EUREKA - Electric Circuits sUccessful REsolution with Kaleidoscopic Aid

Matos. Isabel Filipa V., Almeida. Teresa Mª C. M. M.
filipa.matos@ist.utl.pt, teresamalmeida@ist.utl.pt
Department of Electrical and Computer Engineering
Instituto Superior Técnico

Abstract - EUREKA aims to be an educational software application in the service of students who are first presented to electrical and electronic circuits and their analysis. This application randomly generates a circuit and a question about one of the dimensions of that circuit. The student is led to use one of the techniques of circuit analysis to respond by choosing one of the answer choices given in multiple-choice format. One answer is the right one, having been found by the application using the results of the simulation of the circuit under the conditions of the question, using Ngspice. After choosing one of the answers, the student is immediately informed of whether or not it was the right answer. The student can previously choose which topics of circuit analysis and what level of difficulty he wishes to practice, which allows him to move forward, backwards, or jump over a topic as he prefers. It aims to give these students a training tool of the various techniques of analysis of electrical circuits, allowing a gradual evolution in the number of concepts and degree of complexity of the circuits and questions presented, respecting the rhythm of each student. EUREKA is free and globally available as a web application.

Index Terms—EUREKA, circuit analysis, didactic software, problem generator

I. INTRODUCTION

EUREKA is an online application that generates a virtually infinite number of exercising problems about electric circuit analysis. It is targeted to be used by students learning Circuit Theory for the first time, who can use it to practise the circuit analysis techniques they are introduced through the progress of a typical electrical engineering course.

Just as a kaleidoscope generates innumerable patterns, EUREKA creates countless different circuits for analysis. Such a tool may be seen as an interactive and dynamic exercise book with solutions that will help a student to consolidate his/her knowledge, and practice at his/her own pace until he/she masters each topic.

No predefined topic sequence is imposed to the student, who can navigate through the different topics at his/her pace.

Each problem is composed of a randomly generated circuit, a random question about one of the circuit’s electrical parameters, and a set of possible answers, in a multiple-choice format. After a student chooses one of the possible answers, he/she will immediately be told if that is the correct solution or what would have been the right option. This instantaneous feedback allows an immediate self-evaluation.

EUREKA computes the right answer using the results of the circuit simulation under the conditions established by the question to be answered. The simulation is acquired from an external simulator, NGSPICE [1], which is an open-source SPICE-based simulator. This simulator will be used not only to determine the final solution for the generated problem, but also, when necessary, to find electrical dimensions of the circuit that are needed for the problem generation itself.

As a PHP [2] online application, EUREKA is a portable studying tool that is mainly run on the server side. Its light and responsive interface allows EUREKA to be used on any kind of device with an internet connection and an internet browser.

II. CIRCUIT ANALYSIS CONTENTS OF EUREKA

Before actually starting to develop EUREKA in terms of its code, it was necessary to perform a systematic planning of the organization and contents of EUREKA. First, it was required do decide what circuit analysis topics were to be included and how they should be organized in the program. To meet the needs of the user as a student who is learning circuit analysis for the first time, it is necessary not only that each concept is clearly identified but, above all, that the organization of the concepts is consistent with the sequence in which the student will most probably learn the different concepts.

Using as reference several books that include at least a chapter dedicated to the basics of circuit analysis [3] [4] [5] [6] [7], and taking into account the stated goals of EUREKA, a list of the main concepts that were thought to be essential to include was established. This extensive list of concepts was then organized into eight main Topics. These main Topics correspond to a set of information with logical unity, as a chapter in a book.
The eight Topics covered by EUREKA are:

1. **Basic Concepts** - resistive circuits with constant sources;
2. **Expeditious Analysis** - the circuit analysis rules that allow the simplification of a bigger circuit into a smaller and more manageable one;
3. **Systematic Methods** – the circuit analysis techniques that systematize ways to find all the voltages and currents of any electrical circuit;
4. **Capacitors & Inductors** – circuits with resistor and capacitor or resistor and inductor connected to constant voltage or current sources;
5. **First-order Transient Analysis** – first-order transient analysis of circuits with one capacitor or one inductor and a step input;
6. **AC Steady-State Analysis** – circuits with a sinusoidal voltage source;
7. **Diodes and LEDs** – circuits with one diode or one LED (light emitting diode);
8. **Operational Amplifiers** – common elementary linear applications of operational amplifiers.

These eight Topics will each contain a number of different Concepts related to each other. In a similar way, Concepts are grouped into sets with logical unity. For example Topic 1, which regards circuit analysis basic concepts, is divided into three Concept sets:

- **Concepts A** – includes electric and current voltage, Ohm’s Law, voltage and current constant sources and power;
- **Concepts B** – includes everything from Concepts A and adds the Kirchhoff’s laws;
- **Concepts C** – adds the concept of controlled sources to all the concepts included in Concepts B.

Secondly, different difficulty degrees were considered and it was established and defined that there will be two different difficulty degrees of problems in order to make possible to generate very easy problems for beginner students and problems with a higher degree of difficulty for more experienced students in need of practice. Higher difficulty (Challenging mode) will result on more complex generated circuits for the problems, in comparison with the lower difficulty (Regular mode) ones.

Thirdly, it was necessary to set some definitions and ground rules for the generation of circuits that would make sense from a physical point of view and that are possible to simulate and solve manually in a reasonable amount of time.

The circuit generation must obey the following five rules:

- **A voltage source cannot be placed in parallel with another voltage source** - the voltage between the nodes between which the first source is placed is the parameter of that first source and cannot be forced to have a different value. The particular case when the voltage sources are equal will not present an exception to the rule. Dependent voltage sources are treated the same way, as an implementation decision.
- **A current source cannot be placed in series with another current source** – if the current of a loop is set by the first current source, it cannot be forced to another value by a second source. The particular case when the two sources are equal is ignored. Dependent current sources are treated the same way, as an implementation decision.
- **A diode cannot be placed in parallel with a voltage source** – if the diode is conducting and the ideal diode model is being considered, it would short-circuit the voltage source. In reality, this situation would irreversibly damage both components due to the high currents that would result in this low resistance loop.
- **If a diode can be part of a circuit, then the final circuit must have a diode** – even if there are enough other components that are possible to place in the circuit, a diode must be part of it to make sure that the generated circuit is fit to be part of a problem about diodes (this applies for Topic 7 that deals with diodes). Note that EUREKA only generates circuits with only one diode.
- **If a capacitor or an inductor can be part of the circuit, than the final circuit must have at least one capacitor or inductor** – similar to the previous rule, the circuit must have at least one capacitor or inductor for a problem about one of these components to be possible to be made (applies for Topics 4 and 5 that deal with capacitors and inductors).

Fourthly, a set of possible circuit topologies and characteristics was developed for each part of EUREKA, which will make sure that the generated circuits have a size in accordance with the knowledge level of the student at any given point. These topologies and feature will not jeopardize the randomness of the generated circuits, but only set some boundaries on the size of the circuits and their characteristics. In addition, the type and number of components to use in the generated circuits is related to the knowledge area that is being evaluated.

For example, Topic 1 treats resistive circuits with constant sources, and therefore the components that are accepted in a circuit for a problem of this Topic are DC voltage source, DC current source and resistors. Concepts A, the most basic group of Concepts, will include circuits with only two nodes and two branches on the Regular mode, but will include circuits with two nodes and three branches or with three nodes and three branches on the higher difficulty.

At last, it was indispensable to determine what kind of questions could be made in problems about each topic as one of the primary stages of EUREKA’s development, taking into account that the answers would be numerical or of type ‘yes’ or ‘no’. The type of question has to be in accordance with the concept to be evaluated.

The different types of questions are identified by an
acronym. As an example, the question types accepted in the three first three Topics are:

- **P**: the user is asked to find the power at stake in a given component of the circuit;
- **VJ**: the user is asked what is the voltage in a given node of the circuit with reference to the ground node;
- **VJK**: the user is asked to calculate the voltage difference between any two nodes of the circuit;
- **I**: the user is asked to determine the intensity of the current that flows through a given element, on a given direction;
- **X**: the user is asked to compute what is the parameter value of a component that is not provided (other relevant data is provided in order to be possible to find the parameter value of the component);
- **REQDC**: the user is asked to find what the equivalent resistance a net is, seen by a given component of the circuit.

III. STRUCTURE OF EUREKA

EUREKA’s structure is based on eight different interconnected programming modules, represented on Fig 1. External intervenent agents are represented with dashed lines: the user and the external circuit simulator.

The Data Manager module may be thought of as the heart of the program. It coordinates all other modules, being responsible for the correct sequence of events. It loads information from three different databases, and coordinates the execution and the data flow between modules.

The execution of EUREKA will be dependent on the information retrieved from databases that store static information that is needed for formulating a problem. Three different databases store this information: one for circuit topologies accepted through each stage of the program (dependent of the user’s choices), another one with templates of the questions that can be made about each circuit, and a third with basic configuration parameters.

The Circuit Generator is the module that will randomly build the netlist of the circuit according to a set of rules for the maximum complexity allowed for the circuit for the chosen circuit analysis topic and the set of components that can be included in the circuit. It must verify the physical coherency of the circuit, making sure it is solvable.

In the Question Generator one of the possible type of questions are picked from the information retrieved from the correspondent database and the electrical parameters that will have to be calculated are determined. The final question is prepared to be sent to the Interface, after being adapted to correspond to the generated circuit (it asks about one specific characteristic or electrical parameters of that particular circuit). It also saves the information about the relevant information about the question that will define how the Solution Finder module will be able to determine the right answer for the problem.

The Solution Finder is responsible for creating a correct input to the simulator, with the circuit description and the proper type and description of the analysis to be performed. It will then collect the simulator’s output, which is a list of values for the circuits’ voltages and currents. At last, the results need to be manipulated and computed in order to find the right numeric solution for the question that was created and asked to the user.

Using the correct numeric answer, the Multiple Choice Generator module will create a set of other answers to present to the user. These will be variations of the correct answer with one or more differences from it, like different algebraic signal, different numeric value and different scale factor.

Using the generated circuit’s netlist, the Circuit Representation module builds a readable and correct graphical representation of the circuit to be sent to the Interface and presented to the user as part of the generated problem. This representation has to respect the exact topology and component parameters and electrical variables that were stipulated by the generated circuit netlist.

The Interface module receives the user’s input that reflects the user’s choices of Topic, group of Concepts and difficulty degree, or, later, the user’s answer to the question about a circuit or other user request. The Interface interprets this input and transmits it to the Data Manager module. It is also responsible for the graphical interface that is presented to the user at any given moment, building it according to the information communicated by the Data Manager.

IV. DEVELOPMENT OF EUREKA

The development of EUREKA included not only the
programming of each module, as well as the strategies that were used to make each module able to deal with the several circuit analysis challenges that each new Topic represent.

A. Databases of EUREKA

The three different databases in which the execution of EUREKA is based on are stored using a format in accord to the Extensible Markup Language, XML [8]. The types of questions that are accepted to each Topic and group of Concepts are disposed in the questions database. The different circuit boundaries in terms of number of nodes and branches, type of components allowed and number of components of each type, that are restricted by the choice of Topic, group of Concepts and Difficulty degree, are registered in the topologies database. The third database has basic EUREKA configurations that control some parameters inside the program.

B. Circuit Generator

The circuit generation must meet the following specifications:

- the topology of the circuit to be generated must be picked from between one of the entries of the set of topologies stored in the Topology database for the chosen Level - the number of nodes, n, and branches, b, of the circuit is specified by the randomly chosen topology;
- the types of the components that will be part of the circuit and their number are defined by the same entry of the Topologies database;
- there must be no physical impossibilities in the circuit that would prevent it from being realizable.

The circuit generation for Topics 1 to 7 is performed component by component. After a topology is randomly picked for the circuit, the components, which are also picked randomly from the pool of components allowed for the circuit, are placed, one by one, taking into account that every node of the circuit has to represent the connection of at least two components. After every node is connecting two components (which represents a closed loop integrating all the nodes of the circuit), components start to be placed between any two random nodes of the circuit until there are no more components left.

In order to follow the circuit generation rules stated above, after a component is randomly picked, it is verified if that component type on particular can be placed between the two nodes assigned for it. If, for example, the component to be placed is a voltage source, and there is already another voltage source placed on the circuit, between the same two nodes, the new component has to be replaced by a different type of component to be allowed to integrate the circuit on that place. This process is recursive until a compatible component type is found for that place in the circuit.

Topologies of circuits with operational amplifiers (Topic 8) are fixed. They are six common circuits using operational amplifiers:

- voltage follower;
- inverting amplifier;
- non-inverting amplifier;
- summing amplifier with 2 inputs;
- summing amplifier with 3 inputs;
- differential amplifier.

All the parameters of the components included on the randomly chosen circuit are also randomly chosen, providing an endless number of combinations.

C. Question Generator

After having generated a random circuit adequate for the Level chosen by the student, EUREKA will set a question for the problem it is building. This question will not only be generated to agree with the chosen Topic and group of Concepts, but with the generated circuit as well. Even though there is a bank of questions for each Topic and group of Concepts, these questions include fields that need to be replaced by parameters of the circuit, like the name of a component, the names of nodes of the circuit, time instants, etc.

The Question Generator module proceeds differently according to the type of question it picks from the database. Different questions need different pre-processing before they are validated and ready to be made. Some types of questions even need special treatment, like defining the time constant to generate a logical problem on first-order transient analysis, which means that the circuit simulator is used to set the question. Other questions need additional settings that were not yet specified when building the netlist of the circuit.

D. Solution Finder

Finding the answer to the question that was generated for a given problem greatly depends on the Topic and question type. Generally, it is a process that can be divided into three main steps:

1) Setting the netlist of the circuit to be simulated, writing it into a .cir file and simulating it, all according to the NGSPICE specifications [1];
2) Reading the simulator’s output file and loading it into a program variable, built to be easily manipulated and searchable;
3) Computing the answer: post-processing the simulation results in order to find the answer to the question, that is then sent to the multiple-choice generator.

Some question types do not need all these steps, and some
require some additional computing. A great number of variables must be established and are combined by EUREKA to complete each problem formulation: the Topic, the Concepts group, the Difficulty degree and the question type. The Solution Finder must take into account all these variables, except the Difficulty degree.

The Solution Finder module is subdivided into two main blocks:

- **the Circuit Simulation block** - builds the list of relevant electric dimensions based on the chosen Topic and Concepts as well as the question type;

- **the Answer Computation block** - uses this list to find the value of the answer, which may be numeric or Boolean, based only on the type of question

All the information needed to solve each problem is already defined by the Circuit Generator as well as the Question Generator and will be used by both of these stages.

**E. Multiple-Choice Generator**

The Multiple-Choice Generator is the module of EUREKA responsible for creating a set of possible answers for the generated problem, one of which is the right one, to present to the user. The user has to choose the option that he/she thinks is the correct one.

After the correct answer to the problem is found by the Solution Finder module, the other multiple-choice answers are generated one by one, following an established set of rules. This module uses several of the parameters that are fixed on the Configuration database, which are easily controlled by the developer at any time without having to modify any code.

The first step on this module is to find the scale prefix (i.e. k, m, µ, etc.) and proper unit for the correct answer. This answer is saved as the correct one to be compared to the user’s choice later, and is added to an array of choices to be presented to the user, the other being variations of the correct one: different algebraic signal, incorrect unit or scale, and/or different value.

**F. Circuit Representation**

When presenting a problem about a specific circuit to a student, one of the most important factors to consider is the circuit graphical representation itself. It is the only way to communicate to the student what the circuit topology is and how the components are connected, which component is which, and what node is the question referring to. This representation must be clear and simple, yet contain all the information that is needed for the successful problem resolution.

To keep the representation clear and the most legible possible, the component parameter values are given to the user on a small list under the question itself. This releases the drawing of the circuit of additional numbers and symbols, allowing the circuit to be clearly represented on a smaller area. Fig. 2 shows an example of the representation of a circuit generated by the Circuit Representation module of EUREKA.

![Example of a circuit represented by EUREKA.](image)

EUREKA represents all the topologies of the circuits that are part of the Topologies database, taking into account that for each topology there can be several ways to connect the components, and the components themselves can be of several different types. This results on an infinite number of different representations.

**G. Data Manager Module**

The Data Manager Module is responsible for the coordination of the other modules, and is the link between EUREKA’s internal algorithms and the information presented and retrieved from User’s Interface.

First, this module will open a unique execution session, which allows preserving certain data intact during subsequent accesses. A unique session identification string is created and used, for example, as the filename of the simulation text files. This way, if more than one user is connected at the same time, their sessions will be independent and the files created for one’s session will not be used by another.

The second step is loading the information saved on the different databases for internal variables that will be easier and faster to use during the program’s execution. The Topologies database includes information that is sent to the Interface module to build the homepage of EUREKA: the Topic titles, Concept group titles and descriptions are retrieved from this database and used to generate EUREKA’s menus.

After this is done, a cycle begins: the user’s choice of Topic, group of Concepts and Difficulty degree are retrieved and a problem for that combination of choices is generated. The first module that is called during the problem generation is the Circuit Generator that will create the circuit netlist and save it on the session private global variable to be used throughout the entire execution.

The Question Generator is than called and returns the question array, ready to be sent to the interface. In addition, this module will add to the global variable of the session the
relevant information about the dimension that will be calculated ahead. If a question was found for the topology of the randomly generated circuit, Multiple-choice Generator module is called to create the array of answers to be part of the multiple-choice response. This module, in turn, calls the Solution Finder module to compute the correct answer.

Finally, the Data Manager calls a function that will determine what information about the circuit’s components needs to be given to the user to find the problem’s solution.

When the Data Manager Module receives the user’s choice of answer from the Interface module, it will compare it with the correct answer and send the result of that comparison to the Interface, to inform the user about the correctness of his response.

H. User Interface

EUREKA’s Interface is the only means of communication with users, making it of paramount importance to transmit in an efficient and clear way all the information the users will need to understand how the application works, what is expected of them to choose, and all the information that composes the problem.

The step-by-step application structure of EUREKA allows to divide the user flow into different screens, each dedicated to showing a very specific piece if information to the user in a prescribed sequence: the menu of available Topics and Concept Levels and each problem with its circuit, question, circuit data and multiple-choice answers. This particular characteristic allowed the Interface to be developed using a Wizard like pattern [9], where users are asked to make only a few easy choices at a time.

The Interface module was programmed using HTML and CSS [10], both nowadays widely used description languages, as well as in JavaScript [11], which allows having browser-run scripts. It is based on a Gumby framework [12], which consists on a free and open-source CSS framework that can be easily adapted to build any type of interface. The advantages of using this framework, above which the interface is built, is that it is entirely customizable and is fully responsible, adapting the layout to any screen size including mobile devices, besides having a large set of styles and common interface elements, like buttons, tabs and forms, that can be directly used, customized or replaced [12].

In terms of page layout, the four Gestalt Principles (proximity, similarity, closure and continuation) were taken into account. These principles describe the way people tend to naturally perceive the hierarchy of visual elements organized into groups [9]. In particular, the proximity and the similarity principles are very important for understanding EUREKA’s user flow almost instinctively.

V. Example

The home page of EUREKA asks the student to choose a Topic from the list on the left, which is shown on Fig. 3.

After choosing the Topic, the student must choose the group of Concepts he/she wants, as well as the difficulty degree, as shown by Fig. 4.

The problem is then generated, and is composed by a circuit as the one shown by Fig 2. along with a question, information about the circuit’s parameters, and the multiple choice answers, as represented by Fig. 5.

After submitting his/her choice, the student is immediately told if the answer is correct (Fig. 6) and incorrect (Fig. 7). In the last case, the student has the option of seeing what the correct answer is, or going back to the problem to try again.

After submitting his/her choice, the student is immediately told if the answer is correct (Fig. 6) and incorrect (Fig. 7). In the last case, the student has the option of seeing what the correct answer is, or going back to the problem to try again.
Finally, the user has the option of asking for Help when solving a problem. The Help option shows an exemplificative problem of the same Topic which is solved using the circuit analysis techniques that are being studied on that Topic.

VI. CONCLUSIONS

The greatest goal of EUREKA is to provide students who are in the beginning of their journey of learning Circuit Analysis Theory with a free and easily accessible educational tool. This tool should allow these students to exercise the various circuit analysis techniques that will gradually be presented to them during their learning process, always offering different problems on the topic that is chosen. This implicates a thorough organization of all the techniques and theoretical concepts that a student will find along his/her path and an anticipation of the small steps a student will need to take in order to gradually and firmly build his/her knowledge of Circuit Analysis.

EUREKA’s accessibility is guaranteed by its online nature: being an online application that is fully run on a remote server and the user’s browser, allowing everyone to use EUREKA with virtually any type of machine with an internet connection and a browser.

The random generation of circuits and problems proved to be a challenge, as every circuit and problem had to follow a set of rules to be viable and possible to solve. A set of limiting boundaries had to be defined in order to guarantee coherence on the level of knowledge expected from the students on each stage of their path and well-formed generated circuits that are possible to be solved.

The problems generated by EUREKA are divided into different sets of Topics, like chapters of a textbook. The contents of each Topic and their order tries to follow the typical progress of Circuit Analysis courses and reference books, each adding some new notions to the previous one. In order to be possible to gradually progress inside a Topic, each Topic is sub-divided into smaller groups of Concepts. This strategy allows the user to progress in small steps, and tries to give students the chance to exercise each small new notion that they will learn in each Circuit Analysis class during a semester without letting that too many concepts accumulate before being able to follow to the next level.

Having the need to revise old concepts in mind, for example, when reviewing for an exam, EUREKA has a Challenging Mode. This mode is dedicated to those students who are already familiar with the concepts included in past Topics and groups of Concepts but need to revisit them and practice them, in order to become agile and improve their performance solving circuits of those levels.

EUREKA is thought to be seen like a challenging educative tool, almost like a game. As many games, the user is first asked to choose a level and difficulty degree (as in the Easy, Normal and Hard levels of many computer and smartphone games) and then presented with a challenge he/she has to solve. The interface is clean, appealing and inviting, giving the user all the information he/she needs to solve the “puzzle”. The answer is multiple-choice, making it very easy to input the found answer, even when using EUREKA on a touchscreen device. The user receives immediate feedback on his/her performance, being that if the answer is wrong he/she can choose to see what the correct answer is or not. It is always possible to go back to the “puzzle” and try to solve it again.

EUREKA as it is can be used as an infinite resource both for students as well as for professors who can use it as a source of simple problems for worksheets and class examples. It is able to serve its target users because its free, available online and its interface is in English. Although, it has a great potential to grow even further.

REFERENCES