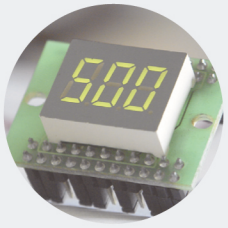
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






U=RIsolve: a web-based tool for teaching & self-learning the Nodal Voltage Method

LINO MANUEL CERQUEIRA SOUSA
novembro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Electrotécnica

Rua Dr. António Bernardino de Almeida, 431, 4249-015 Porto

U=RIsolve: a web-based tool for teaching & self-learning the Nodal Voltage Method

Mestrado em Engenharia Eletrotécnica e de Computadores

Lino Manuel Cerqueira Sousa

Supervision:

Prof. Mário Jorge de Andrade Ferreira Alves

Prof. André Teixeira da Rocha

November, 2020

Resumo

A iniciação à análise de circuitos elétricos durante o primeiro ano de Engenharia Eletrotécnica pode revelar-se uma tarefa desafiante. Desde a correta identificação dos Ramos, Nós e Malhas, à identificação das Leis de Kirchhoff e ainda a aplicação de teoremas de simplificação de circuitos, envolvem alguma complexidade. Num curto espaço de tempo, os alunos familiarizam-se com os simuladores de circuitos elétricos, sendo que estes lhes oferecem inicialmente uma validação para exercícios teóricos e análises experimentais, e mais tarde revelam-se fundamentais para o projeto e aperfeiçoamento de circuitos elétricos. No entanto, os simuladores tradicionais não demonstram como chegar aos resultados nem sequer as leis que os fundamentam. Assim, a presente dissertação aborda o design e implementação de uma *framework* de análise de circuitos elétricos – *U=RIsolve* – que fornece uma solução mais pedagógica para o Método das Tensões nos Nós (MTN). Este método permite obter a tensão em todos os Nós do Circuito em relação a um Nó de referência, e consequentemente a corrente em todos os Ramos.

A ferramenta *U=RIsolve* é uma aplicação web que utiliza um modelo descritivo de um determinado circuito (*netlist*), gerado pelo simulador QUCS, de forma a analisar e gerar uma solução por etapas, que descreve pormenorizadamente a aplicação do MTN. Para além da aplicação, este trabalho apresenta um algoritmo robusto e abrangente, baseado no paradigma dos Supernós, capaz de resolver circuitos de qualquer complexidade, ao contrário do MTN tradicional. O algoritmo proposto fornece ainda uma lógica de execução que facilita a sua implementação em sistemas computacionais e permite obter uma solução passo a passo, completa e intuitiva.

Palavras Chave

Método da Tensão nos Nós, análise de circuitos, abordagem dos Supernós, QUCS, simulador, netlists, aplicação web, auto-aprendizagem.

Abstract

The first steps for rookie Electrical Engineering students to learn circuit analysis are quite challenging. The correct identification of Branches, Nodes and Loops, the expression of Kirchhoff Current and Voltage Laws and the application of analysis/simplification Theorems and Algorithms, all embed tricky and error-prone steps for beginners. Hopefully, students get more and more used to circuit simulators with a very quick learning curve, turning these tools extremely appealing for validating theoretical and experimental analysis and, later on, a paramount help for designing, validating and fine-tuning hardware projects. Nonetheless, traditional circuit simulators neither explain how to obtain those results nor the fundamental laws behind them. In this context, this Thesis addresses the design and implementation of a circuit analysis framework – *U=RIsolve* – for teaching/self-learning the *Nodal voltage Method* (NVM, for short). This method enables to determine the voltage in every Node in the circuit, related to a reference (Ground) Node, and consequently the Current in every Branch.

U=RIsolve (read "you resolve") is a web-based application that uses a circuit description model (*netlist*) generated by the QUCS simulator, to analyse and output a symbolic step-based solution that describes how to apply the NVM and "solve" the circuit. Furthermore, this essay proposes a robust and comprehensive NVM algorithmic approach based on the Supernode paradigm – NVM-SA, which is able to tackle circuits with any complexity level, unlike the baseline NVM algorithm. The proposed NVM-SA algorithm provides a straightforward execution logic, making it suitable for computer implementation and enabling a step-by-step, very complete and intuitive output.

Keywords

Nodal Voltage Method, circuit analysis, Supernode approach, QUCS, circuit simulator, netlists, web-application, self-learning, JavaScript.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

API Application Programming Interface 13, 58

CAL Computer Assisted Learning 12

EE Electrical Engineering 1, 9, 58

GUI Graphical User Interface 12, 48

IVS Isolated Voltage Source 2, 17, 23, 56

KCL Kirchhoff's Current Law 1, 17, 25, 36, 79

KVL Kirchhoff's Voltage Law 1

MCM Mesh-Current Method 1

MNA Modified Nodal Analysis 19, 27

NVM Node-Voltage Method 1, 9, 23, 95

NVM-SA Node-Voltage Method – Supernode Approach 3, 17, 35, 49, 95

QUCS Quite Universal Circuit Simulator 2, 12, 47, 71, 95

SNA Supernode Analysis 20, 27, 35, 62

Acknowledgements

I would first like to express my sincere appreciation to my supervisors, Professor Mário Alves, an individual of an invaluable expertise, who always guided and encouraged me to carry out this work to the best of my ability, and Professor André Rocha whose support and insightful feedback kept pushing me to sharpen my thinking and bring this project to a higher level. It has been an honour working with both of you.

I would like to thank all of my Professors who accompanied me throughout my journey as a student at Instituto Superior de Engenharia do Porto. I would also like to acknowledge some people who helped on testing the *U=RIsolve* application, namely Ana Castedo and Miguel Gouveia. Thank you for your contributions.

In addition, I would like to express my gratitude to my parents who always supported and nurtured me the best they could. I would have never gotten this far without you.

Finally, I would like to extend my deepest gratitude to the most important person in my life, Gabriela, who is undoubtedly the source of all my inspiration. Thank you for your love and unwavering support.

To all of you, my sincere thanks.

Chapter 1

Introduction

Learning circuit analysis involves the study of a wide range of circuit laws and theorems, being a quite challenging task. At the early stages, the correct acquisition of the fundamental concepts and laws of circuit analysis proves to be crucial for continuous success in any Electrical Engineering (EE) course/degree. Notwithstanding, the delicate circumstances the world is currently going through due to the COVID-19 pandemic has made distance and self-learning the everyday life of many students, leveraging the importance of computer applications for EE education.

The present essay outlines a new algorithmic approach for a well-known electrical circuit analysis method named Node-Voltage Method(NVM), which is implemented in a web-based application named U=RIsolve. The U=RIsolve application uses the circuit description to output an interactive step-by-step solution of the NVM. Unlike traditional circuit simulators that merely outputs Branch currents and Node voltages, this tool explains the underlying procedures and calculations to obtain them, aiming to improve teaching and self-learning of electrical circuit analysis

1.1 Research context

During the learning process, students often have theoretical and practical difficulties in all analysis methods, namely the "baseline" *Branch currents Method*, based on the *Kirchhoff's Current Law* (KCL) and *Kirchhoff Voltage Law* (KVL), the *Superposition Theorem Method*, the *Mesh-Current Method* (MCM) – also known as "*Loop-Current Method*" – and the NVM. The latter two methods are particularly appealing for analysing more complex circuits with more Branches and fewer Nodes (NVM) and more Nodes and

fewer Branches (MCM), since they lead to a reduced number of equations/steps (comparing to the "Branch current Method").

Nevertheless, when applying the MCM/NVM methods, beginners often fail in the quantification and selection of the right Loops (for the *Mesh-Current Method*) and Nodes (for the *Node-Voltage Method*). Even when this selection is correctly performed, there is another error-prone step: writing the corresponding equations – KVL based on the Loop currents, for the MCM, and for the selected Nodes, for the NVM. These weaknesses become even more acute when the circuits (DC or AC) under analysis include current sources – for the MCM – or Isolated Voltage Sources (IVS)[1] – for the NVM, although in either case the equation systems get smaller, since the current in a branch with a current source and the voltage between Nodes connected to an IVS are known *a priori*.

In this context, students may take advantage of traditional circuit simulators to perform a validation by comparing their analytical/experimental results against the ones provided by the simulator. However, while simulators are paramount tools for testing, designing and fine-tuning electric/electronic circuits, they purely output "measurements", e.g. Branch current and Node voltage values. In fact, simulators do not show the way to get to those values, i.e., how to apply the analysis methodology, which the student is supposed to master at the end of the course.

To fill the lack of educational tools for teaching and self-learning circuit analysis, this work proposes an application – dubbed *U=RIsolve* (read "you resolve") – which aims at showing how to apply the circuit analysis methods, specifically the NVM. Users must previously implement and simulate the circuit in *Quite Universal Circuit Simulator* (QUCS), explicitly identifying all Nodes with letters (except the *Ground* Node, which has a specific symbol). The *U=RIsolve* application then takes (as input) the "*netlist*", a text file output by QUCS that unequivocally represents the circuit under analysis, and generates a step-based solution of the Node-Voltage analysis method.

1.2 Objectives

The framework covered in this essay aims to enhance both teaching and self-learning of electrical circuit analysis, taking into consideration the most common issues that may constrain these processes, for instance, the misunderstanding of basic concepts and ter-

[1] In this work context, an *Isolated Voltage Source* is assumed as an ideal voltage source (without internal resistance/impedance), connected in a Branch without any other components, fixing the voltage between the two nodes delimiting that Branch.

minology, the electric circuit analysis fundamental laws or simply arithmetic calculations throughout the execution of a certain analysis methodology. Thereby, *U=RIsolve* application enables:

- The student to learn how to apply the NVM to any AC/DC circuit (linear and passive);
- The student to compare the output of the *U=RIsolve* application to his/her own methodology and the Branch currents/Nodes voltage computed in QUCS with the analytical ones;
- The educator to produce practical examples, both off-class (e.g. to prepare exercises, lab scripts or lectures) and in-class (in real-time, demonstrating the application of the NVM through case-studies).

Moreover, this application intends to introduce a didactic component in the electrical circuit simulation tools, providing a comprehensive step-based solution, and not just the results contrarily to traditional simulation software. The goal is to give students a detailed methodology of resolution and consequently improve their learning process.

1.3 Contributions

A preliminary version of the *U=RIsolve* application [1, 2] has been conceived at an earlier stage. Nonetheless, it was originally built on a baseline NVM algorithm that was unable to tackle more complex circuits and had a rudimentary user interface which could not produce a thorough analysis/output. The new framework issued in this work is a complete rebuilt application with different functionalities, in particular:

- A new algorithmic approach to the NVM (*Node-Voltage Method – Supernode Approach* (NVM-SA)) that supports circuits with multiple IVS;
- An interactive, flexible and responsive web-based user interface that adapts itself for any screen size/resolution, thus enabling the execution on different platforms (PC, Tablet, smartphone);
- A comprehensive output of the analysis results, showing users a step-by-step solution;

- Mechanisms of results exportation (as PDF, JSON and TEX) which allow to save and share each circuit analysis;
- An error detection system that warns users about their simulation flaws (in QUCS);
- Ready-to-use circuit examples of different complexity that can be used/analysed with a single click;
- A publicly available website[2] that avoids local installation.

This Master's Thesis encompasses the following contributions:

1. It is proposed a new algorithmic approach to the NVM, dubbed *Node-Voltage Method – Supernode Approach* (NVM-SA), building on the "Supernode paradigm" [3] that simultaneously guarantees: 1) Robustness and generality, since it copes with (passive and linear) electrical circuits of any type/complexity, DC or AC; 2) Step-by-step execution logic, comprehensive to human intuition, building on the fundamental laws of electricity (*Ohm's Law*, *Kirchhoff's Current Law*); 3) Straightforward coding (in any programming language).
2. A web-based application named *U=RIsolve* has been designed and made publicly available, that instantiates the NVM-SA algorithm, enabling to analyse any circuit (that has been previously simulated in QUCS) with a structured step-by-step output that mimics the proposed NVM algorithm and thus is very complete and intuitive to the user.
3. This report/dissertation has been planned and structured to serve as pedagogical material, which resulted in considerable amount of educational elements being offered to students, both in the report and in the application, through the multiple NVM algorithms description and demonstration, theoretical concepts, case-studies and their step-based resolution.
4. The development of the *U=RIsolve* framework along with a new algorithmic approach resulted in the submission of a scientific paper [4] to an international journal (in Appendix B), whose inherent tasks such as journal analysis and selection process, as well as the research and writing phases were an integral part of the work plan and personal growth.

[2]The latest version is currently available at: http://urisolve.ddns.net/urisolve.

The *U=RIsolve* framework is a collaborative effort and each author's contributions are illustrated in the contributions diagram in Fig. 1.1, whereby the project is segmented into three categories. The first is the design and validation of the NVM-SA algorithm that supports the framework. The second addresses the technical implementations of *U=RIsolve*, in particular the algorithm implementation, the software architecture and modules, the user interface, the analysis output design and the testing phase. The third section concerns the submitted manuscript.

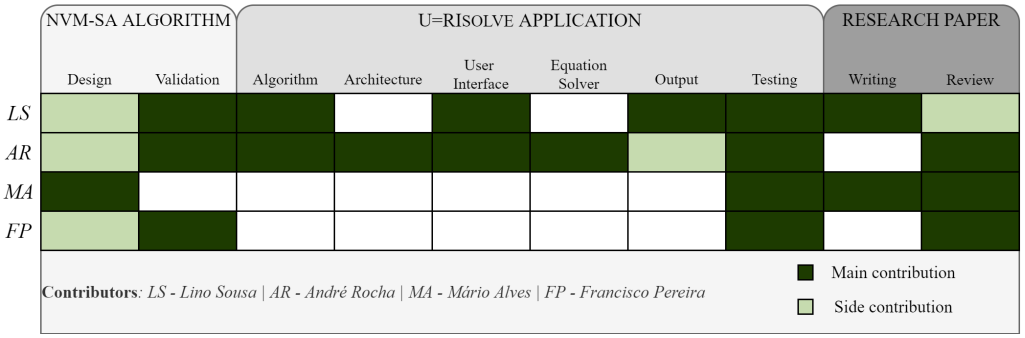


Figure 1.1: Author contributions diagram.

1.4 Research strategy

The development of this application began with the structuring and formalisation of a robust, general-purpose NVM algorithm (NVM-SA) to support the application so that it would be able to solve any AC/DC electrical circuit. Subsequently, it was necessary to deal with the QUCS and *U=RIsolve* interaction issues, related with the *netlist*. Afterwards, the NVM algorithm was employed and fine-tuned into a web-based application through JavaScript. The final stages included the design and development of the user interface and algorithm step-by-step output, as well as extra functionalities, in particular the extraction of results in different formats (PDF, JSON, TEX) and the accessibility of ready-to-use examples. This sequence of tasks were performed in order achieve the work reported in this essay, and it is represented in Fig. 1.2.

1.5 Dissertation structure

The present dissertation is organised into seven chapters. Beginning with a contextualisation and problem statement, Chapter 1 also includes the research approach and tasks scheduling (Gantt chart). Chapter 2 overviews the most popular electrical circuit simu-

lators currently available, as well as some widely known educational tools both in EE and other fields. The Chapter culminates in an examination of the existing NVM algorithm variants in the literature. This leads to the Chapter 3, where a validation of each version of the NVM is performed in order to identify their strengths and shortcomings. Chapter 4 proposes the NVM-SA algorithm (used in the *U=RIsolve* application) along with the definition of fundamental concepts and terminology and a practical model for the algorithm implementation. Chapter 5 addresses the technical implementation aspects of the developed framework, by tackling the issues related to *netlist* information extraction, the creation of new data structures, the analysis methodology, the output and the user interface. Chapter 6 features a set of case-studies that demonstrate the results provided by the *U=RIsolve* application analysis for different circuit size/complexity. Ultimately, Chapter 7 summarises the achievements and contributions together with lessons learned and future work within this framework.

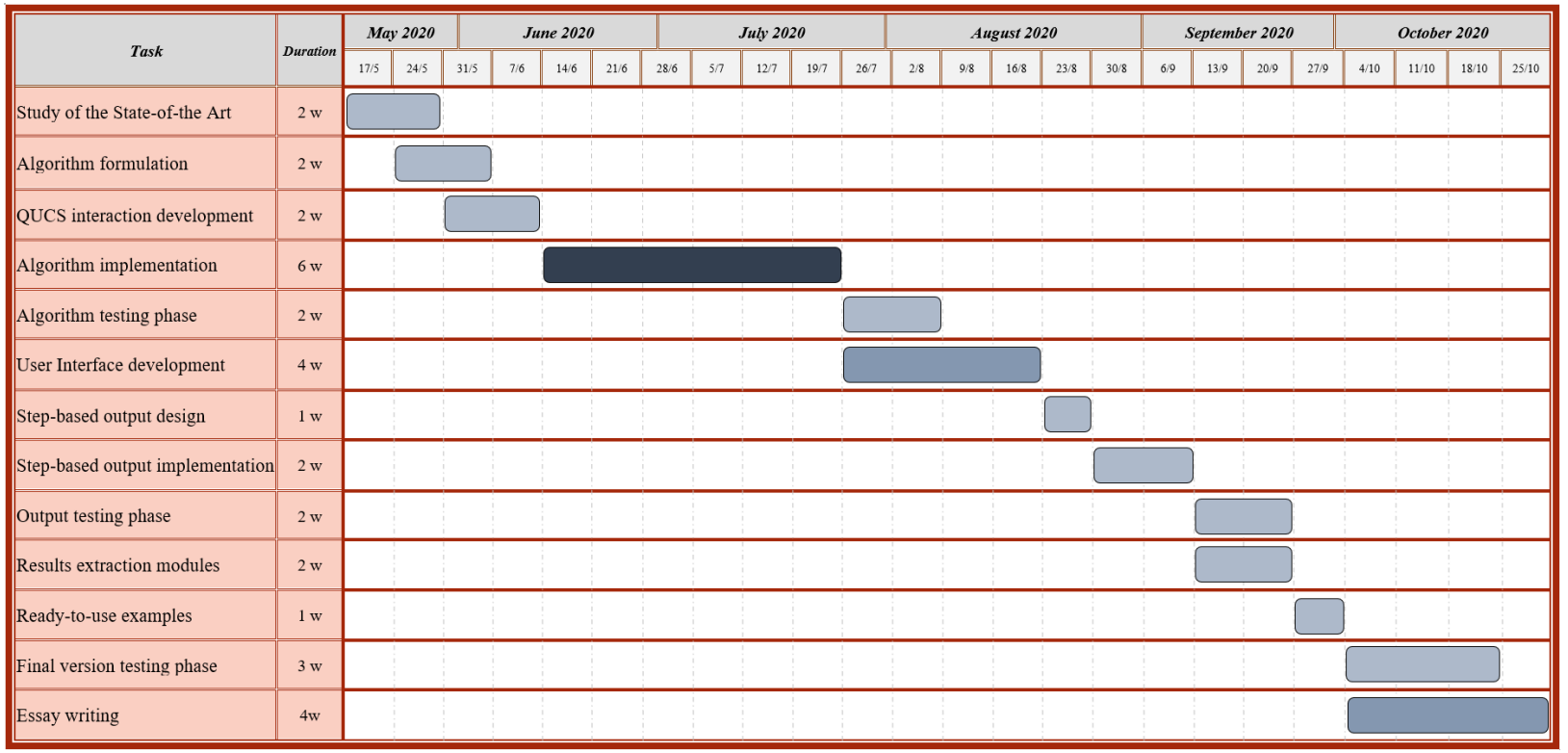


Figure 1.2: Project development schedule.

Chapter 2

State of the Art

The teaching/learning process has undergone continuous mutations over the years. With new technological resources emerging, several teaching methods, tools and applications started to evolve in order to improve the effectiveness of conceptual knowledge acquisition. Significant amount of educational content available nowadays is computer-aided, including simulation software, virtual laboratories and distance learning applications. This chapter addresses computer-based tools for electrical circuit analysis and some other fields. It also overviews the different NVM approaches, as it is the baseline circuit analysis method of the U=RIsolve application.

2.1 Electrical circuit simulators

The selection process of a proper circuit simulator depends on the user's demands. In fact, the requirements for a simulation tool in a professional environment differs from what students seek while learning circuit analysis. Given the context of this work, the analysis of the current electrical circuit simulators focuses on those whose characteristics are mostly relevant to EE students.

EveryCircuit

EveryCircuit [5] is an online circuit simulator that stands out due to the designed graphical interface that allows: real time simulation and parameters adjustment during "run-time" (while the simulation is running); animated representation of dynamic magnitudes such as voltages, currents and charges, providing a detailed visualisation of the basic laws of electricity. Apart from the browser version, EveryCircuit is also available

for Android and iOS and offers multiple pre-designed circuits. The application includes a community tool that gives the opportunity for users to share circuits and import them into their workspace. The use of this application without restrictions costs an annual value of around 13 €, although the free version covers the most important aspects of circuit analysis, however this trial version expires after 24 hours of use, which is the main drawback.

PartSim

PartSim [6] is a web-based circuit simulator with a full SPICE simulation engine. This tool contains two unique features: a graphical wave form viewer, which is a valuable support for the analysis, and a schematic capture tool that allows to export the analysis and the circuit into a PDF document. Moreover, the simulator is entirely free and lets users create their specific SPICE models and introduce them in their simulation. The interface is easy to use, requires no registration and comes with practical examples ready to be executed. Nevertheless, it shows considerable simulation delays for more complex circuits.

LTspice

The LTspice simulator [7] is a high-performance SPICE-based software known for the developments/upgrades on the SPICE algorithm, in particular the fast simulation of switching mode power supplies for enhanced step responses, steady state detection, which enables users to view waveforms for most of switching regulators. The popularity of this platform came mostly from the wide range of SPICE models available and the extensive list of demonstration circuits, since this software is free of charge. The single drawback of LTspice concerns the old-fashioned user interface with a troublesome learning curve.

CircuitLab

CircuitLab [8] provides online tools for schematic capture and circuit simulation. This web-based tool has a well-conceived and easy-to-use user interface, along with a system of electronic design (Smart Wires) to accelerate the process of circuit drafting. The components library is available without charges, however the free version has a maximum limit of 20 components per schematic, which is the main disadvantage of this simulator. Students have an annual paid plan of 20 € that increases the component limit to 30, or a 70 € annual plan with unlimited size circuits.

Circuit Sims

Amongst the first open-source web-based circuit simulators, Circuit Sims [9] is a simple platform ideal for beginners who want to learn the basics of circuit analysis. While the graphical interface does not have any advanced features, the simulation results are intuitive. Furthermore, this application shows the current flow phenomena, which is not easily comprehensible at the early learning stages. The components library is not extensive, though it addresses the basic electrical/electronic elements. Circuit Sims also has several embedded circuit examples ready to explore.

EasyEDA

EasyEDA [10] is a web and cloud-based electronic design software that integrates three main features: mixed-mode circuit simulation; schematic capture; and printed circuit boards layout design, in a cross-platform browser environment. The great advantage of this tool is the large parts library that users can take advantage at no charge. Additionally, there are plenty available circuits across the community, given that EasyEDA is one of the most popular simulators and users can publicly share their works. The downside of this application concerns the difficulties on obtaining the simulation results, since users need to go through several steps before accessing them, even though it is a key part of the simulation which users want to obtain right after it.

Multisim

Multisim [11] also builds on a SPICE simulation engine, supporting an interactive and intuitive interface prepared for real-time circuit behaviour analysis. This platform has an education-oriented version with benchtop instruments, providing different analysis methods, a hands-on learning environment for real and simulated signal comparison, and an educational laboratory to allow seamless integration with hardware. Moreover, the simulator recently introduced a web-based "live" version that uses the same simulation technology, optimised for web browsers, enabling users to create, simulate and share circuits in any device. The free license provides access to a basic library of 78 components and limits the number of elements per circuit in 25.

QUCS

QUCS [12] is an open-source integrated circuit simulator, currently available for almost any operating systems, including mobile. This software comprises a thorough component library, allowing users to add new parts by importing their SPICE models into the simulation. Despite the fact that the *Graphical User Interface* (GUI) has room for improvement, it is clean and remarkably easy to use. Another great advantage of QUCS concerns the manifold simulation types and diagrams that support DC, AC, S-parameter, transient/noise analysis, and sub-circuit hierarchy.

System Vision

System Vision [13] is a cloud-based browser simulator designed for circuit sharing that allows users to freely create unlimited public circuits and embedded live designs. However, if users do not want their work to be publicly available to the community, they will have to opt for the fee based plan, which costs at least 200 € per year. The interface is quite simple and the simulation process is straightforward. Furthermore, the interactive measurement probes facilitate the interpretation. Nonetheless, the free plan has two critical drawbacks: a cloud disk space limit of 200 MB and a runtime credit limit of 50 per year (a runtime credit is measured in normalised computer work, and this software considers 5 minutes of simulation execution as 1 runtime credit).

2.2 Computer-Aided Learning tools

The education process concerns the transmittal of conceptual knowledge and methods of teaching, as well as sustaining a conducive learning environment. One of the most effective contemporary learning methodologies is the *Computer Assisted Learning* (CAL) [14], which uses technology, such as computers, smartphones and other electronic devices to provide didactic instruction. In the past few years, combining technology with course content has proven to be more appealing to students, as it ensures a more interactive learning experience [15].

The *U=RIsolve* major objective is to teach all the steps of a specific electrical circuit analysis method, in particular the NVM. In order to accomplish this goal, it is essential that students are able to produce the equation system without mistakes, which implies making correct substitutions and simplifications. For this reason, a step-based solution is provided. Hereupon, this section evaluates a few well known applications focused on

the mathematical learning process, as well as EE studies and applications aiming at enhancing existing didactic methodologies.

2.2.1 Mathematics education software

Undergraduate students often struggle to reach the correct results for a specific electrical circuit analysis method due to miscalculations, wrong substitutions or errors in the simplification of equation systems. The following tools intend to provide support in mathematical subjects such as trigonometry differential and integral calculus by means of interactive interfaces with step-by-step solutions that guide students through the correct resolution process.

Wolfram Alpha

Wolfram [16] started as a computational platform (*Mathematica*) which addressed topics such as calculus, algebra, statistics, geometry and differential equations. This way, users were able to solve specific maths problems and find the necessary information about these subjects. The Wolfram Alpha's user interface provides Application Programming Interfaces (APIs) to enhance the learning process, mainly plots, graphics, formulas and step-by-step solutions. However, some of these tools are only available in the pro version that costs around 50 € a year. Nowadays, Wolfram Alpha has evolved into a computational knowledge engine that uses Artificial Intelligence to provide answers about a vast set of topics: Physics, Chemistry, Engineering, History, Finances, Entertainment and many more. Answers are obtained through a built-in knowledge base of structured data originating from books, journals and websites.

Symbolab

Released in 2011, Symbolab [17] is an online mathematical education tool and answer engine that provides step-by-step solutions to algebraic, trigonometric and calculus topics. The interface is able to render symbolic expressions, equations and formulas by means of machine learning algorithms, making it possible for the website to understand both meaning and context of users queries. Since the application goal is to turn scientific content universally available, the existing tools cover middle school to college subjects. Symbolab contains a long list of learning tools: how to use examples; a digital notebook to keep track of the math problems; a graphing calculator; and a practice tool with

quizzes and stats about the user performance. It also includes teaching features such as "Groups" that allows the teacher to give quizzes to students and access their results.

Mathway

Mathway [18] is a web-based mathematical problem solver that intends to generate on-demand math assistance accessible to students, parents and teachers. The free version just gives the final numerical results, which means users must pay the subscription fee of 35 € a year in order to get access to the step-based solution. This application also includes a wide variety of math topics such as algebra, calculus, trigonometry and geometry. The interface is easy-to-use, since it has a format of a common message chat; users just need to expose the problem, using the screen scientific keyboard and write the desired operation so that it is possible to get an immediate answer. Furthermore, users can upload or use camera (in case of the mobile app) to take a picture of the problem, and Mathway has tools to recognise the characters and compute the solution.

PhotoMath

PhotoMath [19] is one of the most used mathematics learning apps by students all over the world, solving around 1.2 billions of problems monthly. This mobile app solve a wide range of problems using the camera on the mobile phone. First, PhotoMath recognises handwritten or printed characters on the user-cropped area through advanced optical character recognition. Then, the characters (numbers, symbols and letters) are examined in relation to each other and the mathematical formula is designed. Finally it is applied a solving algorithm and the step-by-step solution is provided. Additionally, the app comprises a few extra learning tools, e.g. animated instructions and calculation steps, a smart calculator with an intuitive math keyboard, and graphs to visualise functions.

Khan Academy

Khan Academy [20] is an online learning web-page that offers interactive courses in multiple subjects, in particular math, science, computing, economics and history. In the math section, users can find almost every subject from the first grade to high school and university level. The available resources vary whether users are students or teachers, nonetheless they include videos, articles covering both theory and examples with step-by-step solutions. In the science section, there is also an EE course that explains circuit analysis, electrostatics, amplifiers, signals and systems, and semiconductor devices. Khan

academy is free, being available in more than 30 languages, and the ideals behind this tool is that anyone should be able to learn, since education is a human right.

2.2.2 Learning tools in Electrical Engineering

The improvement of quality and effectiveness in teaching EE has been focus of research in the last two decades. These studies mainly addressed the creation of new computer-based tools [21, 22, 23, 24, 25, 26] and the enhancement of existing ones with new methods [27, 28, 29, 30]. The *U=RIsolve* framework is part of the latter approach, i.e. it uses QUCS to implement/simulate the circuit under analysis and to generate an input to the application, thus complementing the ordinary simulation procedure with a didactic component, based on a step-by-step solution of the NVM algorithm. This type of instruction has been studied in comparison with traditional teaching methods. Statistical studies [31] have shown that *step-based tutoring* systems had a Cohen d-value[1] of 0.76 standard deviations (σ), which has a roughly accurate approximation to the 0.79σ average outcome of expert human tutors. On the contrary, *answer-based tutoring* systems obtained average d-values of 0.31σ. Furthermore, another study concerning the impact of a *step-based tutoring* system for linear circuit analysis [32] revealed that this step-based trainer had learning improvements of 0.72σ in relation to the conventional publisher-based homework system, for Node voltage analysis. Hence, *U=RIsolve* shows users a complete and straightforward step-by-step solution for any DC/AC circuit with several hints on how to correctly apply the Node Voltage Method.

Among the earliest educational tools for learning electrical circuits was KIRCHHOFF [33], a computed aided learning software package for symbolic analysis of electrical circuits. This tool allowed students to select a circuit from the library, set its parameters such as Nodes and elements labels, Branch currents and voltages. The student would then introduce the Kirchhoff laws in order to obtain the circuit model for nodal or loop methods. Ultimately, KIRCHHOFF software would validate the equations and output an appropriate corrective explanation in case the equations were not properly formulated. Although this software package brought substantial value to *Computer Aided Learning*, its architecture limits its scalability, i.e. students cannot draw their preferred problems, instead they are given a set of predefined circuit examples from an intern library. Moreover, the software was developed specifically for a category of microcomputers (IBM-PC compatible), which is nowadays out of the technological context.

[1] Cohen's d is defined as an effect size used to indicate the standardised mean difference, being calculated through the difference between two means divided by the standard deviation.

With the increasing interest in online education, more web-based applications like *U=RIsolve* started to be developed, as they bring many advantages to undergraduate students, who have real-time access to problems and solutions, which dramatically improves distance-learning. Several works have produced web-based applications to promote success in electrical circuit analysis, providing students with online courses [34], remote homework systems [35] and virtual laboratories for circuit simulation [36, 37, 38].

Examples of educational web-based applications include the work described in [39], where authors designed an interactive textbook of examples (continuous-time linear circuits) that are submitted by administrators (tutors), along with analysis details such as circuit image, *netlist*, problem statement and component values. Solutions are then analysed by users in a symbolic and semi-symbolic form. Another web-based system is presented in [40], which also uses symbolic analysis techniques to teach a set of theorems such as Thévenin's theorem and voltage divider. Every circuit data is stored in a problem-specific database and the students statements are registered so that the instructors can follow the students progress.

However, most of these works have a limited range of circuits and simply vary elements values, Branch currents and Nodes labels. Once the student solves the available set of exercises, and still cannot fully understand the methods, this type of application falls short. Moreover, a finite dataset of examples may not comprehend every aspect of circuit theory that generates a wide range of difficulties in students, which could prevent their progress. There are two different ways of overcoming these problems. The first is to allow an user interface (internal or external to the application) to draw circuit schematics (as *U=RIsolve* does through QUCS), or automatically generate random problems with different layouts. For instance, *SapWin* [41, 42, 43, 44] uses the first approach, providing an interface for users to draw circuit schematics, performs a symbolic analysis through Laplace domain and returns the circuit transfer function. The symbolic method used to solve electronic circuits has the same goal as the *U=RIsolve* application: enhance the learning process.

Although the ideals underlying both applications (*U=RIsolve* and *SapWin*) are very similar, *U=RIsolve* focuses on teaching methods to accomplish a correct circuit analysis, while *SapWin* intends to introduce a new system of linear analog circuit analysis by studying its transfer function. On the other hand, authors from [45, 46] designed a step-based computer-aided tutoring system to teach linear circuit analysis where the first approach is used, i.e. it generates a new circuit diagram after showing students the

correct answers and solutions to the previous one, so that users have access to unlimited examples. These works address the series-parallel simplifications and the Nodal and Mesh analysis.

2.3 NVM algorithm approaches

The *Node-Voltage Method*, or just *Nodal Analysis*, is an electrical circuit analysis method expressed in accordance with KCL, i.e. the algebraic sum of all currents flowing to one Node must be equal to the sum of all currents leaving the same Node. The NVM is based on the concept of Node voltage – the electric potential in a specific Node related to the *Ground* (reference) Node (0 V). The NVM has gone through some adaptations over the years [47] in order to ensure that it suited special circuit scenarios, with IVS. In this section, the baseline NVM algorithm and other variants are addressed, namely the *Direct Matrix Assignment*, the *Modified Nodal Analysis* and the *Supernode Analysis*. Although a formal definition of the fundamental terminology is carried out in section 4.1 given that it serves as the foundation of the proposed NVM-SA, one should notice that a few concepts are exhibited in the following algorithms.

2.3.1 NVM classical approach

The baseline NVM [48] follows a simple set of rules to obtain the Node voltages in accordance with the Algorithm 1:

Algorithm 1 The NVM classical algorithm

1: Identify all the Nodes in the circuit.
2: Choose a reference Node, where the voltage will be zero.
3: Identify the currents in all Branches ($I_1, ..., I_R$) and their flowing direction, being R the number of Branches in the circuit.
4: Write the equation for each current, function of Node voltages ($I_i = f(U_1, ..., U_N)$, $i \in \{1, R\}$), using Ohm's Law and taking into account the chosen directions for currents.
5: Write the KCL for each Node ($U_1, ..., U_N$) based on the Branch currents ($I_1, ..., I_R$).
6: Solve the equation system: $N-1$ equations and $N-1$ variables.
7: Determine each Branch's currents ($I_1, ..., I_R$) from the Node voltages ($U_1, ..., U_N$) computed in Step 6, through the equations from Step 4.

This method has, however, been adapted by inspection of the circuit schematic diagram, since the algorithm is not solvable in certain circumstances. Therefore, work from [49] proposes a new method of writing the *Nodal Analysis* matrix equations for linear circuits, which derives a general matrix solution for Node voltages. The matrix solution includes an independent-source vector, an inspection matrix and naturally the Node-Voltage solution vector.

2.3.2 NVM direct matrix assignment

An alternative to the classical NVM approach concerns the direct matrix assignment of Node voltage equations, being written in the following form:

$$G \cdot v = i \tag{2.1}$$

Where G is the Conductances matrix, the v vector holds the unknown Node voltages and the i vector holds the currents flowing in and out of the Node. As illustrated in Fig. 2.1, the first matrix (G) is filled with the Nodes *Conductance*. The main diagonal features the Conductances that are directly connected to each Node, e.g. G11 to Node 1, G22 to Node 2 and so on. The coefficients outside the main diagonal represent the "common" Conductances between two Nodes. For instance, a Conductance that is connected to both Node 1 and 2 should be in both positions (1,2) and (2,1) of the matrix.

$$\begin{bmatrix} G_{11} & -G_{12} & -G_{13} & \cdots & -G_{1K} \\ -G_{21} & G_{22} & -G_{23} & \cdots & -G_{2K} \\ & & \cdots & & \\ -G_{K1} & -G_{K2} & -G_{K3} & \cdots & G_{KK} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \cdots \\ U_K \end{bmatrix} = \begin{bmatrix} I_{11} \\ I_{22} \\ \cdots \\ I_{KK} \end{bmatrix}$$

Figure 2.1: NVM – Direct matrix assignment

The second matrix, v, which multiplies by the first one, is associated to the variables to be calculated – the Node voltages. The last matrix, i, refers to the sum of all the currents flowing to the associated Node, their direction depending on the voltage source polarity, i.e. the voltage sources electromotive force's direction or the current direction of the current sources that directly influence the Node.

Similarly to the classical NVM approach, this method fails with circuits featuring IVS, since they cause infinite Conductances in the G matrix. As a result, authors from [50] also overcome the limitations of the direct matrix assignment NVM algorithm, by introducing the concept of "virtual current sources" that replace non-convertible (isolated) voltage sources and correspond the current flowing through that particular Branch.

2.3.3 Modified Nodal Analysis

The two aforementioned analysis methodologies do not cover every electrical circuit, since they are susceptible to a singularity caused by infinite Conductances. This situation occurs whenever there are Isolated Voltage Sources (also known as floating sources), that cannot be directly or indirectly (through other voltage sources connected to it) referenced to the Ground Node.

To overcome this problem, a method named *Modified Nodal Analysis* (MNA) was proposed in [51], and then consolidated in [52, 53]. The MNA is a universal method that can handle any type of electrical components and linear circuits thus serving as an extension of the traditional nodal analysis. Therefore, this method is the preferred choice for computerised analysis [54] and has become a commodity for electrical circuit simulators [55]. To our best knowledge MNA was first introduced in PSpice [56] and since then by most Spice-like simulators [57, 58, 59, 60, 12, 7, 11, 6].

The MNA application to an electrical circuit results in a matrix equation represented in the following expression:

$$[A] \times [x] = [z] \tag{2.2}$$

The A matrix, decomposed in Eq. 2.3, is composed of four sub-matrices, each one containing information about the elements directly related to each Node and also parameters related to the existence of isolated voltage and current sources. The x matrix holds the system variables i.e., the voltages in the different Nodes and the currents flowing through the IVS. Finally, the z matrix includes the sum of all the known currents and voltages in the circuit.

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix} \tag{2.3}$$

As previously mentioned, the A matrix contains four matrices: G (identifies all the circuit connections), D (indicates if each voltage source is isolated or not, B and C (both

include the voltage sources connections). The G matrix, is the sum of the elements' Conductances linked to each one of the Nodes (in the first position of the diagonal (1,1) for Node 1, in the second position (2,2) for the Node 2 and so on). The remaining matrix positions are filled with the negative Conductances of Branches connecting two different Nodes.

The Nodal Analysis unknowns are obtained through the solution of the matrix equation from Eq. 2.2. Thus, MNA allows to overcome the infinite Conductances problem caused by IVS, but adds an extra equation to the algorithm. Despite that, its efficiency and universality made it stand out in relation to the classical NVM approach, particularly towards software implementation.

2.3.4 Supernode Analysis

Despite its versatility, MNA has not been well accepted in the teaching of circuit analysis, since it is more a mathematical artefact, not intuitively reflecting the logic of electric circuits and its fundamental laws. Hence, an alternative to MNA – named *Supernode Analysis* (SNA) – was proposed in [3] and later revisited in [61, 62]. SNA is a straightforward algorithm, that intends to turn Nodal Analysis accessible to undergraduate students. This method relies on the concept of a *Supernode*, which is a sub-circuit or cut-set formed by one or more IVS.

Considering the Supernode illustrated in Fig. 2.2, if the voltage source (E) between two different Nodes (A) and (B) has no impedance, then the relation between potentials U_A and U_B is automatically known.

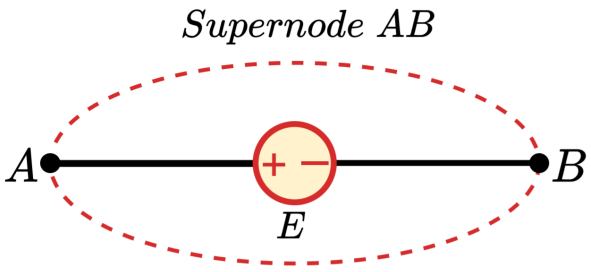


Figure 2.2: Example of a Supernode.

Taking Nodes A and B as a single Supernode AB reduces the degree of freedom of the equation system, but it requires introducing the relation between Nodes A and

B (the Isolated Voltage Source). However, if one of these Nodes' potential is known or Grounded, the other can be immediately determined (through E). For the given example, the equation that correlates both Node voltages is represented in Eq. 2.4.

$$U_B + E = U_A \Leftrightarrow U_A - U_B = E \tag{2.4}$$

The SNA algorithm consists in four main steps, in order to set up the needed equations to solve any electrical circuit by hand, which means it is not ready to be implemented on a computer. These steps are listed in Algorithm 2.

Algorithm 2 The SNA algorithm

1: Identify all the Nodes (N) in the circuit, one of which has to be the reference Node. This leads to $N-1$ Node voltages as independent (unknown) variables.
2: Mark all the Supernodes by enclosing all the cut-sets formed by voltage sources (T) with a closed line.
3: Set up the forced conditions (relation between voltages) for each Supernode. These conditions should be used to eliminate T Node voltages. Take a reference Node voltage from each Supernode as the independent variables.
4: Output the $N-1-T$ equations (one equation per Supernode and another for each regular Node).

One can note that the SNA reduces the number of unknown Node voltages, comparing to the MNA, since it avoids the calculation of currents whose Branches have Supernodes. Additionally, the Supernode equations are simple voltage relationships, which simplifies the algebraic operations. As a result, in relation to MNA, Supernode Analysis is more appropriated in the early circuit analysis learning stages. Nevertheless, given that the SNA has not been adapted for computerised implementation, there is a need to fine-tune the algorithm for this purpose which is one of the main contributions of the present Thesis. The forthcoming chapter performs a practical review of the four NVM variants introduced above.

Chapter 3

NVM Algorithms Validation

Four methods for solving electrical circuits in relation to its Node voltages were presented in the previous chapter. In order to fully understand how the different NVM algorithms get to the solution, as well as their capabilities, advantages and drawbacks, an evaluation process is performed in this chapter. An introductory example without Isolated Voltage Sources is first analysed through each one of the NVM variants, and a more complex circuit with an Isolated Voltage Source is further examined in order to check which methods fail due to the infinite Conductances singularity.

3.1 Circuit without Isolated Voltage Sources

The first circuit to be analysed is represented in Fig. 3.1 and it has no IVS, as mentioned before. The circuit has two unknown Node voltages (U_A and U_B), and the Branch currents flow is set by the black arrows. The two current sources set the remaining Branch currents flow, as well as their value.

Analysis through NVM classical approach

The resolution through the NVM classical algorithm follows the steps from the section 2.3.1, expressed below.

Step 1: All Nodes are correctly identified in the circuit image, namely Nodes A and B.

Step 2: The reference Node is properly placed in one of the circuit's Nodes (on the bottom).

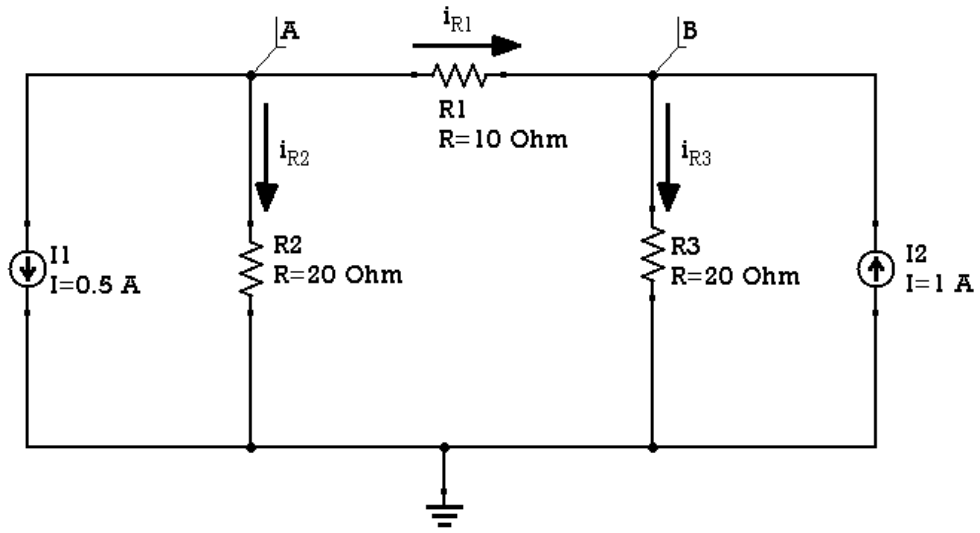


Figure 3.1: NVM algorithms demonstration – basic circuit.

Step 3: The currents identified are i_{R1}, i_{R2}, i_{R3} and the remaining are defined by the current sources $I1$ and $I2$.

Step 4: The currents equations are as follows:

$$\begin{cases} I1 = 0.5\,A \\ I2 = 1\,A \\ i_{R1} = \frac{U_A - U_B}{R1} \\ i_{R2} = \frac{U_A}{R2} \\ i_{R3} = \frac{U_B}{R3} \end{cases} \tag{3.1}$$

Step 5: The KCL equations for each Node are as follows:

$$\begin{cases} I1 + i_{R1} + i_{R2} = 0, & \text{(Node A)} \\ I2 + i_{R1} = i_{R3}, & \text{(Node B)} \end{cases} \tag{3.2}$$

Step 6: The equation system is generated the following way:

$$\begin{cases} 0.5 + \frac{U_A - U_B}{R1} + \frac{U_A}{R2} = 0 \\ 1 + \frac{U_A - U_B}{R1} = \frac{U_B}{R3} \end{cases} \Leftrightarrow \begin{cases} 0.5 + \frac{U_A - U_B}{10} + \frac{U_A}{20} = 0 \\ 1 + \frac{U_A - U_B}{10} = \frac{U_B}{20} \end{cases} \tag{3.3}$$

Leading to the Node voltages:

$$\begin{cases} U_A = 2\ V \\ U_B = 8\ V \end{cases} \tag{3.4}$$

Step 7: The remaining currents can be computed as through the Step 4 equations:

$$\begin{cases} i_{R1} = \frac{2-8}{10} = -0.6\ A \\ i_{R2} = \frac{2}{20} = 0.1\ A \\ i_{R3} = \frac{8}{20} = 0.4\ A \end{cases} \tag{3.5}$$

Analysis through direct matrix assignment

Similarly to the classical approach of the NVM, the direct matrix assignment is based in the KCL equations, but rearranged in a different manner. Taking Eq. 3.3, it is reorganised as follows:

$$\begin{cases} U_A \cdot \left(\frac{1}{R1} + \frac{1}{R2}\right) - U_B \cdot \left(\frac{1}{R1}\right) = -0.5 \\ U_B \cdot \left(\frac{1}{R1} + \frac{1}{R3}\right) - U_A \cdot \left(\frac{1}{R1}\right) = 1 \end{cases} \tag{3.6}$$

Using the definition of Conductance, $G = 1/R$, the KCL equations are transformed into the following:

$$\begin{cases} (G_1 + G_2) \cdot U_A - G_1 \cdot U_B = -0.5 \\ (G_1 + G_3) \cdot U_B - G_1 \cdot U_A = 1 \end{cases} \tag{3.7}$$

The matrix form from Eq. 2.1 is then obtained through the previous KCL equations.

$$\begin{bmatrix} G_1 + G_2 & -G_1 \\ -G_1 & G_1 + G_3 \end{bmatrix} \cdot \begin{bmatrix} U_A \\ U_B \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \tag{3.8}$$

The matrix equation can be simplified by the substitution of the known Conductances and the Node voltages are computed through the matrix equation:

$$\begin{bmatrix} 0.15 & -0.10 \\ -0.10 & 0.15 \end{bmatrix} \cdot \begin{bmatrix} U_A \\ U_B \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \Leftrightarrow \begin{cases} U_A = 2\ V \\ U_B = 8\ V \end{cases} \tag{3.9}$$

Analysis through MNA

Before computing the matrix equation from Eq. 2.2, the four matrices that construct A matrix must be obtained. The conductances matrix G is determined by the interconnections between the passive circuit elements, which means it is the same used in the previous method (direct matrix assignment). The B matrix is formulated by the connection of the voltage sources, being null for this example since no voltage sources are present. Furthermore, given that only ideal (independent) sources are considered for this analysis, then the C matrix is the transpose of B, meaning that it also produces a null matrix. Finally, D is always null, unless dependent sources are considered. The A matrix is then represented in Eq. 3.10.

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix} = \begin{bmatrix} 0.15 & -0.10 & 0 \\ -0.10 & 0.15 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.15 & -0.10 \\ -0.10 & 0.15 \end{bmatrix} \tag{3.10}$$

The x matrix holds the unknown quantities and can be developed as a combination of two column vectors: v, which has the unknown Node voltages; and j, holding the unknown currents through the voltage sources. For the example under analysis, the x matrix is as follows:

$$x = \begin{bmatrix} v \\ j \end{bmatrix} = \begin{bmatrix} U_A \\ U_B \\ 0 \end{bmatrix} = \begin{bmatrix} U_A \\ U_B \end{bmatrix} \tag{3.11}$$

The last matrix is z, containing the voltage and current sources. It is also a combination of two matrices: i, holding the sum of the current sources flowing into the corresponding Node (the signal depends on the current source polarity), and e matrix that contains the values of each voltage source. The z matrix is shown in Eq. 3.11. It is noted that, for this particular example, both current sources are flowing out of the respective Nodes, thus having negative sign. The e matrix is clearly null as there are no voltage sources in the circuit.

$$z = \begin{bmatrix} i \\ e \end{bmatrix} = \begin{bmatrix} -I1 \\ -I2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -I1 \\ -I2 \end{bmatrix} \tag{3.12}$$

It may be inferred that, for circuits without IVS, the MNA performs exactly the same as direct matrix assignment, since the matrix equation resulting from MNA, and demonstrated in Eq. 3.13, is equal to the Eq. 3.9, determined in the previous NVM analysis method.

$$[A]\cdot[x]=[z] \Leftrightarrow \begin{bmatrix} 0.15 & -0.10 \\ -0.10 & 0.15 \end{bmatrix} \cdot \begin{bmatrix} U_A \\ U_B \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \Leftrightarrow \begin{cases} U_A = 2\ V \\ U_B = 8\ V \end{cases} \tag{3.13}$$

Analysis through SNA

The nonexistence of IVS in the example under analysis implies that also no Supernodes are present in the circuit. Such fact drives SNA closer to the NVM classical approach, as demonstrated in the following resolution steps for the SNA algorithm.

Step 1: All Nodes are identified in the circuit image, namely Nodes A, B and *Ground*, which is the reference Node.

Step 2: Given that no IVS are found in the circuit, this means that there are any Supernodes available to mark. This step is then skipped.

Step 3: As stated in the previous step, there are no Supernodes, and consequently no forced conditions to be set, therefore this step is also skipped.

Step 4: The equation system is computed identically as previously done in the NVM classical approach, expressly in Step 6 of the analysis process.

3.2 Circuit with Isolated Voltage Sources

In the preceding circuit analysis, we noticed that MNA was broadly similar do direct matrix assignment, as much as SNA was to the NVM classical approach, since no IVS were present for that particular example. Nevertheless, the main purpose of both MNA and SNA is to tackle the issues brought by IVS to circuit analysis. In order to explore these methods' capabilities and evaluate the difficulties caused by IVS to the remaining analysis methods, the circuit represented in Fig. 3.2 is going to be analysed through the four NVM variants. This circuit has two unknown Node voltages (U_A and U_B) and a Supernode formed Nodes A and B, along with voltage source $V1$, meaning that the current flowing through the Supernode (i_{SN}) is also unknown. Note that Branches with a

current source have their Branch current automatically determined by the source value, which is the case of the Branch with current source $I1$.

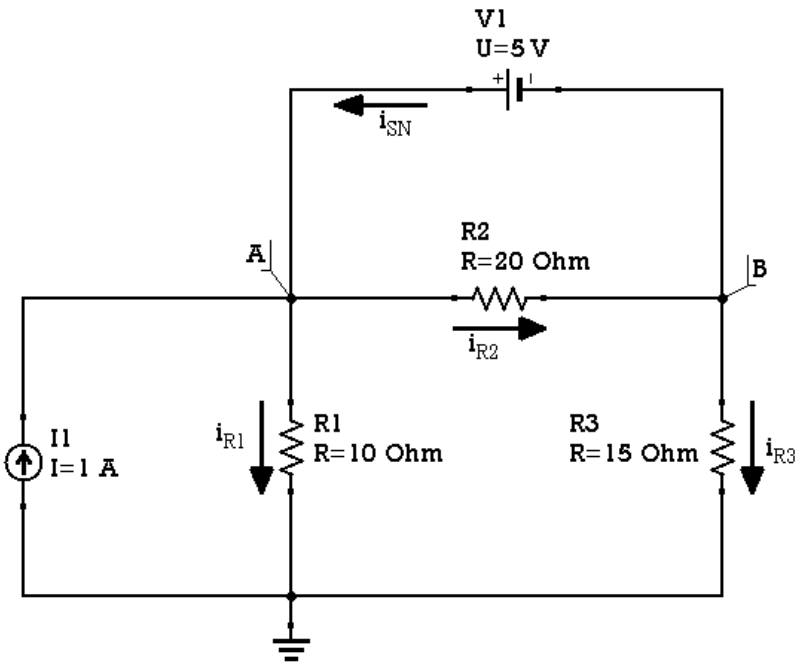


Figure 3.2: NVM algorithms demonstration – complex circuit.

Analysis through NVM classical approach

Step 1: All Nodes are correctly identified in the circuit image, namely Nodes A and B.

Step 2: The reference Node is properly placed in one of the circuit's Nodes.

Step 3: The currents identified are i_{R1}, i_{R2}, i_{R3}, as well as i_{SN} that flows through the Supernode, and $I1$ defined by the current source.

Step 4: The currents equations are represented below. Note that i_{SN} has no equation, since the Branch has neither impedance nor a current source.

$$\begin{cases} I1 = 1\,A \\ i_{R1} = \frac{U_A}{R1} \\ i_{R2} = \frac{U_A - U_B}{R2} \\ i_{R3} = \frac{U_B}{R3} \end{cases} \tag{3.14}$$

Step 5: The KCL equations for each Node are as follows:

$$\begin{cases} I1 + i_{SN} = i_{R1} + i_{R2}, & \text{(Node A)} \\ i_{R2} = i_{R3} + i_{SN}, & \text{(Node B)} \end{cases} \tag{3.15}$$

Step 6: The equation system is generated the following way:

$$\begin{cases} 1 + i_{SN} = \frac{U_A}{R1} + \frac{U_A - U_B}{R2} \\ \frac{U_A - U_B}{R2} = \frac{U_B}{R3} + i_{SN} \end{cases} \Leftrightarrow \begin{cases} 1 + i_{SN} = \frac{U_A}{10} + \frac{U_A - U_B}{20} \\ \frac{U_A - U_B}{20} = \frac{U_B}{15} + i_{SN} \end{cases} \tag{3.16}$$

The NVM classical method resolution originated in an under-determined system with three variables (U_A, U_B and i_{SN}) and just two equations, proving that this analysis methodology is not appropriate for circuits with IVS.

Analysis through direct matrix assignment

Similarly to what was done in the previous example, this method starts by rearranging the KCL equations. Therefore, Eq. 3.16 is changed to the following system:

$$\begin{cases} U_A \cdot \left(\frac{1}{R1} + \frac{1}{R2}\right) - U_B \cdot \left(\frac{1}{R2}\right) = 1 + i_{SN} \\ U_B \cdot \left(\frac{1}{R2} + \frac{1}{R3}\right) - U_A \cdot \left(\frac{1}{R2}\right) = -i_{SN} \end{cases} \tag{3.17}$$

Using the definition of Conductance, $G = 1/R$, the KCL equations are transformed into the following:

$$\begin{cases} (G_1 + G_2) \cdot U_A - G_2 \cdot U_B = 1 + i_{SN} \\ (G_2 + G_3) \cdot U_B - G_2 \cdot U_A = -i_{SN} \end{cases} \tag{3.18}$$

The Branch with an IVS cannot be considered for the interconnections inside the G matrix, since it would cause infinite Conductances. However, analogously to what occurred in the NVM classical approach, the Branch current flowing through the IVS appears as an unknown in the matrix equation, as demonstrated in Eq. 3.19, which leads to an under-determined system. Hence, the direct matrix assignment method cannot deal with circuits comprising IVS.

$$\begin{bmatrix} G_1 + G_2 & -G_2 \\ -G_2 & G_2 + G_3 \end{bmatrix} \cdot \begin{bmatrix} U_A \\ U_B \end{bmatrix} = \begin{bmatrix} 1 + i_{SN} \\ -i_{SN} \end{bmatrix} \tag{3.19}$$

Analysis through MNA

Under the current analysis scenario, the A matrix from MNA equation will differ from the antecedent example, since we now have a voltage source in the circuit. The first matrix to be computed is G, which is the same as in direct matrix assignment method. The G matrix is calculated as follows:

$$G = \begin{bmatrix} \frac{1}{R1} + \frac{1}{R2} & -\frac{1}{R2} \\ -\frac{1}{R2} & \frac{1}{R2} + \frac{1}{R3} \end{bmatrix} = \begin{bmatrix} 0.15 & -0.05 \\ -0.05 & 0.116 \end{bmatrix} \tag{3.20}$$

Further, the B matrix is obtained through the voltage sources information and it is represented in Eq. 3.21. Since the voltage source $V1$ has positive terminal connected to Node A (first line of the matrix) the first B matrix element will have the value 1. The contrary happens to the second element that corresponds to Node B, which is connected to the negative terminal of $V1$.

$$B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{3.21}$$

Considering only ideal sources in this analysis, the C matrix (calculated in Eq. 3.22) is the transpose of B, and D is null.

$$C = B^T = \begin{bmatrix} 1 & -1 \end{bmatrix} \tag{3.22}$$

Grouping the four matrices together, we obtain the following A matrix:

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix} = \begin{bmatrix} 0.15 & -0.05 & 1 \\ -0.05 & 0.116 & -1 \\ 1 & -1 & 0 \end{bmatrix} \tag{3.23}$$

The x matrix holds the unknown quantities, namely Node voltages and currents flowing through voltage sources. The example under analysis has two unknown Node-Voltages, U_A and U_B and a current flowing through the source $V1$. Then, the x matrix is defined as follows:

$$x = \begin{bmatrix} U_A \\ U_B \\ i_{SN} \end{bmatrix} \tag{3.24}$$

Ultimately, the z matrix contains the sum of current sources flowing to each Node (i matrix) and the values of each voltage source (e matrix). For matrix i, since the first element is related to Node A, it includes the current source $I1$, on the other hand the second element is null, as Node B has no current sources flowing in or out of it. Matrix e consists in a single element: $V1$, the exclusive voltage source in the circuit. The calculation of matrix z is demonstrated in Eq. 3.25.

$$z = \begin{bmatrix} i \\ e \end{bmatrix} = \begin{bmatrix} I1 \\ 0 \\ V1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} \tag{3.25}$$

The final form of the MNA matrix equation is represented in Eq. 3.26, as well as the Node voltages results and the current i_{SN} that flows through the IVS.

$$\begin{bmatrix} 0.15 & -0.05 & 1 \\ -0.05 & 0.116 & -1 \\ 1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} U_A \\ U_B \\ i_{SN} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} \Leftrightarrow \begin{cases} U_A = 8\ V \\ U_B = 3\ V \\ i_{SN} = 50\ mA \end{cases} \tag{3.26}$$

Analysis through SNA

Step 1: All Nodes are identified in the circuit image, namely Nodes A, B and *Ground*, which is the reference Node.

Step 2: One Supernode was identified between Nodes A and B, formed by the voltage source $V2$.

Step 3: Given that a reference Node from the Supernode must be chosen in order to generate a proper forced condition, we will consider Node B as a reference. Thus, the forced condition for the found Supernode is as follows:

$$U_A = U_B + V1 = U_B + 5 \tag{3.27}$$

Step 4: The number of equations needed are determined through the following equation:

$$E = N - 1 - T = 3 - 1 - 1 = 1 \tag{3.28}$$

Being N the number of Nodes and T the number of voltage sources. The equation is based on the KCL, however as Node A and B form a Supernode, all the currents flowing in and out of these two Nodes must be taken into account as follows:

$$I1 + i_{SN} + i_{R2} = i_{R1} + i_{R2} + i_{R3} + i_{SN} \tag{3.29}$$

On the left hand side of the equation we have the currents flowing in to the Supernode, and on the right hand side are the currents flowing out of the Supernode. Note that some currents are displayed in both equation terms, which means that they both flow in and out of the Supernode, e.g. I_{R2} flows out of the Node A but flows in to the Node B. As these currents will cancel each others, the KCL equation remains:

$$I1 = i_{R1} + i_{R3} \tag{3.30}$$

Replacing the known currents for its value and the remaining from their respective Ohm's Law, the equation changes into:

$$1 = \frac{U_A}{R1} + \frac{U_B}{R3} \tag{3.31}$$

The Node voltage U_A must then be replaced for an expression in order to the Node voltage from Node B (the reference Node), using the forced condition stated in step 3, resulting in a single variable equation which computes the Node voltage U_B

$$\frac{U_B + 5}{10} + \frac{U_B}{15} = 1 \Leftrightarrow U_B = 3\ V \tag{3.32}$$

The remaining Node voltage is automatically known by the forced condition, as well as the current flowing through the IVS, using the KCL after computing the rest of the currents.

The validation performed in this section allowed to understand the capabilities and disadvantages of each analysis method. In fact, it was observed that neither the classical approach nor the direct matrix assignment can handle circuits with IVS, while MNA and SNA can always be applied.

On one hand, the MNA is undoubtedly the best systematic approach for computerised implementations, since it is focused on mathematical operations and avoids symbolic algebra. On the other hand, this method introduces an unnecessary number of equations (that grows with the number of Supernodes and IVS that forms them), comparing to the SNA. The latter not only induces minor amounts of effort in terms of linear algebra

calculations, but also provides a more intuitive and step-based resolution flow, which is crucial in the educational perspective. These were the primary advantages that made SNA the cornerstone of the *U=RIsolve* algorithm.

Chapter 4

The Proposed NVM-SA Algorithm

The application of the Supernode Analysis is insightful and comprehensive in view of the fact that it is independent of the circuit size and complexity. Despite the usefulness in the teaching and learning of electrical circuit analysis, as far as we know, this method has not been yet computer-implemented. This chapter describes a SNA based algorithm, named NVM-SA, which copes with any (linear and passive) electrical circuit, DC or AC, regardless of its size or complexity. The NVM-SA algorithm is easily implementable in software and it is intuitive to human understanding, as it enables to provide a step-by-step output of the circuit analysis. The chapter starts with a brief explanation of the main terminology used both through the algorithm and the U=RIsolve application, then describes the NVM-SA algorithm and finishes with a demonstration example.

4.1 Concepts and terminology

The following concepts and terminology are paramount to describe the NVM-SA algorithm and will be applied to remaining of this essay through the *U=RIsolve* application description.

- **(electrical) Circuit**: set of components and interconnecting elements that are physically (electrically) connected to compose a closed electrical circuit;
- **(electrical) Component**: (electrical) power source or load;
  - Examples: voltage source, current source, Resistor, Capacitor, Inductor, Lamp, impedance (Resistance, Inductance, Capacitance, pure or combined);

- **Interconnecting Element**: electrical element that interconnects two components or interconnecting elements (with a negligible resistance/impedance);
  - Examples: wire/cable, printed circuit boards/breadboard track/traces, connector/plug, weld;
  - *Note: It may interconnect two components, one component and one interconnecting element or two interconnecting elements.*
- **Ground**: common reference point of an electrical circuit, assuming a 0 V electric voltage;
  - *Note: even if an electrical circuit/diagram does not have an explicit Ground connection, we may reference one of its points (usually nodes) to the Ground, for the sake of the analysis.*
- **Branch**: set of (electrical) components interconnected in series;
  - *Note: a Branch is always delimited by two Nodes, except for the special case of a circuit with a single Loop/Branch.*
- **Node**: point of interconnection between three or more Branches (interconnected by one or more Interconnecting Elements);
  - *Note: In case two or more points of interconnection are directly interconnected via one or more Interconnecting Elements, these points have the same electrical potential (voltage) and constitute a single Node.*
- **Ideal Voltage Source**: voltage source with no internal impedance (0 Ω);
- **Isolated Voltage Source**: Ideal voltage source that is the only component of a Branch;
  - *Note: In case there are more than one ideal voltage source connected in series in a Branch, these should be aggregated into just one (Isolated Voltage Source).*
- **Supernode**: set of two or more Nodes interconnected by Isolated Voltage Sources, including the respective Branches;
  - A Supernode is like a "cloud", hiding its own Nodes/Branches, so that the KCL of a Supernode is formed by the convergent/divergent currents to/from that Supernode (that go in/out of its "cloud");

  - A current circulating in a Branch that belongs to a Supernode (therefore "hidden" inside its cloud) cannot be computed just by solving the NVM equation system, because this current is not going to make part of the KCL of the Supernode; this current can be computed *a posteriori*, after all other currents have been determined, by applying the KCL in one of its two Nodes (that belong to that Supernode).

- **Grounded Supernode**: a Supernode that has one of its Nodes connected to the *Ground*;
  - *Note: the voltages in all its Nodes are numerically known, i.e. can be computed analytically through the relation between the voltage between each Node and the Ground, through the concatenation of (one or more) Isolated Voltage Sources.*
- **Floating Supernode**: a Supernode that has no Node connected the *Ground*.
  - *Note: the voltage in any of its Nodes can be related to the voltage in any other of its Nodes, through the "differential" voltage between each pair of Nodes, through the concatenation of (one or more) Isolated Voltage Sources.*

4.2 The NVM-SA algorithm

The unknown variables, similarly to the other NVM variants, are the Node voltages (U_A, U_B, U_C,...), related to the *Ground* Node. Upon determination of the voltage in every Node, all the currents and (other) voltages in the circuit can be computed. The NVM-SA method is demonstrated through the Algorithm 3.

Algorithm 3 The proposed NVM-SA algorithm

1: **procedure** IDENTIFY THE FUNDAMENTAL VARIABLES FOR THE NVM

1. **Identify & Output** the number of Branches/currents in the circuit $\rightarrow R$
   a) **For each** Branch in the circuit
      i. Identify the Branch current with a symbol, e.g. $I_1, I_2, I_3, \ldots$
      ii. Assume a certain (hypothetical) current direction
      iii. Mark the Branch current (ID + direction) in the circuit schematics
2. **Identify & Output** the number of Nodes $\rightarrow N$
   a) **For each** Node in the circuit
      i. Identify the Node with a voltage, e.g. $U_A, U_B, U_C, \ldots$

ii. Mark the Node voltage ID in the circuit schematics

3. **Identify & Output** the number of Isolated Voltage Sources (IVS) $\rightarrow T$

a) Notes:

- *In case there is more than one ideal voltage source connected in series in a Branch, these should be aggregated and considered as just one Isolated Voltage Source*
- *The Isolated Voltage Sources should be stored/presented as a set, e.g.* $IVS = \{E_X, E_Y\}$

b) **IF** $T = 0$ (*No Isolated Voltage Sources*)

i. Select one of the Nodes as a reference/Ground node, e.g. if Node W is chosen, then $U_W = 0\ V$

ii. Mark that node in the circuit schematics (*use Ground symbol* $\perp$)

iii. Skip procedure 2 (*there are no Supernodes*)

c) **IF** $T = 1$

i. Fix the reference/Ground Node to one of the two Nodes connected to this Isolated Voltage Source

d) **IF** $T \geq 2$

i. Check if there are Branches (with Isolated Voltage Sources) that are connected together, and if so group them together and select the largest group (in case there are more than one group) as the *Grounded Supernode*, in procedure 2: 1).

4. **Compute & Output** the number of KCL equations that will build the equation system to be solved $\rightarrow N - 1 - T$

2: **procedure** IDENTIFY & OUTPUT ALL THE SUPERNODES AND THE VOLTAGE RELATIONS BETWEEN THEIR NODES

1. **Identify & Output** the (single) *Grounded Supernode* (SN_G) and the voltage in its Nodes

a) Identify & Output the *Grounded Supernode* and respective Nodes

- *Note: the Nodes (two or more) can be stored/presented as a set, e.g.* $SN_G = \{U_A, U_B\}$

b) Compute & Output the equations that relate the voltage in each pair of its Nodes and Compute & Output the voltage in each of its Nodes (numerically, in V)

- *Note: the Grounded Supernode voltage equations are not considered in the NVM equation system, as the voltage in its Nodes can be computed a priori.*

2. **Identify & Output** all the Floating Supernodes (SN_F) and corresponding voltage relation equations

   a) **For Each** *Floating Supernode*

      i. Identify & Output the *Floating Supernode* and respective Nodes (two or more)

      - *Note 1: select some kind of index, e.g.* SN_{F1}, SN_{F2}
      - *Note 2: the Floating Supernodes can be stored/presented as a set, e.g.* $SN_{F1} = \{U_A, U_B\}$, $SN_{F2} = \{U_E, U_F, U_G\}$

      ii. Choose a Node from the Floating Supernode as a reference.

      iii. Compute & Output the equations that relate the voltage in each pair of its Nodes, in order to the reference Node chosen in the previous step.

3: **procedure** BUILD & OUTPUT THE $N-1-T$ KCL EQUATIONS, AS A FUNCTION OF THE NODE VOLTAGES

1. Write the $N-1-T$ KCL equations, as a function of Branch currents (I_1, I_2, I_3, ...)

2. **For Each** Branch/current in the circuit $\rightarrow I_x$

   a) **If** the Branch contains more than one impedance connected in series (including internal impedance of one or more voltage sources, aggregate all impedances into just one equivalent impedance $\rightarrow Z_{xeq}$

   b) **If** the Branch contains more than one voltage source connected in series, aggregate all of them into just one equivalent voltage source $\rightarrow E_{xeq}$

   c) Write the Branch current as a function of the voltage across the equivalent impedance divided by the impedance (Ohm's Law)

      i. Subtract the voltage before the equivalent impedance by the voltage after the equivalent impedance $\rightarrow U_{Zxeq}$

      - *Note 1: mind the direction of the Branch current (that has been assumed a priori, in procedure 1: 1. a) ii.)*
      - *Note 2: if the current belongs to a Branch from any Supernode, meaning the Branch has no impedance, the Ohm's Law cannot be used, and*

the current will not have an equation; instead, it will be computed at the end of the algorithm using the known currents

- *Note 3: if the current belongs to a Branch that contains a current source, its value is automatically known (the Ohm's equation can be discarded)*

ii. Store & Output $I_x = \frac{U_{Zxeq}}{Z_{xeq}}$

3. Rewrite the $N-1-T$ KCL equations, (as in Procedure 3: 1.), as a function of Node voltages (U_A, U_B, U_C, ...), using the equations generated in Procedure 3: 1. c) ii.

4: **procedure** COMPUTE THE VOLTAGE IN EVERY NODE

1. **Substitute** every Node voltage from each Floating Supernode by the expression in order to its reference and rewrite the $N-1-T$ KCL equations.
2. **Solve** the equation system & **Output** the voltage in every Node ($U_A = \ldots, U_B = \ldots, U_C = \ldots$)

5: **procedure** (OPTIONAL) COMPUTE & OUTPUT THE BRANCH CURRENTS

1. Use the equations generated in procedure 3: 1. c) ii. to compute the numerical values of the Branch currents.

Final remark: the variables U, E, Z and I are complex variables, however for DC circuits only the real number remains.

4.3 Demonstration of the NVM-SA

We illustrate the NVM-SA algorithm (in the *U=RIsolve* application), through the circuit example represented in Fig. 4.1. This example comprehends the most important details so that the analysis process is complete, such as Node voltages affected by Grounded and Floating Supernodes, as well as Node voltages where KCL is normally applied, i.e. without Supernodes. The analysis will be carried out through the completion of the procedures previously identified in Algorithm 3.

Procedure 1: Identify the fundamental variables for the NVM

1. Number of Branches: $R = 13$

a) Identified currents: $\{I_{R1}, I_{R2}, \ldots, I_{R13}\}$

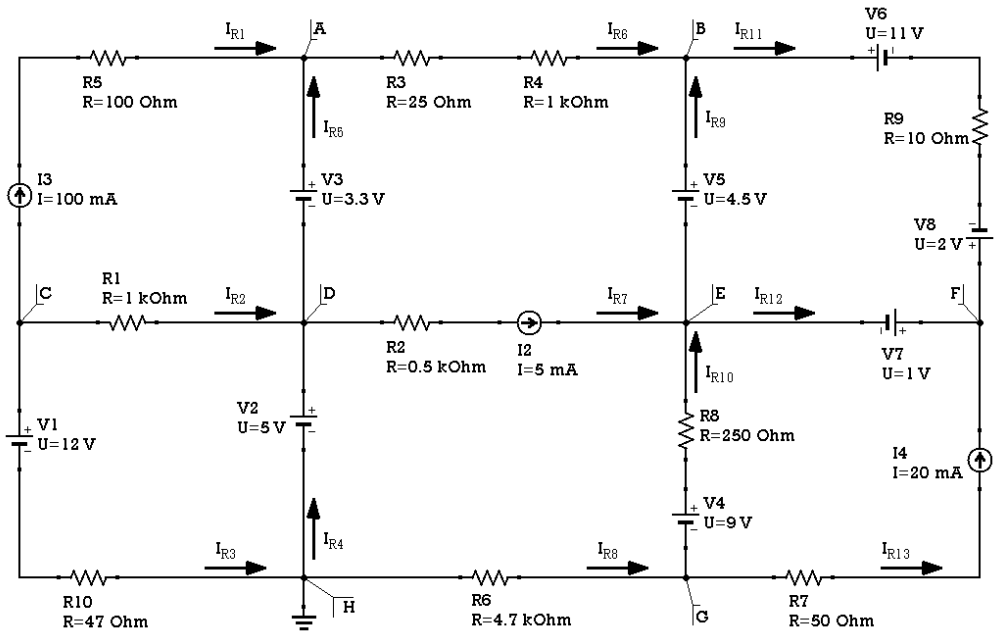


Figure 4.1: Example circuit for NVM-SA algorithm demonstration.

b) The currents direction: defined in Fig. 4.1

c) currents direction: represented in Fig. 4.1

2. Number of Nodes: $N = 8$

   a) Identified Node voltages: $\{U_A, U_B, ..., U_G\}$, U_H is selected as the reference (Ground) $\rightarrow U_H = 0\ V$

   b) Wire labels $(A, B, ..., G)$ are used to identify/mark Node voltages.

3. Number of Isolated Voltage Sources: $T = 4$

   - Condition $T \geq 2$ is **true**
     - Both pairs of voltage sources $\{V2, V3\}$ and $\{V5, V7\}$ are connected together to form two Supernodes. As there is no largest group of sources, we choose the first pair to form the *Grounded Supernode*, and the ground is placed at node H, as Fig. 4.1 shows.

4. Number of KCL Equations: 3

$$N - 1 - T = 8 - 1 - 4 = 3 \tag{4.1}$$

Procedure 2: Identify & Output all the Supernodes and the voltage relations between their Nodes

1. The *Grounded Supernode* is identified in Fig. 4.2 (rotated to horizontal, for convenience).

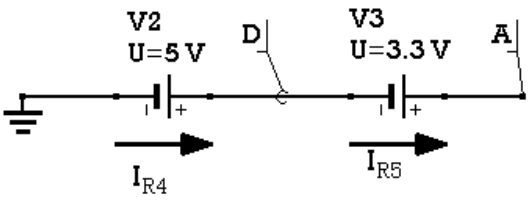


Figure 4.2: *Grounded Supernode* (from the example circuit).

   a) The *Grounded Supernode* is formed by two nodes (plus *Ground*) and can be represented as: $SN_G = \{U_D, U_A\}$.

   b) The equations expressing the voltage relation between the Nodes inside the *Grounded Supernode* are as follows:

$$\begin{cases} U_D = U_{GND} + V2 \\ U_A = U_D + V3 \end{cases} \Leftrightarrow \begin{cases} U_D = 5\ V \\ U_A = 8.3\ V \end{cases} \tag{4.2}$$

2. There is only one *Floating Supernode* and it is represented in Fig. 4.3 (flattened, for convenience).

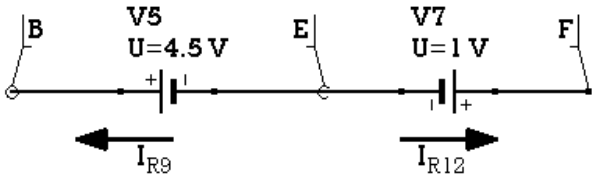


Figure 4.3: *Floating Supernode* (from the example circuit).

   a) The *Floating Supernode* is formed by three nodes and can be represented as: $SN_{F0} = \{U_B, U_E, U_F\}$.

   b) The reference chosen was Node E.

   c) The equations that relate each Node from the *Floating Supernode* (in order to reference E) are as follows:

$$\begin{cases} U_B = U_E + V5 \\ U_F = U_E + V7 \end{cases} \Leftrightarrow \begin{cases} U_B = U_E + 4.5 \\ U_F = U_E + 1 \end{cases} \tag{4.3}$$

Procedure 3: Build & Output the $N-1-T$ KCL equations, as a function of the Node voltages

1. The KCL equations, as a function of Branch currents are listed below:

$$\begin{cases} 0 = I_{R1} + I_{R2} + I_{R3}, & (\text{Node } C) \\ I_{R8} = I_{R10} + I_{R13}, & (\text{Node } G) \\ I_{R6} + I_{R7} + I_{R10} + I_{R13} = 0, & (\text{Node } SN_{F0}) \end{cases} \tag{4.4}$$

2. Determination of the Branch currents (excluding the Supernodes Branch currents):

   - Multiple impedances in the same Branch found:

$$Z_{R6EQ} = R3 + R4 \tag{4.5}$$

   - voltage sources in series found:

$$U_{R11EQ} = V6 - V8 \tag{4.6}$$

   - Computed Branch currents:

$$\begin{cases} I_{R1} = I3 \\ I_{R2} = \frac{U_C - U_D}{R1} \\ I_{R3} = \frac{U_C - V1 - U_{GND}}{R10} \\ I_{R6} = \frac{U_A - U_B}{Z_{R6EQ}} \\ I_{R7} = I2 \\ I_{R8} = \frac{U_{GND} - U_G}{R6} \\ I_{R10} = \frac{U_G + V4 - U_E}{R8} \\ I_{R11} = \frac{U_B - U_{R11EQ} - U_F}{R9} \\ I_{R13} = I4 \end{cases} \Leftrightarrow \begin{cases} I_{R1} = 0.1\,A \\ I_{R2} = \frac{U_C - 5}{1000} \\ I_{R3} = \frac{U_C - 12}{47} \\ I_{R6} = \frac{8.3 - U_B}{1025} \\ I_{R7} = 0.005\,A \\ I_{R8} = -\frac{U_G}{4700} \\ I_{R10} = \frac{U_G + 9 - U_E}{250} \\ I_{R11} = \frac{U_B - 9 - U_F}{10} \\ I_{R13} = 0.02\,A \end{cases} \tag{4.7}$$

3. Rewrote KCL equations as function of Node voltages:

$$\begin{cases} 0 = 0.1 + \frac{U_C - 5}{1000} + \frac{U_C - 12}{47} \\ -\frac{U_G}{4700} = \frac{U_G + 9 - U_E}{250} + 0.02 \\ \frac{8.3 - U_B}{1025} + 0.005 + \frac{U_G + 9 - U_E}{250} + 0.02 = 0 \end{cases} \tag{4.8}$$

Procedure 4: Compute the Voltage in every Node

1. The reference for the Supernode SN_{F0} is Node E (chosen in procedure 2: 2. b)). In the equation system, the potential U_B is then replaced with its expression in order to Node E.

$$\begin{cases} 0 = 0.1 + \frac{U_C - 5}{1000} + \frac{U_C - 12}{47} \\ -\frac{U_G}{4700} = \frac{U_G + 9 - U_E}{250} + 0.02 \\ \frac{8.3 - (U_E + 4.5)}{1025} + 0.005 + \frac{U_G + 9 - U_E}{250} + 0.02 = 0 \end{cases} \tag{4.9}$$

2. voltages in every Node:

   a) As already mentioned, the following Node voltages can be immediately determined (*Grounded Supernode*):

$$\begin{cases} U_D = 5\ V \\ U_A = 8.3\ V \end{cases} \tag{4.10}$$

   b) The following Node voltages are obtained by resolving the equation system (Eq. 4.9):

$$\begin{cases} U_C = 7.2\ V \\ U_G = -3.99\ V \\ U_E = 9.8\ V \end{cases} \tag{4.11}$$

   c) The remaining Node voltages can be computed from the *Floating Supernode* Equations:

$$\begin{cases} U_B = U_E + 4.5 \\ U_F = U_E + 1 \end{cases} \Leftrightarrow \begin{cases} U_B = 14.3\ V \\ U_F = 10.8\ V \end{cases} \tag{4.12}$$

Procedure 5: Compute and Output the Branch currents

1. The following Branch currents are known *a priori* (current sources):

$$\begin{cases} I_{R1} = 0.1\,A \\ I_{R7} = 0.005\,A \\ I_{R13} = 0.02\,A \end{cases} \tag{4.13}$$

2. The following Branch currents can be obtained from the equations in procedure 3: 2.:

$$\begin{cases} I_{R2} = 0.002\,A \\ I_{R3} = -0.1\,A \\ I_{R6} = -0.006\,A \\ I_{R8} = 0.001\,A \\ I_{R10} = -0.019\,A \\ I_{R11} = -0.55\,A \end{cases} \tag{4.14}$$

3. The remaining Branch currents can be computed through KCL:

$$\begin{cases} I_{R4} = I_{R5} + I_{R7} - I_{R2} \\ I_{R5} = I_{R6} - I_{R1} \\ I_{R9} = I_{R11} - I_{R6} \\ I_{R12} = -I_{R11} - I_{R13} \end{cases} \Leftrightarrow \begin{cases} I_{R4} = -0.103\,A \\ I_{R5} = -0.106\,A \\ I_{R9} = -0.544\,A \\ I_{R12} = 0.53\,A \end{cases} \tag{4.15}$$

The previous analysis enabled to understand how the NVM-SA is applied to a circuit comprising IVS, along with the execution logic which sequentially outputs the necessary resolution details. These outputs create the progressive, step-based solution provided by *U=RIsolve* which is addressed in the upcoming chapter.

Chapter 5

Implementation Approach

The U=RIsolve application is addressed in this chapter. Starting with the preliminary input phase that tackles both QUCS as the background simulator and its netlist files characteristics, this section then explores the application architecture, providing a description of each relevant software module that conducts the analysis process. Lastly, the aspects related to user interface and the application step-based output are demonstrated through an analysis example circuit which contains the most important details to be displayed in the results panel, such as Grounded and Floating Supernodes.

5.1 QUCS as the benchmark simulator

The students' learning process in DC/AC circuit analysis is usually consolidated in a way that theoretical models can be instantiated and validated in practice (through practical experiments in lab classes) and also through simulation. Therefore, undergraduate students often use circuit simulators not merely as a self-learning tool but also to prepare their lab classes (*a priori*), by performing the theoretical (analytical) and simulation analysis of the circuits they are going to implement/experiment in the lab.

Several reasons led to the choice of QUCS as the basis simulation tool of the *U=RIsolve* project, some of them already identified in Chapter 2. Essentially, the fact that QUCS is extremely easy to use, multi-platform, freeware/open-source and features basic functionality for teaching/learning DC/AC circuit analysis made this software stand out in the current landscape of simulation tool-sets. QUCS has also been adopted has an active learning tool that promotes conceptual change when studying circuit analysis [63, 64]. Furthermore, the QUCS project has been quite active, involving continuous upgrades

[65] and a large community of contributors that implemented new software modules [66, 67, 68, 69, 70].

QUCS is therefore the selected circuit simulator to implement the circuit under analysis and to generate its description (*netlist*), which users can upload to *U=RIsolve*. For the time being, only the QUCS *netlist* is supported for the analysis, however future implementations are envisaged to extend the application to other simulators, by adding specific *netlist* parsing functions for each simulation software.

5.2 The netlist files and parsing

Most Spice-based circuit simulators, such as QUCS, generate a circuit description during simulation and provide it in a format of a text file, entitled *netlist*. In preliminary versions of Spice, *netlists* were usually employed as an input to the simulation [71], notwithstanding, from a users' perspective, this method long lost terrain against more appealing and easy to use GUI. *U=RIsolve* relies on the QUCS *netlist* as an input to the analysis algorithm, which requires users to perform an *a priori* simulation to generate the *netlist* file.

An illustration of a *netlist* content (similar to QUCS *netlist*) is given in Fig. 5.1 (right), and the circuit schematic (left) with every component labelled and interconnections defined with numbers (1, 2, 3).

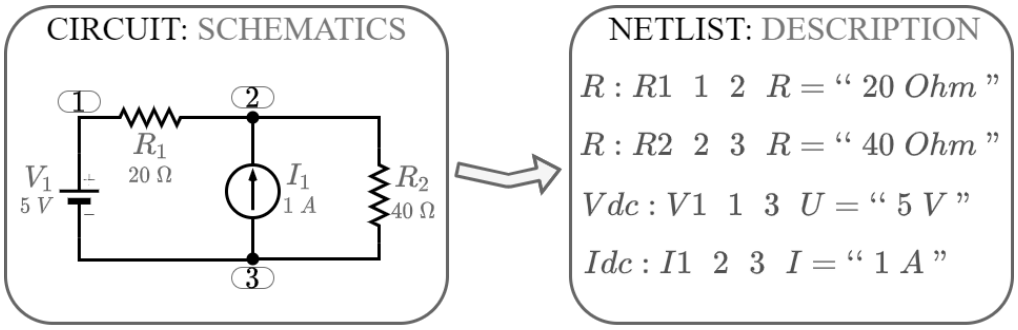


Figure 5.1: Electrical circuit (left) and its *netlist* description (right).

Each *netlist* line describes an existing electrical component in the circuit schematic with the following essential information:

1. Component type: identifies the class of the component ("R" for resistors, "Vdc" for DC voltage sources, "Idc" for DC current sources, "L" for Inductors, "C" for

Capacitors, “Vac” for AC voltage sources and “Iac” for AC current sources).

2. Component label: an unique attribute/representation of the component assigned in the simulator, by the user (e.g. R_1).

3. Component connections: determines the two interconnection points (start and end) of the element.

4. Component unit and value: specifies the electrical quantity associated to the element.

Towards the implementation of the NVM-SA algorithm to a specific circuit using its description, two main issues arise. The first is how to identify Branches in the *netlist*, and the second is how to extract sources polarities, out of the *netlist* information.

The first issue is accomplished through the components interconnections. Here, two types of interconnections are introduced: virtual Nodes, and real Nodes (or just “Nodes” as formalised in Chapter 4). The variation between these types of Nodes is that virtual Nodes just interconnect two components, while real Nodes join together three or more components. In our application, real Nodes must be identified with a letter (*wire label*, in QUCS). In the example from Fig. 5.1, Node 1 is a virtual Node, connecting V_1 and R_1, while Nodes 2 and 3 are real Nodes, as they connect R_1, source I_1 and R_2; and V_1, source I_1, and R_2, respectively. Knowing that a Branch is always delimited by two real Nodes, enables to identify them by evaluating the components interconnections in the *netlist*. On one hand, Branches with only one component are automatically recognised, since the component's interconnections are both real Nodes (the case of I_1 and R_2 in the Fig. 5.1 example). On the other hand, Branches with multiple components require a more complex procedure, as follows:

1. If one component has one real Node and one virtual Node as interconnections, start the Branch in the real Node.

2. Find in the *netlist* the component with the remaining instance of the virtual Node (note that there are only two instances of each virtual Node in the *netlist*) and check the other interconnection from the component found.

3. Repeat Step 2 until a real Node is found, so that the Branch can be completed.

This procedure could be illustrated with the example provided. Suppose the algorithm finds the resistor R_1 in the *netlist* (which appears in the first line), it is observed that its

interconnections consist of a real Node (2) and a virtual one (1). The algorithm assumes the Branch starts at Node 2, and searches for Node 1 in the *netlist*. Node 1 appears at component V_1, and its remaining interconnection is Node 3, which is a real Node, meaning the Branch starts at Node 2 and ends at Node 3.

The second issue refers to both current and voltage sources polarity. The algorithm identifying the voltage sources positive and negative poles and the current sources direction is crucial for a correct analysis. Usually, simulators define a specific interconnection to be the positive pole (in case of a voltage source), or the flow direction (in case of a current source). In QUCS *netlist*, as the one shown in this example, that interconnection is always the first Node. In Fig. 5.2, it is demonstrated how *netlist* represents sources polarities. On the left, it is observed that the voltage source positive pole interconnection is always the first Node in the *netlist*. Similarly, on the right hand side, the direction to where the current flows is represented by the first Node.

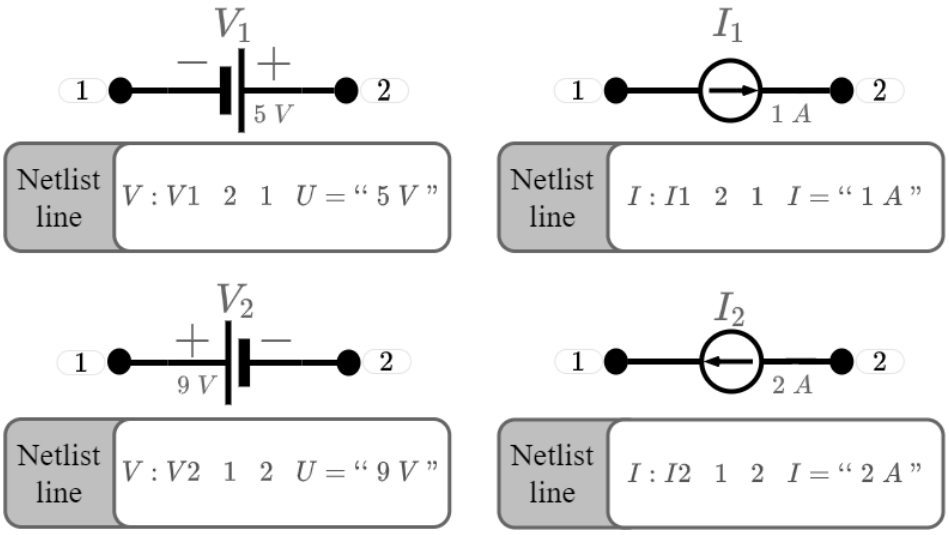


Figure 5.2: Changes in *netlist* with polarity inversion.

Accomplished these two challenges, any electrical circuit can be accurately analysed through the NVM-SA approach, as the algorithm is provided all the necessary circuit information in order to model it.

5.3 Software architecture

The *U=RIsolve* application receives the circuit information (*netlist*) as an input, and outputs the solution through the NVM-SA. In between, the software architecture is segmented into twelve steps (1–12), illustrated in Fig. 5.3 and described next.

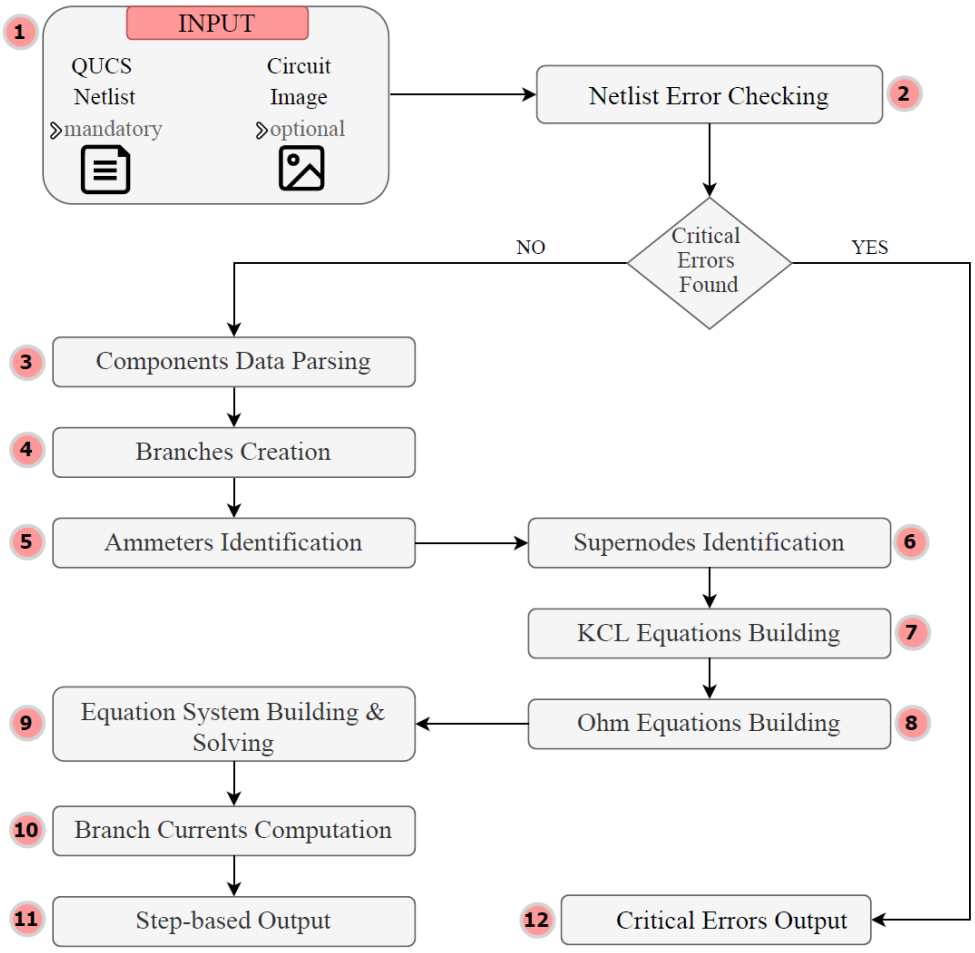


Figure 5.3: *U=RIsolve* software architecture.

1. Application inputs

The system accepts both circuit description (*netlist*: text file) and schematics (image file). The circuit *netlist* models the circuit, which is mandatory to perform the analysis. The circuit schematics (image), is optional, but adds more intuition to users, since it is displayed in the output along with the analysis results.

2. Netlist error checking

In the early learning phases, using circuit simulators can be a complex task, and students often make mistakes. These mistakes lead to error-prone *netlists* that need to be handled by the *U=RIsolve* application. In this step, the *netlist* is analysed in order to detect errors that are split into *warnings* (problems in *netlist* that can be filtered and solved during the execution time) and *critical errors* (problems that prevent the algorithm to proceed). In case warnings are found, information is attached to the output; however, when critical errors are found, the algorithm stops and these problems are reported to users. The critical errors that *U=RIsolve* can currently diagnose are illustrated in Fig. 5.4 and described next:

- **E1**: This error is detected every time that a real Node is found without an identification. All the circuit Nodes must be labelled with a letter (A, B, ...), otherwise, even though the algorithm could still solve the circuit, users would not be able to associate the Node voltage computed to a Node.

- **E2**: The second flaw is detected whenever a wrong *Ground* ID is detected. The reference Node should always be marked with the symbol ⏚ available in QUCS, which generates the ID "*gnd*". In case users try to identify *Ground* with a *wire label* and write a different ID (e.g. "GND" or "ground"), it triggers an error.

- **E3**: The simulation block defines the type of the analysis, despite that, users can introduce an AC component on a DC circuit by mistake, which prevents the analysis to proceed.

- **E4**: In order to correctly generate the multiples of each components value, the unit string must be in conformity with the standard prefixes. If any other types are detected (e.g "kiloHz" instead of "kHz"), a critical error is issued.

- **E5**: It is a basic rule that current sources cannot be physically coupled in series, in case this scenario occurs (reflected in the *netlist*) the algorithm stops and generates a critical error.

- **E6**: The last critical issue that may occur is a duplicated Node label introduced by the user (which QUCS allows to).

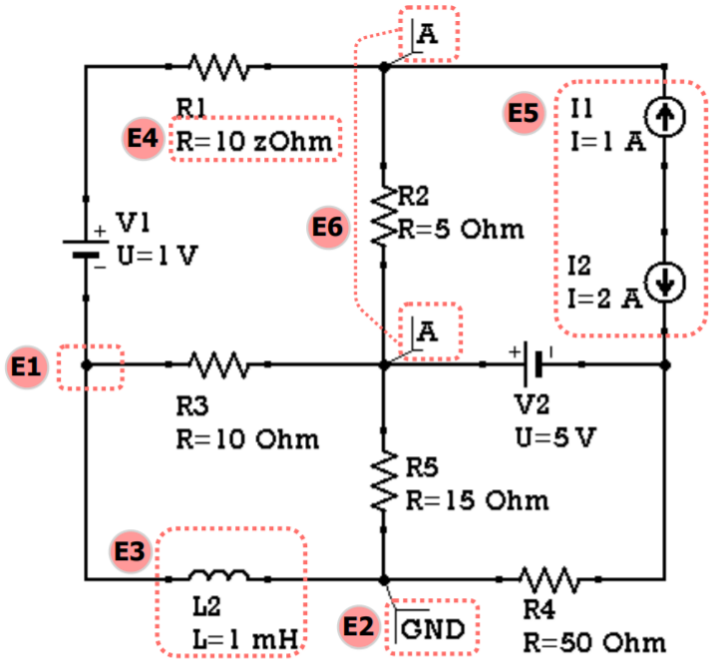


Figure 5.4: Critical errors detected by *U=RIsolve*.

Additionally to the critical errors, a few warnings may be displayed to users. This errors do not cause any problems in the analysis process, but it is an adequate policy for users to fix them in future simulations. The warnings that can be detected by *U=RIsolve* are listed next:

- Ground Node not found: this error occurs when users do not introduce the *Ground* symbol ($\perp$) in the circuit; the application solves it by placing the *Ground* in the best position (that reduces the number of variables). *U=RIsolve* then informs users in which Node the *Ground* has been located.

- Virtual Nodes mistakenly identified: if users wrongly label a virtual Node (not a real Node), *U=RIsolve* execution is not affected, since it is changed to "*net_i*" being i the smallest available index for virtual Nodes. Nevertheless the error is reported to the user.

- Invalid real Node ID: some Nodes IDs should be avoidable as they could introduce ambiguities to results (e.g "VA" is taken as an invalid ID, because it may be misinterpreted as Node voltage).

- Multiple Ammeters in the same Branch: in case this error occurs, *U=RIsolve* simply chooses one of the Ammeters and deletes the remaining, adjusting the netlist.

- Multiple AC frequencies detected: if *U=RIsolve* founds AC voltage sources with different frequencies, it favours the controller frequency. Nonetheless, if the controller does not have a valid frequency (e.g an array of frequencies), the algorithm chooses a random frequency from an AC voltage source and warns the user.

- Duplicated measuring instrument IDs: although QUCS forbids this operation, users could manually change the IDs directly in the *netlist* and mistakenly set two or more equal IDs. *U=RIsolve* detects this problem and reports it to the user.

- Multiple Voltmeters measuring the same potential difference: even though it does not affect the simulation, for learning purposes, the algorithm warns users about Voltmeters inserted between the same Nodes.

3. Components data parsing

At this step, every component from the *netlist* is parsed into an object with specific fields depending on its type. The standard component object has five main fields:

- *ID*: an unique value that represents the element;

- *Positive Node*: the positive interconnection;

- *Negative Node*: the negative interconnection;

- *Reference*: user-defined label in QUCS;

- *Value*: a number that represents the magnitude;

- *Unit*: a string defining the multiple of the unit value.

Note that the positive and negative Node fields are only useful for elements with polarity, since it will directly affect the equations.

Fig. 5.5 illustrates the creation of the components data structure, resulting from the parsing of the Fig. 5.1 *netlist*. It is observed that multiple elements of the same class are grouped together, which is the case of resistors `R1` and `R2`.

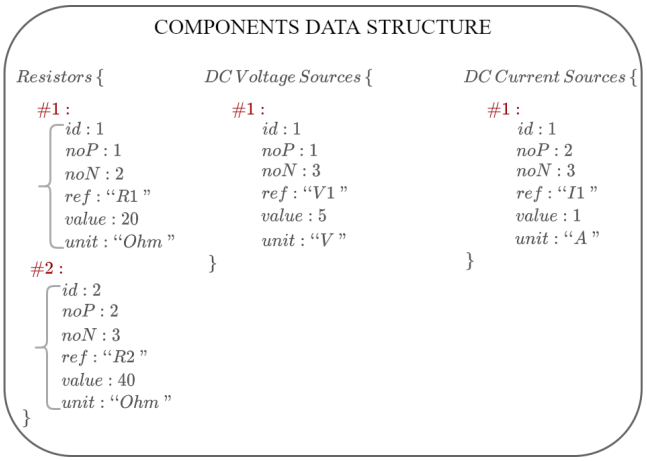


Figure 5.5: Components data structure (from the previous *netlist*).

4. Branches creation

Each Branch is identified through the components interconnection data, as explained beforehand in section 5.2. Since it is acknowledged that a Branch is formed by two real Nodes, interconnection information will determine which components belong to a single Branch. Branches also have multiple attributes, in particular *start* and *end* Nodes, *equivalent impedance* and *voltage*, an *unique ID*, as well as a list of elements that belong to the specific Branch.

5. Ammeters identification

Right now, *U=RIsolve* has a limitation concerning the assignment of each current's direction/flow, due to the fact that the application cannot yet provide a visual demonstration of the current (for example inserting an arrow in the circuit schematics). If the algorithm generates random currents, it is only possible to describe them textually to users, for instance: "*current* I_1 *flows from Node* A *to Node* B". However, users can alternatively define the Branch currents direction through Ammeters (in QUCS), which is quite interest in the teaching/self-learning context. In case no Ammeter exists in a Branch with a current source, the current source direction is used to determine the flow. If neither Ammeter or current source are found, then the current direction is randomly generated by the application.

6. Supernodes identification

Supernodes are detected using the voltage sources interconnection data. First, the IVS are determined by checking the delimiting Nodes type. If a voltage source has two real Nodes as boundaries, then we are in presence of an IVS, which automatically belongs to a Supernode. Supernodes with multiple voltage sources are processed by evaluating the common connections of each IVS with others. It is also in this step that *U=RIsolve* computes the best *Ground* position (Node), which returns the circuit Nodes that minimise the number of unknowns (Node voltages). In case the user does not choose the best Node as *Ground*, a tip is displayed along with the computed best *Ground* positions.

7. KCL equations building

These equations are computed based on the starting and ending Nodes from the currents objects. As previously mentioned, each Branch current direction is determined manually by users with Ammeters or randomly generated (if no Ammeter is found). Simply put, for each unknown Node voltage, if its inner Node is found to be the starting Node of a current, that means that the current is flowing out from the Node and the current label will appear in the second term of the KCL equation. On the other hand, if the end Node is found, meaning that the current is flowing in to the Node, its label will be in the first term of the equation.

8. Ohm equations building

Every current from the previous step must be replaced by its equivalent Ohm equation, which introduces the system variables we want to solve for, i.e. the Node voltages. As mentioned before, currents objects have both equivalent voltage and equivalent impedance. The Ohm's equation numerator is then computed using the current starting Node, the equivalent voltage and the ending Node. The Ohm's equation denominator is equal to the equivalent impedance of the Branch.

9. Equation System Building and Solving

A step solution like the one provided in *U=RIsolve* requires consecutive simplifications and adjustments to the equations at each step. The building of the equation system is then segmented into several stages; it begins with raw KCL equations with the currents labels, then the known currents are substituted, thereafter each current is replaced with

its Ohm's equation. The last substitution occurs if any Floating Supernodes are found, which replaces the Node voltages from a particular Supernode with an expression in order to the reference Node.

These equations adjustments presuppose tools for expression parsing/evaluation and symbolic computation. In order to build the solution parts, *U=RIsolve* uses two well-known JavaScript libraries for mathematical string manipulations: Math.js [72] and Algebra.js [73]. Furthermore, a symbolic JavaScript equation system solver [74] for real and complex numbers is employed to obtain the Node voltages results.

10. Branch Currents computation

Although this is an optional part of the algorithm, it is important to show students how to compute the currents, for a complete analysis of the circuit. In this step, the currents' calculation is divided into three categories:

- current with a known value *a priori*, i.e. there is a current source in the Branch.
- current in a Branch with a finite impedance ($> 0\ \Omega$) so that it is possible to formulate an Ohm's equation and compute it using the solution of the NVM algorithm.
- current in a Branch that belongs to a Supernode, i.e. its value can only be obtained through KCL current equations. In this scenario, the algorithm must find which KCL equation to use in order to sequentially calculate the currents. For instance, supposing that a KCL equation has three currents and one of them flows through an IVS (i_{SN}), the value of the other two currents must be previously calculated before solving the KCL equation in order to the single variable (i_{SN}).

11. Step-based output

The solution's purpose is to conduct the principles of the NVM algorithm in a simple, complete and attractive way. In *U=RIsolve* results, students can find all the Supernodes information, the circuit currents flow/direction, the KCL equations together with a canvas illustrating how they are formulated and every step of how to generate the final equations system of the NVM algorithm.

The output also includes notes and tips so that users understand how NVM works, as well as warnings that may improve their simulation skills. The mathematical content in the solution is presented in TEX typesetting format, by means of a JavaScript lightweight

parser and renderer KaTeX [75]. Additionally, the output provides methods for download the analysis in different formats:

- *JSON*: the functionality of producing a JSON file with the complete analysis of the circuit open doors for a future integration API between *U=RIsolve* and other software (e.g. an external software may be able to send an HTTP request with the *netlist* and obtain a structured circuit analysis);
- *TEX*: this file allows to use the analysis content in an academic context, for instance by students in laboratory reports, during EE course/degree;
- *PDF*: could be particularly useful to students, allowing them to print and share the analysis results, but also to teachers to prepare class exercises.

12. Critical errors output

This stage occurs if critical errors are found after the error check. Each error (critical or warning) has a specific code that is evaluated before the execution of the NVM algorithm. If any of the critical error code is found, the program immediately stops and prompts the information to users. Multiple errors and warnings can be displayed simultaneously, in a modal dialogue box.

5.4 User interface

The *U=RIsolve* application features a simple web-based user interface adaptive (responsive) to any screen size/resolution, which enables its use in laptop/desktop personal computers, tablets and smart-phones, being totally agnostic to the underlying hardware and operating system.

The homepage, illustrated in Fig. 5.6, allows users to upload both *netlist* and circuit image and perform the NVM-SA analysis. Once the user executes the NVM-SA and check the simulation results, he/she can refresh the page or upload new files and reanalyse. Work in progress addresses extending the *U=RIsolve* application with other circuit analysis methods (such as the *Loop Current Method* and the *Branch Current Method*), options that are still disabled (gray buttons).

An instructions section explains, step-by-step, the process of generating and uploading the *netlist* file to *U=RIsolve*. These guidelines start with the circuit design in QUCS and explain how to insert the Nodes labels, as well as the reference Node. It also clarifies

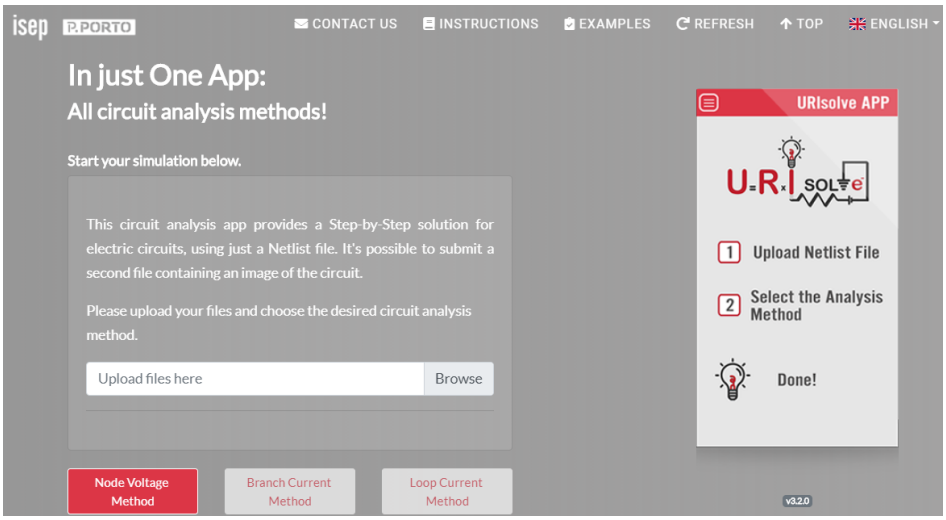


Figure 5.6: *U=RIsolve* home page.

the simulation procedure, and how to export the *netlist* file from QUCS. Moreover, instructions include a video demonstration of the circuit and *netlist* generation, along with the analysis in *U=RIsolve*, with every step described with subtitles. The instructions panel is shown in Fig. 5.7.

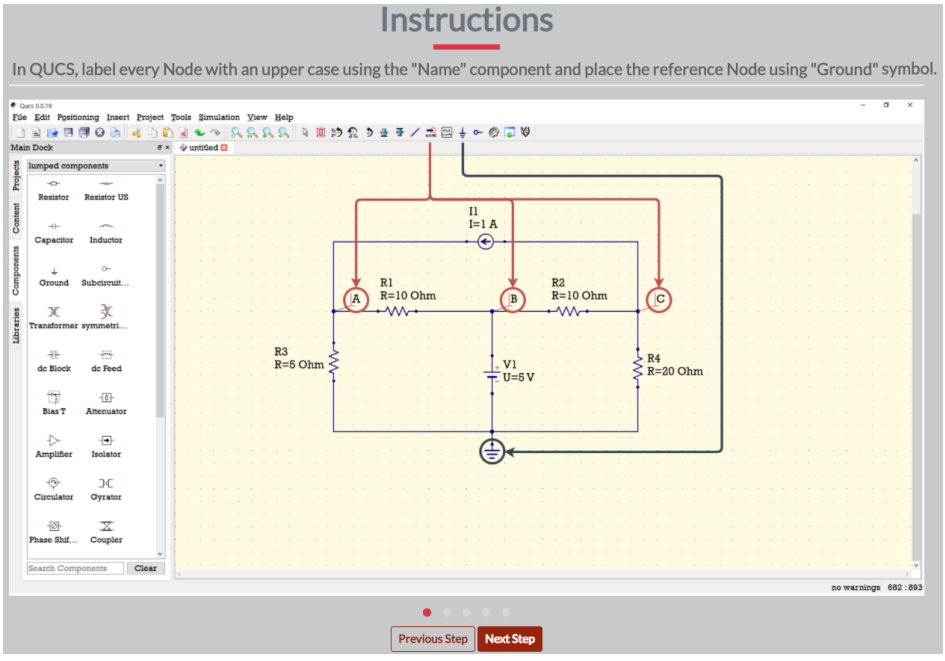


Figure 5.7: *U=RIsolve* instructions section.

Additionally, the user interface has a few ready-to-use circuits that users can analyse. These examples are ordered by complexity level and include the circuit image, a brief description of the circuit, the difficulty level and the analysis button. Two samples from this section are shown in Fig. 5.8, which shows a dynamic navigation card with four examples that alternate the visibility depending on which circuit the user selects.

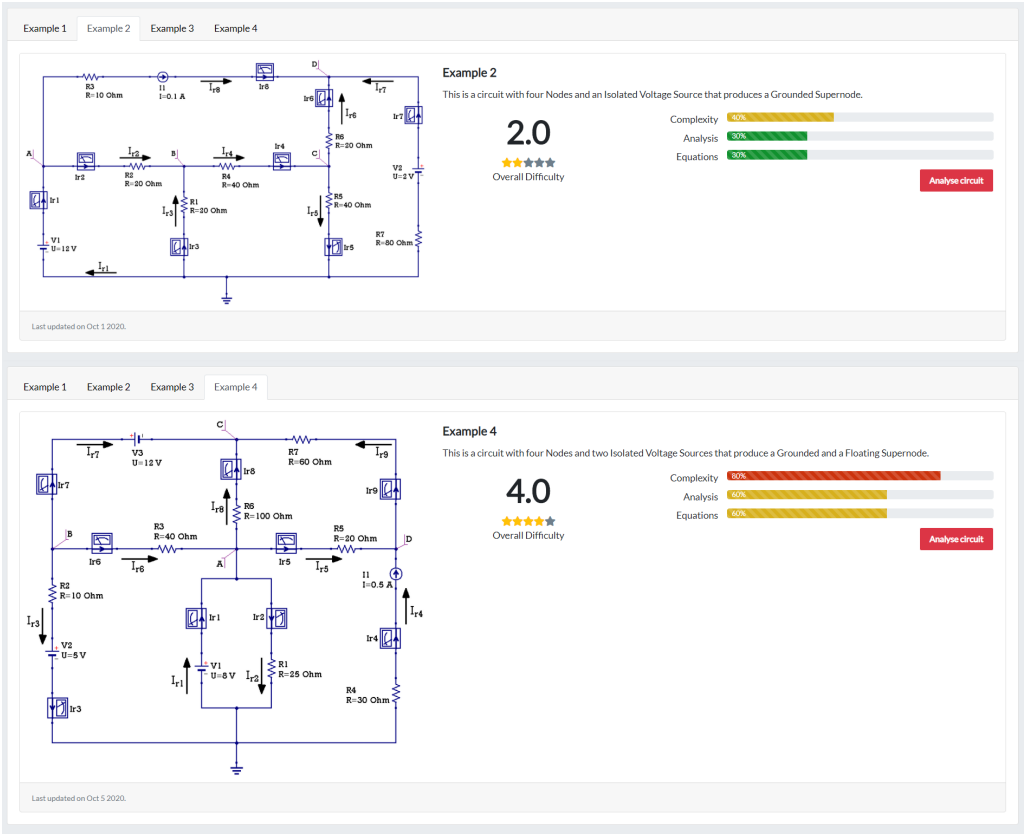


Figure 5.8: *U=RIsolve* examples section.

5.5 Analysis output

The *U=RIsolve* output is a step-by-step solution to the NVM-SA. This solution is presented in a modal dialog box, once one selects the *Node Voltage Method* button, and it is organised in nine sections. In order to demonstrate the results stages, the circuit from Fig. 4.1 (used in the NVM-SA algorithm explanation) is analysed below.

5.5.1 Circuit schematics

In this section, the uploaded image of the circuit is presented. In case users do not provide an image, this part of the output is hidden. As mentioned above, in this demonstration the circuit from Fig. 4.1 is going to be submitted to the *U=RIsolve* analysis, so its schematics will appear in this output section. Here, we opted for introducing Ammeters in all Branches (to force the selection of current directions), as shown in Fig. 5.9. The "Expand results" button allows users to view the entire solution at once, instead of seeing it step-by-step.

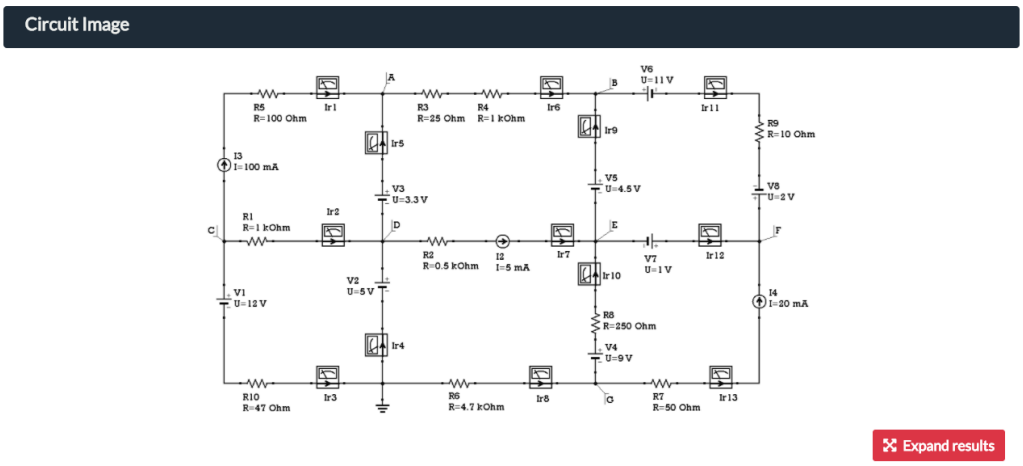


Figure 5.9: *U=RIsolve* output: circuit schematics section.

5.5.2 Fundamental variables and circuit information

These subdivisions provide details about the circuit, in particular the necessary variables to compute the number of equations needed for the uploaded circuit, which consist in the number of Branches, Nodes, IVS and NVM equations. Furthermore, extra information is given to users, such as the AC frequency (0 Hz for DC), the number of current sources (that will help students to realise how many known currents exist), the number of Ammeters (to declare the Branch currents direction), and finally the simulation type (AC or DC). In case any of the circuit Branches does not contain an Ammeter, a tip is displayed to users. Fig. 5.10 demonstrate both fundamental variables and circuit information.

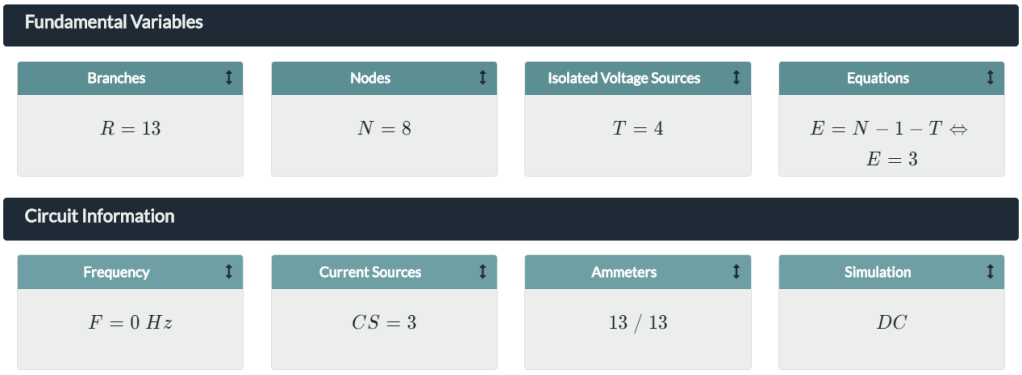


Figure 5.10: *U=RIsolve* output: fundamental variables and circuit information.

5.5.3 Supernodes

Each Supernode is described in this section. As demonstrated in Fig. 5.11, a Supernode "card" contains the reference, type (Grounded or Floating), constituent Nodes and their respective equations (or forced conditions, as introduced from the SNA algorithm). An extra description (white containers at the bottom of each Supernode card) is shown when users press the "Show Steps" button, illustrating how the equations were obtained.

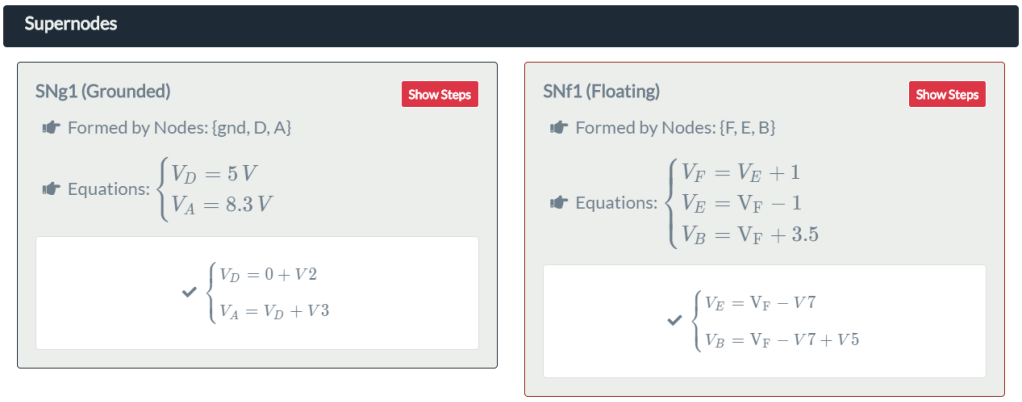


Figure 5.11: *U=RIsolve* output: Supernodes.

5.5.4 Branch Currents

This section outputs all currents and their direction, according to the Ammeters that have been placed in the simulation or generated automatically if Ammeters are not used. Fig. 5.12 denotes it outputs both starting and ending Nodes of each current, as well as its label (which is also obtained from the Ammeters, if any). For instance, current `Ir1`

flows from Node C to Node A and passes through the components $I3$ and $R5$. Since this circuit is quite extensive, some currents were not included in Fig. 5.12 for simplification reasons.

Figure 5.12: *U=RIsolve* output: Branch currents.

5.5.5 Equivalent Impedances and Voltages

This section purpose is to indicate students that the program found one or more simplifications that could have been applied to the circuit under analysis, such as in-series Resistors, Inductors and Capacitors (equivalent impedances) or in-series voltage sources (equivalent voltages). Fig. 5.13 represents the occurrences of equivalent voltages/impedances found in the example circuit, namely $ZeqAB$ (an equivalent impedance formed by $R3$ and $R4$) and $VeqBF$ (an equivalent impedance formed by $V6$ and $V8$). Future work includes the detection and exhibition of parallel impedances and current sources.

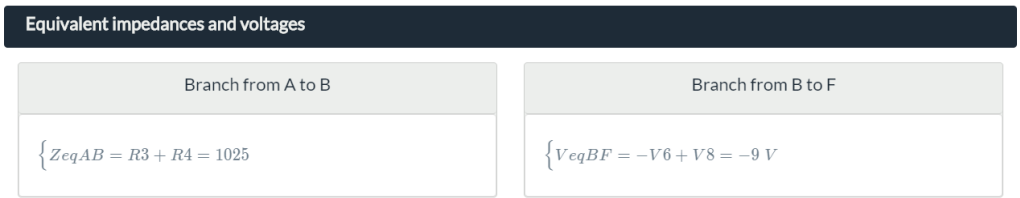


Figure 5.13: *U=RIsolve* output: equivalent impedances and voltages.

5.5.6 KCL currents equations

This section outputs the KCL equation for all Nodes, as shown in Fig. 5.14, including an illustration of the currents flowing in (drawn in green) and out (drawn in red) of each Node. The canvas objective is to offer further intuition on which currents should be in the first or in the second term of their KCL equation. Note that the KCL equation for Supernodes is shown in the simplified form, i.e. the currents that flow both in and out of the Supernode were already removed from the equation.

Figure 5.14: *U=RIsolve* output: KCL currents equations.

5.5.7 Equations system

The equations system section shows the final set of equations that enable to compute the Node voltages, as demonstrated in Fig. 5.15. Importantly, the output explains how to determine them, through six steps that describe the process of creating the system, starting from the KCL equations and making the necessary adjustments and substitutions, until finally obtaining the final set of equations. This way, it is ensured that users thoroughly understand the methodology.

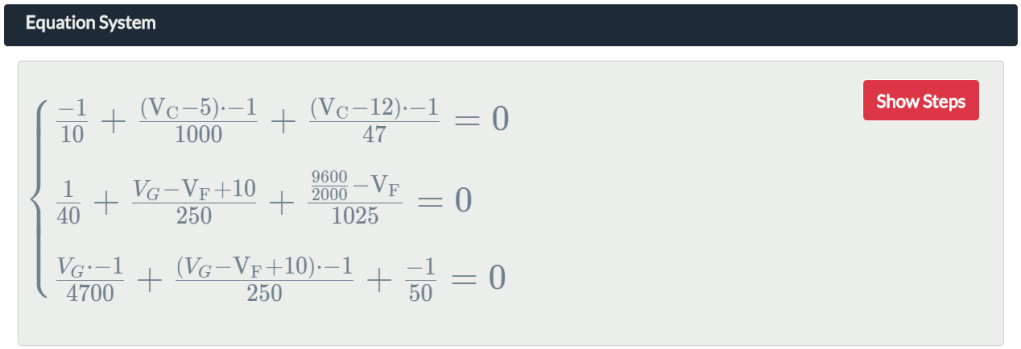


Figure 5.15: *U=RIsolve* output: equations system.

Some steps may occasionally be omitted depending on the circuit that is being simulated; for instance if the circuit does not include any Supernodes. The equations system steps pop-up once the user clicks the "Show Steps" button (Fig. 5.15).

The six steps that describe the equations system are expanded individually by users, since their purpose is to instruct the most important actions before obtaining the final set of equations. These steps are described below:

Step 1: The first step is illustrated in Fig. 5.16, and shows how to rearrange the KCL equations, so that every current is in the first term (some calculators or online linear system solvers may require this operation).

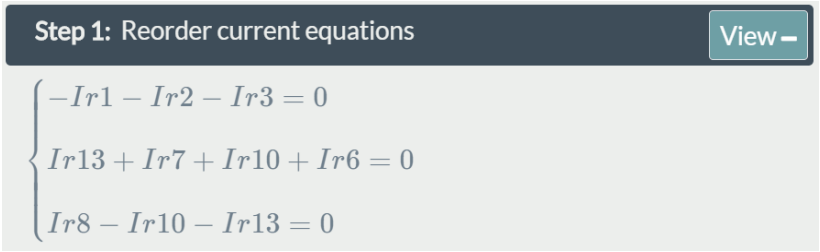


Figure 5.16: *U=RIsolve* output: equations system (Step 1).

Step 2: The second step purpose is to substitute the known current variables by their numerical value, i.e. whose Branches have a current source defining its Branch current. For the circuit under analysis, the known currents will be $Ir1$, $Ir7$ and $Ir13$. The direct substitution is represented in Fig. 5.17.

Step 3: The remaining currents in the equation system should then be replaced by their Ohm Law equation, thus introducing the actual unknowns in the system: the

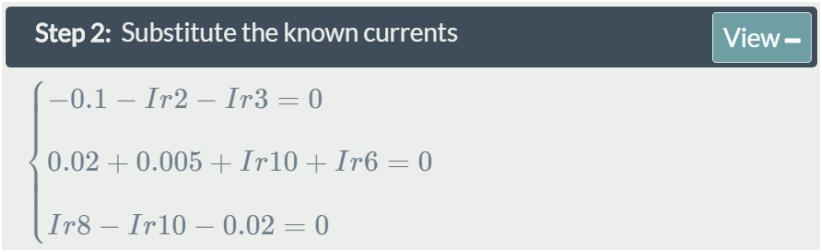


Figure 5.17: *U=RIsolve* output: equations system (Step 2).

Node voltages. As simple as Ohm's Law may seem, there is a common error-prone situation that occurs when the potential difference in a Branch with multiple voltage sources in alternate orientations needs to be computed. This step shows users how to compute the remaining currents in the system through Ohm's Law, as Fig. 5.18 outlines.

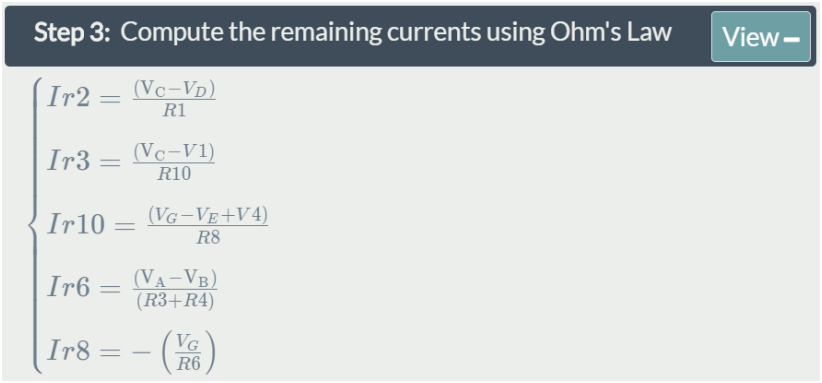


Figure 5.18: *U=RIsolve* output: equations system (Step 3).

Step 4: This step addresses the substitution of the currents in the equations system by their respective expressions, computed in Step 3. Such operation is displayed in Fig. 5.19.

Step 5: The next substitution covered in this step is related to known voltages. As observed in the previous step, equations include voltage sources $V1$ and $V4$ whose values are automatically acknowledged. As shown in Fig 5.20, every voltage source variable is replaced by its numerical value.

Step 6: The last stage involves setting a reference for each Floating Supernode, which is necessary to turn the system solvable. For instance, in the last version of the equations system from step 5, there are four unknowns (V_C, V_G, V_E and V_B),

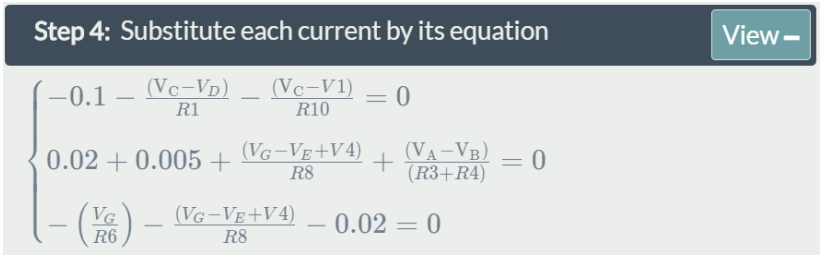


Figure 5.19: *U=RIsolve* output: equations system (Step 4).

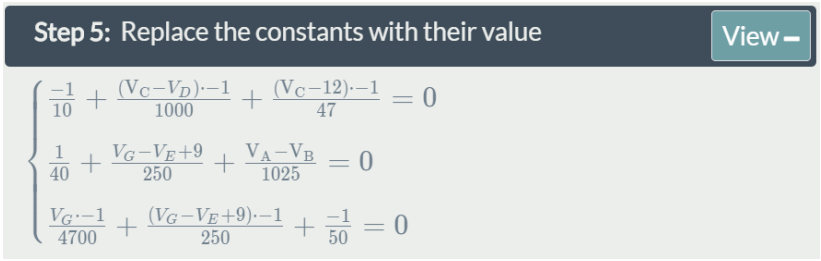


Figure 5.20: *U=RIsolve* output: equations system (Step 5).

since V_A and V_D are already computed from the Grounded Supernode. However, given that the system has just three equations, and V_E plus V_B belong to the same Supernode, it is necessary to reference them to a single potential. In this example, the program chose Node F as reference (note that in Fig. 5.11, V_E and V_B are already referenced to V_F), and this information is output as shown in Fig. 5.21.

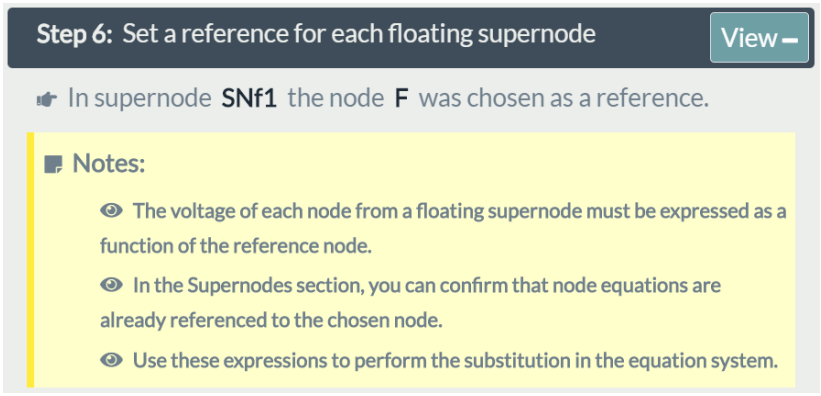


Figure 5.21: *U=RIsolve* output: equations system (Step 6).

These steps, when completed, will lead students to the exact same final equation system as the one that is shown at the beginning of the "Equation System" section, establishing an essential coherence to the learning process.

5.5.8 Results

The last section of the step-based solution are the Results, which includes the Node voltages and the Branch currents. For the example, the Node voltages output is shown in Fig. 5.22.

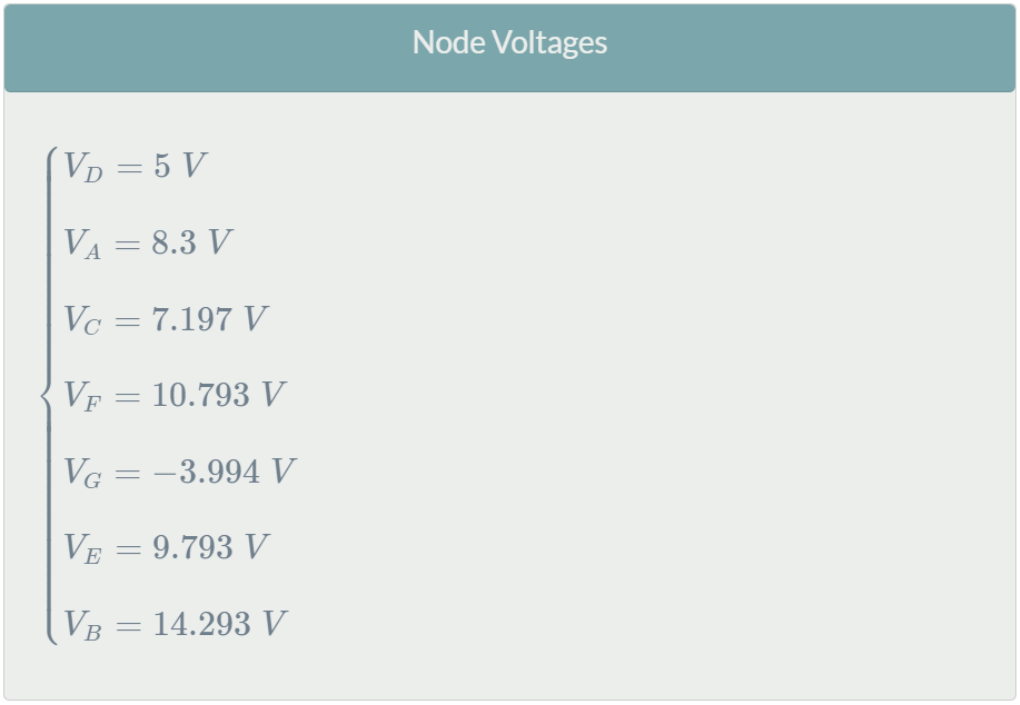


Figure 5.22: *U=RIsolve* output: Node voltages results.

As explained before, in NVM-SA algorithm, currents are segmented into three types: obtained by an existing current source in the Branch; computed through the Ohm's Law equation; and obtained through the KCL equation (in case the current belongs to a Branch with one or more IVS). Each current is provided with the expression (Ohm's Law or KCL equation) and the respective result in order to clarify users which equation was used to acquire such results. Furthermore, both Branch currents and Node voltages results are evaluated by the algorithm, which computes the appropriate unit for them to be displayed, e.g. if a Node voltage has a value of 0.001 V, it will be displayed in mV (1 mV). The results for the Branch currents are demonstrated in Fig. 5.23.

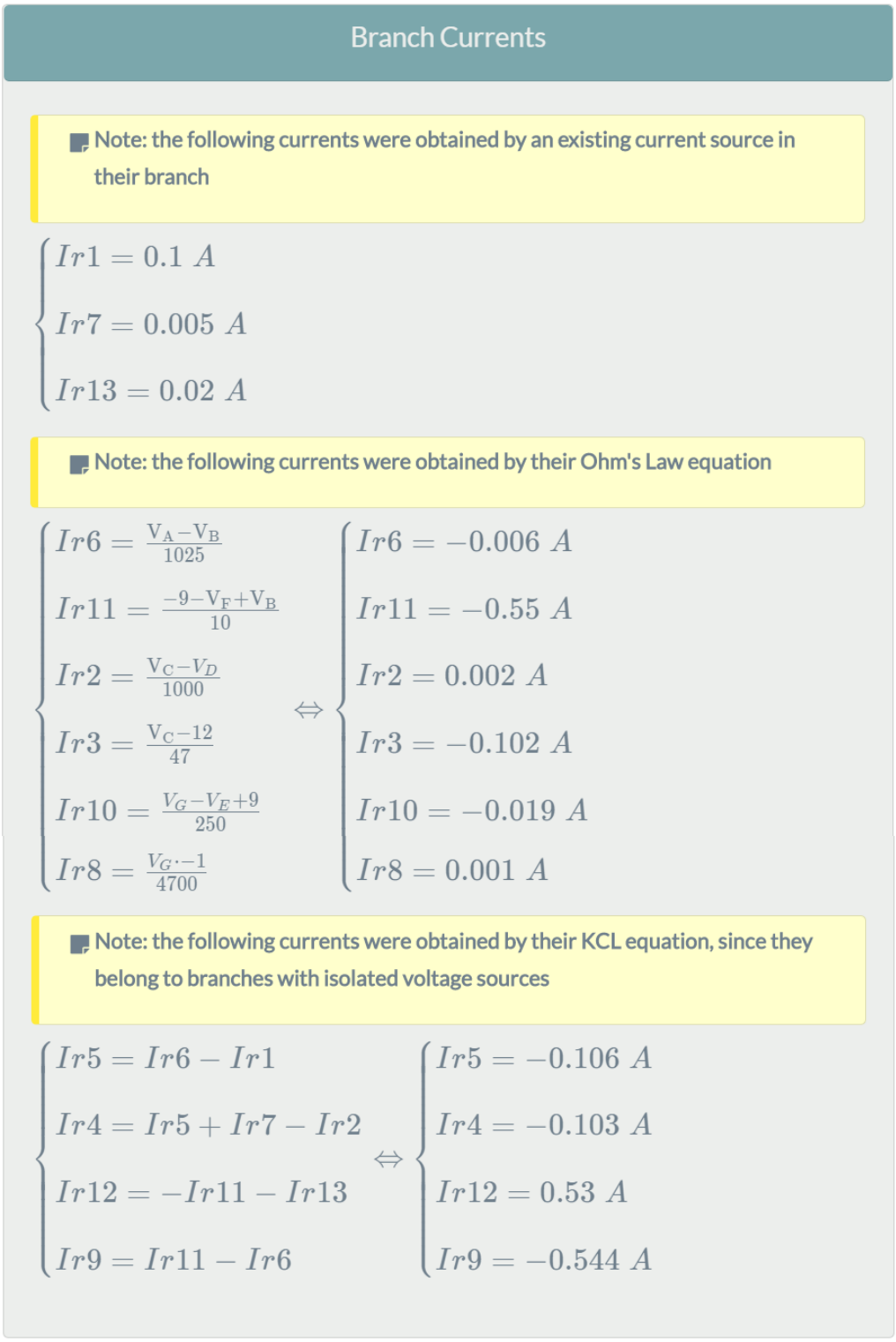


Figure 5.23: *U=RIsolve* output: Branch currents results.

The presented NVM-SA output grants a progressive analysis process, starting by revealing some details and characteristics of the circuit, it then addresses the Supernodes and Branch currents, providing as well an illustration of the KCL equations. A step-by-step demonstration of how to calculate the equation system is also provided and the solution ends with the results for Node voltages and Branch currents (which are segmented into three types so that the user understand how they where computed). This output is further evaluated in the following Chapter with a few case studies.

Chapter 6

Case Studies

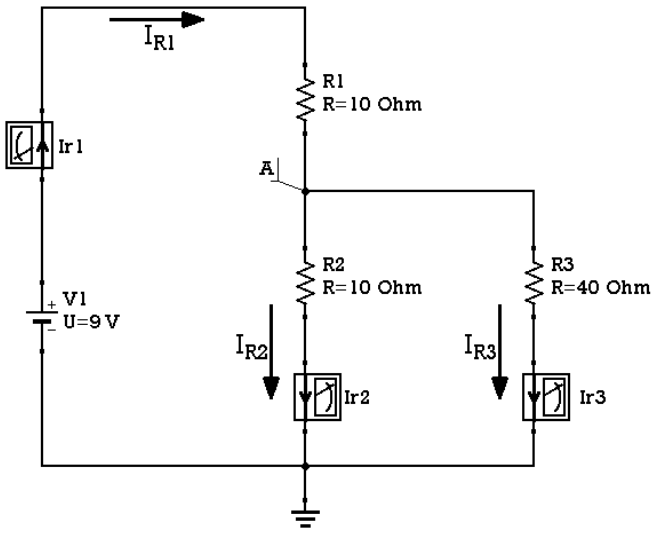
This chapter provides an insightful evaluation of the U=RIsolve application output. A set of electrical circuits is presented and simulated in U=RIsolve, ranging from DC to AC circuits with different complexity levels (e.g. no Supernode, a Grounded Supernode, a Floating Supernode). Interestingly, the analysis results are directly pasted to this report from the PDF output available in the application (the graphical version in the GUI (that users see) is illustrated in Appendix A. The circuit schematics are directly imported from QUCS and show the simulated "measurements" for the Node voltages and Branch currents, to enable to match them with the numerical results generated by the U=RIsolve output.

6.1 DC circuit without Supernodes

The first circuit to be analysed is a simple voltage divider with two Nodes and consequently just a single unknown Node voltage. Each current direction is determined with an Ammeter and an additional demonstration is given with the arrows that represent the currents flow.

Fig. 6.1 includes the QUCS results for Node voltages and Branch currents. The *U=RIsolve* output is presented below, however the web-based graphical result is given in the Appendix A.1.

Circuit Image



Node Voltages

number	A.V
1	4

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I
1	0.5	0.4	0.1

Figure 6.1: DC Circuit without Supernodes

Fundamental Variables

Branches [R]	Nodes [N]	IVS [T]	Equations [E]
R=3	N=2	T=0	E=N-T-1=1

Circuit Information

Frequency [F]	Current Sources [I]	Ammeters [A]	Simulation
F=0	I=0	3/3	DC

Circuit Currents

General information

Table 6.1: List of the Currents and their properties for the first case-study

Reference	Start Node	End Node	Components
Ir3	A	gnd	R3
Ir2	A	gnd	R2
Ir1	gnd	A	V1, R1

Equations

Equations using the Kirchhoff Currents Law (KCL)

Node A

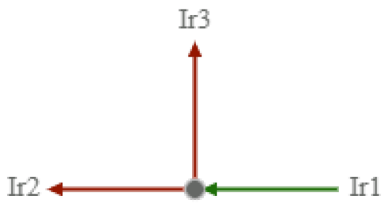


Figure 6.2: Case-study 1 – Node A currents.

$$\text{Equation}: \quad Ir1 = Ir3 + Ir2 \tag{6.1}$$

Equation System

Equations:

$$\frac{V_A \cdot -50}{400} + \frac{9 - V_A}{10} = 0 \tag{6.2}$$

Steps:

Step 1: Reorder current equations

$$Ir1 - Ir3 - Ir2 = 0 \tag{6.3}$$

Step 2: Substitute the known currents

$$Ir1 - Ir3 - Ir2 = 0 \tag{6.4}$$

Step 3: Compute the remaining currents using Ohm's Law

$$\begin{cases} Ir1 = \frac{(V1 - \mathrm{V_A})}{R1} \\ Ir3 = \frac{\mathrm{V_A}}{R3} \\ Ir2 = \frac{\mathrm{V_A}}{R2} \end{cases} \tag{6.5}$$

Step 4: Substitute each current by its equation

$$\frac{(V1 - \mathrm{V_A})}{R1} - \frac{\mathrm{V_A}}{R3} - \frac{\mathrm{V_A}}{R2} = 0 \tag{6.6}$$

Step 5: Replace the constants with their value

$$\frac{\mathrm{V_A} \cdot -50}{400} + \frac{9 - \mathrm{V_A}}{10} = 0 \tag{6.7}$$

Results

Node Voltages

$$V_A = 4\ V \tag{6.8}$$

Branch Currents

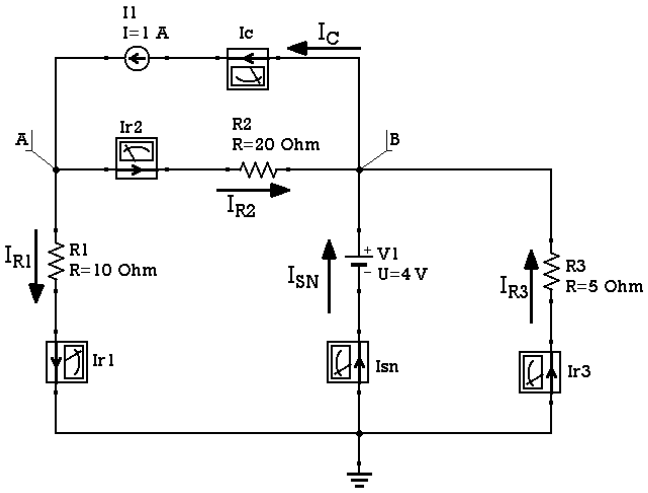
$$\begin{cases} Ir3 = \frac{\mathrm{V_A}}{40} \\ Ir2 = \frac{\mathrm{V_A}}{10} \\ Ir1 = \frac{9 - \mathrm{V_A}}{10} \end{cases} \Leftrightarrow \begin{cases} Ir3 = 0.1\ A \\ Ir2 = 0.4\ A \\ Ir1 = 0.5\ A \end{cases} \tag{6.9}$$

Note: These currents were obtained by their Ohm's Law equation.

6.2 DC Circuit with a Grounded Supernode

The next circuit has two Nodes and the *Ground*, but just one unknown Node voltage since there exist a Supernode which is connected to the reference Node, meaning it defines a Grounded Supernode. Each current's direction is established by the Ammeters, inclusively the Branch with a current source. Fig. 6.3 illustrates the example circuit, followed by its resolution given by *U=RIsolve*.Appendix A.2 represents the graphical output provided at *U=RIsolve* application.

Circuit Image



Node Voltages

number	A.V	B.V
1	8	4

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Isn.I	Ic.I
1	0.8	0.2	-0.8	1.6	1

Figure 6.3: DC Circuit with a Grounded Supernode.

Fundamental Variables

Branches [R]	**Nodes [N]**	**IVS [T]**	**Equations [E]**
R=5	N=3	T=1	E=N-T-1=1

Circuit Information

Frequency [F]	Current Sources [I]	Ammeters [A]	Simulation
F=0	I=1	5/5	DC

Supernodes

Grounded Supernode

SNg

Formed by Nodes: gnd, B

Equations:

$$V_B = 4\,V \tag{6.10}$$

Steps:

$$V_B = 0 + V1 \tag{6.11}$$

Circuit Currents

General information

Table 6.2: List of the Currents and their properties for the second case-study

Reference	Start Node	End Node	Components
Ir2	A	B	R2
Ir1	A	gnd	R1
Ic	B	A	I1
Ir3	gnd	B	R3
Isn	gnd	B	V1

Equations

Equations using the Kirchhoff Currents Law (KCL)

Node A

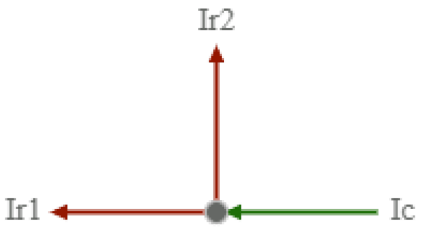


Figure 6.4: Case-study 2 – Node A currents.

$$\text{Equation}: \quad Ic = Ir2 + Ir1 \tag{6.12}$$

Equation System

Equations:

$$\frac{(V_A - 4)\cdot -1}{20} + \frac{V_A \cdot -1}{10} + 1 = 0 \tag{6.13}$$

Steps:

Step 1: Reorder current equations

$$Ic - Ir2 - Ir1 = 0 \tag{6.14}$$

Step 2: Substitute the known currents

$$1 - Ir2 - Ir1 = 0 \tag{6.15}$$

Step 3: Compute the remaining currents using Ohm's Law

$$\begin{cases} Ir2 = \frac{(V_A - V_B)}{R2} \\ Ir1 = \frac{V_A}{R1} \end{cases} \tag{6.16}$$

Step 4: Substitute each current by its equation

$$1 - \frac{(V_A - V_B)}{R2} - \frac{V_A}{R1} = 0 \tag{6.17}$$

Step 5: Replace the constants with their value

$$\frac{(V_A - V_B) \cdot -1}{20} + \frac{V_A \cdot -1}{10} + 1 = 0 \tag{6.18}$$

Results

Node Voltages

$$\begin{cases} V_B = 4\ V \\ V_A = 8\ V \end{cases} \tag{6.19}$$

Branch Currents

$$Ic = 1\ A \tag{6.20}$$

Note: These currents were obtained by an existing Current Source in their Branch.

$$\begin{cases} Ir2 = \frac{V_A - V_B}{20} \\ Ir1 = \frac{V_A}{10} \\ Ir3 = \frac{V_B \cdot -1}{5} \end{cases} \Leftrightarrow \begin{cases} Ir2 = 0.2\ A \\ Ir1 = 0.8\ A \\ Ir3 = -0.8\ A \end{cases} \tag{6.21}$$

Note: These currents were obtained by their Ohm's Law equation.

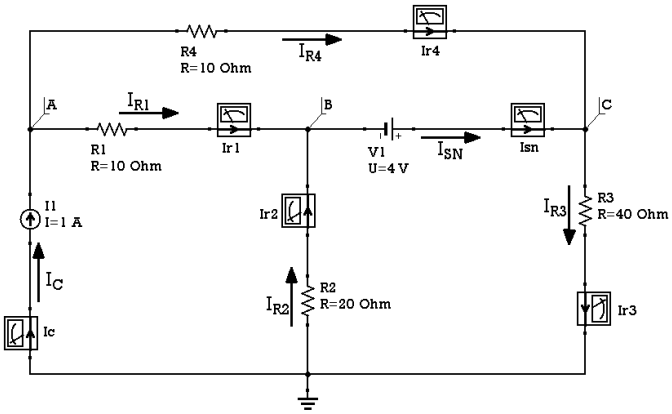
$$Isn = Ic - Ir2 - Ir3 \Leftrightarrow Isn = 1.6\ A \tag{6.22}$$

Note: These currents were obtained by their KCL equation, since they belong to Branches with Isolated Voltage Sources.

6.3 DC Circuit with a Floating Supernode

The following circuit has three unknown Node voltages, nevertheless, considering that there exist a Floating Supernode between Nodes B and C, only two KCL equations are needed to solve the circuit. Furthermore, an additional step appears at the equation system, since a reference must be chosen for the existing Supernode. This example circuit is demonstrated in Fig. 6.5, followed by its resolution given by *U=RIsolve*. The graphical output is shown in Appendix A.3.

Circuit Image



Node Voltages

number	A.V	B.V	C.V
1	19	12	16

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Ir4.I	Isn.I	Ic.I
1	0.7	-0.6	0.4	0.3	0.1	1

Figure 6.5: DC Circuit with a Floating Supernode.

Fundamental Variables

Branches [R]	**Nodes [N]**	**IVS [T]**	**Equations [E]**
R=6	N=4	T=1	E=N-T-1=2

Circuit Information

Frequency [F]	**Current Sources [I]**	**Ammeters [A]**	**Simulation**
F=0	I=1	6/6	DC

Supernodes

Floating Supernode

SNf1

Formed by Nodes: C, B

Equations:

$$V_B = \mathrm{V_C} - 4 \tag{6.23}$$

Steps:

$$V_B = \mathrm{V_C} - V1 \tag{6.24}$$

Circuit Currents

General information

Table 6.3: List of the Currents and their properties for the third case-study

Reference	Start Node	End Node	Components
Ir1	A	B	R1
Ir4	A	C	R4
Ic	gnd	A	I1
Ir2	gnd	B	R2
Isn	B	C	V1
Ir3	C	gnd	R3

Equations

Equations using the Kirchhoff Currents Law (KCL)

Node A

$$\text{Equation}: \quad Ic = Ir1 + Ir4 \tag{6.25}$$

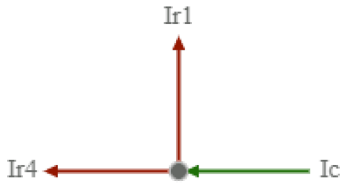


Figure 6.6: Case-study 3 – Node A currents.

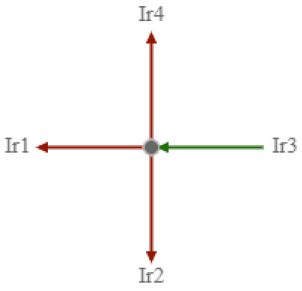


Figure 6.7: Case-study 3 – Node Snf1 currents.

Node Snf1

$$\text{Equation}: \quad Ir3 = Ir4 + Ir1 + Ir2 \tag{6.26}$$

Equation System

Equations:

$$\begin{cases} \frac{(2 \cdot V_A - 2 \cdot V_C + 4) \cdot -1}{10} + 1 = 0 \\ \frac{(2 \cdot V_A - 2 \cdot V_C + 4) \cdot -1}{10} + \frac{V_C}{40} + \frac{V_C - 4}{20} = 0 \end{cases} \tag{6.27}$$

Steps:

Step 1: Reorder current equations

$$\begin{cases} Ic - Ir1 - Ir4 = 0 \\ Ir3 - Ir4 - Ir1 - Ir2 = 0 \end{cases} \tag{6.28}$$

Step 2: Substitute the known currents

$$\begin{cases} 1 - Ir1 - Ir4 = 0 \\ Ir3 - Ir4 - Ir1 - Ir2 = 0 \end{cases} \tag{6.29}$$

Step 3: Compute the remaining currents using Ohm's Law

$$\begin{cases} Ir1 = \frac{(V_A - V_B)}{R1} \\ Ir4 = \frac{(V_A - V_C)}{R4} \\ Ir3 = \frac{V_C}{R3} \\ Ir2 = -\left(\frac{V_B}{R2}\right) \end{cases} \tag{6.30}$$

Step 4: Substitute each current by its equation

$$\begin{cases} 1 - \frac{(V_A - V_B)}{R1} - \frac{(V_A - V_C)}{R4} = 0 \\ \frac{V_C}{R3} - \frac{(V_A - V_C)}{R4} - \frac{(V_A - V_B)}{R1} + \left(\frac{V_B}{R2}\right) = 0 \end{cases} \tag{6.31}$$

Step 5: Replace the constants with their value

$$\begin{cases} \frac{(2 \cdot V_A - V_B - V_C) \cdot -1}{10} + 1 = 0 \\ \frac{(2 \cdot V_A - V_C - V_B) \cdot -1}{10} + \frac{V_C}{40} + \frac{V_B}{20} = 0 \end{cases} \tag{6.32}$$

Step 6: Set a reference for each floating Supernode

In Supernode **SNf1** the node **C** was chosen as a reference.

Notes:

- The voltage of each node from a floating Supernode must be expressed as a function of the reference node.
- In the Supernodes section, you can confirm that node equations are already referenced to the chosen node.
- Use these expressions to perform the substitution in the equation system.

Results

Node Voltages

$$\begin{cases} V_A = 19\ V \\ V_C = 16\ V \\ V_B = 12\ V \end{cases} \tag{6.33}$$

Branch Currents

$$Ic = 1\ A \tag{6.34}$$

Note: These currents were obtained by an existing Current Source in their Branch.

$$\begin{cases} Ir1 = \frac{V_A - V_B}{10} \\ Ir4 = \frac{V_A - V_C}{10} \\ Ir2 = \frac{V_B \cdot -1}{20} \\ Ir3 = \frac{V_C}{40} \end{cases} \Leftrightarrow \begin{cases} Ir1 = 0.7\ A \\ Ir4 = 0.3\ A \\ Ir2 = -0.6\ A \\ Ir3 = 0.4\ A \end{cases} \tag{6.35}$$

Note: These currents were obtained by their Ohm's Law equation.

$$Isn = Ir1 + Ir2 \Leftrightarrow Isn = 0.1\ A \tag{6.36}$$

Note: These currents were obtained by their KCL equation, since they belong to Branches with Isolated Voltage Sources.

6.4 AC circuit with a Grounded Supernode

The next example is an AC circuit with two Nodes and one Grounded Supernode, which means there is a single unknown, computed through one KCL equation. This circuit demonstrates the flexibility of *U=RIsolve* on dealing with operations containing complex numbers. Fig. 6.5 shows the example circuit, followed by its resolution given by *U=RIsolve*. The graphical output is exhibited in Appendix A.4

Circuit Image

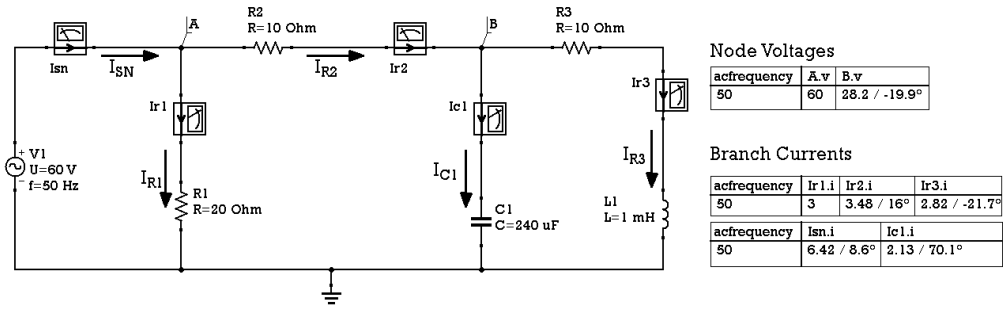


Figure 6.8: AC Circuit with a Grounded Supernode

Fundamental Variables

Branches [R]	**Nodes [N]**	**IVS [T]**	**Equations [E]**
R=5	N=3	T=1	E=N-T-1=1

Circuit Information

Frequency [F]	**Current Sources [I]**	**Ammeters [A]**	**Simulation**
F=50 Hz	I=0	5/5	AC

Supernodes

Grounded Supernode

SNg

Formed by Nodes: gnd, A

Equations:

$$V_A = 60\,V \tag{6.37}$$

Steps:

$$V_A = 0 + V1 \tag{6.38}$$

Circuit Currents

General information

Table 6.4: List of the Currents and their properties for the fourth case-study

Reference	Start Node	End Node	Components
Ir1	A	gnd	R1
Ir2	A	B	R2
Isn	gnd	A	V1
Ir3	B	gnd	L1, R3
Ic1	B	gnd	C1

Equivalent Impedances and Voltages

Branch from B to gnd

$$ZeqBgnd = L1 + R3 = 10 + 0.314i \tag{6.39}$$

Equations

Equations using the Kirchhoff Currents Law (KCL)

Node B

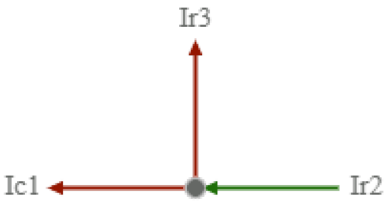


Figure 6.9: Case-study 4 – Node B currents.

$$\text{Equation}: \quad Ir2 = Ir3 + Ic1 \tag{6.40}$$

Equation System

Equations:

$$\frac{60 - V_B}{10} - \left(\frac{1}{\frac{i \cdot 157}{500} + 10} + \frac{1}{\frac{i \cdot -6.631 \cdot 10^{+5}}{50000}} \right) \cdot V_B = 0 \tag{6.41}$$

Steps:

Step 1: Reorder current equations

$$Ir2 - Ir3 - Ic1 = 0 \tag{6.42}$$

Step 2: Substitute the known currents

$$Ir2 - Ir3 - Ic1 = 0 \tag{6.43}$$

Step 3: Compute the remaining currents using Ohm's Law

$$\begin{cases} Ir2 = \frac{(V_A - V_B)}{R2} \\ Ir3 = \frac{V_B}{(XL1 + R3)} \\ Ic1 = \frac{V_B}{XC1} \end{cases} \tag{6.44}$$

Step 4: Substitute each current by its equation

$$\frac{(V_A - V_B)}{R2} - \frac{V_B}{(XL1 + R3)} - \frac{V_B}{XC1} = 0 \tag{6.45}$$

Step 5: Replace the constants with their value

$$\frac{V_A - V_B}{10} - \left(\frac{1}{\frac{i \cdot 157}{500} + 10} + \frac{1}{\frac{i \cdot -6.631 \cdot 10^{+5}}{50000}} \right) \cdot V_B = 0 \tag{6.46}$$

Results

Node Voltages

$$\begin{cases} V_A = 60\ V \\ V_B = 28.227 \angle -19.874°\ V \end{cases} \tag{6.47}$$

Branch Currents

$$\begin{cases} Ir1 = \frac{V_A}{20} \\ Ir2 = \frac{V_A - V_B}{10} \\ Ir3 = \frac{V_B}{\frac{i \cdot 3.141 \cdot 10^{+5}}{1 \cdot 10^{+6}} + 10} \\ Ic1 = \frac{V_B}{\frac{i \cdot -2.0723 \cdot 10^{+7}}{1.562 \cdot 10^{+6}}} \end{cases} \Leftrightarrow \begin{cases} Ir1 = 3\ A \\ Ir2 = 3.48 \angle 16.013°\ A \\ Ir3 = 2.821 \angle -21.673°\ A \\ Ic1 = 2.129 \angle 70.118°\ A \end{cases} \tag{6.48}$$

Note: These currents were obtained by their Ohm's Law equation.

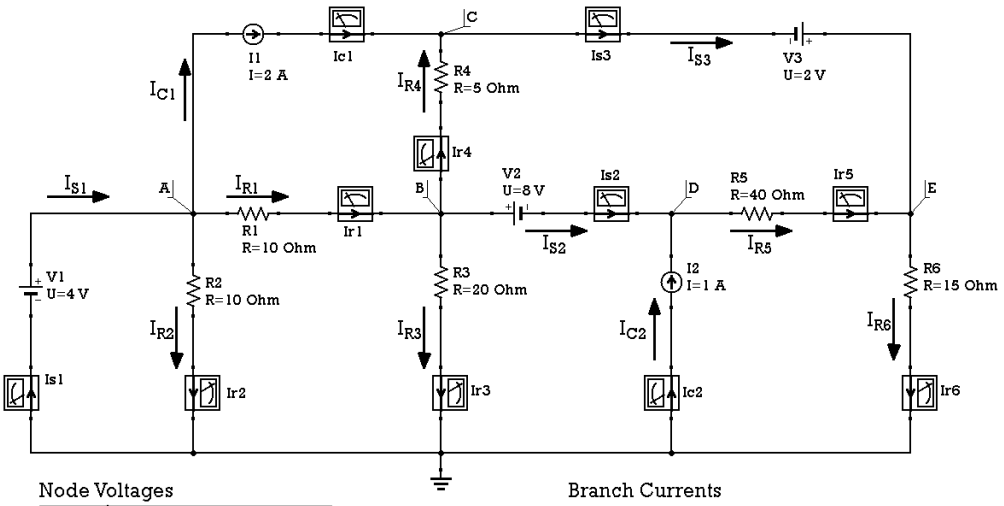
$$Isn = Ir1 + Ir2 \Leftrightarrow Isn = 6.417 \angle 8.604°\ A \tag{6.49}$$

Note: These currents were obtained by their KCL equation, since they belong to Branches with Isolated Voltage Sources.

6.5 DC Circuit with multiple Supernodes

The last case-study is a circuit with five Nodes, one Grounded Supernode and two Floating Supernodes, which imposes four unknown Node voltages and a system of two equations to obtain them. This example circuit demonstrates that *U=RIsolve* is able to solve more complex circuits and it is illustrated in Fig. 6.10. Accordingly, its graphical resolution given by *U=RIsolve* is shown in Appendix A.5 and the textual solution is presented hereinafter.

Circuit Image



Node Voltages

number	A.V	B.V	C.V	D.V	E.V
1	4	14.4	16.6	6.38	18.6

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Ir4.I	Ir5.I	Ir6.I
1	-1.04	0.4	0.719	-0.451	-0.306	1.24

number	Ic1.I	Ic2.I	Is1.I	Is2.I	Is3.I
1	2	1	1.36	-1.31	1.55

Figure 6.10: DC Circuit with a multiple Supernodes.

Fundamental Variables

Branches [R]	Nodes [N]	IVS [T]	Equations [E]
R=11	N=6	T=3	E=N-T-1=2

Circuit Information

Frequency [F]	Current Sources [I]	Ammeters [A]	Simulation
F=0	I=2	11/11	DC

Supernodes

Grounded Supernode

SNg1

Formed by Nodes: gnd, A

Equations:

$$V_A = 4\,V \tag{6.50}$$

Steps:

$$V_A = 0 + V1 \tag{6.51}$$

Floating Supernode

SNf1

Formed by Nodes: B, D

Equations:

$$V_D = \mathrm{V_B} - 8 \tag{6.52}$$

Steps:

$$V_D = \mathrm{V_B} - V2 \tag{6.53}$$

Floating Supernode

SNf2

Formed by Nodes: E, C

Equations:

$$V_C = V_E - 2 \tag{6.54}$$

Steps:

$$V_C = V_E - V3 \tag{6.55}$$

Circuit Currents

General information

Table 6.5: List of the Currents and their properties for the fifth case-study

Reference	Start Node	End Node	Components
Ir1	A	B	R1
Ir2	A	gnd	R2
Is1	gnd	A	V1
Ic1	A	C	I1
Ir4	B	C	R4
Ir3	B	gnd	R3
Is2	B	D	V2
Is3	C	E	V3
Ir5	D	E	R5
Ic2	gnd	D	I2
Ir6	E	gnd	R6

Equations

Equations using the Kirchhoff Currents Law (KCL)

Node Snf1

$$\text{Equation}: \quad Ir1 + Ic2 = Ir4 + Ir3 + Ir5 \tag{6.56}$$

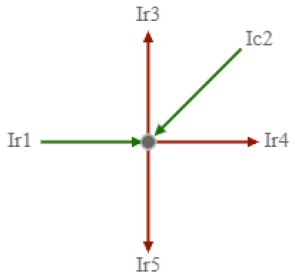


Figure 6.11: Case-study 5 – Node Snf1 currents.

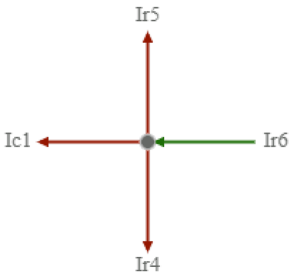


Figure 6.12: Case-study 5 – Node Snf2 currents.

Node Snf2

$$\text{Equation}: \quad Ir6 = Ir5 + Ic1 + Ir4 \tag{6.57}$$

Equation System

Equations:

$$\begin{cases} \frac{V_E - 2 - \mathrm{V_B}}{5} + \frac{\mathrm{V_B} \cdot -1}{20} + \frac{4 - \mathrm{V_B}}{10} + \frac{(\mathrm{V_B} - 8 - V_E) \cdot -1}{40} + 1 = 0 \\ \frac{V_E}{15} - 2 + \frac{(\mathrm{V_B} - 8 - V_E) \cdot -1}{40} + \frac{(\mathrm{V_B} - V_E + 2) \cdot -1}{5} = 0 \end{cases} \tag{6.58}$$

Steps:

Step 1: Reorder current equations

$$\begin{cases} -Ir4 - Ir3 + Ir1 - Ir5 + Ic2 = 0 \\ Ir6 - Ir5 - Ic1 - Ir4 = 0 \end{cases} \tag{6.59}$$

Step 2: Substitute the known currents

$$\begin{cases} -Ir4 - Ir3 + Ir1 - Ir5 + 1 = 0 \\ Ir6 - Ir5 - 2 - Ir4 = 0 \end{cases} \tag{6.60}$$

Step 3: Compute the remaining currents using Ohm's Law

$$\begin{cases} Ir4 = \frac{(V_B - V_C)}{R4} \\ Ir3 = \frac{V_B}{R3} \\ Ir1 = \frac{(V_A - V_B)}{R1} \\ Ir5 = \frac{(V_D - V_E)}{R5} \\ Ir6 = \frac{V_E}{R6} \end{cases} \tag{6.61}$$

Step 4: Substitute each current by its equation

$$\begin{cases} \frac{-(V_B - V_C)}{R4} - \frac{V_B}{R3} + \frac{(V_A - V_B)}{R1} - \frac{(V_D - V_E)}{R5} + 1 = 0 \\ \frac{V_E}{R6} - \frac{(V_D - V_E)}{R5} - 2 - \frac{(V_B - V_C)}{R4} = 0 \end{cases} \tag{6.62}$$

Step 5: Replace the constants with their value

$$\begin{cases} \frac{V_C - V_B}{5} + \frac{V_B \cdot -1}{20} + \frac{V_A - V_B}{10} + \frac{(V_D - V_E) \cdot -1}{40} + 1 = 0 \\ \frac{V_E}{15} - 2 + \frac{(V_D - V_E) \cdot -1}{40} + \frac{(V_B - V_C) \cdot -1}{5} = 0 \end{cases} \tag{6.63}$$

Step 6: Set a reference for each floating Supernode

In Supernode **SNf1** the node **B** was chosen as a reference.

In Supernode **SNf2** the node **E** was chosen as a reference.

Notes:

- The voltage of each node from a floating Supernode must be expressed as a function of the reference node.
- In the Supernodes section, you can confirm that node equations are already referenced to the chosen node.
- Use these expressions to perform the substitution in the equation system.

Results

Node Voltages

$$\begin{cases} V_A = 4\,V \\ V_B = 14.383\,V \\ V_E = 18.638\,V \\ V_D = 6.383\,V \\ V_C = 16.638\,V \end{cases} \tag{6.64}$$

Branch Currents

$$\begin{cases} Ic1 = 2\,A \\ Ic2 = 1\,A \end{cases} \tag{6.65}$$

Note: These currents were obtained by an existing Current Source in their Branch.

$$\begin{cases} Ir1 = \frac{V_A - V_B}{10} \\ Ir2 = \frac{V_A}{10} \\ Ir4 = \frac{V_B - V_C}{5} \\ Ir3 = \frac{V_B}{20} \\ Ir5 = \frac{V_D - V_E}{40} \\ Ir6 = \frac{V_E}{15} \end{cases} \Leftrightarrow \begin{cases} Ir1 = -1.038\,A \\ Ir2 = 0.4\,A \\ Ir4 = -0.451\,A \\ Ir3 = 0.719\,A \\ Ir5 = -0.306\,A \\ Ir6 = 1.243\,A \end{cases} \tag{6.66}$$

Note: These currents were obtained by their Ohm's Law equation.

$$\begin{cases} Is1 = Ir1 + Ir2 + Ic1 \\ Is2 = -Ir4 - Ir3 + Ir1 \\ Is3 = Ic1 + Ir4 \end{cases} \Leftrightarrow \begin{cases} Is1 = 1.362\,A \\ Is2 = -1.306\,A \\ Is3 = 1.549\,A \end{cases} \tag{6.67}$$

Note: These currents were obtained by their KCL equation, since they belong to Branches with Isolated Voltage Sources.

Chapter 7

Conclusions

7.1 Outcomes

The presented dissertation addressed the design and implementation of a web-based application – *U=RIsolve* – which aims at motivating and improving the self-learning skills of undergraduate Electrical Engineering students during the early stages of circuit analysis, specifically concerning the *Node Voltage Method*. The application relies on a robust and comprehensive NVM algorithm building on the Supernode paradigm – dubbed NVM-SA – that has not been consolidated and implemented in software to date, to our best knowledge. This application introduces a didactic component to circuit simulators, enabling to explore innovative and more efficient teaching and self-learning methodologies. *U=RIsolve* relies on the QUCS simulator to provide the electrical circuit mathematical model (*netlist* file), including the identification (labelling) of the circuit Nodes, which voltage is to be determined by the NVM. Then the *U=RIsolve* application provides a step-based solution that includes circuit information, the NVM analysis process and the numerical results for Branch currents and Node voltages, allowing students to fully understand how this method is applied, step-by-step. Furthermore, *U=RIsolve* is also being used by teachers for the development of new circuit analysis exercises and as a complementary way of in-class instruction.

7.2 Lessons learned

With the course of the work described in this essay, a few challenges have arisen. The first difficulty was concerned to the exhaustive continuous evaluation and testing of the application, given that there exist countless circuit layouts with various complexities,

which required the framework to encompass multiple exceptions and peculiarities. Thus, with the purpose of providing a robust analysis tool for the largest number of circuits, an in-depth experimentation phase was necessary before releasing a stable "bullet-proof" version.

Moreover, as a global pandemic was declared in the early stages of this work, a lot of procedures had to be readjusted, given the conditions of house confinement and social distancing, such as team reunions, discussions and decision makings. However impossible it may seem, a few positive outcomes were encountered from this situation, mostly the improvement of soft skills such like organisation, effective planning and communication, as well as the development of the work ethic, in order to ensure a steady pace of work through this project development.

In addition to the above-mentioned contributions, this dissertation allowed to consolidate not merely computational skills regarding web development, but also scientific knowledge through the evaluation of the assorted circuit analysis algorithms. In fact, there was the possibility to extend the personal knowledge and expertise as a result of studying and implementing front-end technologies that produced an interactive and responsive user interface to the *U=RIsolve* application. Otherwise, the inherent proceedings to a scientific paper submission, in particular the background study, the writing and the planning/selection of target journals allowed to take a step forward in the academic life, given the personal growth and development.

7.3 Future work

The *U=RIsolve* framework has an enthusiastic margin for improvements. One of the main goals is to turn the application completely independent of third-party simulation tools, by designing a graphical interface module for drawing and modelling circuits. In addition, it is planned to extend the application to other circuit analysis methods, namely the *Branch Current Method*, the *Mesh/Loop Current Method*, as well as circuit simplification methods e.g. based on the *Superposition*, *Thévenin*, *Norton* and *Millman* Theorems. Apart from these enhancements, a forward-looking strategy concerns the development of a computer vision software module to recognise and model any computer or hand-drawn circuit as a complement to the computer-based circuit modelling required by *U=RIsolve*.

Bibliography

[1] L. Sousa, "U=risolve - aplicativo de análise de circuitos elétricos simulados no qucs." Licenciatura em Engenharia Electrotécnica e de Computadores, Projeto Final, ISEP, 2018. [cited in p. 3]

[2] L. Sousa, M. Alves, and F. Pereira, "U=risolve: teaching & self-learning the nodal analysis method the easy way," in *CASHE - Conference on Academic Success in Higher Education*, Porto School of Engineering (ISEP), 2019. Extended abstract. Best presentation award. Available at: `https://www.isep.ipp.pt/files/CASHE_Talk11.pdf`. [cited in p. 3]

[3] R. Sommer, D. Ammermann, and E. Hennig, "More efficient algorithms for symbolic network analysis: Supernodes and reduced loop analysis," *Analog Integrated Circuits and Signal Processing, Springer*, vol. 3, pp. 73–83, 1993. [cited in p. 4, 20]

[4] L. Sousa, A. Rocha, M. Alves, and F. Pereira, "Revisiting the nodal voltage method for both human comprehension and software implementation: towards a teaching/self-learning simulation tool," *Under submission to an international journal*, 2020. [cited in p. 4]

[5] MuseMaze Inc., "Everycircuit – circuits are better animated," 2020. Retrieved from: `https://everycircuit.com`. [cited in p. 9]

[6] Aspencore, "Partsim – circuit simulation made easy," 2020. Retrieved from: `https://www.partsim.com`. [cited in p. 10, 19]

[7] Analog Devices, "Ltspice simulator," 2020. Retrieved from: `https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html`. [cited in p. 10, 19]

[8] C. Inc., "Circuitlab – circuit simulation and schematics," 2020. Retrieved from: `https://www.circuitlab.com`. [cited in p. 10]

[9] P. Falstad, "Circuit simulator applet," 2020. Retrieved from: `https://www.falstad.com/circuit`. [cited in p. 11]

[10] EasyEDA, "Easyeda – an easier and powerful online pcb design tool," 2020. Retrieved from: `https://easyeda.com`. [cited in p. 11]

[11] National Instruments, "Multisim – discover electronics with online spice simulation," 2020. Retrieved from: `https://www.multisim.com`. [cited in p. 11, 19]

[12] Sourceforge, "Qucs project – quite universal circuit simulator," 2017. Retrieved from: `http://qucs.sourceforge.net`. [cited in p. 12, 19]

[13] System Vision, "System vision cloud – find. design. simulate. collaborate," 2020. Retrieved from: `https://www.systemvision.com`. [cited in p. 12]

[14] A. Huczynski and S. P. Johnston, "Engineering students' use of computer assisted learning (cal)," *European Journal of Engineering Education, Taylor & Francis*, vol. 30, no. 2, pp. 287–298, 2005. [cited in p. 12]

[15] B. Haßler, L. Major, and S. Hennessy, "Tablet use in schools: a critical review of the evidence for learning outcomes," *Journal of Computer Assisted Learning, Wiley*, vol. 32, no. 2, pp. 139–156, 2016. [cited in p. 12]

[16] Wolfram, "Wolfram|alpha – computational intelligence," 2020. Retrieved from: `https://www.wolframalpha.com`. [cited in p. 13]

[17] EqsQuest, "Symbolab math solver – step by step calculator," 2017. Retrieved from: `https://www.symbolab.com`. [cited in p. 13]

[18] Mathway, "Mathway – algebra problem solver," 2020. Retrieved from: `https://www.mathway.com/Calculus`. [cited in p. 14]

[19] Photomath, "Photomath – math superpowers for every student," 2020. Retrieved from: `https://photomath.app`. [cited in p. 14]

[20] K. Academy, "Khan academy – free online courses, lessons & practice," 2020. Retrieved from: `https://www.khanacademy.org`. [cited in p. 14]

[21] Z. Pudlowski, "Developing computer-aided education in electrical engineering," *The International Journal of Electrical Engineering & Education, Sage*, vol. 31, pp. 111–127, 1994. [cited in p. 15]

[22] O. Zavalani, "Computer-based simulation development of a design course project in electrical engineering," *Computer Applications in Engineering Education, Wiley*, vol. 23, pp. 587–595, 2015. [cited in p. 15]

[23] K. Kayisli, S. Tuncer, and M. Poyraz, "An educational tool for fundamental dc-dc converter circuits and active power factor correction applications," *Computer Applications in Engineering Education, Wiley*, vol. 21, pp. 113–134, 2013. [cited in p. 15]

[24] A. Morelato, "A computer tool for helping engineering students in their learning of electrical energy basics," *Education, IEEE Transactions on*, vol. 44, p. 3 pp., 06 2001. [cited in p. 15]

[25] D. Y. Northam, "Introducing computer tools into a first course in electrical engineering," *IEEE Transactions on Education*, vol. 38, no. 1, pp. 13–16, 1995. [cited in p. 15]

[26] S. R. Cvetkovic, R. J. A. Seebold, K. N. Bateson, and V. K. Okretic, "Cal programs developed in advanced programming environments for teaching electrical engineering," *IEEE Transactions on Education*, vol. 37, no. 2, pp. 221–227, 1994. [cited in p. 15]

[27] G. Rubner, "First-year undergraduate teaching of electrical and electronic engineering: innovation and inspiration," *The International Journal of Electrical Engineering & Education, Sage*, vol. 54, pp. 281–282, 2017. [cited in p. 15]

[28] K. Zeashan and M. Abid, "Role of laboratory setup in project-based learning of freshmen electrical engineering," *The International Journal of Electrical Engineering & Education, Sage*, vol. 54, pp. 150–163, 2017. [cited in p. 15]

[29] A. Magana, S. Brophy, and G. Bodner, "Instructors' intended learning outcomes for using computational simulations as learning tools," *Journal of Engineering Education, Wiley*, vol. 101, pp. 220–241, 2012. [cited in p. 15]

[30] O. Campbell, J. Bourne, P. Mosterman, and A. Brodersen, "The effectiveness of learning simulations for electronic laboratories," *Journal of Engineering Education, Wiley*, vol. 91, pp. 81–87, 2002. [cited in p. 15]

[31] K. Vanlehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, pp. 197–221, 2011. [cited in p. 15]

[32] B. Skromme, V. Seetharam, X. Gao, B. Korrapati, B. E. McNamara, Y. F. Huang, and D. H. Robinson, "Impact of step-based tutoring on student learning in linear circuit courses," in *FIE 2016 - Frontiers in Education 2016: The Crossroads of Engineering and Business*, (United States), Institute of Electrical and Electronics Engineers Inc., 2016. [cited in p. 15]

[33] F. de Coulon, E. Forte, and J. M. Rivera, "Kirchhoff: an educational software for learning the basic principles and methodology in electrical circuits modeling," *IEEE Transactions on Education*, vol. 36, no. 1, pp. 19–22, 1993. [cited in p. 15]

[34] L. Palma, R. F. Morrison, P. N. Enjeti, and J. W. Howze, "Use of web-based materials to teach electric circuit theory," *IEEE Transactions on Education*, vol. 48, no. 4, pp. 729–734, 2005. [cited in p. 16]

[35] J. Becker and C. Plumb, "Towards a web-based homework system for promoting success of at-risk students in a basic electric circuits course," *ASEE Annual Conference & Exposition, American Society for Engineering Education*, 2017. [cited in p. 16]

[36] O. Yang, Y. Dong, M. Zhu, H. Yuewei, and M. S. M. Yunjie, "Ecvlab: A web-based virtual laboratory system for electronic circuit simulation," *Lecture Notes in Computer Science, Springer*, vol. 3514, pp. 1027–1034, 2005. [cited in p. 16]

[37] S. Khaddaj and A. Marmar, "Electric circuit interactive laboratory," *The International Journal of Electrical Engineering & Education, Sage*, vol. 53, 10 2015. [cited in p. 16]

[38] A. Moura and A. Moura, "Use of virtual industry and laboratory machines to teach electric circuit theory," *The International Journal of Electrical Engineering & Education, Sage*, vol. 53, 02 2016. [cited in p. 16]

[39] J. Hospodka and M. Koubik, "Special web-based application for electric circuit analysis," in *Proceedings of the 10th WSEAS International Conference on E-Activities*, pp. 140–145, World Scientific and Engineering Academy and Society (WSEAS), 2011. [cited in p. 16]

[40] L. Weyten, P. Rombouts, and J. D. Maeyer, "Web-based trainer for electrical circuit analysis," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 185–189, 2009. [cited in p. 16]

[41] A. Liberatore, A. Luchetta, S. Manetti, and M. C. Piccirilli, "A new symbolic program package for the interactive design of analog circuits," in *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, vol. 3, pp. 2209–2212, 1995. [cited in p. 16]

[42] G. Fedi, R. Giomi, A. Luchetta, S. Manetti, and M. C. Piccirilli, "Sapwin 2.0: a symbolic software tool for educational purposes in analysis and synthesis of analog circuits," in *International Conference on Simulation and Multimedia in Engineering Education ICSEE'99*, pp. 33–36, 1999. [cited in p. 16]

[43] A. Luchetta, S. Manetti, and A. Reatti, "Sapwin - a symbolic simulator as a support in electrical engineering education," *IEEE Transactions on Education*, vol. 44, no. 2, 2001. [cited in p. 16]

[44] F. Grasso, A. Luchetta, S. Manetti, M. C. Piccirilli, and A. Reatti, "Sapwin 4.0–a new simulation program for electrical engineering education using symbolic analysis," *Computer Applications in Engineering Education, Wiley*, vol. 44, pp. 44–57, 2016. [cited in p. 16]

[45] B. J. Skromme, P. J. Rayes, C. D. Whitlatch, Q. Wang, A. Barrus, J. M. Quick, R. K. Atkinson, and T. S. Frank, "Computer-aided instruction for introductory linear circuit analysis," in *2013 IEEE Frontiers in Education Conference (FIE)*, pp. 314–319, 2013. [cited in p. 16]

[46] B. J. Skromme, P. J. Rayes, B. E. McNamara, V. Seetharam, X. Gao, T. Thompson, X. Wang, B. Cheng, Y.-F. Huang, and D. H. Robinson, "Step-based tutoring system for introductory linear circuit analysis," in *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, 2015. [cited in p. 16]

[47] R. DeCarlo and P.-M. Lin, "Nodal and loop analyses," in *Linear Circuits: Time Domain, Phasor and Laplace Transform Approaches*, ch. 3, pp. 107–154, Kendall Hunt, 3rd ed., 2009. [cited in p. 17]

[48] R. Smith and R. Dorf, "Circuit laws," in *Circuits, devices and systems: a first course in Electrical Engineering*, ch. 2.2, pp. 34–47, John Wiley & Sons, 5th ed., 1991. [cited in p. 17]

[49] J. G. Gottling, "Node and mesh analysis by inspection," *IEEE Transactions on Education*, vol. 38, no. 4, pp. 312–316, 1995. [cited in p. 18]

[50] G. E. Chatzarakis and M. D. Tortoreli, "Node-voltage method using 'virtual current sources' technique for special cases," *The International Journal of Electrical Engineering & Education, Sage*, vol. 41, no. 3, pp. 230–243, 2004. [cited in p. 19]

[51] C. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, pp. 505–509, 1975. [cited in p. 19]

[52] K. Lee and S. Park, "Reduced modified nodal approach to circuit analysis," *IEEE Transactions on Circuits and Systems*, vol. 32, no. 10, pp. 1056–1060, 1985. [cited in p. 19]

[53] L. M. Wedepohl and L. Jackson, "Modified nodal analysis: an essential addition to electrical circuit theory and analysis," *Engineering Science and Education Journal*, vol. 11, pp. 84–92, 1992. [cited in p. 19]

[54] M. Shokouhifar and A. Jalali, "Automatic simplified symbolic analysis of analog circuits using modified nodal analysis and genetic algorithm," *Journal of Circuits System and Computers, World Scientific*, vol. 24, pp. 1–20, 2015. [cited in p. 19]

[55] R. Pratap, V. Agarwal, and R. K. Singh, "Review of various available spice simulators," in *2014 International Conference on Power, Control and Embedded Systems (ICPCES)*, pp. 1–6, 2014. [cited in p. 19]

[56] Cadence Design Systems, "Orcad pspice designer," 2020. Retrieved from: `https://www.orcad.com/pt/products/orcad-pspice-designer/overview`. [cited in p. 19]

[57] Cadence Design Systems, "Spectre x simulator," 2020. Retrieved from: `https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-x-simulator.html`. [cited in p. 19]

[58] Synopsys, "Hspice – the gold standard for accurate circuit simulation," 2020. Retrieved from: `https://www.synopsys.com/verification/ams-verification/hspice.html`. [cited in p. 19]

[59] Texas Instruments, "Tina-ti – spice-based analog simulation program," 2016. Retrieved from: `http://www.ti.com/tool/TINA-TI`. [cited in p. 19]

[60] T. Tuma and A. Buermen, "Circuit simulation with spice opus," *Theory and Practice*, 2009. [cited in p. 19]

[61] E. H.-A. Gerbracht, "Supernodal analysis revisited," *Proceedings of the International Workshop on Symbolic Methods and Applications in Circuit Design*, pp. 113–116, 2004. [cited in p. 20]

[62] R. R. Chen, A. M. Davis, and M. Simaan, "Kirchhoff's voltage and current laws," in *The Circuits and Filters Handbook* (W.-K. Chen, ed.), ch. 18.1, pp. 536–573, CRC Press, 2nd ed., 2002. [cited in p. 20]

[63] M. Baser, "Effects of conceptual change and traditional confirmatory simulations on pre-service teachers' understanding of direct current circuits," *Journal of Science Education and Technology*, vol. 15, no. 5, pp. 367–381, 2006. [cited in p. 47]

[64] M. Baser, "Promoting conceptual change through active learning using open source software for physics simulations," *Australasian Journal of Educational Technology*, vol. 22, no. 3, pp. 336–354, 2006. [cited in p. 47]

[65] M. Brinson and V. Kuznetsov, "Qucs-0.0.19s: A new open-source circuit simulator and its application for hardware design," in *2016 International Siberian Conference on Control and Communications (SIBCON)*, pp. 1–5, 2016. [cited in p. 48]

[66] A. Silva, "Design, implementation and integration of new functionalities in the qucs simulator." Licenciatura em Engenharia Electrotécnica e de Computadores, Projeto Final, ISEP, 2017. [cited in p. 48]

[67] N. Martins, "Desenvolvimento de módulos para o qucs." Licenciatura em Engenharia Electrotécnica e de Computadores, Projeto Final, ISEP, 2016. [cited in p. 48]

[68] P. Macedo, "Análise comparativa e melhoramento de simuladores de circuitos elétricos." Licenciatura em Engenharia Electrotécnica e de Computadores, Projeto Final, ISEP, 2015. [cited in p. 48]

[69] L. Kechiev, N. Kruchkov, and V. Kuznetsov, "New active filter synthesis tool for qucs open-source circuit simulator," in *2016 International Siberian Conference on Control and Communications (SIBCON)*, pp. 1–4, 2016. [cited in p. 48]

[70] M. Brinson and V. Kuznetsov, "Improvements in qucs-s equation-defined modelling of semiconductor devices and ic's," in *2017 MIXDES - 24th International Conference "Mixed Design of Integrated Circuits and Systems*, pp. 137–142, 2017. [cited in p. 48]

[71] J. Keown, *PSpice and Circuit Analysis*. Prentice Hall PTR 16, 2nd ed., 1994. Pages 13–65. [cited in p. 48]

[72] J. de Jong and E. Mansfield, "Math.js: An advanced mathematics library for javascript," *Computing in Science Engineering, IEEE*, vol. 20, no. 1, pp. 20–32, 2018. `https://mathjs.org/`. [cited in p. 57]

[73] N. White, "Algebra.js - build, display, and solve algebraic equations in javascript," 2016. Retrieved from: `https://algebra.js.org`. [cited in p. 57]

[74] A. Rocha, "Syseqsolver.js - linear equations system solver," 2020. Retrieved from: `https://github.com/txroot/syseqsolver`. [cited in p. 57]

[75] E. Eisenberg and S. Alpert, "Katex - the fastest math typesetting library for the web," 2020. Retrieved from: `https://katex.org`. [cited in p. 58]

Appendix A

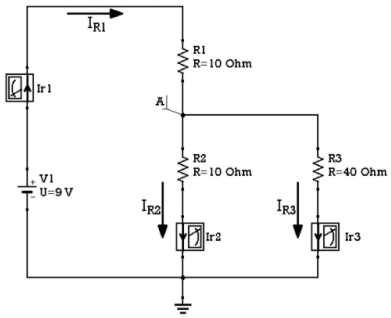
Case-Studies Graphical Outputs

This annex illustrates the *U=RIsolve* output exactly as it is shown to the users, for the examples in Sections 6.1, 6.2, 6.3, 6.4 and 6.5 respectively.

A.1 DC circuit without Supernodes

URIsolve Analysis Results

Circuit Image



Node Voltages

number	A.V
1	4

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I
1	0.5	0.4	0.1

Expand results

Fundamental Variables

Branches	Nodes	Isolated Voltage Sources	Equations
$R = 3$	$N = 2$	$T = 0$	$E = N - 1 - T \Leftrightarrow E = 1$

Circuit Information

Frequency	Current Sources	Ammeters	Simulation
$F = 0\ Hz$	$CS = 0$	$3\ /\ 3$	DC

Branch Currents

ID: $Ir3$
Flow: $A \rightarrow gnd$
Components: $R3$

ID: $Ir2$
Flow: $A \rightarrow gnd$
Components: $R2$

ID: $Ir1$
Flow: $gnd \rightarrow A$
Components: $V1, R1$

Current Equations (Kirchhoff Currents Law - KCL)

Node A

$$Ir1 = Ir3 + Ir2$$

Equation System

$$\left\{\frac{V_A \cdot -50}{400} + \frac{9 - V_A}{10} = 0\right.$$

Show Steps

Step 1: Reorder current equations — View –

$$\left\{Ir1 - Ir3 - Ir2 = 0\right.$$

Step 2: Substitute the known currents — View –

$$\left\{Ir1 - Ir3 - Ir2 = 0\right.$$

Step 3: Compute the remaining currents using Ohm's Law — View –

$$\begin{cases} Ir1 = \frac{(V1 - V_A)}{R1} \\ Ir3 = \frac{V_A}{R3} \\ Ir2 = \frac{V_A}{R2} \end{cases}$$

Step 4: Substitute each current by its equation — View –

$$\left\{\frac{(V1 - V_A)}{R1} - \frac{V_A}{R3} - \frac{V_A}{R2} = 0\right.$$

Step 5: Replace the constants with their value — View –

$$\left\{\frac{V_A \cdot -50}{400} + \frac{9 - V_A}{10} = 0\right.$$

Results

Node Voltages

$\{V_A = 4\ V$

Branch Currents

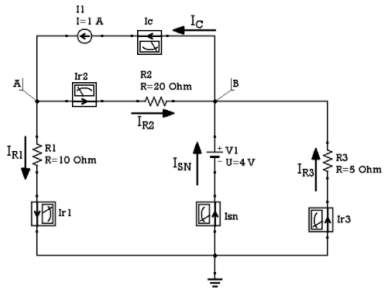
Note: the following currents were obtained by their Ohm's Law equation

$$\begin{cases} Ir3 = \frac{V_A}{40} \\ Ir2 = \frac{V_A}{10} \\ Ir1 = \frac{9-V_A}{10} \end{cases} \Leftrightarrow \begin{cases} Ir3 = 0.1\ A \\ Ir2 = 0.4\ A \\ Ir1 = 0.5\ A \end{cases}$$

A.2 DC circuit with a Grounded Supernode

URIsolve Analysis Results

Circuit Image



Node Voltages

number	A.V	B.V
1	8	4

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Isn.I	Ic.I
1	0.8	0.2	-0.8	1.6	1

Expand results

Fundamental Variables

Branches	Nodes	Isolated Voltage Sources	Equations
$R = 5$	$N = 3$	$T = 1$	$E = N - 1 - T \Leftrightarrow E = 1$

Circuit Information

Frequency	Current Sources	Ammeters	Simulation
$F = 0\ Hz$	$CS = 1$	$5\ /\ 5$	DC

Supernodes

SNg1 (Grounded) Show Steps

Formed by Nodes: {gnd, B}

Equations: $\{V_B = 4\,V$

$\{V_B = 0 + V1$

Branch Currents

ID: $Ir2$
Flow: $A \rightarrow B$
Components: $R2$

ID: $Ir1$
Flow: $A \rightarrow gnd$
Components: $R1$

ID: Ic
Flow: $B \rightarrow A$
Components: $I1$

ID: $Ir3$
Flow: $gnd \rightarrow B$
Components: $R3$

ID: Isn
Flow: $gnd \rightarrow B$
Components: $V1$

Current Equations (Kirchhoff Currents Law - KCL)

Node A

$Ic = Ir2 + Ir1$

Equation System

$\{\frac{(V_A - 4)\cdot -1}{20} + \frac{V_A \cdot -1}{10} + 1 = 0$

Show Steps

Step 1: Reorder current equations View –

$\{Ic - Ir2 - Ir1 = 0$

Step 2: Substitute the known currents View –

$\{1 - Ir2 - Ir1 = 0$

Step 3: Compute the remaining currents using Ohm's Law — View

$$\begin{cases} Ir2 = \frac{(V_A - V_B)}{R2} \\ Ir1 = \frac{V_A}{R1} \end{cases}$$

Step 4: Substitute each current by its equation — View

$$\left\{1 - \frac{(V_A - V_B)}{R2} - \frac{V_A}{R1} = 0\right.$$

Step 5: Replace the constants with their value — View

$$\left\{\frac{(V_A - V_B)\cdot -1}{20} + \frac{V_A \cdot -1}{10} + 1 = 0\right.$$

Results

Node Voltages

$$\begin{cases} V_B = 4\ V \\ V_A = 8\ V \end{cases}$$

Branch Currents

Note: the following currents were obtained by an existing current source in their branch

$$\{Ic = 1\ A$$

Note: the following currents were obtained by their Ohm's Law equation

$$\begin{cases} Ir2 = \frac{V_A - V_B}{20} \\ Ir1 = \frac{V_A}{10} \\ Ir3 = \frac{V_B \cdot -1}{5} \end{cases} \Leftrightarrow \begin{cases} Ir2 = 0.2\ A \\ Ir1 = 0.8\ A \\ Ir3 = -0.8\ A \end{cases}$$

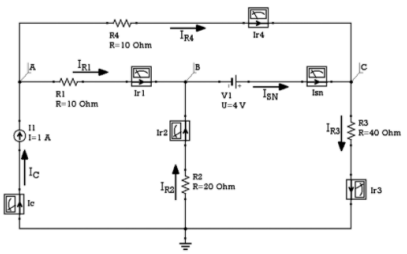
Note: the following currents were obtained by their KCL equation, since they belong to branches with isolated voltage sources

$$\{Isn = Ic - Ir2 - Ir3 \Leftrightarrow \{Isn = 1.6\ A$$

A.3 DC circuit with a Floating Supernode

URIsolve Analysis Results

Circuit Image



Node Voltages

number	A.V	B.V	C.V
1	19	12	16

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Ir4.I	Isn.I	Ic.I
1	0.7	-0.6	0.4	0.3	0.1	1

Expand results

Fundamental Variables

Branches	Nodes	Isolated Voltage Sources	Equations
$R = 6$	$N = 4$	$T = 1$	$E = N - 1 - T \Leftrightarrow E = 2$

Circuit Information

Frequency	Current Sources	Ammeters	Simulation
$F = 0\ Hz$	$CS = 1$	$6\ /\ 6$	DC

Supernodes

SNf1 (Floating) Show Steps

Formed by Nodes: {C, B}

Equations: $\{V_B = \mathrm{V_C} - 4$

$\{V_B = \mathrm{V_C} - V1$

Branch Currents

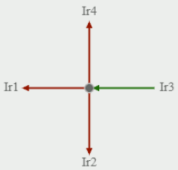
ID: $Ir1$ Flow: $A \rightarrow B$ Components: $R1$	ID: $Ir4$ Flow: $A \rightarrow C$ Components: $R4$	ID: Ic Flow: $gnd \rightarrow A$ Components: $I1$
ID: $Ir2$ Flow: $gnd \rightarrow B$ Components: $R2$	ID: Isn Flow: $B \rightarrow C$ Components: $V1$	ID: $Ir3$ Flow: $C \rightarrow gnd$ Components: $R3$

Current Equations (Kirchhoff Currents Law - KCL)

Node A

$Ic = Ir1 + Ir4$

Node SNf1



$Ir3 = Ir4 + Ir1 + Ir2$

Equation System

$$\begin{cases} \frac{(2\cdot V_A - 2\cdot V_C + 4)\cdot -1}{10} + 1 = 0 \\ \frac{(2\cdot V_A - 2\cdot V_C + 4)\cdot -1}{10} + \frac{V_C}{40} + \frac{V_C - 4}{20} = 0 \end{cases}$$

Show Steps

Step 1: Reorder current equations — View –

$$\begin{cases} Ic - Ir1 - Ir4 = 0 \\ Ir3 - Ir4 - Ir1 - Ir2 = 0 \end{cases}$$

Step 2: Substitute the known currents — View –

$$\begin{cases} 1 - Ir1 - Ir4 = 0 \\ Ir3 - Ir4 - Ir1 - Ir2 = 0 \end{cases}$$

Step 3: Compute the remaining currents using Ohm's Law — View –

$$\begin{cases} Ir1 = \frac{(V_A - V_B)}{R1} \\ Ir4 = \frac{(V_A - V_C)}{R4} \\ Ir3 = \frac{V_C}{R3} \\ Ir2 = -\left(\frac{V_B}{R2}\right) \end{cases}$$

Step 4: Substitute each current by its equation — View –

$$\begin{cases} 1 - \frac{(V_A - V_B)}{R1} - \frac{(V_A - V_C)}{R4} = 0 \\ \frac{V_C}{R3} - \frac{(V_A - V_C)}{R4} - \frac{(V_A - V_B)}{R1} + \left(\frac{V_B}{R2}\right) = 0 \end{cases}$$

Step 5: Replace the constants with their value — View –

$$\begin{cases} \frac{(2\cdot V_A - V_B - V_C)\cdot -1}{10} + 1 = 0 \\ \frac{(2\cdot V_A - V_C - V_B)\cdot -1}{10} + \frac{V_C}{40} + \frac{V_B}{20} = 0 \end{cases}$$

Step 6: Set a reference for each floating supernode — View –

In supernode **SNf1** the node **C** was chosen as a reference.

Notes:

- The voltage of each node from a floating supernode must be expressed as a function of the reference node.
- In the Supernodes section, you can confirm that node equations are already referenced to the chosen node.
- Use these expressions to perform the substitution in the equation system.

Results

Node Voltages

$$\begin{cases} V_A = 19\ V \\ V_C = 16\ V \\ V_B = 12\ V \end{cases}$$

Branch Currents

Note: the following currents were obtained by an existing current source in their branch

$$\{Ic = 1\ A$$

Note: the following currents were obtained by their Ohm's Law equation

$$\begin{cases} Ir1 = \frac{V_A - V_B}{10} \\ Ir4 = \frac{V_A - V_C}{10} \\ Ir2 = \frac{V_B - -1}{20} \\ Ir3 = \frac{V_C}{40} \end{cases} \Leftrightarrow \begin{cases} Ir1 = 0.7\ A \\ Ir4 = 0.3\ A \\ Ir2 = -0.6\ A \\ Ir3 = 0.4\ A \end{cases}$$

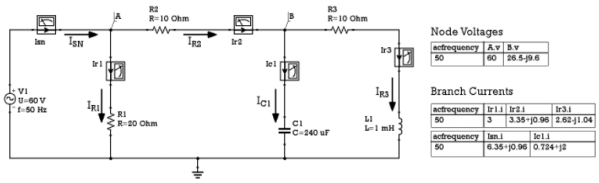
Note: the following currents were obtained by their KCL equation, since they belong to branches with isolated voltage sources

$$\{Isn = Ir1 + Ir2 \Leftrightarrow \{Isn = 0.1\ A$$

A.4 AC circuit with a Grounded Supernode

URIsolve Analysis Results

Circuit Image



Expand results

Fundamental Variables

Branches	Nodes	Isolated Voltage Sources	Equations
$R = 5$	$N = 3$	$T = 1$	$E = N - 1 - T \Leftrightarrow E = 1$

Circuit Information

Frequency	Current Sources	Ammeters	Simulation
$F = 50\ Hz$	$CS = 0$	$5\ /\ 5$	AC

Supernodes

SNg1 (Grounded) Show Steps

Formed by Nodes: {gnd, A}

Equations: $\{V_A = 60\,V$

$\{V_A = 0 + V1$

Branch Currents

ID: $Ir1$
Flow: $A \rightarrow gnd$
Components: $R1$

ID: $Ir2$
Flow: $A \rightarrow B$
Components: $R2$

ID: Isn
Flow: $gnd \rightarrow A$
Components: $V1$

ID: $Ir3$
Flow: $B \rightarrow gnd$
Components: $L1, R3$

ID: $Ic1$
Flow: $B \rightarrow gnd$
Components: $C1$

Equivalent impedances and voltages

Branch from B to gnd

$\{ZeqBgnd = L1 + R3 = 10 + 0.314i$

Current Equations (Kirchhoff Currents Law - KCL)

Node **B**

$Ir2 = Ir3 + Ic1$

Equation System

$$\left\{\frac{60-\mathrm{V_B}}{10}-\left(\frac{1}{\frac{i\cdot 157}{500}+10}+\frac{1}{\frac{i\cdot -6.631\cdot 10^{+5}}{50000}}\right)\cdot \mathrm{V_B}=0\right.$$

Show Steps

Step 1: Reorder current equations — View

$$\left\{Ir2-Ir3-Ic1=0\right.$$

Step 2: Substitute the known currents — View

$$\left\{Ir2-Ir3-Ic1=0\right.$$

Step 3: Compute the remaining currents using Ohm's Law — View

$$\begin{cases} Ir2=\frac{(\mathrm{V_A}-\mathrm{V_B})}{R2} \\ Ir3=\frac{\mathrm{V_B}}{(XL1+R3)} \\ Ic1=\frac{\mathrm{V_B}}{XC1} \end{cases}$$

Step 4: Substitute each current by its equation — View

$$\left\{\frac{(\mathrm{V_A}-\mathrm{V_B})}{R2}-\frac{\mathrm{V_B}}{(XL1+R3)}-\frac{\mathrm{V_B}}{XC1}=0\right.$$

Step 5: Replace the constants with their value — View

$$\left\{\frac{\mathrm{V_A}-\mathrm{V_B}}{10}-\left(\frac{1}{\frac{i\cdot 157}{500}+10}+\frac{1}{\frac{i\cdot -6.631\cdot 10^{+5}}{50000}}\right)\cdot \mathrm{V_B}=0\right.$$

Results

Node Voltages

$$\begin{cases} V_A=60\ V \\ V_B=28.227\angle -19.874^\circ\ V \end{cases}$$

Branch Currents

Note: the following currents were obtained by their Ohm's Law equation

$$\begin{cases} Ir1=\frac{\mathrm{V_A}}{20} \\ Ir2=\frac{\mathrm{V_A}-\mathrm{V_B}}{10} \\ Ir3=\frac{\mathrm{V_B}}{\frac{i\cdot 3.141\cdot 10^{+5}}{1\cdot 10^{+6}}+10} \\ Ic1=\frac{\mathrm{V_B}}{\frac{i\cdot -2.0723\cdot 10^{+7}}{1.562\cdot 10^{+6}}} \end{cases} \Leftrightarrow \begin{cases} Ir1=3\ A \\ Ir2=3.48\angle 16.013^\circ\ A \\ Ir3=2.821\angle -21.673^\circ\ A \\ Ic1=2.129\angle 70.118^\circ\ A \end{cases}$$

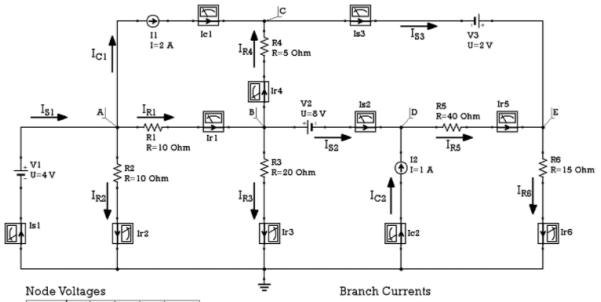
Note: the following currents were obtained by their KCL equation, since they belong to branches with isolated voltage sources

$$\left\{Isn=Ir1+Ir2 \Leftrightarrow \left\{Isn=6.417\angle 8.604^\circ\ A\right.\right.$$

A.5 DC circuit with multiple Supernodes

URIsolve Analysis Results

Circuit Image



Node Voltages

number	A.V	B.V	C.V	D.V	E.V
1	4	14.4	16.6	6.38	18.6

Branch Currents

number	Ir1.I	Ir2.I	Ir3.I	Ir4.I	Ir5.I	Ir6.I
1	-1.04	0.4	0.719	-0.451	-0.306	1.24

number	Ic1.I	Ic2.I	Is1.I	Is2.I	Is3.I
1	2	1	1.36	-1.31	1.55

Expand results

Fundamental Variables

Branches	Nodes	Isolated Voltage Sources	Equations
$R = 11$	$N = 6$	$T = 3$	$E = N - 1 - T \Leftrightarrow E = 2$

Circuit Information

Frequency	Current Sources	Ammeters	Simulation
$F = 0\ Hz$	$CS = 2$	$11\ /\ 11$	DC

Supernodes

SNg1 (Grounded) Show Steps

- Formed by Nodes: {gnd, A}
- Equations: $\{V_A = 4\,V$

✔ $\{V_A = 0 + V1$

SNf1 (Floating) Show Steps

- Formed by Nodes: {B, D}
- Equations: $\{V_D = \mathrm{V_B} - 8$

✔ $\{V_D = \mathrm{V_B} - V2$

SNf2 (Floating) Show Steps

- Formed by Nodes: {E, C}
- Equations: $\{V_C = V_E - 2$

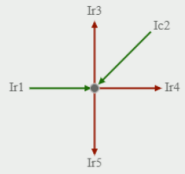
✔ $\{V_C = V_E - V3$

Branch Currents

ID: $Ir1$ Flow: $A \rightarrow B$ Components: $R1$	ID: $Ir2$ Flow: $A \rightarrow gnd$ Components: $R2$	ID: $Is1$ Flow: $gnd \rightarrow A$ Components: $V1$
ID: $Ic1$ Flow: $A \rightarrow C$ Components: $I1$	ID: $Ir4$ Flow: $B \rightarrow C$ Components: $R4$	ID: $Ir3$ Flow: $B \rightarrow gnd$ Components: $R3$
ID: $Is2$ Flow: $B \rightarrow D$ Components: $V2$	ID: $Is3$ Flow: $C \rightarrow E$ Components: $V3$	ID: $Ir5$ Flow: $D \rightarrow E$ Components: $R5$
ID: $Ic2$ Flow: $gnd \rightarrow D$ Components: $I2$	ID: $Ir6$ Flow: $E \rightarrow gnd$ Components: $R6$	

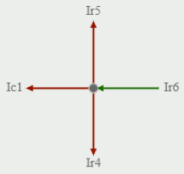
Current Equations (Kirchhoff Currents Law - KCL)

Node SNf1



$$Ir1 + Ic2 = Ir4 + Ir3 + Ir5$$

Node SNf2



$$Ir6 = Ir5 + Ic1 + Ir4$$

Equation System

$$\begin{cases} \frac{V_E - 2 - V_B}{5} + \frac{V_B \cdot -1}{20} + \frac{4 - V_B}{10} + \frac{(V_B - 8 - V_E) \cdot -1}{40} + 1 = 0 \\ \frac{V_E}{15} - 2 + \frac{(V_B - 8 - V_E) \cdot -1}{40} + \frac{(V_B - V_E + 2) \cdot -1}{5} = 0 \end{cases}$$

Show Steps

Step 1: Reorder current equations — View

$$\begin{cases} -Ir4 - Ir3 + Ir1 - Ir5 + Ic2 = 0 \\ Ir6 - Ir5 - Ic1 - Ir4 = 0 \end{cases}$$

Step 2: Substitute the known currents — View

$$\begin{cases} -Ir4 - Ir3 + Ir1 - Ir5 + 1 = 0 \\ Ir6 - Ir5 - 2 - Ir4 = 0 \end{cases}$$

Step 3: Compute the remaining currents using Ohm's Law — View

$$\begin{cases} Ir4 = \frac{(V_B - V_C)}{R4} \\ Ir3 = \frac{V_B}{R3} \\ Ir1 = \frac{(V_A - V_B)}{R1} \\ Ir5 = \frac{(V_D - V_E)}{R5} \\ Ir6 = \frac{V_E}{R6} \end{cases}$$

Step 4: Substitute each current by its equation — View

$$\begin{cases} \frac{-(V_B - V_C)}{R4} - \frac{V_B}{R3} + \frac{(V_A - V_B)}{R1} - \frac{(V_D - V_E)}{R5} + 1 = 0 \\ \frac{V_E}{R6} - \frac{(V_D - V_E)}{R5} - 2 - \frac{(V_B - V_C)}{R4} = 0 \end{cases}$$

Step 5: Replace the constants with their value — View

$$\begin{cases} \frac{V_C - V_B}{5} + \frac{V_B \cdot -1}{20} + \frac{V_A - V_B}{10} + \frac{(V_D - V_E) \cdot -1}{40} + 1 = 0 \\ \frac{V_E}{15} - 2 + \frac{(V_D - V_E) \cdot -1}{40} + \frac{(V_B - V_C) \cdot -1}{5} = 0 \end{cases}$$

Step 6: Set a reference for each floating supernode — View

- In supernode **SNf1** the node **B** was chosen as a reference.
- In supernode **SNf2** the node **E** was chosen as a reference.

Notes:

- The voltage of each node from a floating supernode must be expressed as a function of the reference node.
- In the Supernodes section, you can confirm that node equations are already referenced to the chosen node.
- Use these expressions to perform the substitution in the equation system.

Results

Node Voltages

$$\begin{cases} V_A = 4\ V \\ V_B = 14.383\ V \\ V_E = 18.638\ V \\ V_D = 6.383\ V \\ V_C = 16.638\ V \end{cases}$$

Branch Currents

Note: the following currents were obtained by an existing current source in their branch

$$\begin{cases} Ic1 = 2\ A \\ Ic2 = 1\ A \end{cases}$$

Note: the following currents were obtained by their Ohm's Law equation

$$\begin{cases} Ir1 = \frac{V_A - V_B}{10} \\ Ir2 = \frac{V_A}{10} \\ Ir4 = \frac{V_B - V_C}{5} \\ Ir3 = \frac{V_B}{20} \\ Ir5 = \frac{V_D - V_E}{40} \\ Ir6 = \frac{V_E}{15} \end{cases} \Leftrightarrow \begin{cases} Ir1 = -1.038\ A \\ Ir2 = 0.4\ A \\ Ir4 = -0.451\ A \\ Ir3 = 0.719\ A \\ Ir5 = -0.306\ A \\ Ir6 = 1.243\ A \end{cases}$$

Note: the following currents were obtained by their KCL equation, since they belong to branches with isolated voltage sources

$$\begin{cases} Is1 = Ir1 + Ir2 + Ic1 \\ Is2 = -Ir4 - Ir3 + Ir1 \\ Is3 = Ic1 + Ir4 \end{cases} \Leftrightarrow \begin{cases} Is1 = 1.362\ A \\ Is2 = -1.306\ A \\ Is3 = 1.549\ A \end{cases}$$

Appendix B

U=RIsolve research article (under submission)

Revisiting the Nodal Voltage Method for both human comprehension and software implementation: towards a teaching/self-learning simulation tool

Lino Sousa, André Rocha, Mário Alves, Francisco Pereira
Politécnico do Porto (ISEP)
{1140355,anr,mjf,fdp}@isep.ipp.pt

***Abstract*—This paper outlines an application — dubbed *U=RIsolve* (read as "you resolve") — for helping students to learn the Nodal (or Node) Voltage Analysis Method (NVM, for short) for electrical circuit analysis. The NVM enables to determine the voltage in every Node in the circuit, related to a reference (Ground) Node. In turn, this enables to compute the Current in every Branch (and therefore, the voltage across every component). The *U=RIsolve* application goes far beyond the capabilities of traditional circuit simulators; as it outputs the fundamental circuit information, the methodology, the equations and the results related to the NVM for a given circuit previously simulated in QUCS (*Quite Universal Circuit Simulator*). QUCS can generate a text file (called *netlist*) which identifies all the circuit components/connections and that serves as an input to the *U=RIsolve* application. Additionally, users can also upload the circuit schematics to facilitate the analysis of the U=RIsolve output. Then, the user just needs to click a button to get all the reasoning of the NVM analysis. From a preliminary feedback we received from a group of students/professors that volunteered to help us testing, debugging and optimising this tool, we feel enthusiastic about the way it may improve teaching and self-learning of electrical circuit analysis.**

***Keywords*—Nodal analysis, Node-Voltage Method, electrical circuits, self-learning, QUCS, circuit simulator, Supernode analysis, JavaScript, web-application.**

I. Introduction

Teaching DC and AC electrical circuit analysis is a challenging task. Along over 20 years teaching experience in electrical engineering, we constantly have to deal with students' theoretical and practical difficulties in all analysis methods, namely the "baseline" Branch Currents Method (based on the Kirchhoff Current and Voltage laws — KCL/KVL) the Superposition Theorem Method, the Mesh-Current Method (MCM, aka "Loop-Current Method") and the Node-Voltage Method (NVM). The latter two methods are particularly appealing for analysing more complex circuits with more Branches and fewer Nodes (NVM) and more Nodes and fewer Branches (MCM), since they lead to a reduced number of equations/steps (comparing to the "Branch Current Method").

Nevertheless, when applying the MCM/NVM methods, beginners often fail in the quantification and selection of the right Loops (for the Mesh-Current method) and Nodes (for the Node-Voltage method). Even when this selection is correctly performed, there is another error-prone step: writing the corresponding equations — KVL based on the Loop Currents, for the MCM, and KCL for the selected Nodes, for the NVM. These weaknesses become even more acute when the circuits (DC or AC) under analysis include Current Sources — for the MCM — or isolated Voltage Sources ("isolated" in a Branch) — for the NVM, although in either case the equation systems get smaller, since the Current in a branch with a Current Source and the Voltage between Nodes connected to an isolated voltage source are known *a priori*.

In order to better consolidate the students' learning process, we have been supporting the DC/AC circuit analysis teaching/learning process in a way that theoretical models can be instantiated and validated in practice (through practical experiments in lab classes) and also through simulation. We motivate students to use circuit simulators as a self-learning tool and also to prepare their lab classes (*a priori*), by performing the theoretical (analytical) and simulation analysis of the circuits they are going to implement/experiment in the lab. We have also been demonstrating circuit analysis through simulation in Theoretical classes, together with traditional expositive/slides-based teaching.

Concerning our educational context and the current landscape of simulation toolsets and their technical characteristics, we have been favoring the use of the QUCS simulator [1]–[4]. QUCS is easy-to-use, multi-platform, freeware/open-source and features basic functionality for teaching/learning DC/AC circuit analysis. The QUCS project has been quite active, involving a large community of contributors and averaging over 10000 downloads per month [5]. In this context, we highlight add-ons to QUCS that have been implemented in previous works by the authors [6]–[8] and other contributors [9], [10].

Nonetheless, there is a fundamental and open problem. While simulators are paramount tools for testing, designing and fine-tuning electric/electronic circuits, they purely output "measurements", e.g. Branch Current and Node Voltage values in DC/AC circuit analysis. In fact, simulators do not show the way to get to those values, i.e., how to apply the analysis methodology, which the student is supposed to master at the end of the course. Bottom line, with traditional simulators the student is only able to compare results (analytical/experimental against simulation).

To overcome this gap, we have designed an application — dubbed *U=RIsolve* (read "you resolve") — which aims at showing how to apply the circuit analysis methods, specifically

the NVM in this paper. The user must previously implement and simulate the circuit in QUCS, explicitly identifying all Nodes with letters (except the *Ground* Node, which has a specific symbol). The *U=RIsolve* application then takes (as input) the "*netlist*", a text file output by QUCS and that unequivocally represents the circuit under analysis, and generates all steps/equations of the Node-Voltage analysis method.

To our best knowledge, no such type of application has been developed to date, as it enables:

1) The student to learn how to apply the NVM to any DC/AC circuit;
2) The student to compare the output of the *U=RIsolve* application to his/her own methodology and the Branch Currents/Nodes Voltage computed in QUCS with the analytical ones (by solving the equation system that is generated by the NVM);
3) The educator to produce practical examples, both off-class (e.g. to prepare exercises, lab scripts or lectures) and in-class (in real-time, demonstrating the application of the NVM through case-studies).

A preliminary version of the *U=RIsolve* application [11], [12] has been conceived at an earlier stage, based on a baseline NVM algorithm that was unable to tackle more complex circuits with multiple isolated Voltage Sources. That "alpha" version had a rudimentary user interface, produced a less complete analysis/output and it required installation in a PC. The main contributions of the work reported in this paper are two-fold:

1) We propose a new algorithmic approach to the NVM, dubbed NVM-SA, building on the Supernode paradigm [13] that simultaneously guarantees: a) robustness and generality, since it copes with (passive and linear) electrical circuits of any type/complexity, DC or AC; b) step-by-step execution logic, comprehensive to human intuition, building on the fundamental laws of electricity (Ohm Law, Kirchhoff Current Law); c) straightforward coding (in any programming language).
2) We provide a web-based application - dubbed *U=RIsolve* - that instantiates this algorithm, enabling to analyse any circuit (that has been previously simulated in QUCS) with a structured step-by-step output that mimics the proposed NVM algorithm and thus is very complete and intuitive to the user.

As a side contribution, we have designed a JavaScript library for solving linear equation systems suporting real and complex variables, turning it available to the community as open-source [14].

The remainder of the paper is organised as follows. Section II outlines related work. Section III revisits the Node-Voltage Analysis Method, while Chapter IV elaborates on the proposed NVM-SA algorithm, which is used by the *U=RIsolve* application. In Section V we describe the *U=RIsolve* application design and architecture and Section VI illustrates its graphical user interface and step-by-step output. Finally, we summarise the problems encountered and lessons learned as well as some conclusions and future work, in Section VII.

II. Related Work

Studies on improving quality and effectiveness in teaching Electrical Engineering, mainly addressed the creation of new computer-based tools [15]–[21] and the enhancement of existing ones [22]–[25]. We have opted (for the moment) for the latter approach, i.e. we use QUCS to implement/simulate the circuit under analysis and to generate the input to *U=RIsolve*.

Statistical studies [26] have shown that *step-based tutoring* systems had a Cohen *d*-value of 0.76 standard deviations (σ), which has a roughly accurate approximation to the 0.79σ average outcome of expert human tutors. On the contrary, *answer-based tutoring* systems obtained average *d*-values of 0.31σ. Furthermore, another study concerning the impact of a *step-based tutoring* system for linear circuit analysis [27] revealed that this step-based trainer had learning improvements of 0.72σ in relation to the conventional publisher-based homework system, for Node Voltage analysis. Hence, *U=RIsolve* shows users a complete and straightforward step-by-step solution for any DC/AC circuit with several hints on how to correctly apply the Node Voltage Method.

Among the earliest educational tools for learning electrical circuits was KIRCHHOFF [28], a computed aided learning software package for symbolic analysis of electrical circuits. This tool allowed students to select a circuit from the library, set its parameters such as nodes and elements labels, branch currents and voltages. The student would then introduce the Kirchhoff laws in order to obtain the circuit model for nodal or loop methods. Ultimately, KIRCHHOFF software would validate the equations and output an appropriate corrective explanation in case the equations were not properly formulated. Although this software package brought substantial value to the Computer Aided Learning, it's architecture limits its scalability, i.e. students cannot draw their preferred problems, instead they are given a set of predefined circuit examples from an intern library. Besides, the software was developed specifically for IBM-PC compatible microcomputers which is nowadays out of the technological context.

With the increasing interest in online education, more web-based applications like *U=RIsolve* started to be developed, as they bring many advantages to undergraduate students, who have real-time access to problems and solutions, which dramatically improves distance learning. Several works have produced web-based applications to promote success in electrical circuit analysis, providing students with online courses [29], remote homework systems [30] and virtual laboratories for circuit simulation [31]–[33]. One example is the application described in [34], where authors designed an interactive textbook of examples (continuous-time linear circuits) that are submitted by administrators (tutors) along with analysis details such as circuit image, *netlist*, problem statement and component values. Solutions are then analysed by users in a symbolic and semi-symbolic form. Another web-based system

is presented in [35], which also uses symbolic analysis techniques to teach a set of theorems such as Thévenin's theorem and voltage divider. Every circuit data is stored in a problem-specific database and the students statements are registered so that the instructors can follow the students progress.

However, most of these works have a limited range of circuits and simply vary elements values, Branch Currents and Nodes labels. Once the student solves the available set of exercises, and still cannot fully understand the methods, this type of application falls short. Moreover, a finite dataset of examples may not comprehend every aspect of circuit theory that generates a wide range of difficulties in students, which could prevent their progress. There are two different ways of overcoming these problems. The first is to allow an user interface (internal or external to the application) to draw circuit schematics (as *U=RIsolve* does through QUCS), or automatically generate random problems with different layouts. For instance, *SapWin* [36]–[39] uses the first approach, it provides an interface for users to draw circuit schematics, performs a symbolic analysis through Laplace domain and returns the circuit's transfer function. The symbolic method used to solve electronic circuits has the same goal as the *U=RIsolve* application: enhance the learning process. Although the ideals underlying both applications (*U=RIsolve* and *SapWin*) are very similar, *U=RIsolve* focuses on teaching methods to accomplish a correct circuit analysis, while *SapWin* intends to introduce a new system of linear analog circuit analysis by studying its transfer function. On the other hand, authors from [40], [41] designed a step-based computer-aided tutoring system to teach linear circuit analysis where the first approach is used, i.e. it generates a new circuit diagram after showing students the correct answers and solutions to the previous one, so that users have access to unlimited examples. These works address the series-parallel simplifications and the Nodal and Mesh analysis.

III. The NVM algorithms

A. The NVM algorithm – classical approach

The Node-Voltage Method [42], [43] (or just Nodal Analysis) is an electrical circuit analysis method that follows a simple set of rules in order to determine the electric potential in every Node in the circuit. It is based on Khirchhoff's Current Law (KCL), i.e. the algebraic sum of all Currents flowing to one Node must be equal to the sum of all Currents leaving the same Node. In this method, the variables are the Node Voltages towards a reference *Ground* Node, which potential is 0 V. The "baseline" NVM algorithm is as follows:

1) Identify all the Nodes in the circuit.
2) Choose a reference Node, where the voltage will be zero.
3) Identify the Currents in all Branches ($I_1, \ldots, I_R$) and their flowing direction, being R the number of Branches in the circuit.
4) Write the equations for each current, function of the Node Voltages ($I_i = f(U_1, \ldots, U_N), i \in \{1, R\}$), using Ohm's Law and taking into account the chosen directions for Currents.
5) Write the KCL for every Node ($U_1, \ldots, U_N$) based on the Branch Currents ($I_1, \ldots, I_R$).
6) Solve the equation system: $N-1$ equations and $N-1$ variables.
7) Determine each branch's currents ($I_1, \ldots, I_R$) by the Node voltages ($U_1, \ldots, U_N$) obtained in Step 6, through the equations from Step 4.

Alternatively, NVM can also be applied by direct matrix assignment, where the Node voltage equations are written in the form:

$$Gv = i \tag{1}$$

Where G is the conductances' matrix, v holds the unknown node voltages and i matrix are the currents flowing in and out of the Node. This method is outlined in Appendix A.

B. The Modified Nodal Analysis (MNA) and the Supernode Analysis (SA)

The traditional analysis methods (outlined in subsection A) do not cover every electrical circuit, since they are susceptible to a singularity caused by infinite conductances. This situation occurs whenever there are isolated Voltage Sources, also known as floating Sources, that cannot be directly or indirectly (through other Voltage Sources connected to it) referenced to the Ground Node.To overcome this problem, a method named *Modified Nodal Analysis* (MNA) was proposed in [44], and then consolidated in [45], [46]. The MNA is a universal method that can handle any type of electrical components and linear circuits. A brief description of this method is provided in Appendix B.

Importantly, the MNA method has become a commodity for electrical circuit simulators [47]. To our best knowledge, MNA has been firstly used in PSpice and since then by most Spice-like simulators [48]–[54]. Despite its versatility, MNA has not been well accepted in the teaching of circuit analysis, since it is more a mathematical artefact, not intuitively reflecting the logic of electric circuits and its fundamental laws.

Hence, an alternative to MNA — named Supernodal Analysis — was proposed in [13] and later revisited in [55], [56]. SA is a straightforward algorithm, that intends to turn Nodal Analysis accessible to undergraduate students. This method relies on the concept of a *Supernode* (detailed in Section IV), which is a sub-circuit formed by one or more isolated Voltage Sources [1]. In this scenario, if a Voltage Source (E) between two different Nodes (A) and (B) is considered an electric potential difference, then the relation between potentials U_A and U_B is automatically known. Considering Nodes A and B as a single Supernode AB would reduce the degree of freedom of the equation system, but in the other hand it would introduce the relation between Nodes A and B (the isolated Voltage Source). However, if one of these Nodes' potential is known

[1]In this paper context, an isolated Voltage Source is assumed as an ideal Voltage Source, connected in a Branch without any other components, fixing the Voltage between the two nodes delimiting that Branch.

or Grounded, the other would be immediately known by the relation between the isolated Voltage Source (E).

Further work have been issued in the past years in order to generalise the Nodal-Voltage Method, tackling the problem of the isolated Voltage Sources. In [57], it is proposed a method to write the nodal analysis matrix equations for linear circuits by deriving a general matrix solution for the Node-Voltage vector. Similarly, authors from [58] also overcomes the limitations of the traditional NVM, by introducing the concept of "virtual current sources" that replace non-convertible (isolated) Voltage Sources and correspond the Current flowing through that particular Branch. Notwithstanding the fact that this is a more effective algorithm in the pedagogical point of view over the MNA, we believe the logic and insight of the SA still stand out.

Although the SA approach has not yet been implemented in any circuit analysis software (to our best knowledge), we opted to do so. This way we can take benefit from its advantages (robustness, generality and execution logic), providing users with an intuitive step-based output.

IV. The proposed NVM-SA approach

In this section we propose and describe the NVM-SA algorithm, which copes with any (linear and passive) electrical circuit, DC or AC, independently of its size and complexity, and that is easily implementable in software and intuitive to human understanding, as it enables to provide a step-by-step output of the circuit analysis.

A. Concepts and terminology

The following concepts and terminology are paramount to describe the NVM-SA algorithm.

- **(electrical) Circuit**: set of components and interconnecting elements that are physically (electrically) connected to compose a closed electrical circuit;
- **(electrical) Component**: (electrical) power source or load
  - Examples: Voltage Source, Current Source, Resistor, Capacitor, Inductor, Lamp, Switch, measuring instruments, Impedance (Resistance, Inductance, Capacitance, pure or combined);
- **Interconnecting Element**: electrical element that interconnects two components or interconnecting elements
  - It may interconnect two components, one component and one interconnecting element or two interconnecting elements;
  - Examples: wire/cable, printed circuit boards/breadboard track/traces, connector/plug, weld;
- **Ground**: common reference point of an electrical circuit, assuming a 0 V electric voltage
  - *Note: even if an electrical circuit/diagram does not have an explicit Ground connection, we may reference one of its points (usually nodes) to the Ground*;
- **Branch**: set of (electrical) components interconnected in series
  - *Note: a Branch is always delimited by two Nodes, except for the special case of a circuit with a single Loop/Branch*;
- **Node**: point of interconnection between three or more Branches (interconnected by one or more Interconnecting Elements)
  - In case two or more points of interconnection are directly interconnected via one or more Interconnecting Elements, these points constitute a single Node;
- **Ideal Voltage Source**: Voltage Source with no internal Impedance (0 ohm);
- **Isolated Voltage Source**: Ideal Voltage Source that is the only component of a Branch
  - In case there are more than one Ideal Voltage Source connected in series in a Branch, these should be aggregated into just one (Isolated Voltage Source);
- **Supernode**: set of two or more Nodes interconnected by Isolated Voltage Sources, including the respective Branches
  - A Supernode is like a "cloud", hiding its own Nodes/Branches, so that the KNL of a Supernode is formed by the convergent/divergent currents to/from that Supernode (that go in/out of its "cloud");
  - A current circulating in a Branch that belongs to a Supernode (hidden inside its cloud) cannot be computed just by solving the NVM equation system, because this current is not going to make part of the KNL of the Supernode; this current can be computed *a posteriori*, after all other currents have been determined, by applying the KNL in one of its two Nodes (that make part of that Supernode);
- **Grounded Supernode**: a Supernode that has one of its Nodes connected to the *Ground*
  - Note: the voltages in all its Nodes are numerically known, i.e. can be computed analytically through the relation between the Voltage between each Node and the *Ground*, through the concatenation of (one or more) Isolated Voltage Sources;
- **Floating Supernode**: a Supernode that has no Node connected the *Ground*
  - Note: the voltage in any of its Nodes can be related to the voltage in any other of its Nodes, through the "differential" Voltage between each pair of Nodes, through the concatenation of (one or more) Isolated Voltage Sources;

B. The NVM-SA algorithm

The "unknown" variables are the Node Voltages (U_A, U_B, U_C,...), related to the *Ground* Node. Upon determination of the voltage in every Node, all the currents and voltages in the circuit can be computed. The algorithm is as follows:

Algorithm 1 Proposed NVM-SA

1: **procedure** IDENTIFY THE FUNDAMENTAL VARIABLES FOR THE NVM

1) **Identify & Output** the number of Branches/Currents in the circuit $\rightarrow R$
   a) **For each** Branch in the circuit
      i) Identify the Branch Current with a symbol, e.g. $I_1, I_2, I_3, \ldots$
      ii) Assume a certain (hypothetical) Current direction
      iii) Mark the Branch Current (ID + direction) in the circuit schematics
2) **Identify & Output** the number of Nodes $\rightarrow N$
   a) **For each** Node in the circuit
      i) Identify the Node with a voltage, e.g. U_A, U_B, U_C, ...
      ii) Mark the Node Voltage ID in the circuit schematics
3) **Identify & Output** the number of Isolated Voltage Sources (IVS) $\rightarrow T$
   a) Notes:
      - *In case there is more than one Ideal Voltage Source connected in series in a Branch, these should be aggregated and considered as just one Isolated Voltage Source*
      - *The Isolated Voltage Sources should be stored/presented as a set, e.g.* $IVS = \{E_X, E_Y\}$
   b) **IF** $T = 0$ (*No Isolated Voltage Sources*)
      i) Select one of the Nodes as a reference/Ground node, e.g. if Node W is chosen, then $U_W = 0\ V$
         - *Note: as the voltage in this node is fixed to 0 V, the number of unknowns becomes equal to* $N - 1 - T$
      ii) Mark that node in the circuit schematics (*use Ground symbol* $\perp$)
      iii) Skip procedure 2 (*There are no Supernodes*)
   c) **IF** $T = 1$
      i) Fix the reference/Ground Node to one of the two Nodes connected to this Isolated Voltage Source
   d) **IF** $T \geq 2$
      i) Check if there are Branches (with Isolated Voltage Sources) that are connected together, and if so group them together and select the largest group (in case there are more than one group) as the *Grounded Supernode*, in procedure 2: 1).
4) **Compute & Output** the number of KNL equations that will build the equation system to be solved $\rightarrow N-1-T$

2: **procedure** IDENTIFY & OUTPUT ALL THE SUPERNODES AND THE VOLTAGE RELATIONS BETWEEN THEIR NODES

1) **Identify & Output** the (only) *Grounded Supernode* (SN_G) and the voltage in its Nodes
   a) Identify & Output the *Grounded Supernode* and respective Nodes
      - *Note: the Nodes (two or more) can be stored/presented as a set, e.g.* $SN_G = \{U_A, U_B\}$
   b) Compute & Output the equations that relate the voltage in each pair of its Nodes and Compute & Output the voltage in each of its Nodes (numerically, in V)
      - *Note: the Grounded Supernode voltage equations are not considered in the NVM equation system, as the voltage in its Nodes can be computed a priori.*
2) **Identify & Output** all the Floating Supernodes (SN_F) and corresponding voltage relation equations
   a) **For Each** *Floating Supernode*
      i) Identify & Output the *Floating Supernode* and respective Nodes (two or more)
         - *Note: select some kind of index, e.g.* SN_{F1}, SN_{F2}
         - *Note: the Floating Supernodes can be stored/presented as a set, e.g.* $SN_{F1} = \{U_A, U_B\}$, $SN_{F2} = \{U_E, U_F, U_G\}$
      ii) Choose a Node from the Floating Supernode as a reference.
      iii) Compute & Output the equations that relate the voltage in each pair of its Nodes, in order to the reference Node chosen in the previous step.

3: **procedure** BUILD & OUTPUT THE $N - 1 - T$ KNL EQUATIONS, AS A FUNCTION OF THE NODE VOLTAGES

1) Write the $N - 1 - T$ KNL equations, as a function of Branch Currents ($I_1, I_2, I_3, \ldots$)
2) **For Each** Branch/Current in the circuit $\rightarrow I_x$
   a) **If** the Branch contains more than one Impedance connected in series (including internal impedance of one or more Voltage Sources, aggregate all impedances into just one equivalent Impedance $\rightarrow Z_{xeq}$
   b) **If** the Branch contains more than one Voltage Source connected in series, aggregate all of them into just one equivalent Voltage Source $\rightarrow E_{xeq}$
   c) Write the Branch Current as a function of the voltage across the equivalent Impedance divided by the Impedance (Ohm's Law)
      i) Subtract the voltage before the equivalent Impedance by the voltage after the equivalent Impedance $\rightarrow U_{Zxeq}$
         - *Note: mind the direction of the Branch Current (that has been assumed a priori, in procedure 1: 1) ii))*
         - *Note: if the current belongs to a Branch from any Supernode, meaning the Branch has no impedance, the Ohm's Law cannot be used, and the current will not have an equation. Instead, it will be computed at the end of the algorithm using the known currents*

- *Note: if the current belongs to a Branch that contains a Current Source, its value is automatically known and the Ohm's equation can be discarded*

ii) Store & Output $I_X = \frac{U_{Zxeq}}{Z_{xeq}}$

3) Rewrite the $N-1-T$ KNL equations, (as in Procedure 3: 1)), as a function of Node Voltages (U_A, U_B, U_C, ...), using the equations generated in Procedure 3: c) ii)

4: **procedure** COMPUTE THE VOLTAGE IN EVERY NODE

1) **Substitute** every Node Voltage from each Floating Supernode by the expression in order to its reference and rewrite the $N-1-T$ KNL equations.
2) **Solve** the equation system & **Output** the Voltage in every Node ($U_A = \ldots, U_B = \ldots, U_C = \ldots$)

5: **procedure** (OPTIONAL) COMPUTE & OUTPUT THE BRANCH CURRENTS

1) Use the equations generated in procedure 3: c) ii) to compute the numerical values of the Branch Currents.

C. Example Circuit

In order to illustrate the NVM algorithm use in the *U=RIsolve* application, we further analyse the circuit represented in Fig. 1. The analysis will be carried out through the completion of the procedures previously identified in Algorithm 1.

1) ***Procedure 1: Identify the fundamental variables for the NVM****:*

1) Number of Branches: $R = 13$
   - Identified Currents: $\{I_{R1}, I_{R2}, \ldots, I_{R13}\}$
   - Currents direction: identified in Fig. 1
2) Number of Nodes: $N = 8$
   a) Identified Node Voltages: $\{U_A, U_B, \ldots, U_G\}$
   b) Wire labels ($A, B, \ldots, G$) are used to identify/mark Node Voltages.
3) Number of Isolated Voltage Sources: $T = 4$
   - Condition $T \geq 2$ is **true**
     - Both pairs of Voltage Sources $\{V2, V3\}$ and $\{V5, V7\}$ are connected together to form two Supernodes. As there is no largest group of sources, we chose the first pair to form the *Grounded Supernode*, and the ground is placed at node H, as Fig. 1 shows.
4) Number of KNL Equations: $3 (N-1-T = 8-1-4 = 3)$

2) ***Procedure 2: Identify & Output all the Supernodes and the voltage relations between their Nodes****:*

1) The *Grounded Supernode* is identified in Fig. 2 (rotated to horizontal, for convenience).

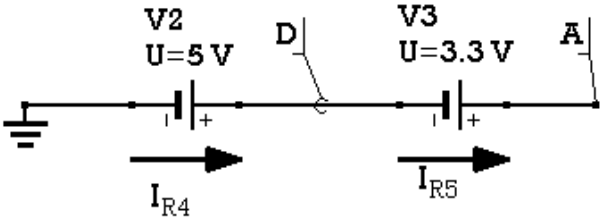


Fig. 2. *Grounded Supernode* from the example circuit

a) The *Grounded Supernode* is formed by two nodes (plus *Ground*) and can be represented as: $SN_G = \{U_D, U_A\}$.
b) The equations expressing the Voltage relation between the Nodes inside the *Grounded Supernode* are as follows:

$$\begin{cases} U_D = U_{GND} + V2 \\ U_A = U_D + V3 \end{cases} \Leftrightarrow \begin{cases} U_D = 5\ V \\ U_A = 8.3\ V \end{cases}$$

2) There is only one *Floating Supernode* and it is represented in Fig. 3.

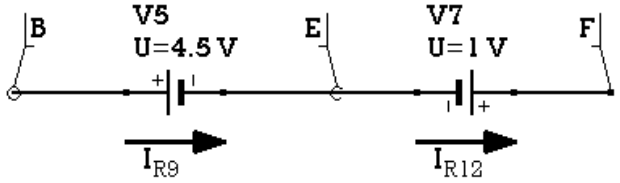


Fig. 3. Floating Supernode from the example circuit

a) The Floating Supernode is formed by three nodes and can be represented as: $SN_{F0} = \{U_B, U_E, U_F\}$.
b) The reference chosen was Node E.
c) The equations that relate each Node from the *Floating Supernode* (in order to reference E) are as follows:

$$\begin{cases} U_B = U_E + V5 \\ U_F = U_E + V7 \end{cases} \Leftrightarrow \begin{cases} U_B = U_E + 4.5 \\ U_F = U_E + 1 \end{cases}$$

3) ***Procedure 3: Build & Output the $N-1-T$ KNL equations, as a function of the Node voltages****:*

1) The KNL equations, as a function of Branch Currents are listed below:

$$\begin{cases} 0 = I_{R1} + I_{R2} + I_{R3}, & (Node\ C) \\ I_{R8} = I_{R10} + I_{R13}, & (Node\ G) \\ I_{R6} + I_{R7} + I_{R10} + I_{R13} = 0, & (Node\ SN_{F0}) \end{cases}$$

2) Determination of the Branch Currents (excluding the Supernodes Branch Currents):
   - Multiple impedances in the same Branch found: $Z_{R6EQ} = R3 + R4$
   - Voltage Sources in series found: $U_{R11EQ} = V6 - V8$

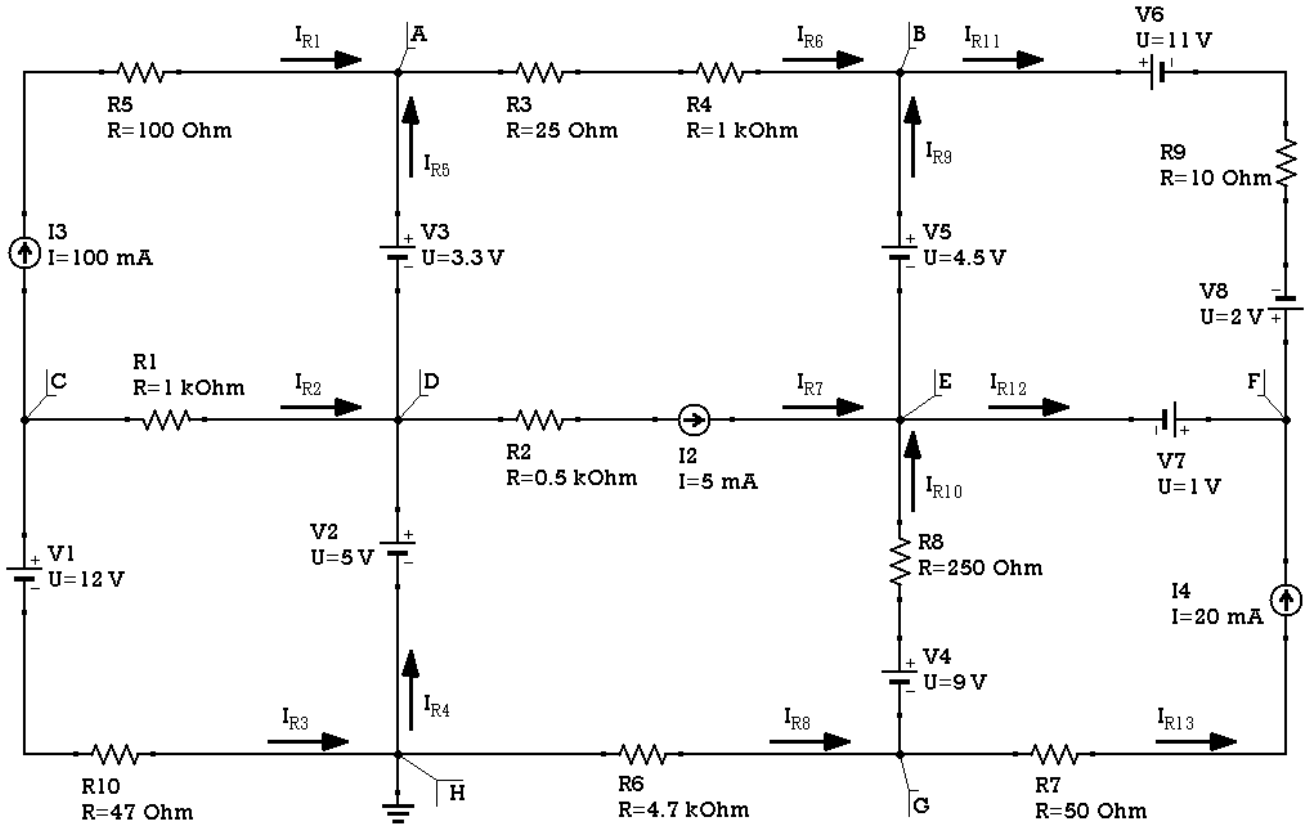


Fig. 1. Example circuit for algorithm demonstration (drawn in QUCS).

- Computed Branch Currents:

$$\begin{cases} I_{R1} = I3 \\ I_{R2} = \frac{U_C - U_D}{R1} \\ I_{R3} = \frac{U_C - V1 - U_{GND}}{R10} \\ I_{R6} = \frac{U_A - U_B}{Z_{R6EQ}} \\ I_{R7} = I2 \\ I_{R8} = \frac{U_{GND} - U_G}{R6} \\ I_{R10} = \frac{U_G + v4 - U_E}{R8} \\ I_{R11} = \frac{U_B - U_{R11EQ} - U_F}{R9} \\ I_{R13} = I4 \end{cases} \Leftrightarrow \begin{cases} I_{R1} = 0.1\ A \\ I_{R2} = \frac{U_C - 5}{1000} \\ I_{R3} = \frac{U_C - 12}{47} \\ I_{R6} = \frac{8.3 - U_B}{1025} \\ I_{R7} = 0.005\ A \\ I_{R8} = -\frac{U_G}{4700} \\ I_{R10} = \frac{U_G + 9 - U_E}{250} \\ I_{R11} = \frac{U_B - 9 - U_F}{10} \\ I_{R13} = 0.02\ A \end{cases}$$

3) Rewrited KNL equations as function of Node Voltages:

$$\begin{cases} 0 = 0.1 + \frac{U_C - 5}{1000} + \frac{U_C - 12}{47} \\ -\frac{U_G}{4700} = \frac{U_G + 9 - U_E}{250} + 0.02 \\ \frac{8.3 - U_B}{1025} + 0.005 + \frac{U_G + 9 - U_E}{250} + 0.02 = 0 \end{cases}$$

*4) **Procedure 4: Compute the Voltage in every Node**:*

1) The reference for the Supernode SN_{F0} is Node E (chosen in procedure 2) 2) b)). In the equation system, the potential U_B is then replaced with its expression in order to Node E.

$$\begin{cases} 0 = 0.1 + \frac{U_C - 5}{1000} + \frac{U_C - 12}{47} \\ -\frac{U_G}{4700} = \frac{U_G + 9 - U_E}{250} + 0.02 \\ \frac{8.3 - (U_E + 4.5)}{1025} + 0.005 + \frac{U_G + 9 - U_E}{250} + 0.02 = 0 \end{cases}$$

2) Voltages in every Node:

a) The following Node Voltages can be immediately determined (*Grounded Supernode*):

$$\begin{cases} U_D = 5\ V \\ U_A = 8.3\ V \end{cases}$$

b) The following Node Voltages are obtained by resolving the equation system:

$$\begin{cases} U_C = 7.2\ V \\ U_G = -3.99\ V \\ U_E = 9.8\ V \end{cases}$$

c) The remaining Node Voltages can be computed from

the *Floating Supernode* Equations:

$$\begin{cases} U_B = U_E + 4.5 \\ U_F = U_E + 1 \end{cases} \equiv \begin{cases} U_B = 14.3\ V \\ U_F = 10.8\ V \end{cases}$$

5) ***Procedure 5: Compute and Output the Branch Currents****:*

1) The following Branch Currents are known *a priori* (Current Sources):

$$\begin{cases} I_{R1} = 0.1\ A \\ I_{R7} = 0.005\ A \\ I_{R13} = 0.02\ A \end{cases}$$

2) The following Branch Currents can be obtained from the equations in procedure 3) 2):

$$\begin{cases} I_{R2} = 0.002\ A \\ I_{R3} = -0.1\ A \\ I_{R6} = -0.006\ A \\ I_{R8} = 0.001\ A \\ I_{R10} = -0.019\ A \\ I_{R11} = -0.55\ A \end{cases}$$

3) The remaining Branch Currents can be computed through KNL:

$$\begin{cases} I_{R4} = I_{R5} + I_{R7} - I_{R2} \\ I_{R5} = I_{R6} - I_{R1} \\ I_{R9} = I_{R11} - I_{R6} \\ I_{R12} = -I_{R11} - I_{R13} \end{cases} \Leftrightarrow \begin{cases} I_{R4} = -0.103\ A \\ I_{R5} = -0.106\ A \\ I_{R9} = -0.544\ A \\ I_{R12} = 0.53\ A \end{cases}$$

V. Application design and architecture

A. The netlist files and parsing

Most Spice-based circuit simulators, such as QUCS, generate a circuit description during simulation and provide it in a format of a text file, entitled *netlist*. In preliminary versions of Spice, *netlists* were usually employed as an input to the simulation [59], however, from a users' perspective, this method long lost terrain against more appealing and easy to use graphical user interfaces. In *U=RIsolve* we rely on the QUCS *netlist* as an input to the analysis algorithm, which requires users to perform an *a priori* simulation and to generate the *netlist* file.

An illustration of a *netlist* content (similar to QUCS *netlist*) is given in Fig. 4 (right), and the circuit schematic (left) with every component labelled and interconnections defined with numbers (1, 2, 3). Each *netlist* line describes an existing electrical component in the circuit schematic with the following essential information:

1) Component type: identifies the class of the component ("R" for resistors, "Vdc" for DC Voltage Sources, "Idc" for DC Current Sources, and so on).
2) Component label: an unique attribute/representation of the component assigned in the simulator (e.g. R1).
3) Component connections: determines the two interconnection points (start and end) of the element.
4) Component unit and value: specifies the electrical quantity associated to the element.

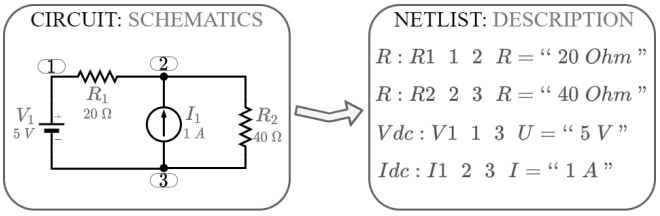


Fig. 4. Electrical circuit (left) and its *netlist* description (right).

As we aim to apply our NVM algorithm to the circuit, two main issues arise: how to identify Branches and how to extract Sources polarities, out of the netlist information.

The first issue is accomplished through the components interconnections. Here we create two types of interconnections: virtual Nodes, and real Nodes (or just "Nodes" as formalised in Section IV). The variation between these types of Nodes is that virtual Nodes just interconnect two components, while real Nodes join together three or more components. In our application, real Nodes must be identified with a letter (wire label, in QUCS). In the example from Fig. 4, Node 1 is a virtual Node, connecting V_1 and R_1, while Nodes 2 and 3 are real Nodes, as they connect R_1, source I_1 and R_2; and V_1, source I_1, and R_2, respectively. Knowing that a Branch is always delimited by two real Nodes, enables to identify them by evaluating the components interconnections in the *netlist*. On one hand, Branches with only one component are automatically recognised, since the component's interconnections are both real Nodes (the case of I_1 and R_2) in the Fig. 4 example. On the other hand, Branches with multiple components require a more complex procedure, as follows:

1) If one component has one real Node and one virtual Node as interconnections, start the Branch in the real Node.
2) Find in the *netlist* the component with the remaining instance of the virtual Node (note that there exist only two instances of each virtual Node in the *netlist*) and check the other interconnection from the component found.
3) Repeat Step 2 while a real Node is found, so that the Branch can be completed.

This procedure could be illustrated with the example we provided. Suppose we find the resistor R_1 in the *netlist* (which appears in the first line), we observe that its interconnections consist of a real Node (2) and a virtual one (1). The algorithm assumes the Branch starts at Node 2, and searches for Node 1 in the *netlist*. Node 1 appears at component V_1, and its remaining interconnection is Node 3, which is a real Node, meaning the Branch starts at Node 2 and ends at Node 3.

The second issue refers to both Current and Voltage sources polarity. The algorithm identifying the Voltage sources positive

and negative poles and the Current sources direction is crucial for a correct analysis. Usually, simulators define a specific interconnection to be the positive pole (in case of a Voltage source), or the flow direction (in case of a Current source). In QUCS *netlist*, as the one shown in this example, that interconnection is always the first Node. In Fig. 5, it is demonstrated how *netlist* represents sources polarities. On the left, we observe that the Voltage Source positive pole interconnection is always the first Node in the *netlist*. Similarly, on the right hand side, the direction to where the current flows is represented by the first Node.

With these issues solved, any electrical circuit can be accurately analysed through our NVM approach, as we have all the necessary information to model it.

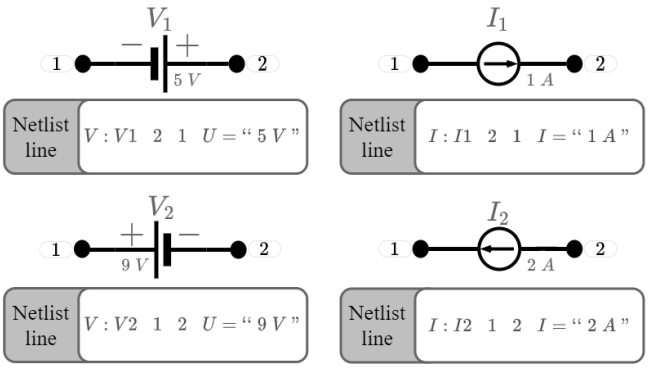


Fig. 5. Changes in *netlist* with polarity inversion.

B. Software Architecture

The *U=RIsolve* application receives the circuit information (*netlist*) as an input, and outputs the solution through the NVM. Thus, we segment the software algorithm into twelve (12) steps, illustrated in Fig. 6 and described next.

1) Application inputs: The system accepts both circuit description (*netlist*; text file) and schematics (image file). The circuit *netlist* models the circuit, which is mandatory to perform the analysis. Circuit schematics (image), on the other hand, adds more intuition to users, since it is displayed in the output along with results. This latter file is optional.

2) Netlist error check: In the early learning phases, using circuit simulators can be a complex task, and students often make mistakes. These mistakes lead to error-prone netlists that need to be handled by our application. In this step the *netlist* is analysed in order to detect errors that are split into warnings (problems in *netlist* that can be solved at the execution time) and critical errors (problems that prevent the algorithm to proceed). In case warnings are found, information is attached to the output; however, when critical errors are found, the algorithm stops and these problems are reported to users.

The critical errors that *U=RIsolve* can currently diagnose are illustrated in Fig. 7 and described next:

- **E1**: Non-identified real Nodes (all Nodes must be labelled with letters).

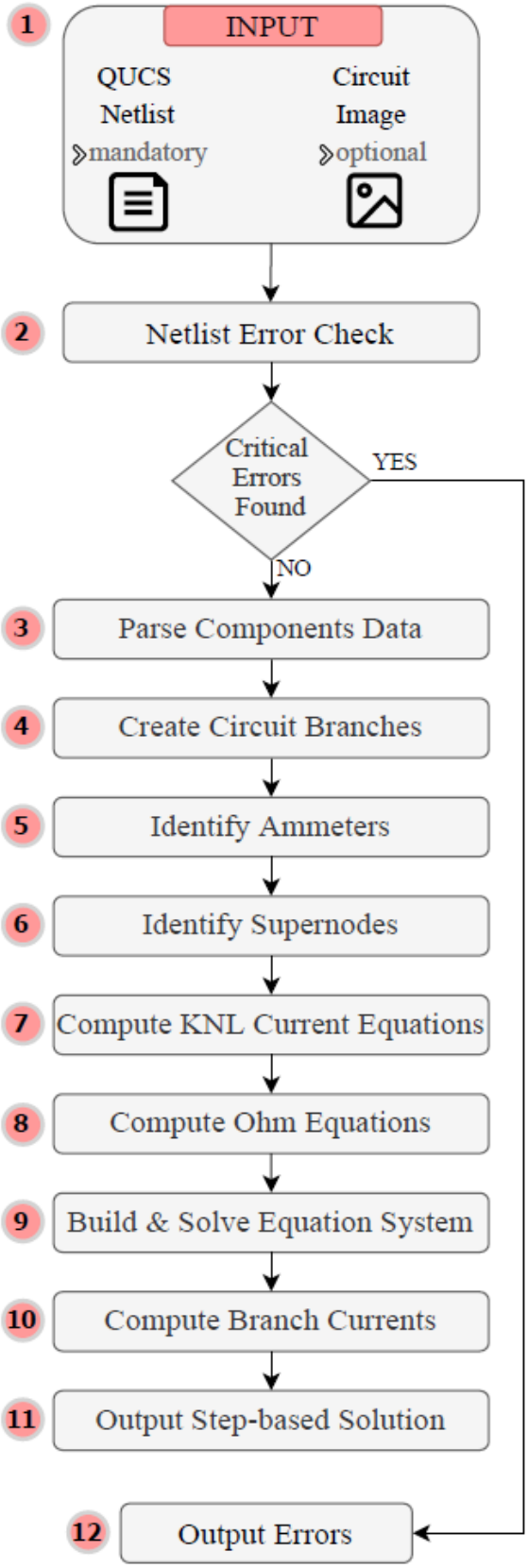


Fig. 6. U=RIsolve software architecture.

- **E2**: Invalid Ground ID/label.
- **E3**: AC component in DC circuit.
- **E4**: Invalid component unit.
- **E5**: Multiple Current Sources in the same Branch.
- **E6**: Duplicated Node causing short-circuits.

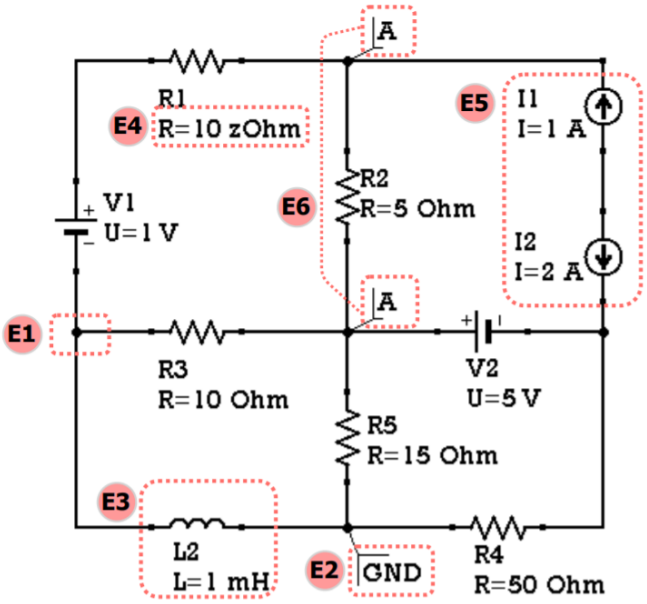


Fig. 7. Critical errors detected by *U=RIsolve*.

And the warning errors that can be detected are the following:

- Ground Node not found: this error occurs when users do not introduce the *Ground* symbol ($\frac{\perp}{=}$) in the circuit and the software solves it by placing the *Ground* in the best position (that reduces the number of variables). *U=RIsolve* then informs users in which Node the *Ground* is located.
- Virtual Nodes mistakenly identified: if users wrongly label a virtual Node, *U=RIsolve* execution is not affected, since it is changed to "*net_i*" being *i* the smallest available index for virtual Nodes. Nevertheless the error is reported to users.
- Invalid real Node ID: some Nodes IDs should be avoidable as they could introduce ambiguities to results (e.g "VA" is taken as an invalid ID, because it may be misinterpreted as Node Voltage).
- Multiple Ammeters in the same Branch: in case this error occurs, *U=RIsolve* simply chooses an Ammeter and deletes the remaining, adjusting the netlist.
- Multiple AC frequencies detected: if *U=RIsolve* founds AC Voltage Sources with different frequencies, it favours the controller frequency. However, if the controller does not have a valid frequency (e.g an array of frequencies), the algorithm chooses a random frequency from an AC Voltage Source and warns users.
- Duplicated measuring instrument IDs: although QUCS forbids this operation, users could manually change the IDs directly in the *netlist* and mistakenly set two or more equal IDs. *U=RIsolve* detects this problem and reports it to users.
- Multiple Voltmeters measuring the same potential difference: even though it does not affect the simulation, for learning purposes, the algorithm warns users about Voltmeters inserted in the exact same Nodes.

3) Components data parsing: At this step, every component from the *netlist* is parsed into an object with specific fields depending on their type. For instance, a DC Voltage Source has properties such as reference (ID), positive and negative Node, unit (V, mV) and value (number). Each component type is extracted from the *netlist* through their unique label assigned by QUCS (e.g. *R* for Resistors, *L* for Inductors, *C* for Capacitors) and then grouped with other components of the same type.

4) Branches creation: Each Branch is identified through the components interconnection data. Since it is acknowledged that a Branch is formed by two real Nodes, interconnection information will determine which components belong to a single Branch. Branches also have multiple attributes, in particular *start* and *end* Nodes, *equivalent impedance* and *voltage*, an *unique ID* and every component that belongs to it.

5) Ammeters identification: Right now, U=RIsolve has a limitation concerning the assignment of each Current's direction/flow, due to the fact that we cannot yet provide a visual demonstration of the Current, for example inserting an arrow in the circuit schematics. If we generate random Currents, it is only possible to describe them textually to users, for instance: "*Current* I_1 *flows from Node* A *to Node* B". However, users may alternatively define the Currents direction through Ammeters (in QUCS), so we offer an additional solution: using Ammeters to determine the Current direction in Branches. This way users can define the direction by themselves which will lead to their preferred solution to the NVM. In case no Ammeter exists in a Branch with a Current Source, the Current Source direction is used to determine the flow. If neither Ammeter or Current Source are found, then the Current direction is randomly generated.

6) Supernodes identification: Supernodes are detected using the Voltage Sources interconnection data. First, we determine the isolated Voltage Sources by checking the delimiting Nodes type. If a Voltage Source has two real Nodes as boundaries, then we are in presence of an isolated Voltage Source, which automatically belongs to a Supernode. Supernodes with multiple Voltage Sources are processed by evaluating the common connections of each isolated Voltage Source with others. It is also at this step that *U=RIsolve* computes the best *Ground* position (Node), which returns the circuit Nodes that minimizes the number of unknowns (Node Voltages). In case the user does not choose the best Node as *Ground*, a tip is displayed with the best *Ground* positions.

7) KNL Current equations: These equations are computed based on starting and ending Nodes from the Currents objects. As previously mentioned, each Branch Current direction is

determined manually by users with Ammeters or randomly generated (if no Ammeter is found). Simply put, for each unknown Node Voltage, if its inner Node is found to be the starting Node of a Current, that means that the Current is flowing out from the Node and the Current label will appear in the second term of the KNL equation. On the other hand, if the end Node is found, meaning that the Current is flowing in to the Node, its label will be at the first term of the equation.

8) Ohm equations: Every current from the previous step must be replaced by its equivalent Ohm equation, which introduces the system variables we want to solve for, i.e. the Node Voltages. As mentioned before, Currents objects have both equivalent voltage and equivalent impedance. The Ohm's equation numerator is then computed using the current starting node, the equivalent voltage and the ending node. The Ohm's equation denominator is equal to the equivalent impedance of the Branch.

9) Equation System Building and Solving: A step solution like the one we provide in *U=RIsolve* requires consecutive simplifications and adjustments to the equations at each step, which requires tools for expression parsing/evaluation and symbolic computation. In order to build the solution parts, *U=RIsolve* uses two well-known JavaScript libraries for mathematical string manipulations: Math.js [60] and Algebra.js [61]. Importantly, we have designed a JavaScript library for linear equation system solving [14] that supports both real and complex numbers, as existing solutions [62] were not providing correct results for complex equation systems.

10) Branch Currents calculation: Although this is an optional part of the algorithm, it is important to show students how to compute the Currents, for a complete analysis of the circuit. In this step we divide the Currents calculation into three categories:

- Current with a known value *a priori*, i.e. there is a Current Source in the Branch.
- Current in a Branch with a finite impedance ($> 0\ \Omega$) so that it is possible to formulate an Ohm's equation and compute it using the solution of the NVM algorithm.
- Current in a Branch that belongs to a Supernode; its value can only be obtained through KNL Current equations; the algorithm must find which KNL equation to use in order to sequentially calculate these Currents.

11) Step-based solution: The solution's purpose is to conduct the principles of the NVM algorithm in a simple, complete and attractive way. In *U=RIsolve* results, students can find all the Supernodes information, the circuit Currents flow/direction, the KNL equations together with a canvas illustrating how they are formulated and every step of how to generate the final equations system of the NVM algorithm. The output also includes warnings, notes and tips so that users understand how NVM works and may improve their simulation skills. The mathematical content in the solution is presented in TEX typesetting format, by means of a JavaScript lightweight parser and renderer KaTeX [63].

VI. User Interface and Output

A. User interface

The *U=RIsolve* application features a simple User Interface (UI) that allows users to upload both *netlist* and circuit image and perform the NVM-SA analysis, as illustrated in the home page screenshot in Fig. 8. Importantly, the web-based user interface is adaptive (responsive) to any screen size/resolution, which enables its use in laptop/desktop personal computers, tablets and smart-phones, being totally agnostic to the underlying hardware and operating system. An instructions section explains how to generate the *netlist* in QUCS and the examples section provides ready-to-use circuits ordered by complexity.

Once students execute the NVM-SA and check the simulation results, they can refresh the page or upload new files and reanalyse. Work in progress addresses extending the *U=RIsolve* application with other circuit analysis methods (such as the *Loop Current Method* and the *Branch Current Method*), options that are still disabled (gray buttons). We also envisage the design of a new module for drawing circuit schematics, to turn the *U=RIsolve* application "self-sufficient", i.e. cutting the umbilical cord with the QUCS simulator (most students manifested their preference for this possibility).

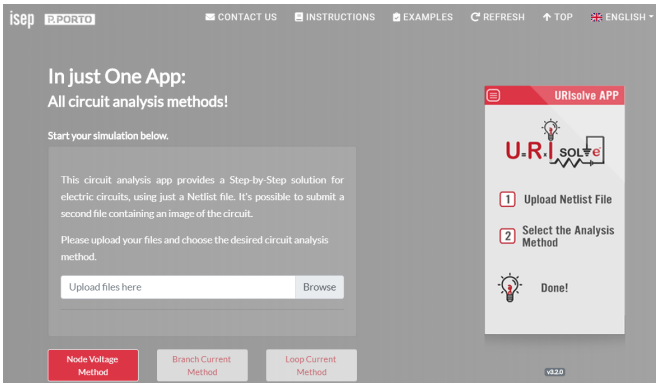


Fig. 8. *U=RIsolve* home page.

B. NVM-SA analysis output

As previously mentioned, the *U=RIsolve* output is a step-by-step solution to the NVM-SA. This solution is presented in a modal dialog box, once one selects the *Node Voltage Method* button, and it is organised in nine sections, as illustrated next, for the circuit in Fig. 1.

1) Circuit schematics: In this section the uploaded image of the circuit is presented. In case users do not provide an image, this part of the output is hidden. In this demonstration we are not showing the circuit image output, since it is very similar to the one in Fig 1, with a single variation that is the introduction of Ammeters in all Branches (to force the selection of Current directions).

2) Fundamental variables: Here we show users the necessary variables to compute the number of equations needed for the uploaded circuit. As shown in Fig. 9, the fundamental

variables consist in the number of Branches, Nodes, isolated Voltage Sources and NVM equations.

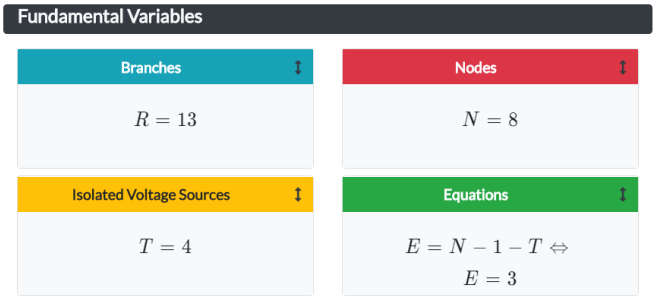


Fig. 9. *U=RIsolve* output: fundamental variables.

3) Circuit information: This region provides more details about the circuit: the AC frequency , the number of Current Sources (that will help students to realise how many known currents exist), the number of Ammeters (to declare the Currents direction), and finally the simulation type (AC or DC). Fig. 10 demonstrate this step.

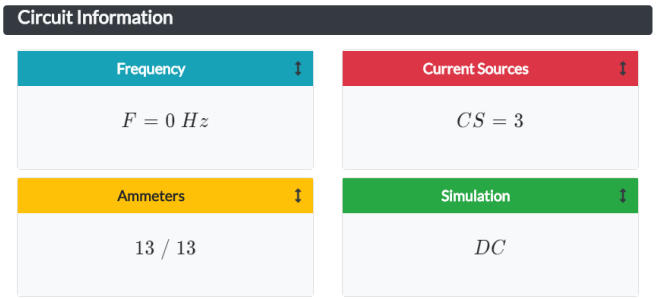


Fig. 10. *U=RIsolve* output: circuit information.

4) Supernodes: This part describes the Supernodes that have been automatically generated (Fig. 11, i.e which Nodes constitute each Supernode and their equations. An extra description (white containers at the bottom of each Supernode card) is shown when users press the "Show Steps" button, illustrating how each Supernode equation has been obtained.

5) Branch Currents: This section outputs all Currents and their direction, according to the Ammeters that have been placed in the simulation or generated automatically if Ammeters are not used (Fig. 12. It is output both starting and ending Nodes of each current, as well as its label (which is also obtained from the Ammeters, if any). For instance, Current $Ir1$ flows from Node C to Node A and passes through the components $I3$ and $R5$.

6) Equivalent Impedances and Voltages: This section's purpose is to indicate students that the program found one or more simplifications that could have been applied to the circuit under analysis, such as in-series Resistors, Inductors and Capacitors (equivalent Impedances) or in-series voltage sources (equivalent Voltages). Fig. 13 represents the occurrences of equivalent Voltages/Impedances found in the example circuit, namely $ZeqAB$ (an equivalent impedance formed by $R3$ and $R4$) and $VeqBF$ (an equivalent impedance formed by $V6$ and $V8$).

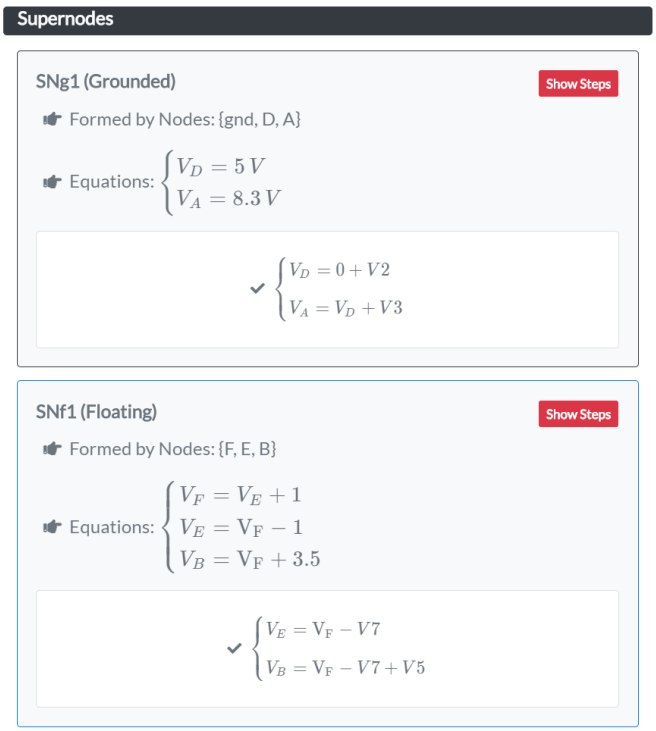


Fig. 11. *U=RIsolve* output: Supernodes.

7) KNL Current equations: This section outputs the KNL equation for all Nodes, as shown in Fig. 14, including an illustration of the Currents flowing in (drawn in green) and out (drawn in red) of each Node. The canvas objective is to offer further intuition on which Currents should be in the first or in the second term of their KNL equation, for each Node.

8) Equations system: The equations system section shows the final set of equations that enable to compute the Node Voltages, as demonstrated in Fig. 15. Importantly, the output explains how to determine them, through six steps that describe the process of creating the system, starting from the KNL equations and making the necessary adjustments and substitutions, until finally obtaining the final set of equations. This way, we ensure that users thoroughly understand the methodology. Some steps may occasionally be omitted depending on the circuit that is being simulated; for instance if the circuit does not include any Supernodes. The equations system steps pop-up once the user clicks the "Show Steps" button (Fig. 15).

a) Step 1: The first step is illustrated in Fig. 16, and shows how to rearrange the KNL equations, so that every Current is in the first term (some calculators or online linear system solvers may require this operation).

b) Step 2: The second step purpose is to substitute the known Current variables by their numerical value, i.e. whose Branches have a Current Source defining its Branch Current. For our example, the known Currents will be $Ir1$, $Ir7$ and $Ir13$. The direct substitution is represented in Fig. 17.

c) Step 3: The remaining Currents in the equation system should then be replaced by their Ohm Law equation, thus

Branch Currents

ID: $Ir1$
Flow: $C \rightarrow A$
Components: $I3, R5$

ID: $Ir6$
Flow: $A \rightarrow B$
Components: $R3, R4$

ID: $Ir5$
Flow: $D \rightarrow A$
Components: $V3$

ID: $Ir9$
Flow: $E \rightarrow B$
Components: $V5$

ID: $Ir11$
Flow: $B \rightarrow F$
Components: $V6, V8, R9$

ID: $Ir2$
Flow: $C \rightarrow D$
Components: $R1$

ID: $Ir3$
Flow: $C \rightarrow gnd$
Components: $V1, R10$

ID: $Ir7$
Flow: $D \rightarrow E$
Components: $I2, R2$

ID: $Ir4$
Flow: $gnd \rightarrow D$
Components: $V2$

ID: $Ir10$
Flow: $G \rightarrow E$
Components: $V4, R8$

ID: $Ir12$
Flow: $E \rightarrow F$
Components: $V7$

ID: $Ir13$
Flow: $G \rightarrow F$
Components: $I4, R7$

ID: $Ir8$
Flow: $gnd \rightarrow G$
Components: $R6$

Fig. 12. *U=RIsolve* output: Branch Currents.

Equivalent impedances and voltages

Branch from A to B

$$\{ ZeqAB = R3 + R4 = 1025$$

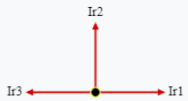
Branch from B to F

$$\{ VeqBF = -V6 + V8 = -9V$$

Fig. 13. *U=RIsolve* output: equivalent Impedances and Voltages.

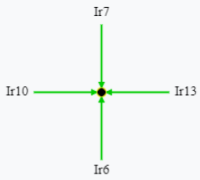
Current Equations (Kirchhoff Nodes Law - KNL)

Node C



$$0 = Ir1 + Ir2 + Ir3$$

Node SNf1



$$Ir13 + Ir7 + Ir10 + Ir6 = 0$$

Node G

$$Ir8 = Ir10 + Ir13$$

Fig. 14. *U=RIsolve* output: KNL Current equations.

Equation System

$$\begin{cases} \frac{-1}{10} + \frac{(\mathrm{V_C}-5)\cdot -1}{1000} + \frac{(\mathrm{V_C}-12)\cdot -1}{47} = 0 \\ \frac{1}{40} + \frac{G-\mathrm{V_F}+10}{250} + \frac{\frac{9600}{2000}-\mathrm{V_F}}{1025} = 0 \\ \frac{G\cdot -1}{4700} + \frac{(G-\mathrm{V_F}+10)\cdot -1}{250} + \frac{-1}{50} = 0 \end{cases}$$

Show Steps

Fig. 15. *U=RIsolve* output: equations system (main window).

Step 1: Reorder current equations — View –

$$\begin{cases} -Ir1 - Ir2 - Ir3 = 0 \\ Ir13 + Ir7 + Ir10 + Ir6 = 0 \\ Ir8 - Ir10 - Ir13 = 0 \end{cases}$$

Fig. 16. *U=RIsolve* output: equations system (Step 1).

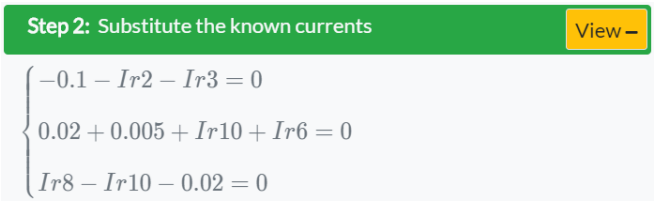


Fig. 17. *U=RIsolve* output: equations system (Step 2).

introducing the actual unknowns in the system: the Node Voltages. As simple as Ohm's Law may seem, there is a common error-prone situation that occurs when the potential difference in a Branch with multiple Voltage Sources in alternate orientations needs to be computed. This step shows users how to compute the remaining Currents in the system through Ohm's Law, as Fig. 18 outlines.

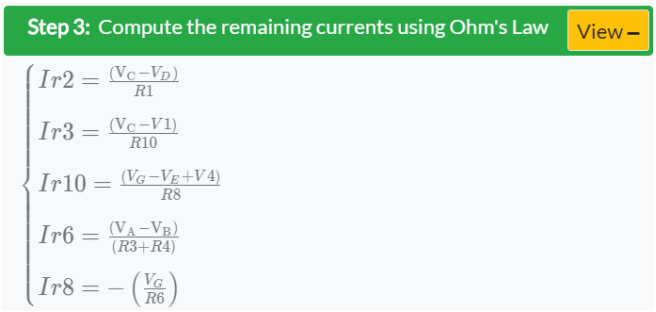


Fig. 18. *U=RIsolve* output: equations system (Step 3).

d) Step 4: This step addresses the substitution of the Currents in the equations system by their expressions, computed in Step 3. This operation is displayed in Fig. 19.

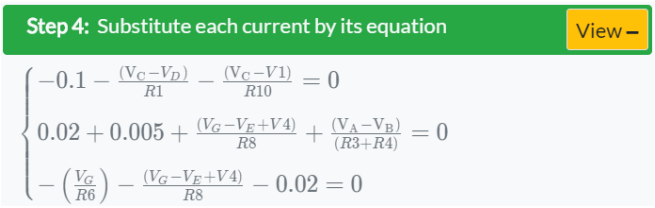


Fig. 19. U=*RIsolve* output: equations system (Step 4).

e) Step 5: The next substitution covered in this step is related to known Voltages. As observed in the previous step, equations include Voltage Sources $V1$ and $V4$ whose values are automatically acknowledged. As shown in Fig 20, every Voltage Source variable is replaced by its numerical value.

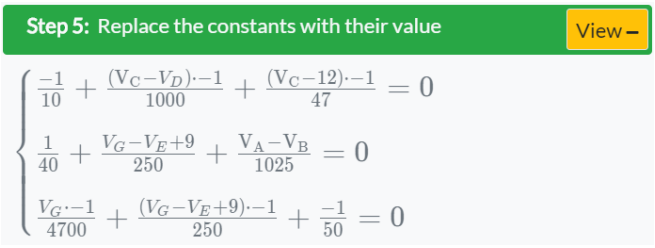


Fig. 20. *U=RIsolve* output: equations system (Step 5).

f) Step 6: The last stage involves setting a reference for each Floating Supernode, which is necessary to turn the system solvable. For instance, in the last version of the equations system from step 5, there are four unknowns (V_C, V_G, V_E and V_B), since V_A and V_D are already computed from the Grounded Supernode. However, given that the system has just three equations, and V_E plus V_B belong to the same Supernode, it is necessary to reference them to a single potential. In this example, the program chose Node F as reference (note that in Fig. 11, V_E and V_B are already referenced to V_F), and this information is output as shown in Fig. 21.

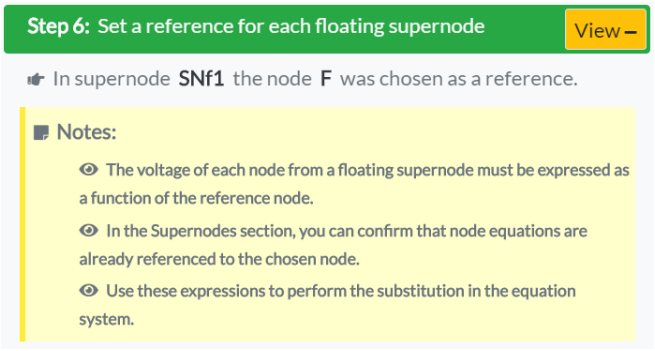


Fig. 21. *U=RIsolve* output: equations system (Step 6).

These steps, when completed, will lead students to the exact same final equation system as the one that is shown at the beginning of the "Equation System" section, establishing an essential coherence to the learning process.

9) Results: The last section of the step-based solution are the Results, which includes the Node Voltages and the Branch Currents. For our example, the Node Voltages output is shown in Fig. 22.

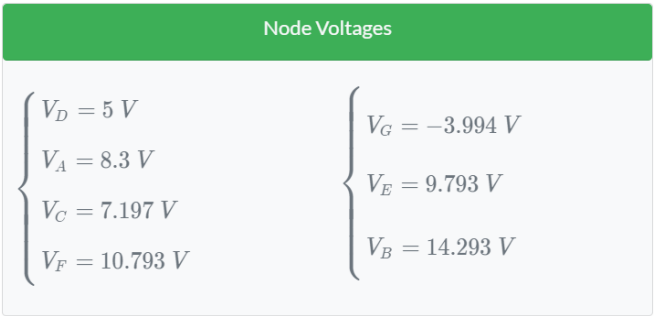


Fig. 22. *U=RIsolve* output: Node Voltage results.

As explained before, in our algorithm we segment the Currents into three types: obtained by an existing Current Source in the Branch; computed through the Ohm Law equation; and obtained through the KNL equation (in case the Current belongs to a Branch with one or more isolated Voltage Sources). The results for the Branch Currents are demonstrated in Fig. 23.

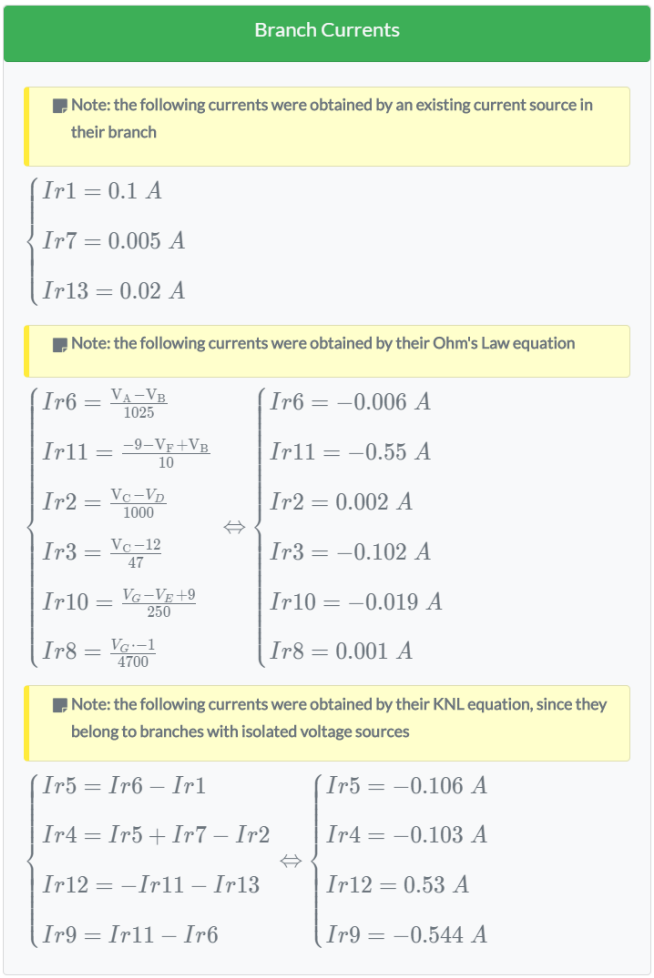


Fig. 23. *U=RIsolve* output: Branch Currents results.

VII. Conclusions

This work outlines a web-based application — *U=RIsolve* — that aims at motivating and improving the self-learning skills of undergraduate Electrical Engineering students, concerning the *Node Voltage Method* for electrical circuit analysis. This application relies on a very robust and comprehensive NVM algorithm that builds on the Supernode paradigm - dubbed *NVM-SA* – that has never been consolidated and implemented in software before, to our best knowledge.

This application introduces a didactic component to circuit simulators, enabling to explore innovative and more efficient teaching and self-learning methodologies. *U=RIsolve* depends on the QUCS simulator to provide the electrical circuit mathematical model (*netlist* file), including the identification (labelling) of the circuit Nodes, which voltage is to be determined by the NVM. Then the *U=RIsolve* application provides a step-based solution that includes circuit information, the NVM analysis process and the numerical results for Branch Currents and Node Voltages, allowing students to fully understand how this method is applied, step-by-step. Furthermore, *U=RIsolve* is also being used by teachers for the development of new circuit analysis exercises and a complementary way of in-class instruction.

The countless circuit layouts and complexity required our application to encompass multiple exceptions and peculiarities. Thus, to overcome this challenge and provide a robust analysis tool for the largest number of circuits, an in-depth experimentation phase was necessary before releasing a stable "bullet-proof" version, as reported in this paper and now publicly available[2]. There is still enthusiastic room for improvements and add-ons in the *U=RIsolve* framework. One of our main goals is to turn the application completely independent of third-party simulation tools (QUCS, in our case), by designing a graphical interface module for drawing and modelling circuits. In addition, we aim at extending the application with other circuit analysis methods, namely the *Branch Current Method*, the *Mesh/Loop Current Method*, as well as circuit simplification methods e.g. based on the *Superposition*, *Thévenin*, *Norton* and *Millman* Theorems.

References

[1] Sourceforge. Qucs project: Quite universal circuit simulator, 2017.

[2] M. Baser. Effects of conceptual change and traditional confirmatory simulations on pre-service teachers' understanding of direct current circuits. *Journal of Science Education and Technology*, 15(5):367–381, 2006.

[3] M. Baser. Promoting conceptual change through active learning using open source software for physics simulations. *Australasian Journal of Educational Technology*, 22(3):336–354, 2006.

[4] M. Brinson and V. Kuznetsov. Qucs-0.0.19s: A new open-source circuit simulator and its application for hardware design. In *2016 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–5, 2016.

[5] Sourceforge. Qucs download statistics, 2020.

[6] A. Silva. Design, implementation and integration of new functionalities in the qucs simulator. Licenciatura em Engenharia Electrotécnica e de Computadores, ISEP, 2017.

[7] N. Martins. Desenvolvimento de módulos para o qucs. Licenciatura em Engenharia Electrotécnica e de Computadores, ISEP, 2016.

[8] P. Macedo. Análise comparativa e melhoramento de simuladores de circuitos elétricos. Licenciatura em Engenharia Electrotécnica e de Computadores, ISEP, 2015.

[9] L. Kechiev, N. Kruchkov, and V. Kuznetsov. New active filter synthesis tool for qucs open-source circuit simulator. In *2016 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–4, 2016.

[10] M. Brinson and V. Kuznetsov. Improvements in qucs-s equation-defined modelling of semiconductor devices and ic's. In *2017 MIXDES - 24th International Conference "Mixed Design of Integrated Circuits and Systems*, pages 137–142, 2017.

[11] L. Sousa. U=risolve - aplicativo de análise de circuitos elétricos simulados no qucs. Licenciatura em Engenharia Electrotécnica e de Computadores, ISEP, 2018.

[12] L. Sousa, M. Alves, and F. Pereira. U=risolve: teaching & self-learning the nodal analysis method the easy way. In *CASHE - Conference on Academic Success in Higher Education*. Porto School of Engineering (ISEP), 2019. Best presentation award.

[13] R. Sommer, D. Ammermann, and E. Hennig. More efficient algorithms for symbolic network analysis: Supernodes and reduced loop analysis. *Analog Integrated Circuits and Signal Processing*, 3:73–83, 1993.

[14] A. Rocha. Syseqsolver.js - linear equations system solver. https://github.com/txroot/syseqsolver.

[15] Z. Pudlowski. Developing computer-aided education in electrical engineering. *International Journal of Electrical Engineering Education*, 31:111–127, 1994.

[2]The latest version is currently available at: http://urisolve.ddns.net/urisolve.

[16] O. Zavalani. Computer-based simulation development of a design course project in electrical engineering. *Computer Applications in Engineering Education*, 23:587–595, 2015.

[17] K. Kayisli, S. Tuncer, and M. Poyraz. An educational tool for fundamental dc-dc converter circuits and active power factor correction applications. *Computer Applications in Engineering Education*, 21:113–134, 2013.

[18] A. Müsing, U. Drofenik, and J. W. Kolar. New circuit simulation applets for online education in power electronics. In *2011 5th IEEE International Conference on E-Learning in Industrial Electronics (ICELIE)*, pages 70–75, 2011.

[19] Andre Morelato. A computer tool for helping engineering students in their learning of electrical energy basics. *Education, IEEE Transactions on*, 44:3 pp., 06 2001.

[20] D. Y. Northam. Introducing computer tools into a first course in electrical engineering. *IEEE Transactions on Education*, 38(1):13–16, 1995.

[21] S. R. Cvetkovic, R. J. A. Seebold, K. N. Bateson, and V. K. Okretic. Cal programs developed in advanced programming environments for teaching electrical engineering. *IEEE Transactions on Education*, 37(2):221–227, 1994.

[22] Geoff RUBNER. First-year undergraduate teaching of electrical and electronic engineering: innovation and inspiration. *International Journal of Electrical Engineering Education*, 54:281–282, 2017.

[23] K. Zeashan and M. Abid. Role of laboratory setup in project-based learning of freshmen electrical engineering. *The International Journal of Electrical Engineering & Education*, 54:150–163, 2017.

[24] A. Magana, S. Brophy, and G. Bodner. Instructors' intended learning outcomes for using computational simulations as learning tools. *Journal of Engineering Education*, 101:220–241, 2012.

[25] O. Campbell, J. Bourne, P. Mosterman, and A. Brodersen. The effectiveness of learning simulations for electronic laboratories. *Journal of Engineering Education*, 91:81–87, 2002.

[26] K. Vanlehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46:197–221, 2011.

[27] B. Skromme, V. Seetharam, X. Gao, B. Korrapati, B. E. McNamara, Y. F. Huang, and D. H. Robinson. Impact of step-based tutoring on student learning in linear circuit courses. In *FIE 2016 - Frontiers in Education 2016: The Crossroads of Engineering and Business*, United States, 2016. Institute of Electrical and Electronics Engineers Inc.

[28] F. de Coulon, E. Forte, and J. M. Rivera. Kirchhoff: an educational software for learning the basic principles and methodology in electrical circuits modeling. *IEEE Transactions on Education*, 36(1):19–22, 1993.

[29] L. Palma, R. F. Morrison, P. N. Enjeti, and J. W. Howze. Use of web-based materials to teach electric circuit theory. *IEEE Transactions on Education*, 48(4):729–734, 2005.

[30] J.P. Becker and C. Plumb. Towards a web-based homework system for promoting success of at-risk students in a basic electric circuits course. 2017.

[31] O. Yang, Y. Dong, M. Zhu, H. Yuewei, and M. Songand M. Yunjie. Ecvlab: A web-based virtual laboratory system for electronic circuit simulation. *Lecture Notes in Computer Science*, 3514:1027–1034, 2005.

[32] Sara Khaddaj and Ali Marmar. Electric circuit interactive laboratory. *International Journal of Electrical Engineering Education*, 53, 10 2015.

[33] Ailson Moura and Adriano Moura. Use of virtual industry and laboratory machines to teach electric circuit theory. *International Journal of Electrical Engineering Education*, 53, 02 2016.

[34] J. Hospodka and M. Koubik. Special web-based application for electric circuit analysis. In *Proceedings of the 10th WSEAS International Conference on E-Activities*, pages 140–145. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[35] L. Weyten, P. Rombouts, and J. De Maeyer. Web-based trainer for electrical circuit analysis. *IEEE Transactions on Education*, 52(1):185–189, 2009.

[36] A. Liberatore, A. Luchetta, S. Manetti, and M. C. Piccirilli. A new symbolic program package for the interactive design of analog circuits. In *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, volume 3, pages 2209–2212, 1995.

[37] G. Fedi, R. Giomi, A. Luchetta, S. Manetti, and M. C. Piccirilli. Sapwin 2.0: a symbolic software tool for educational purposes in analysis and synthesis of analog circuits. In *International Conference on Simulation and Multimedia in Engineering Education ICSEE'99*, pages 33–36, 1999.

[38] A. Luchetta, S. Manetti, and A. Reatti. Sapwin - a symbolic simulator as a support in electrical engineering education. *IEEE Transactions on Education*, 44(2), 2001.

[39] F. Grasso, A. Luchetta, S. Manetti, M. C. Piccirilli, and A. Reatti. Sapwin 4.0–a new simulation program for electrical engineering education using symbolic analysis. *Computer Applications in Engineering Education*, 44:44–57, 2016.

[40] B. J. Skromme, P. J. Rayes, C. D. Whitlatch, Q. Wang, A. Barrus, J. M. Quick, R. K. Atkinson, and T. S. Frank. Computer-aided instruction for introductory linear circuit analysis. In *2013 IEEE Frontiers in Education Conference (FIE)*, pages 314–319, 2013.

[41] B. J. Skromme, P. J. Rayes, B. E. McNamara, V. Seetharam, X. Gao, T. Thompson, X. Wang, B. Cheng, Y.-F. Huang, and D. H. Robinson. Step-based tutoring system for introductory linear circuit analysis. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2015.

[42] R. Smith and R. Dorf. Circuit laws. In *Circuits, devices and systems: a first course in Electrical Engineering*, chapter 2.2, pages 34–47. John Wiley & Sons, 5th edition, 1991.

[43] R. DeCarlo and Pen-Min Lin. Nodal and loop analyses. In *Linear Circuits: Time Domain, Phasor and Laplace Transform Approaches*, chapter 3, pages 107–154. Kendall Hunt, 3rd edition, 2009.

[44] C. Ho, A. Ruehli, and P. Brennan. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, 22:505–509, 1975.

[45] K. Lee and S. Park. Reduced modified nodal approach to circuit analysis. *IEEE Transactions on Circuits and Systems*, 32(10):1056–1060, 1985.

[46] L. M. Wedepohl and L. Jackson. Modified nodal analysis: an essential addition to electrical circuit theory and analysis. *Engineering Science and Education Journal*, 11:84–92, 1992.

[47] M. Shokouhifar and A. Jalali. Automatic simplified symbolic analysis of analog circuits using modified nodal analysis and genetic algorithm. *Journal of Circuits System and Computers*, 24:1–20, 2015.

[48] R. Pratap, V. Agarwal, and R. K. Singh. Review of various available spice simulators. In *2014 International Conference on Power, Control and Embedded Systems (ICPCES)*, pages 1–6, 2014.

[49] T. Tuma and Á. Buermen. Circuit simulation with spice opus. *Theory and Practice*, 2009.

[50] Texas Instruments. Spice-based analog simulation program - tina-ti, 2016.

[51] OrCAD. Orcad pspice designer, 2014.

[52] Synopsys. Hspice the gold standard for accurate circuit simulation, 2019.

[53] Linear Technology. Ltspice iv getting started guide, 2011.

[54] Cadence. Spectre simulation platform, 2019.

[55] E. H.-A. Gerbracht. Supernodal analysis revisited. *Proceedings of the International Workshop on Symbolic Methods and Applications in Circuit Design*, pages 113–116, 2004.

[56] R. R. Chen, A. M. Davis, and M. Simaan. Kirchhoff's voltage and current laws. In Wai-Kai Chen, editor, *The Circuits and Filters Handbook*, chapter 18.1, pages 536–573. CRC Press, 2nd edition, 2002.

[57] J. G. Gottling. Node and mesh analysis by inspection. *IEEE Transactions on Education*, 38(4):312–316, 1995.

[58] George E. Chatzarakis and Marina D. Tortoreli. Node-voltage method using 'virtual current sources' technique for special cases. *The International Journal of Electrical Engineering & Education*, 41(3):230–243, 2004.

[59] J. Keown. *PSpice and Circuit Analysis*. Prentice Hall PTR 16, 2nd edition, 1994. Pages 13–65.

[60] J. de Jong and E. Mansfield. Math.js: An advanced mathematics library for javascript. *Computing in Science Engineering*, 20(1):20–32, 2018. https://mathjs.org/.

[61] N. White. Algebra.js - build, display, and solve algebraic equations in javascript. https://algebra.js.org.

[62] M. Donk. Nerdamer - symbolic math for javascript. https://github.com/jiggzson/nerdamer.

[63] E. Eisenberg and S. Alpert. Katex - the fastest math typesetting library for the web. https://katex.org.

APPENDIX A
NVM - DIRECT MATRIX ASSIGNMENT

This appendix briefly outlines how the NVM can be alternatively applied by directly filling the indices in the matrix, as illustrated in Fig. 24.

$$\begin{bmatrix} G_{11} & -G_{12} & -G_{13} & \cdots & -G_{1K} \\ -G_{21} & G_{22} & -G_{23} & \cdots & -G_{2K} \\ & & \cdots & & \\ -G_{K1} & -G_{K2} & -G_{K3} & \cdots & G_{KK} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \cdots \\ U_K \end{bmatrix} = \begin{bmatrix} I_{11} \\ I_{22} \\ \cdots \\ I_{KK} \end{bmatrix}$$

Fig. 24. NVM - Direct matrix assignment

The first matrix (G) is filled with the Nodes *Conductance*. The main diagonal features the Conductances that are directly connected to each Node, e.g. G11 to Node 1, G22 to Node 2 and so on. The coefficients outside the main diagonal represent the "common" Conductances between two Nodes. For instance, a Conductance that is connected to both Node 1 and 2 should be in both positions (1,2) and (2,1) of the matrix.

The second matrix, which multiplies by the first one, is associated to the variables to be calculated — the Node voltages. The last matrix refers to the sum of all the currents flowing to the associated Node, their direction depending on the voltage source polarity, i.e. the voltage sources electromotive force's direction or the current direction of the current sources that directly influence the Node.

APPENDIX B
MODIFIED NODAL ANALYSIS

The MNA application to an electrical circuit results in a matrix equation represented in the following expression:

$$[A] \times [x] = [z] \tag{2}$$

The A matrix, decomposed in Eq. 3, is formed by four submatrices, each one containing information about the elements directly related to each Node and analysis parameters related to the existence of isolated Voltage and Current Sources. The x matrix holds the system variables i.e., the Voltages in the different Nodes and the Currents flowing through the isolated Voltage Sources. Finally, the z matrix includes the sum of all the known Currents and Voltages in the circuit.

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix} \tag{3}$$

As previously mentioned, the A matrix contains four matrices: G (identifies all the circuit connections), D (indicates if each Voltage Source is isolated or not, B and C (both include the Voltage Sources connections).

The G matrix, is the sum of the elements' Conductances linked to each one of the Nodes (in the first position of the diagonal (1,1) for Node 1, in the second position (2,2) for the Node 2 and so on). The remaining matrix positions are filled with the negative Conductances of branches connecting two different nodes. For a better understanding of this method, we take a simple circuit from Figure 25.

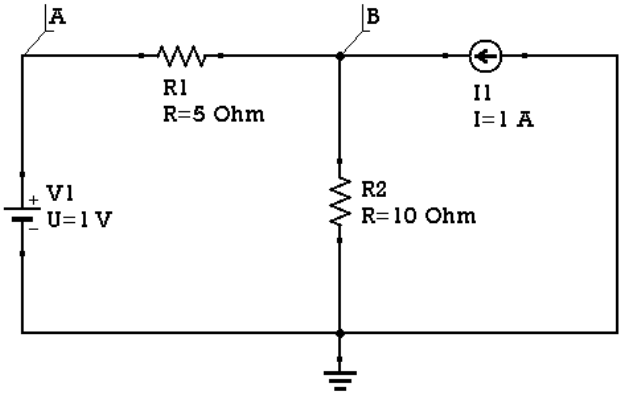


Fig. 25. Example circuit for demonstrating the MNA algorithm.

For this example, the G matrix for the Figure 4 circuit is as follows:

$$G = \begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} \\ -\frac{1}{R_1} & \frac{1}{R_1+R_2} \end{bmatrix} = \begin{bmatrix} 0.2 & -0.2 \\ -0.2 & 0.3 \end{bmatrix} \tag{4}$$

The D matrix would be null matrix, since there aren't any isolated Voltage Sources. The B matrix, which is also the transpose matrix of C, will have the values 0, 1 or -1 depending on the Voltage Sources' polarity. For the example circuit, the source $V1$ has the positive terminal connected to Node A, therefore the element of the B matrix associated to this Node will be 1, and the element referred to the second node will be 0 due to the nonexistence of a Voltage Source connected to the Node, as shown in the expression 5.

$$B = C^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{5}$$

Lastly, the x and z matrices are demonstrated in Eq. 6. The x matrix, as referred before, has the circuit variables.

$$x = \begin{bmatrix} V_A \\ V_B \\ I_{V_A} \end{bmatrix} \tag{6}$$

In the z matrix (displayed in Eq. 7), the first two lines belong to the Current Sources whose Current is flowing to the specific Node (I1 for Node B and none for Node A), the third line represents the Voltage Source potential.

$$z = \begin{bmatrix} 0 \\ I1 \\ V1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{7}$$

The circuit unknowns can be obtained by solving the matrix equation from Eq. 8.

$$\begin{bmatrix} 0.2 & -0.2 \\ -0.2 & 0.3 \end{bmatrix} \times \begin{bmatrix} V_A \\ V_B \\ I_{V_A} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{8}$$

This method allows to overcome the infinite Conductances problem caused by isolated Voltage Sources, but adds an extra equation to the algorithm. Despite that, its efficiency and universality made it stand out in relation to the classical NVM approach, particularly towards software implementation.