework. Having obtained it with a simulator made by myself, made me in addition worthy of the respect of Professor Yee, whom I hold in very high esteem.

Here the homework ends. However, suppose we had also been asked for the phase diagram. There is no need to run the simulation again, because the variables `v4` and `v5` are still in memory. It is enough to run the `bode` tool again:

```
sq\bode()
```

The previously entered values are still there. This is thanks to certain variables the tool generates, which can be deleted at the end. We leave everything the same, but select Phase Plot.

We press ENTER. A message appears, announcing that the units of the vertical axis are radians, and those of the horizontal axis are the logarithm of the frequency in radians/second.

We press ENTER. After about a minute, the diagram appears.

Having finished, we run the tool again and choose Clean & Exit to clean up the created variables.

12.3 The plot tool

The `plot` tool allows graphing functions of time t. It is executed like this:

```
sq\plot()
```

It is also very easy to use. Let us learn to use this tool by solving a real problem. We will use a problem that Professor Eliane Boulet de Cabrera, of the Technological University of Panama, presented to her students in an examination. In that examination the graph of the answers was not requested, but we can use it to illustrate the use of the tool.

Problem Nº 088

Problem: Given the following circuit, find the complete response for $v_2(t)$ and $v_1(t)$ for $t > 0$. Then, graph the value of $v_2(t)$ and $v_1(t)$ vs time, from 0 to 7 seconds.

Solution: We will use exact values for precision, but we will save time using FD instead of TR, because in the middle of an examination time is very valuable. We give the command to run the simulation in direct current, to find the initial conditions:

```
sq\dc("e1,3,0,18;r1,3,1,8;cc1,1,0,1/6;r2,1,0,12;r3,1,2,18;
      r4,2,0,6;cc2,2,0,1/3")
```

For `vcc1` we obtain 9. For `vcc2` we obtain 9/4. These will be our initial conditions for the next interval. We give the command to run the simulation in the frequency domain:

```
sq\fd("cc1,1,0,1/6,9;r2,1,0,12,0;r3,1,2,18,0;r4,2,0,6,0;
      cc2,2,0,1/3,9/4;js,0,2,10*e^(-t)*sin(2t+30°),0")
```

We press ENTER. The simulation executes, with the participation of the (Impala) Mode. Now we are going to obtain the answers, and at the same time we will store them in variables, to be able to invoke them later from the `plot` tool. We use `sq\fd_to_tr(v1)`. We obtain a very voluminous symbolic expression. For that reason, we use `expand(approx(ans(1)),t)→v1t` and obtain the complete answer in approximate form for the voltage in the capacitor on the left:

```
-.667*e^(-t)*cos(2.*t)-2.33*e^(-t)*sin(2.*t)
   +13.8*e^(-t/2)-4.16*e^(-t)
```

With `expand(approx(sq\fd_to_tr(v2)),t)→v2t` we obtain the complete approximate answer for the voltage in the capacitor on the right:

```
-13.7*e^(-t)*cos(2.*t)+5.17*e^(-t)*sin(2.*t)
   +13.8*e^(-t/2)+2.08*e^(-t)
```

Now we can execute the `plot` tool:

```
sq\plot()
```

A menu appears, in which we must insert three data:

Function. It is the function of time t we wish to graph. In our case, `v2t`. This is the name we gave the variable in which we stored the voltage in the time domain.

Minimum time. It is the minimum time of the graph, that is, the left end of the horizontal axis. In our case, 0.

Maximum time. It is the maximum time of the graph, that is, the right end of the horizontal axis. It must be greater than the minimum. In our case, 7.

We press ENTER. The graph appears.

Once we have this graph ready on the screen, we can apply over it all the graphing commands the calculator has. For example, the maximum value command. For more information on these commands, consult the User's Manual.

To obtain the graph of the voltage v_1, the same procedure is repeated, but entering `v1t` as the function to be graphed.

With this, we have finished the problem. As was to be expected, the `plot` tool can also graph functions that are not Symbulator answers. For example, if we enter as the function the following: $e^{-t}\sin(10t)$, and graph between 0 and 3 seconds, we will obtain its graph.

In this way, we have finished this book about the Symbulator.

Conclusions

Symbulator is a useful tool for the analysis of linear circuits. It allows the use of numerical and symbolic values. It can simulate in direct current, alternating current, the frequency domain and transient analysis. It gives as answers all the values that interest the user: voltages at the nodes, and currents, voltage drops and powers in the elements. It makes it easy to find the Thévenin, Norton and two-port equivalents of a network.

Because of its capabilities, Symbulator can be of great help in the teaching, learning and practice of Electrical Engineering, in the area of linear circuit analysis, especially in those circuit problems that require symbolic answers.

It is to be hoped that the technology developed in Symbulator, and matured during these two years of polishing, will serve as the basis for developing more and better products in the future, including programs for new platforms such as personal computers and portable computers.

Recommendations

I do not write these recommendations as the author, but based on my own experience as a user, and on the testimonials I have received in writing from the hundreds of students and professionals who have used Symbulator as a learning and verification tool, during its two years of development.

I recommend that professors use Symbulator as a teaching instrument in their circuit theory courses. I believe this simulator has a place in the classroom, due to its characteristics and portability. Professors can use it to verify the answer keys of their examinations before grading them, and as part of their classes.

I consider that the student should learn, in the first place, the manual techniques of circuit solving, and that in evaluations the professors should require them to present all calculations and steps manually. At the same time, I recommend that the student be allowed to use instruments like Symbulator, as a verification tool in classes, homework and examinations. I think that having a way to verify their answers will favor the learning and the self-confidence of the student. Especially because circuit theory textbooks do not include answers to all the problems they present, and because some of the answers they do present are wrong.

References

- *Engineering Circuit Analysis* (Análisis de circuitos en ingeniería), Hayt & Kemmerly, fifth edition (third edition in Spanish), McGraw-Hill, 1993.
- *User's Manual — TI-89 Calculator*, Texas Instruments, Banta Book Group, 1998.
- *User's Manual — TI-92 Plus Calculator*, Texas Instruments, Banta Book Group, 1999.

Appendix A

A brief history of Symbulator

The original thesis carried, as its first appendix, a brief history of Symbulator and a note on the language of the program and its documentation. When the translators began their work, they moved that material forward into the body of the book, where you have already read it: it is section 1.4, *Brief history of Symbulator*, and its subsection on language. It is not repeated here; this appendix remains in the plan of the book as a signpost, so that the numbering of the remaining appendices matches the original.

Appendix B

Symbulator and symbolic simulation

The knowledge that man has of electricity has been developing for centuries, from Volta's frogs to quantum transistors. The tools of electrical engineering analysis have developed as well. Symbolic simulation, being a very recent invention, is among the newest of these tools.

In 1998 I came upon the first symbolic circuit simulator I had seen in my life (and in fact the only one I know of apart from Symbulator). Its name is Syrup, and it is designed to run in the environment of the mathematical program Maple V for personal computers. Although I never came to master this simulator, thanks to it I understood that symbolic simulation is a very powerful tool for learning and design. At that time I was a student in the Circuit Theory II course at the Azuero Regional Center of the Technological University, and I felt it would be very useful to have a symbolic simulator available in the classroom. I struck up a good friendship with Joe Riel, the author of Syrup, and he explained to me the underlying concept of this program, an extremely simple concept in truth. Unlike numerical simulators, including some I myself had programmed up to that date, Syrup did not use matrices and modified nodal analysis. On the contrary, it generated nodal equations, solved them and presented them symbolically. With this idea in mind, on March 30, 1999, I undertook the programming of a similar simulator on my TI-89 calculator, which I initially called SCS (for Symbolic Circuit Simulator) and later Symbulator. Syrup was an initial inspiration, but not a model to follow, because my idea of the perfect simulator for the classroom differed in some respects from Syrup. Since, during the two years it took me to perfect it, I had no contact with any other symbolic simulator, I was forced to invent and develop by my own means the techniques of symbolic simulation that I present today, and which have rendered excellent results to the thousands of users that Symbulator has in the whole world. I have made this clarification because I consider that the work presented in this book should be taken for what it truly is: a pioneering contribution to a virgin field of electrical engineering. Perhaps that is why it earned first place in the IEEE Student Paper Contest 2000 for Latin America (Region 9).

Appendix C

When is it preferable not to use Symbulator?

An important skill that a Symbulator user must develop is that of recognizing when the simulation represents a saving of time and when it does not. This skill manifests itself as a kind of scent, of intuition, that tells you whether, due to the nature of a problem, it will be more complicated to solve it with a symbolic simulation than with manual techniques. It is acquired after much practice. Only your own experience will provide you with this special intuition.

Below, I present an example that illustrates a case in which it is advisable to decline the use of the simulator. It is a circuit that, at first sight, seems to be a perfect candidate for symbolic simulation in Symbulator. However, it turns out to be much simpler to use a single line with the calculator's `solve` command to solve it.

Problem Nº 089

Problem: In the circuit shown, it is known that $\text{abs}(I_1) = 7\ A$ *and* $\text{abs}(I_2) = 5\ A$. *Find* I_1 *and* I_2.

Solution: We will simply use the `cSolve` command:

```
cSolve(abs(i1)=7 and abs(i2)=5 and i1+i2=10,
       {i1=1+1i,i2=1-1i})
```

Note two aspects of this line:

We have written three equations to find two unknowns. This might seem to be an error, since one should have as many equations as unknowns. However, when we remember that each complex unknown is really two unknowns, we understand that we wish to solve for four unknowns and not two. The third equation can be considered, from this perspective, as two equations instead of one, since in it the following two equations are implicitly expressed: `real(i1)+real(i2)=10 and imag(i1)+imag(i2)=0`. So, in the command line shown above we have virtually four equations with four unknowns.

We have added an equal sign and a value following the name of each variable, when listing the unknowns. Instead of saying `{i1,i2}`, we have said `{i1=1+1i,i2=1-1i}`. These values are known as initial guesses, and their function is to give the calculator a starting point for the numerical resolution of the equations. The closer the starting point is to the real value, the faster the answer will be found. Observe that we have given a positive imaginary value to I_1, and a negative imaginary value to I_2. The reason is simple: we know beforehand that, since the branch of I_2 has a capacitor, its imaginary part will be negative, while the imaginary part of I_1 will be positive, since there is an inductance in that branch.

We press ENTER. The answers appear. Depending on the modes of the calculator, the answers will be shown in rectangular or polar form. Since we wish to see them in polar form, regardless of the modes, we use the `absang` tool, like this:

```
sq\absang(i1)|ans(1)
```

`ans(1)` refers to the answers obtained by the previous command. The number in parentheses indicates which previous answer we are referring to. We press ENTER. We obtain $7.\angle 27.66°$.

```
sq\absang(i2)|ans(2)
```

We press ENTER. We obtain $5.\angle -40.54°$.

Note how we have obtained these answers easily, using `cSolve`. Using Symbulator, the effort would have been much greater. That is why, in this case and in many other similar cases, it is preferable to decline the use of the simulator, and take another, simpler road.

Appendix D

Origin of the problems used

The problems in this book were, in their majority, taken from the following book: *Análisis de circuitos en ingeniería*, Hayt & Kemmerly, fifth edition (third edition in Spanish), McGraw-Hill, 1993. The exceptions have been noted. The original thesis closes with a table giving, for each of the 89 problems, the page of that book from which it was taken; the table is preserved in the archived Spanish original, and its exceptions are worth recording here: Problem 057 was contributed by Nevin McChesney (USA); Problem 066 by Jake Adams (USA); Problem 087 was set by a professor of the Technological University of Panama, as was Problem 088 (see Chapter 12).