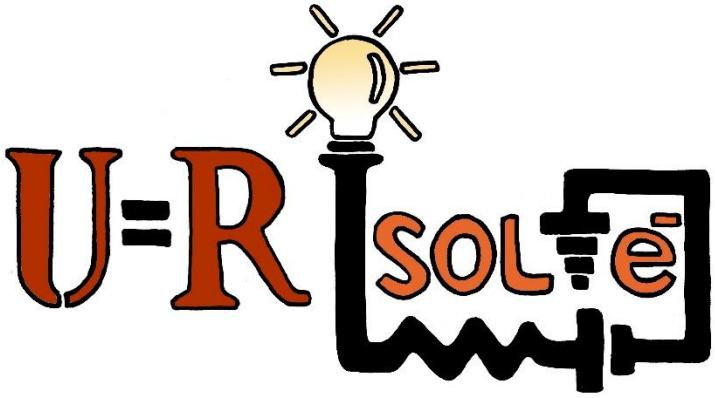
APLICAÇÃO PARA GERAÇÃO DA METODOLOGIA/EQUAÇÕES DO MÉTODO DAS CORRENTES NAS MALHAS, A PARTIR DO QUCS



Miguel Ângelo da Rocha Duarte

isep Instituto Superior de Engenharia do Porto

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2019

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores

Candidato: Miguel Ângelo da Rocha Duarte, Nº 1131201, 1131201@isep.ipp.pt
Orientador: Mário Jorge de Andrade Ferreira Alves, mjf@isep.ipp.pt
Coorientador: Francisco José Dias Pereira, fdp@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
5 de setembro de 2019

Agradecimentos

Exprimo nesta secção os meus sinceros agradecimentos a todos aqueles que, de alguma forma, contribuíram para a realização deste projeto:

À minha namorada e companheira, pelo seu apoio incondicional, por todos os sacrifícios e por acreditar sempre em mim.

À minha família pelo carinho e apoio incansável desde o começo até ao último segundo na minha formação.

Aos orientadores do projeto, Professor Mário Alves e Professor Francisco Pereira, que se mostraram sempre disponíveis para me ajudar.

Ao colega Lino Sousa, pela sua disponibilidade e ao colega Joshua Soares pela ajuda na implementação do código.

Resumo

O projeto apresentado neste documento consiste numa aplicação de análise de circuitos elétricos a integrar a plataforma "U=RISOLVE", voltada para o âmbito educativo, sobretudo a alunos que iniciam o seu percurso académico em Engenharia Eletrotécnica (EE). É uma ferramenta que visa ajudar o aluno na aprendizagem de circuitos elétricos em Corrente Contínua e Corrente Alternada através do Método das Correntes nas Malhas (MCM).

A cada simulação de um qualquer circuito desenvolvido no *software* de simulação *Quite Universal Circuit Simulator* (QUCS), é gerado um ficheiro de texto, denominado *netlist*, com toda a informação referente aos componentes do circuito, bem como as ligações entre si. Deste modo, é possível ao utilizador enviar para a plataforma a *netlist* do seu circuito, dando lugar ao processamento da informação e apresentação das respetivas soluções resultantes da análise do MCM.

Assim como a aplicação para o Método das Tensões nos Nós (MTN), integrante na plataforma, a aplicação do MCM diferencia-se pela forma didática e de fácil leitura, como expõe a resolução do seu método, expressando o número de Nós, Ramos e Malhas possíveis no circuito, ao contrário das várias ferramentas de simulação de circuitos, que apenas devolvem ao utilizador os valores, sem qualquer clarificação ou exposição de uma possível resolução.

O processo da geração das Malhas baseia-se na elaboração de combinações de Ramos do circuito e consequente teste dessas mesmas combinações, verificando se correspondem a uma malha fechada ou não.

Ao se tratar de uma plataforma desenvolvida em *HyperText Markup Language* (HTML), o *script* para análise computacional de circuitos elétricos foi implementado em JavaScript (JS), usando as funcionalidades da versão ES6. Esta linguagem de programação de alto novel permite que a análise seja efetuada em *browser*, o que simplifica e cativa a sua utilização.

Abstract

The project presented on this document consists of a circuit analysis application to integrate the platform "U = RISOLVE", aiming the educational scope, mainly students who are beginning their academic journey in Electrical Engineering (EE). It is a tool that was designed to help in the learning process of electrical circuits in direct current and alternating current, using the Mesh Current Method.

Every simulation of a circuit developed in Quite Universal Circuit Simulator (QUCS) creates a text file called netlist, with all the information regarding the circuit components, as well as the links between each other. This allows the user to send a netlist of their circuit to the platform described on this project, processing the information and presenting Mesh Current Method analytics solutions.

In association with the application of the Node Voltage Method, which is part of the platform, the application of Mesh Current Method differentiates from the Node Voltage Method in the didactic form that presents results. Not only does it expose the solution of the method but also reveals the number of nodes and possible meshes on the circuit, unlike various circuit simulation tools, which return the values to the user without any clarification or possible resolution.

The mesh generation process is based on the creation of combinations between the different branches of the circuit and consequent tests of these combinations, verifying if it corresponds to a closed loop or not.

When dealing with a platform developed in HyperText Markup Language (HTML), the script for computational analysis of electrical circuits was implemented in JavaScript (JS) using the features of the version ES6. This high-performance programming language allows an analysis to be accomplished on the browser, what simplifies and captivates its use.

Keywords

Mesh Analysis, Kirchhoff Laws, Mesh Analysis, Mesh Equation, Circuit Simulator, QUCS, Browser App, JavaScript, HTML, Electrical Engineering.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

API – Application Programming Interface

CSS – Cascading Style Sheets

EE – Engenharia Eletrotécnica

GPL – GNU General Public License

GUI – Graphical User Interface

HTML – HyperText Markup Language

JS – JavaScript

LKM – Lei de Kirchhoff das Malhas

LKN – Lei de Kirchhoff dos Nós

MCM – Método das Correntes nas Malhas

MNA – Modified Nodal Analysis

MTN – Método das Tensões nos Nós

PCB – Printed Circuit Board

QUCS – Quite Universal Circuit Simulator

SPICE – Simulated Program with Integrated Circuits Emphasis

W3C – World Wide Web Consortium

SI - Sistema Internacional de Unidades

1. INTRODUÇÃO

Neste capítulo introdutório serão divulgados os principais objetivos e motivações deste projeto. Expõem-se, também, as contribuições a que esta iniciativa se destina e quais as suas finalidades.

No fim do capítulo, será evidenciada a cronologia dos trabalhos e explicada a organização do relatório.

1.1. CONTEXTUALIZAÇÃO

O projeto aqui apresentado despontou da necessidade de melhorar e complementar a metodologia de ensino existente em Engenharia Eletrotécnica, pretendendo ajudar a diminuir a dificuldade sentida pelos alunos na aprendizagem de metodologias de análise de circuitos. Num primeiro impacto com circuitos elétricos, os alunos tendem a complexificar o conteúdo aprendido, causando-lhes algumas dificuldades na resolução de problemas com circuitos. Alguns obstáculos comuns passam por identificar corretamente Ramos e Nós, sentidos de Correntes, polaridades de Fontes de Tensão e Fontes de Corrente quanto ao sentido da Malha em questão, sentidos Correntes em Malhas Independentes, revelando-se uma complicação maior quando lhes é exigido transpor as variáveis para uma equação de um Nó ou de uma Malha.

No âmbito de continuar a complementar o trabalho relativo ao simulador de circuitos QUCS, elaborado por outros estudantes [1][2][3][4], será acrescentada a possibilidade da análise de circuitos elétricos pelo Método das Correntes nas Malhas (MCM). Esta

ferramenta estará incorporada na aplicação U=RISOLVE. Deste modo, a aplicação passará a contemplar dois métodos de análise de circuitos distintos na mesma interface, tendo o utilizador a opção de analisar o circuito pelo Método das Tensões nos Nós (MTN) e pelo MCM.

Motivada pela falta de explicação de métodos de análise de circuitos, a aplicação U=RISOLVE vem oferecer ao meio uma solução estruturada e de leitura acessível de todos os passos necessários para a obtenção do resultado, ao contrário das ferramentas de simulação, que apenas se limitam a exibir as soluções, sem qualquer menção ao seu método de resolução.

1.2. OBJETIVOS

O principal objetivo é a criação de uma ferramenta de ensino e de autoaprendizagem que possa ser útil tanto para professores, no caso de exemplificação em aula e preparação de exercícios, como para alunos, no caso de aprendizagem do método das Correntes nas Malhas e verificação de exercícios teóricos e práticos. Esse propósito foi dividido pelas seguintes tarefas:

- Integrar a aplicação U=RISOLVE, já com o MTN desenvolvido, na mesma plataforma;
- Reutilizar e melhorar algoritmo de aquisição de dados;
- Desenvolver método didático para visualização do MCM passo a passo;
- Aprimorar *layout* da plataforma e página de resultados.

O facto de, até ao momento, não existir nenhuma ferramenta que clarifique e demonstre a análise de um qualquer circuito pelo MCM, torna-se revelador do grau da complexidade da questão em estudo.

Este projeto, como mencionado anteriormente, dedica-se ao estudo do MCM e dele resultam as equações das Malhas, assim como o valor das suas Correntes, no caso da análise em Corrente Contínua. Ao contrário do MTN, que inspirou a criação do método da análise nodal modificada – *Modified Nodal Analysis* (MNA) – usado como base nos algoritmos de análise de circuitos em simuladores, como por exemplo o QUCS, o MCM

não é utilizado em nenhuma forma de algoritmo base de análise devido à sua arbitrariedade quanto à definição de Malhas e dos seus sentidos. Consequentemente, haverá um maior desafio em construir de raiz um algoritmo capaz de construir Malhas, validá-las, e selecioná-las automaticamente, sendo essencial conceber um método para tornar o *script* autocrático.

1.3. CONTRIBUIÇÕES

O projeto realizado contribui de forma clara como complemento ao ensino em EE, servindo de ferramenta didática de apoio ao estudo, incentivando a autoaprendizagem na validação de resultados obtidos, com maior rapidez e menor probabilidade de erro. Para além da principal operação da geração de Malhas, o aluno obtém igualmente informações úteis como o número de Nós, de Ramos, de Malhas (M) possíveis no circuito e de forma intuitiva, o número de Malhas necessárias para a resolução pelo MCM, apresentando a sua fórmula. É de realçar a importância do cálculo de número de Malhas possíveis visto que o seu cálculo é de difícil obtenção, ou seja, não existe nenhuma equação que o defina. Apenas uma aplicação como a desenvolvida sabe o número máximo de Malhas possíveis.

Em síntese, são expostas de seguida as contribuições deste projeto:

1. Adição de funcionalidade à aplicação U=RISOLVE:

    - Apresentação do número de Ramos (R), Nós reais (N) e Fonte de Corrente (FC) do circuito, para resolver a fórmula Malhas(M);
    - Cálculo do número de combinações de Malhas possíveis;
    - Apuração do número de Malhas/equações Independentes e número de Malhas Auxiliares;
    - Validação e seleção de conjuntos de Malhas adequados à resolução do circuito em questão;
    - Programação de estratégia otimizada de arbitrariedade na apuração de sentidos para as Correntes de Malha;
    - Constituição de equações de Malha pela manipulação de *strings*;

- Resolução numérica de sistemas de equações;
- Deteção de erros nas ligações do circuito pela *netlist*;
- Adaptação de aquisição de dados à versão do alterada do QUCS;
- Apresentação de resultados em formato *LaTeX*;

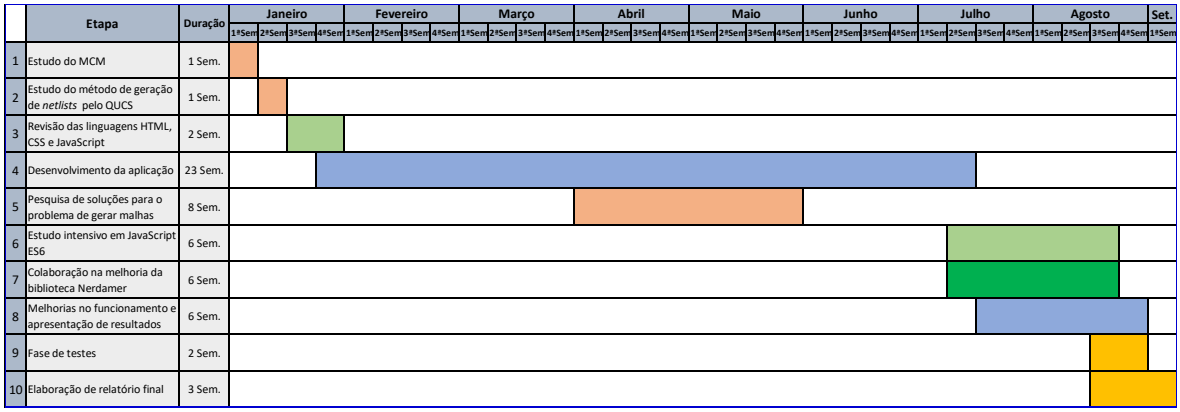
2. Constatação do facto de uma Malha ser capaz de conter no máximo o número de Nós do Circuito, o qual não consta na definição clássica de Malhas elétricas;

3. Participação na melhoria do *Nerdamer*, biblioteca de operações matemáticas baseada em JavaScript. Sendo esta a biblioteca que a aplicação se apoia na resolução de sistemas de equações, no desenvolvimento do projeto, quando se tentava resolver sistemas com complexos, descobriu-se um erro na biblioteca, não previsto na documentação da mesma. Em colaboração com o criador da ferramenta, foi possível desenvolver uma versão melhorada, embora continue limitada a sistemas de equações mais simples, usada sobretudo na análise de circuitos em Corrente Contínua.

1.4. CALENDARIZAÇÃO

Este projeto foi concebido ao longo de trinta e três semanas, divididas em várias tarefas apresentadas na Tabela 1. O trabalho começou pelo estudo do MCM, de modo a entender qual seria a melhor abordagem ao problema. Posteriormente foi essencial perceber o modo de funcionamento do QUCS relativamente à produção de *netlists*. Em seguida, foi fundamental relembrar as ferramentas usadas na implementação da aplicação e, por fim, desenvolver o algoritmo de geração, seleção e análise de Malhas, assim como a interface gráfica de apresentação de resultados.

Tabela 1 Calendarização do projeto



1.5. ORGANIZAÇÃO DO RELATÓRIO

O relatório é constituído por seis capítulos, que descrevem todo o desenvolvimento do projeto. Esta distribuição dos capítulos procura contribuir para a perfeita compreensão da pesquisa e trabalhos efetuados no desenvolvimento da aplicação.

O primeiro capítulo expõe as motivações que levaram à realização do projeto, as funcionalidades que este deverá ter, as suas contribuições do ponto de vista científico, tecnológico e didático e a ordem como as várias tarefas foram praticadas para a realização do mesmo. Desta maneira, são mostradas, no diagrama de Gantt, a calendarização e a duração de cada encargo.

O segundo capítulo cobre o tema dos simuladores de circuitos elétricos, apontando as melhores ferramentas para o efeito disponíveis na atualidade, contando com uma tabela comparativa, evidenciando a escolha do QUCS como ferramenta ideal a usar em ambiente académico.

O terceiro capítulo baseia-se na demonstração das etapas a percorrer na análise de circuitos pelo Método das Correntes nas Malhas

O capítulo quatro e o capítulo cinco são reservados ao funcionamento da aplicação, da abordagem ao problema de geração de Malhas, método de apresentação de resultados e demonstração de resultados com exercícios práticos.

No sexto e último capítulo efetuam-se as considerações finais, mencionando metas alcançadas tal como como as dificuldades que se foram manifestando ao longo da realização do projeto.

2. SIMULADORES DE CIRCUITOS ELÉTRICOS

A escolha de simuladores de circuitos elétricos é, nos dias de hoje, bastante ampla. Constituindo uma ferramenta fundamental em várias áreas, no ensino estas são imprescindíveis. Este capítulo irá enumerar alguns dos simuladores mais influentes no mercado, tanto pagos com *Open Source*, tentando descrever as principais razões para a sua utilização.

2.1. SIMULADORES DE CIRCUITOS ELÉTRICOS EXISTENTES

Analisando os mais populares simuladores existentes, é possível comparar entre si as suas maiores vantagens e desvantagens. Devido à natureza do trabalho, os tópicos de estudo serão orientados numa perspetiva de utilizador académico, de forma a instruir os alunos na seleção do melhor simulador que servirá de ferramenta de autoaprendizagem.

Os parâmetros de análise reincidem nas plataformas onde estas ferramentas estão disponíveis, na que forma como são disponibilizadas e qual a sua licença, isto é, se a aplicação é paga – *software* proprietário -, se oferece algum tipo de versão *trial*, se se trata de *freeware* e se é *Open Source*. Também se expõe o tamanho das bibliotecas de componentes e modelos em cada um dos simuladores, a sua interface e o suporte da

comunidade de cada uma na iniciação à ferramenta, assim como a qualidade da sua documentação oficial.

CircuitLab

O *CircuitLab* [6] é um simulador em *browser*, o que exige ligação à Internet para ser usado. Oferece uma versão gratuita para fins não comerciais muito completa com alguns limites no que toca à simulação de circuitos mais complexos por apresentar uma biblioteca menos rica comparativamente com outros simuladores. Por cerca de 22€ por ano, é possível usufruir de mais funcionalidades, podendo o utilizador aceder a complementos como documentação variada e tutoriais gratuitos. Dispõe ainda de um fórum para a comunidade de utilizadores expor possíveis *bugs* e eventuais adversidades. A Figura 1 mostra como é composto o *layout* da ferramenta.

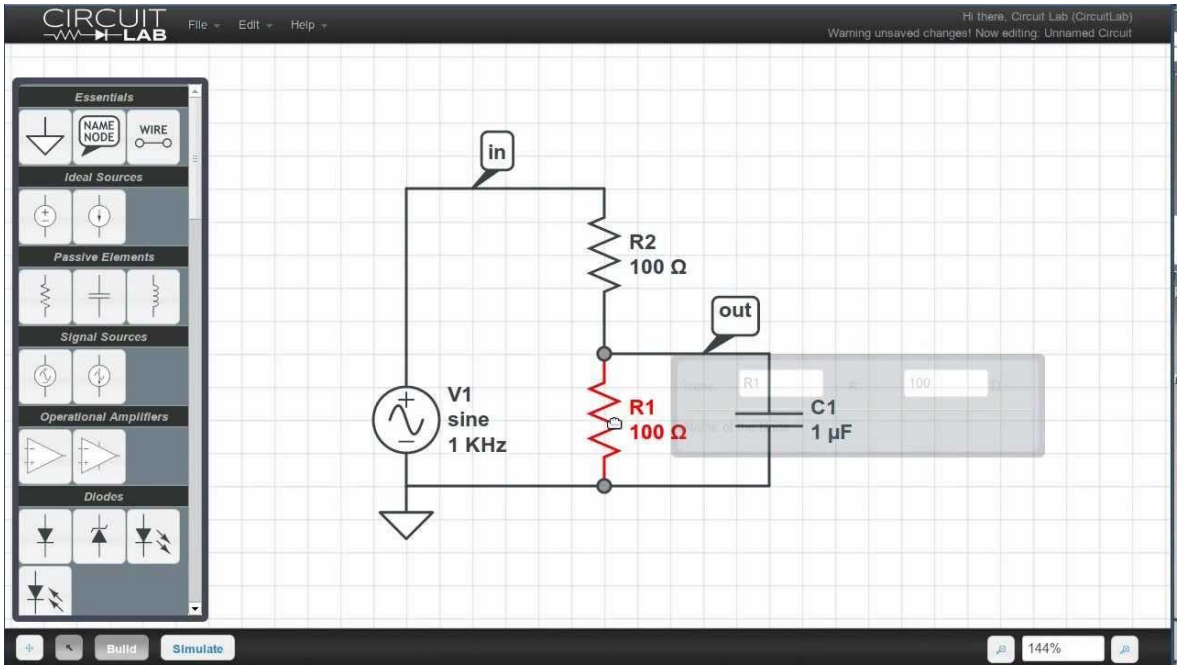


Figura 1 Exemplo da aplicação *CircuitLab* [6]

EasyEDA

Com a sua primeira versão estável lançada em 2017, o *EasyEDA* [7] é um simulador, como demonstrado na Figura 7, simples e interativo que está acessível tanto como programa para Linux, macOS e Windows, como plataforma *web* em *browser*. Com licença para uso comercial *free*, a sua utilização é gratuita, embora com anúncios, e já conta com mais de 800 mil utilizadores. Atualizada todas as semanas pela sua equipa, tem várias bibliotecas de componentes e módulos de circuitos *standard*, e ainda permite a conversão do esquemático produzido em formato PCB. O *software* ainda permite importar arquivos nos formatos de outros simuladores, o que permite uma verificação rápida e cómoda de qualquer ficheiro de esquemático de circuitos. Oferece além disso, uma ampla lista de reprodução de vídeos tutoriais e toda a documentação explicada intuitivamente para incentivar a iniciação ao seu uso

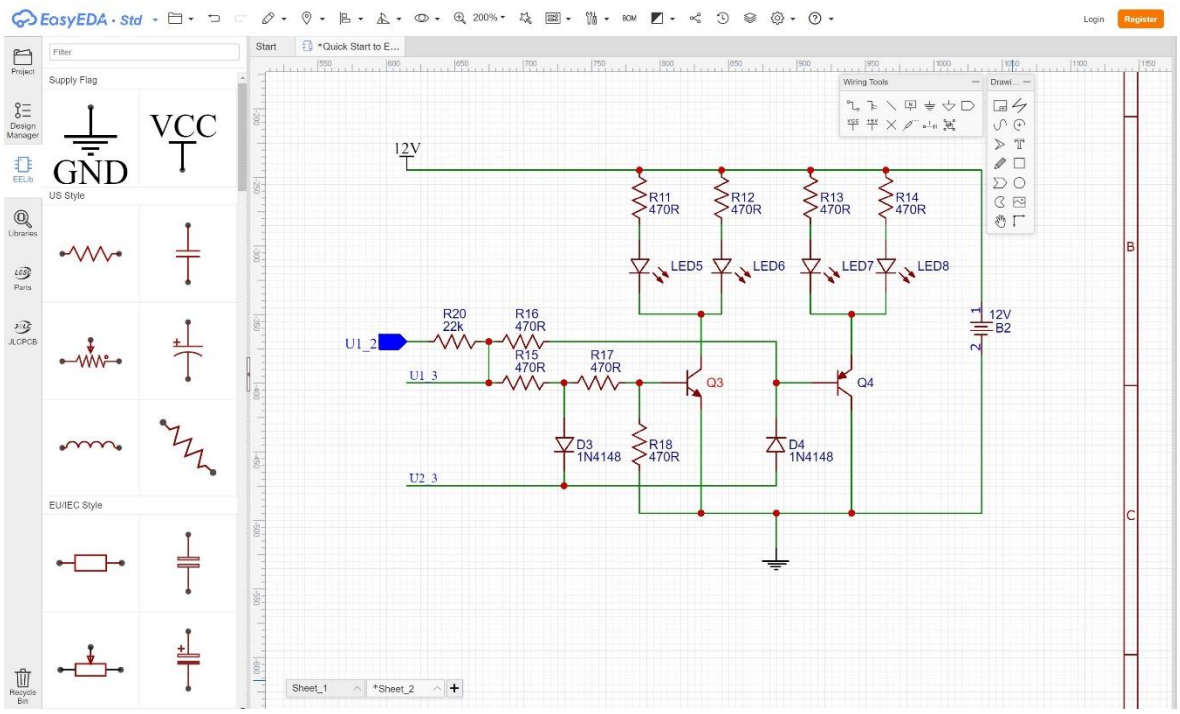


Figura 2 Exemplo da aplicação *EasyEDA* [7]

EveryCircuit

O *EveryCircuit* [8] é provavelmente a aplicação mais usada em plataformas móveis visto ter sido inicialmente desenvolvida para iOS e *Android*, numa altura em que não existia praticamente opções do género. Posteriormente, lançaram uma versão compatível com o *browser Google Chrome*, possibilitando desta forma uma maior transversalidade no seu uso. Contempla uma versão gratuita exclusivamente em plataformas móveis, para uma utilização com recurso a componentes básicos e a circuitos simples. Sendo uma aplicação de licença proprietária, tem um preço único de 15€, e nesta modalidade já é permitido ter acesso a bibliotecas mais completas. Como demonstrado na Figura 3, destaca-se pela sua apelativa interface gráfica, com visualização dinâmica e em tempo real de gráficos de medição e sentidos de Corrente.

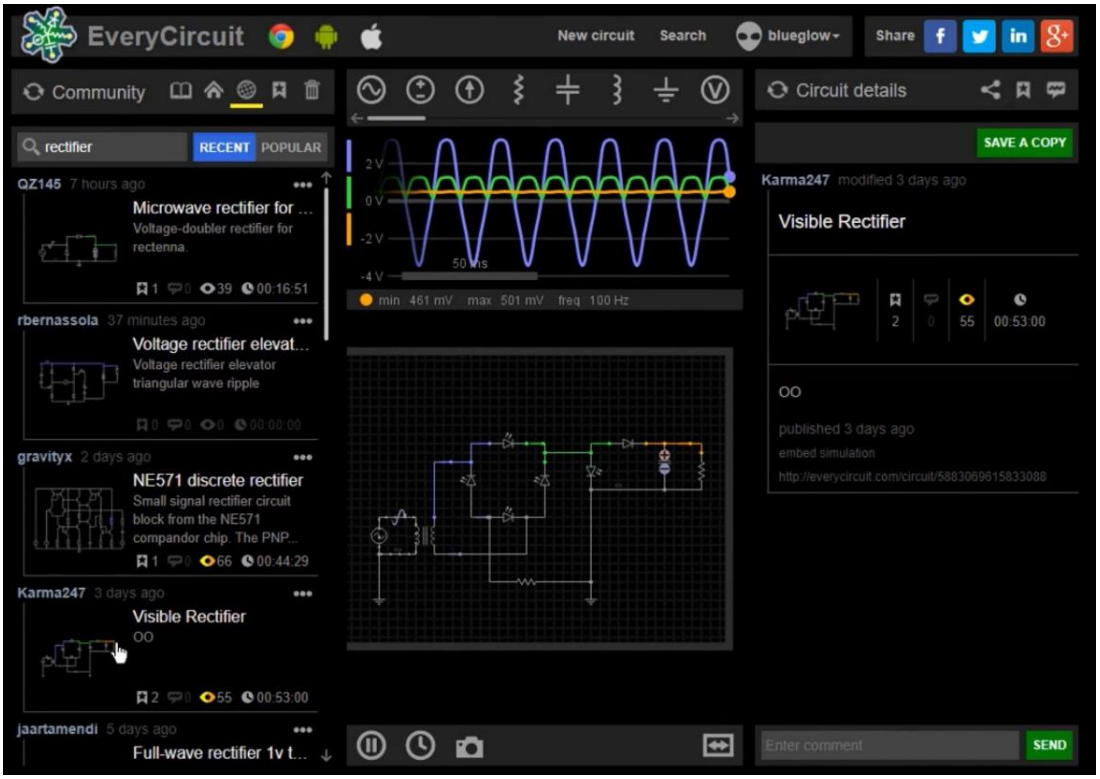


Figura 3 Exemplo da aplicação *EveryCircuit* [8]

KTechLab

O *KTechLab* [9] é um *software Open Source* desenvolvido apenas para a plataforma Linux, escrito em C++, especialmente concebido para simulação e de circuitos elétricos e projeção de circuitos de eletrónica digital com microcontroladores PIC, da *Microchip Technology*,

como demonstrado na Figura 4. Lançada a versão estável em janeiro de 2018, este programa tem vindo a ganhar cada vez mais utilizadores por a cada atualização tentarem implementar o maior número de ferramentas. Destaca-se a integração do *Flowcode*, *software* da *Matrix TSL*, que permite a programação e controlo de microcontroladores num ambiente gráfico. A sua interface é de algum modo intuitiva o que poderá facilitar o seu manuseamento por novos utilizadores.

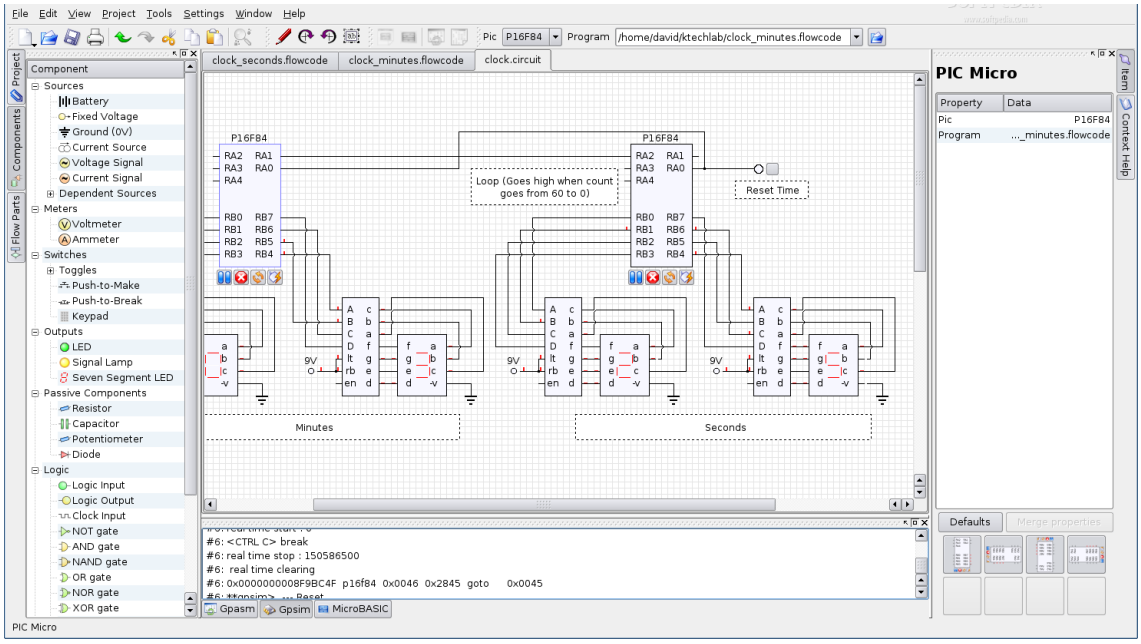


Figura 4 Exemplo do *software KTechLab* [9]

LTspice

O *LTspice* [10], desenvolvido à base do algoritmo SPICE pela fabricante de semicondutores *Linear Technology*, da *Analog Devices*, é possivelmente o *software* mais usado pela comunidade académica por se tratar de *freeware* disponível para macOS e Windows. Segundo os desenvolvedores, é a ferramenta à base de SPICE mais usada na indústria e das únicas a não limitar o *software* em código, ou seja, não limita o número de Nós nem de componentes. Apesar de robusto, o programa é bastante leve no que diz respeito à utilização de recursos da máquina em que corre. As suas bibliotecas são atualizadas todos os meses, adicionando os novos modelos dos componentes de que a marca é proprietária. De maneira a corrigir potenciais erros de *software* reportados pela comunidade, é prática comum lançarem atualizações mensais. Na Figura 5, é notável que a interface gráfica não é de todo envolvente.

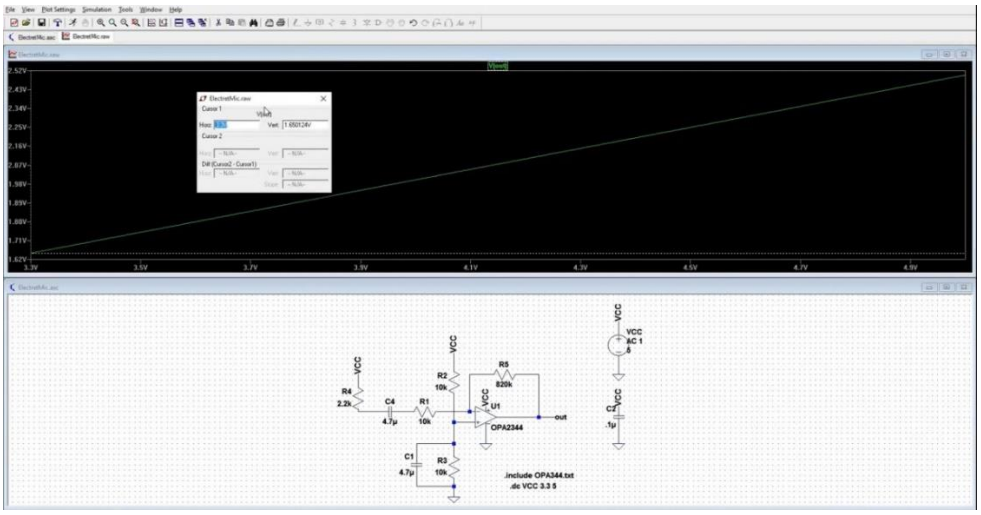


Figura 5 Exemplo do *software LTspice* [10]

NI Multisim

O *Multisim* [11] da *National Instruments*, integra a simulação SPICE padrão da indústria e está disponível apenas para o sistema operativo *Windows*. Integrando as bibliotecas mais completas, em relação a outros simuladores, o *Multisim* dispõe de funcionalidades como a conversão de esquemáticos em PCB para outro programa também de *software* proprietário, o *NI Ultiboard*. Ao poder ser usado tanto no âmbito académico como no profissional, por permitir um manuseamento tanto de circuitos simples como complexos, este torna-se numa das aplicações mais caras existentes no mercado, chegando a custar 685€ para a edição de ensino e 4985€ para a licença profissional.

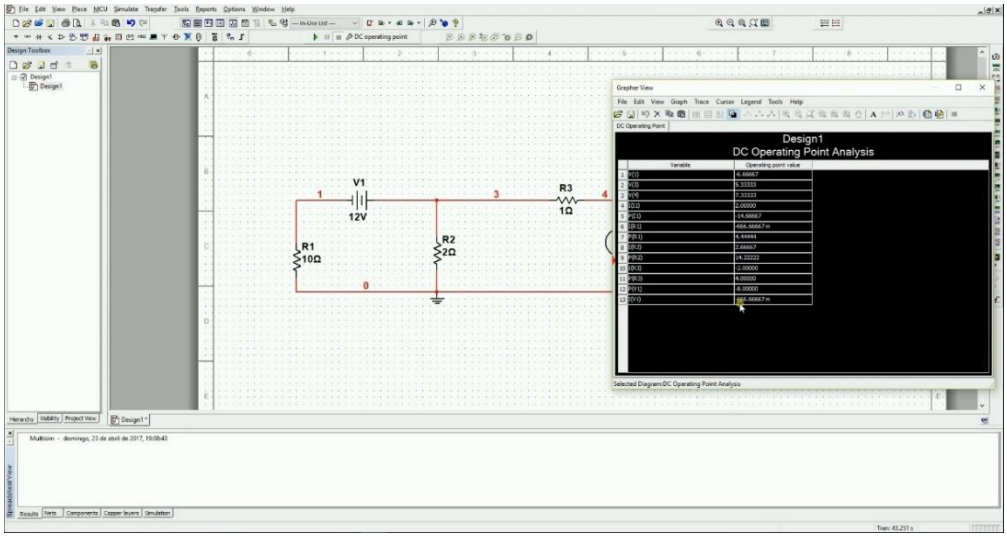


Figura 6 Exemplo do *software NI Multisim* [11]

Oregano

O *Oregano* [12] é um OSS (*Open Source* S*oftware*) disponível somente para sistemas *Linux*. Baseado no poderoso *software* SPICE, esta aplicação é bastante usada como ferramenta de desenho de circuitos por oferecer uma basta biblioteca de componentes, que é atualizada regularmente. A interface gráfica, mostrada na Figura 7, é deveras simples e minimalista, permitindo aos utilizadores menos experientes na área uma boa adaptação à ferramenta. Contudo, sendo uma aplicação tão simples, torna-se numa ferramenta quase obsoleta quando se quer manipular e simular circuitos mais complexos. Peca também pela falta de documentação oficial. A equipa está neste momento a desenvolver uma aplicação *web*, *ahoi.io*, para poder chegar a mais utilizadores.

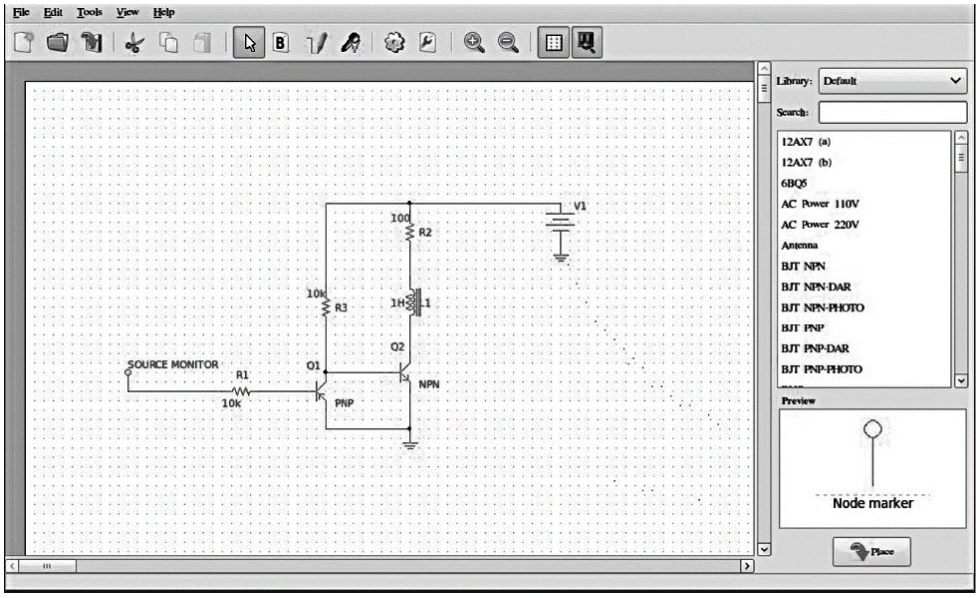


Figura 7 Exemplo do *software Oregano* [12]

PSIM

Acessível unicamente para a plataforma da *Microsoft Windows*, o PSIM [13] é um dos programas mais conhecidos na sua categoria. Apesar de não ser gratuito, dispõe de uma versão experimental de 30 dias com todas as suas funcionalidades básicas acessíveis, limitando apenas o tamanho máximo do circuito em 34 componentes. A sua interface gráfica, retratada na Figura 8, é relativamente simples e bastante similar com a de outros simuladores. Apresenta várias bibliotecas, mas perde pela rara envolvência da comunidade no fórum oficial.

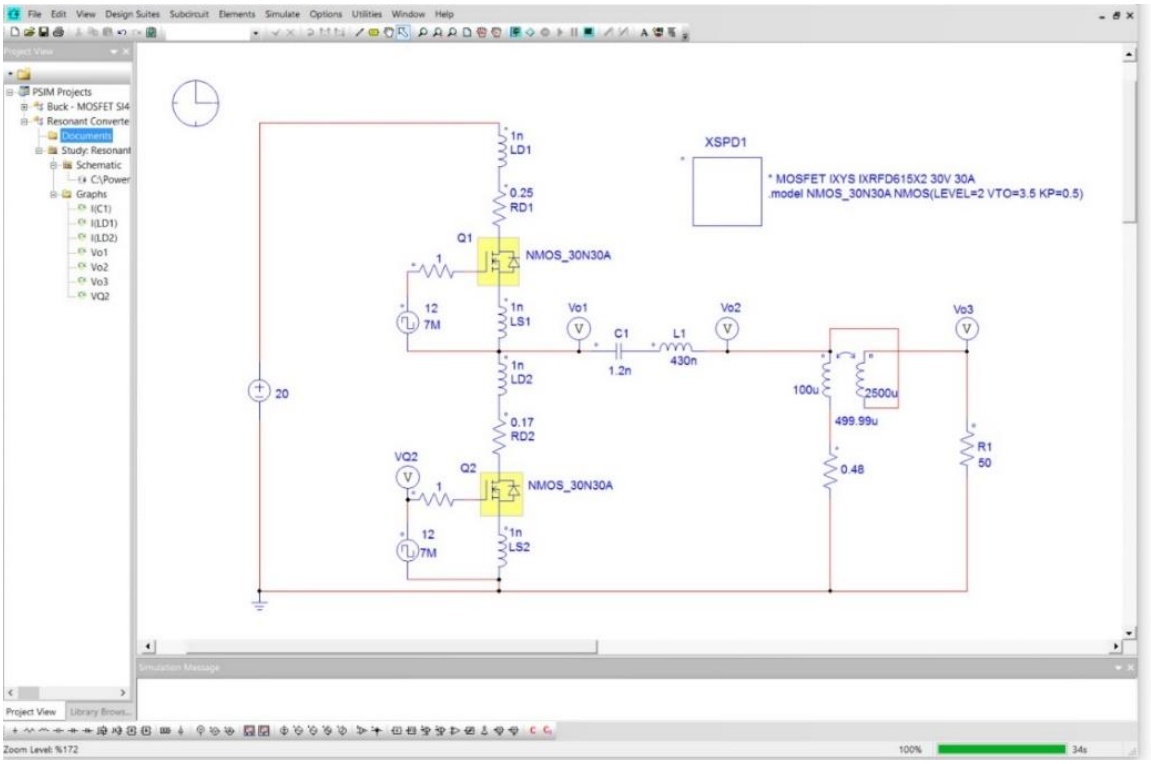


Figura 8 Exemplo do *software* PSIM [13]

PSpice

O *PSpice* [14], da *Cadence*, é provavelmente o *software* de simulação de circuitos mais completo do mercado, disponibilizando uma extensa lista de bibliotecas de componentes e modelos de circuitos. Por esta razão, fica justificado o preço elevado da ferramenta. A versão de compra única tem um custo de 6000€, sendo esta na verdade composta por 6 aplicações, a *OrCAD Capture*, *OrCAD Capture CIS*, *PSpice A/D*, *PSpice Advanced Analysis*, *OrCAD PCB* e *SPECCTRA*. Existe a possibilidade de fazer o *download* da versão *trial* (30 dias) e da versão *lite*, que limita o uso de componentes mais complexos. A sua interface, retratada na Figura 9, revela-se pouco apelativa para iniciantes e os tutoriais existentes para manipulação da mesma encontram-se desatualizados.

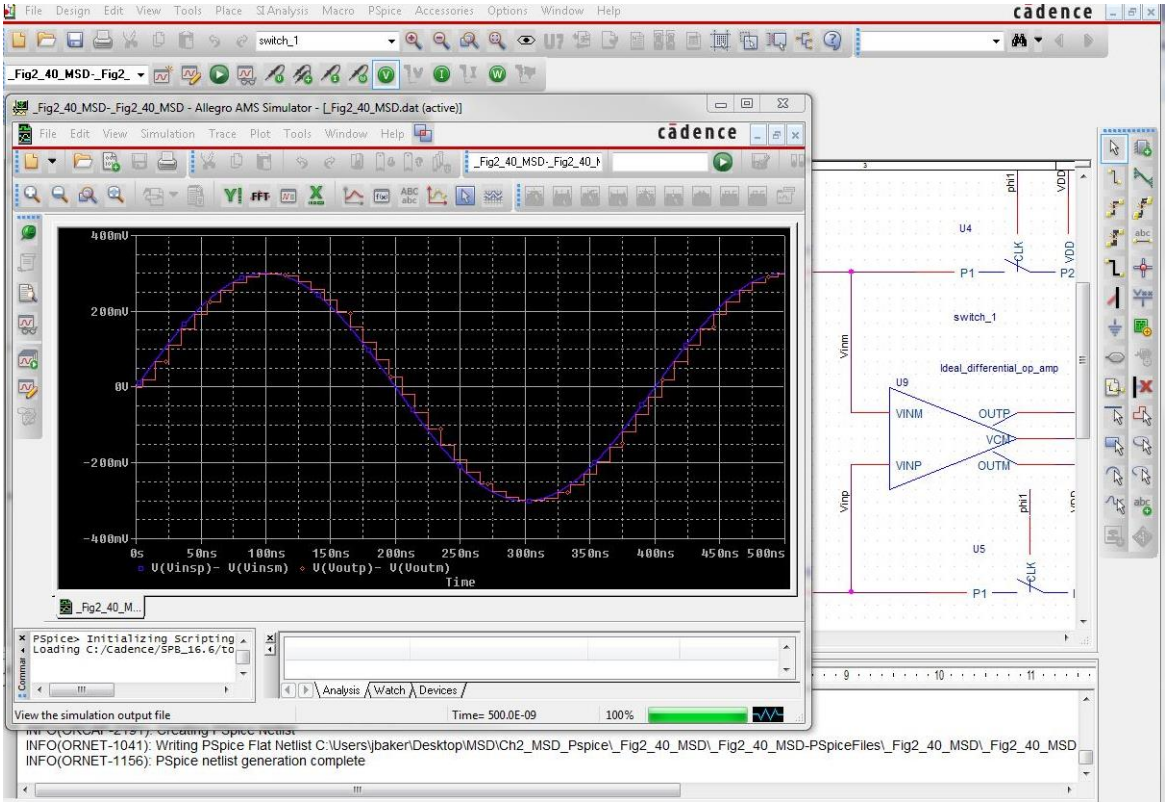


Figura 9 Exemplo do *software PSpice* [14]

QUCS

O QUCS [15] é um simulador *Open Source* lançado em 2003 que desde então não parou de evoluir, seja por parte dos criadores, seja por desenvolvedores voluntários. Disponível nos sistemas operativos *Linux*, *macOS* e *Windows*, o simulador apresenta uma interface gráfica simples, representada na Figura 10, que permite ao utilizador menos experiente projetar facilmente circuitos e simular o seu funcionamento. Dispondo de várias bibliotecas, também elas *Open Source*, o QUCS possui uma das maiores comunidades de desenvolvimento voluntário em comparação com as restantes ferramentas. Esta coletividade onde todos podem participar no melhoramento da interface, na criação de novas funcionalidades e até mesmo na tradução da aplicação para outras línguas, permitindo à aplicação subsistir até aos dias de hoje de forma exemplar. O facto de estar hospedado no *GitHub*, permitindo a rápida resolução de problemas, e de existir um fórum para discutir melhorias a implementar, aponta para a continuação do processo de evolução do programa.

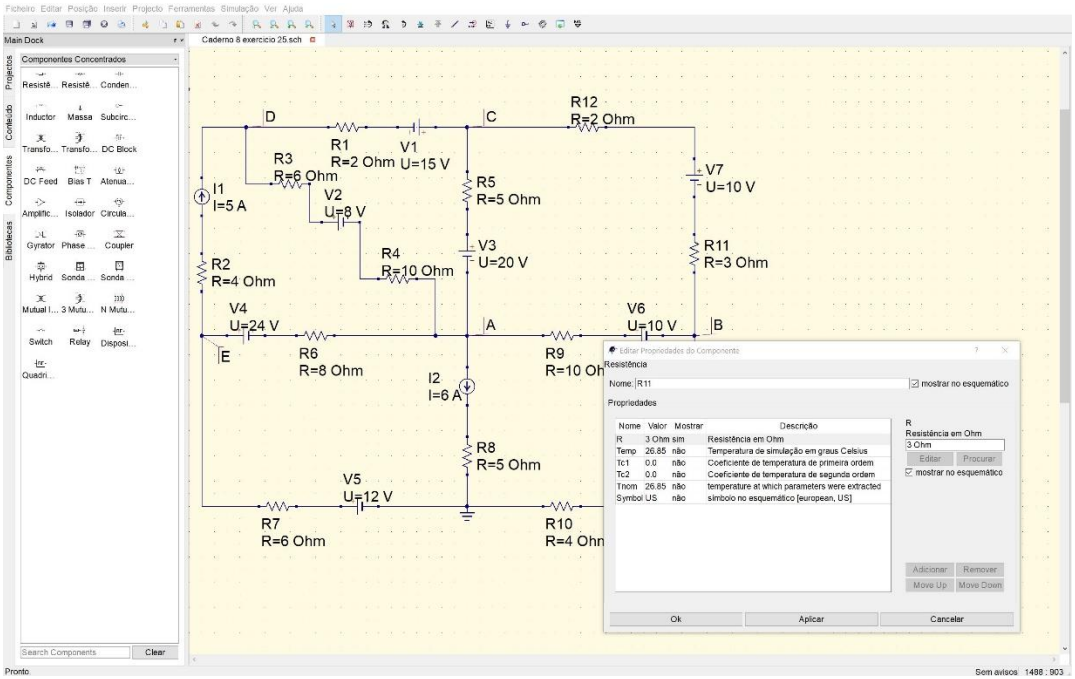


Figura 10 Exemplo do *software* QUCS [15]

2.2. COMPARAÇÃO DE SIMULADORES

Após a análise dos diferentes simuladores, existe a necessidade de avaliar o contexto em que cada um se deve inserir, visto que nem todos são elaborados com o intuito de se enquadrarem no âmbito académico. Posto isto, de modo a resumir a informação sobre as ferramentas, na tabela 2, são atribuídas as avaliações em função dos padrões escolhidos, tipo de licença e preço, resultando numa classificação final global. A avaliação foi atribuída numa escala de 1 a 5 estrelas. Elegeram-se os parâmetros de avaliação para cada simulador tais como o tipo de licença, o seu preço, os sistemas operativos que operam, a quantidade de bibliotecas e módulos disponíveis, a facilidade de adaptação à ferramenta, o suporte oficial, documentação e dimensão da comunidade.

Tabela 2 Tabela comparativa entre os simuladores de circuitos elétricos

	Licença	Preço	Plataforma	Bibliotecas	Facilidade de adaptação à ferramenta	Suporte e comunidade	Classificação global
CircuitLab	Software Proprietário	22,00 € / Ano	Browser	★★★	★★★★	★★★	★★
EasyEDA	Software Proprietário	Gratuito com Anúncios	Browser, Linux, macOS; Windows	★★★	★★★★	★★★	★★★
EveryCircuit	Software Proprietário	15,00 € Compra Única	Android, Google Chrome Browser, iOS	★★★	★★★★	★★★★	★★★
KTechLab	GPL	Gratuito	Linux	★★★	★★★★	★★	★★★
LTspice	Freeware	Gratuito	macOS, Windows	★★★	★★	★★	★★
NI Multisim	Software Proprietário	685,00 € Compra Única	Windows	★★★★★	★★★	★★★★★	★★★
Oregano	GPL	Gratuito	Linux	★★★	★★★	★	★★
PSIM	Software Proprietário	Versão trial 30 dias	Windows	★★★	★★★	★★★★	★★
PSpice	Software Proprietário	6.000,00 € Compra Única	Windows	★★★★	★★	★★★★	★★
QUCS	GPL	Gratuito	Linux, macOS; Windows	★★★	★★★★★	★★★	★★★★

O *NI Multisim* e o *OrCAD PSpice* são sem sombra de dúvida as aplicações mais completas a nível tecnico, mas para o caso, foram desvalorizadas por serem extremamente caras na ótica de utilização para autoaprendizagem e pelo facto de não serem *Open Source*.

Analisadas todas as alternativas, o QUCS, para além de ser uma aplicação flexível no ponto de vista de desenvolvimento, por ter o seu código fonte disponível publicamente, é considerado, por comparação, o *software Open Source* com melhor avaliação.

3. MÉTODO DAS CORRENTES NAS MALHAS

Enquanto as leis de Kirchhoff nos fornecem o método básico para analisar qualquer circuito elétrico complexo, existem diferentes maneiras de melhorar esse método usando a análise de Correntes nas Malhas ou a análise da Tensão nos Nós, que resultam numa diminuição da matemática envolvida e quando se trata de circuitos mais complexos essa redução pode ser uma grande vantagem. Este método permite obter a Corrente de cada Malhas do circuito em análise. Define-se por Malha um caminho fechado sem repetição, ou seja, o seu trajeto só percorre os Nós uma vez.

3.1. MÉTODO PARA A ANÁLISE MCM

O MCM é um método organizado que segue um conjunto de técnicas simples para a obtenção das equações do circuito. É baseado na Lei de Kirchhoff das Malhas (LKM), isto é, a soma algébrica das forças eletromotrizes (f.e.m.) em qualquer Malha deverá ser igual à soma algébrica das quedas de tensão contidas nessa Malha. Neste método as variáveis a determinar são as Correntes fictícias das Malhas selecionadas.

A análise pelo Método das Correntes das Malha é executada através da seguinte sequência de tarefas [14]:

1. Identificar o número de Ramos (R), Nós (N) do circuito e Fontes de Corrente (FC);

2. Definir o número de Malhas (M) Independentes necessárias, para formular o sistema de equações, pela fórmula M=R-(N-1)-FC;

3. Definir FC Malhas Auxiliares, incluindo em cada Malha apenas uma Fonte de Corrente e atribuir à Corrente da Malha o valor da Fonte de Corrente e o seu sentido;

4. Escolher M Malhas lineamente Independentes sem Fontes de Corrente e atribuir a cada Malha uma Corrente fictícia de circulação, com sentido arbitrário, denominada de Corrente de Malha;

5. Escrever as M equações das Malha, com base na LKM;

6. Resolver o sistema de equações resultante da combinação de todas as equações das Malhas, em função das Correntes de Malha.

3.2. ANÁLISE DE CIRCUITO EXEMPLO

Para exemplificar este método, serão aplicados todos os passos evidenciados no circuito apresentado na Figura 11.

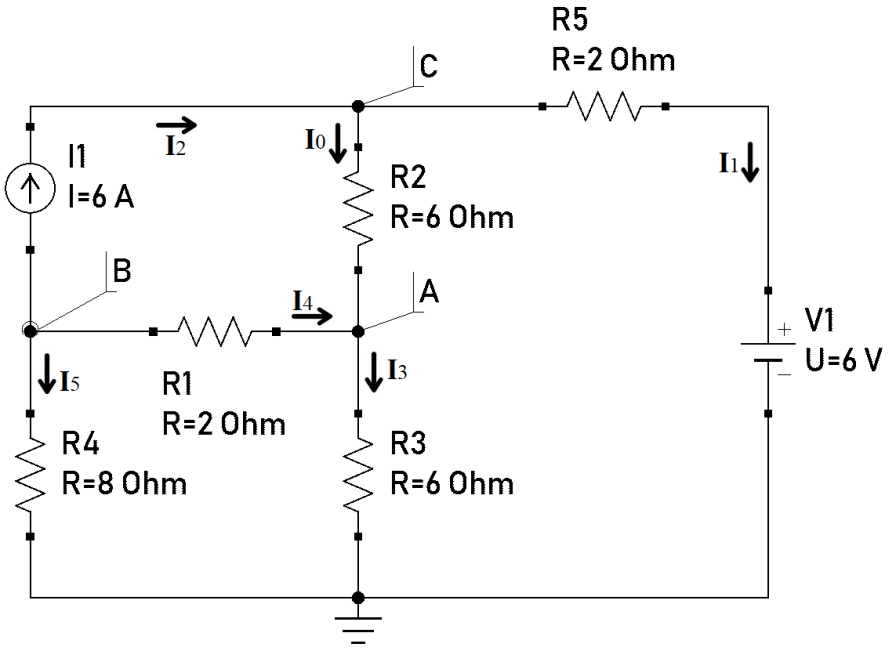


Figura 11 Exemplo de circuito para análise MCM

- **Passo 1**

  R: 6 Ramos;

  N: 4 Nós;

  FC: 1 Fonte de Corrente.

- **Passo 2**

  M = R – (N – 1) – FC = 6 – (4 – 1) – 1 = 2 Malhas Independentes.

- **Passo 3**

  FC = 1 Malha Auxiliar;

  Componentes da Malha Auxiliar: I1, R2 e R3;

  Sentido da Malha igual ao sentido da Fonte de Corrente (do Nó B para o Nó C).

- **Passo 4 (representado na Figura 12)**

  Malha 1:

  Composta por: R1, R3 e R4;

  Sentido da Corrente $I_{11:}$ Do Nó B para o Nó A.

  Malha 2:

  Composta por: R3, R2, R5 e V1;

  Sentida da Corrente I_{22}: Do Nó A para o Nó C.

- **Passo 5**

  Equações das Malhas:

  Malha 1:

  $$R3 \cdot (I_{11} - I_{22}) + R4 \cdot I_{11} + R1 \cdot (I_{11} - I1) = 0$$

  Malha 2:

$$R3 \cdot (I_{22} - I_{11}) + R2 \cdot (I_{22} - I1) + R5 \cdot I_{22} = V1$$

- **Passo 6**

  Substituindo os valores conhecidos:

$$\begin{cases} 6 \cdot (I_{11} - I_{22}) + 8 \cdot I_{11} + 2 \cdot (I_{11} - 6) = 0 \\ 6 \cdot (I_{22} - I_{11}) + 6 \cdot (I_{22} - 6) + 2 \cdot I_{22} = 6 \end{cases}$$

  Resolvendo em ordem às Correntes de Malha I_{11} e I_{22}:

$$\begin{cases} I_{11} = 2{,}23\ A \\ I_{22} = 3{,}95\ A \end{cases}$$

Com estes valores, torna-se logo possível o calculo das Correntes nos Ramos, dado que estas se podem obter pela soma algébrica das Correntes nas Malhas que passam nesse Ramo.

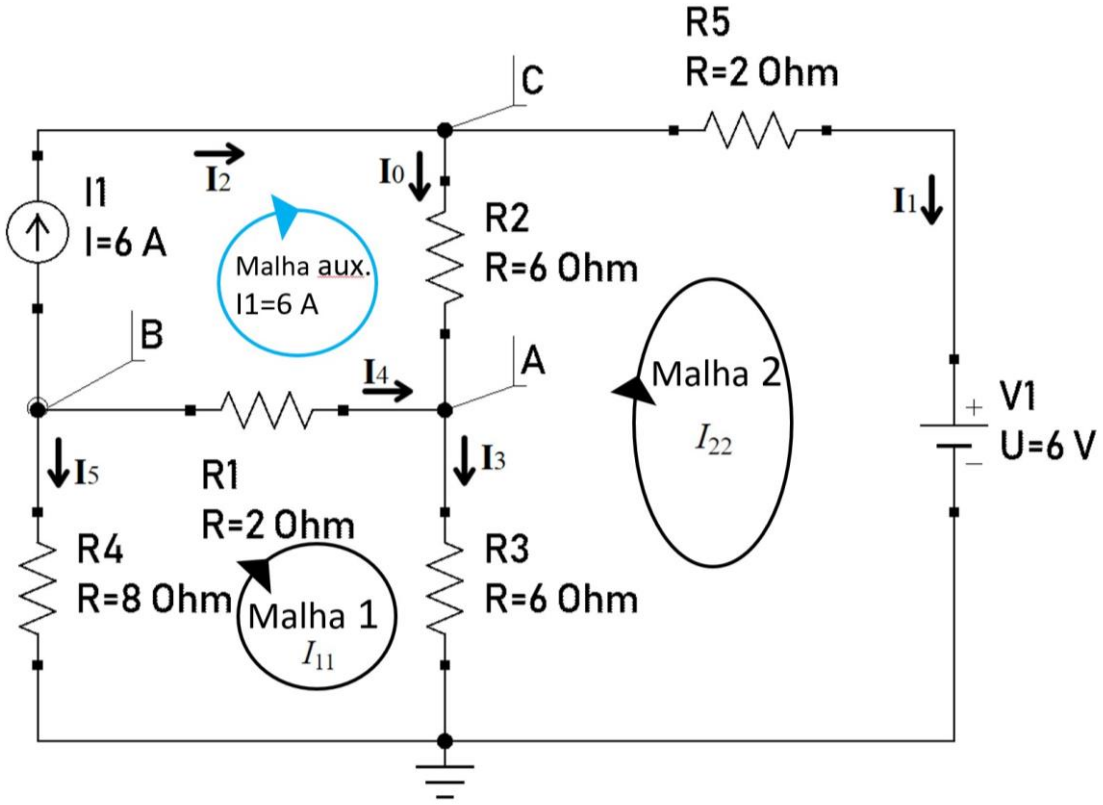


Figura 12 Circuito com as Malhas definidas

É importante perceber que, na seleção das Malhas, devemos garantir as seguintes condições:

- Todos os elementos integrantes devem estar incluídos em pelo menos uma Malha, pois cada elemento precisa de ter a oportunidade de influenciar a solução;
- As Malhas devem conter pelo menos um Ramo que não integre outra Malha;
- As Malhas devem partilhar pelo menos um Ramo com qualquer outra Malha.

4. APLICAÇÃO U=RISOLVE: MÓDULO DO MÉTODO DAS CORRENTES NAS MALHAS

A aplicação desenvolvida pretende acrescentar à plataforma U=RISOLVE a possibilidade da análise pelo Método das Correntes nas Malhas. Procura incentivar o utilizador no momento da aprendizagem do método, munindo-o com uma ferramenta de autoaprendizagem, concebida de forma a apresentar uma resolução passo a passo de qualquer circuito. Neste capítulo será descrita a aplicação desenvolvida e como esta interage com o utilizador, bem como as tecnologias quer permitem adquirir a informação do circuito.

4.1. OBJETIVOS E ARQUITETURA

A plataforma U=RISOLVE é uma aplicação inteiramente desenvolvida em JavaScript com interface gráfica simples desenvolvida em HTML5 e CSS. Surge de outro projeto desenvolvido no âmbito da disciplina de PESTA [4], onde se desenvolveu a

funcionalidade da Análise de circuitos pelo Método das Tensões nos Nós. O objetivo é adicionar outra funcionalidade à plataforma, que, no caso deste projeto, será o MCM. Esta nova implementação permite ao utilizador visualizar as equações das Malhas de um qualquer circuito e o resultado das suas Correntes fictícias. Como o objetivo é de ter uma ferramenta prática e simples, o seu processo de interação é de igual modo simples e pode-se traduzir nas seguintes tarefas:

1. Gerar uma *netlist* para o circuito simulado no QUCS, tanto no menu de simulação como pressionado a tecla F6;

2. Guardar a *netlist* em formato de texto (.txt) no diretório desejado pelo utilizador;

3. Fazer *upload* da *netlist* gerada na plataforma U=RISOLVE e pressionar o botão correspondente à análise pretendida.

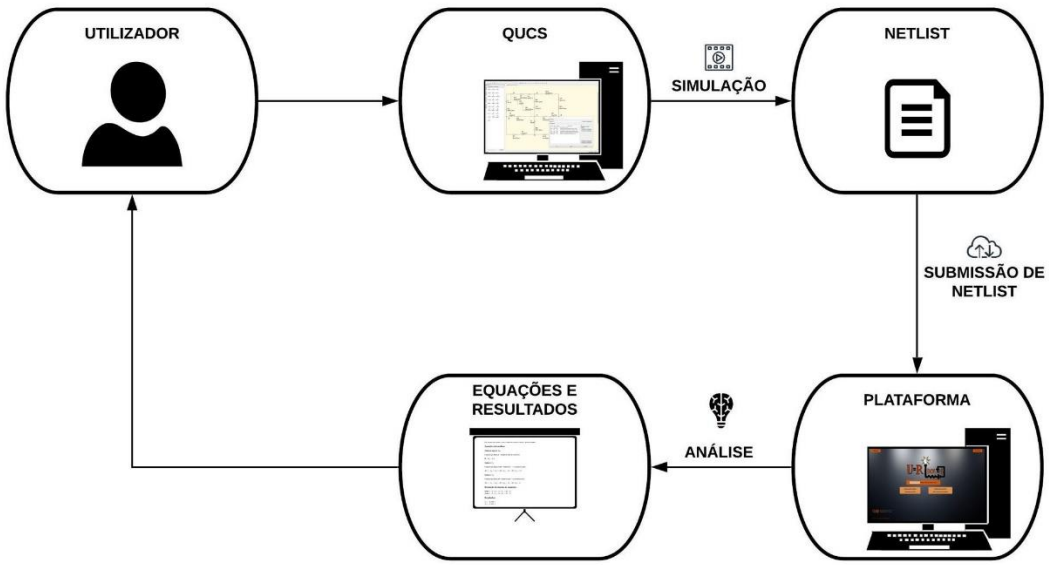


Figura 13 Lógica de funcionamento da plataforma U=RISOLVE

A apresentação do resultado da análise é feita numa janela *pop-up* que se abre no browser sempre que o utilizador pressione um dos botões das diferentes análises disponíveis e que tenha feito o upload da *netlist*. Deste modo, é permitido ao utilizador executar a ferramenta com diferentes circuitos sem a necessidade de abrir uma nova

janela da aplicação ou extrair os resultados das simulações anteriores. O resultado é exibido por etapas, tentando reproduzir o método de resolução à mão, de modo a facilitar a sua leitura ao utilizador e tornar a aplicação mais atrativa no ponto de vista da sua utilidade e conveniência. A exposição dá-se pela seguinte ordem:

1. Inicialmente são apresentadas as características do circuito submetido, tais como o número dos seus Ramos, Nós e Fontes de Corrente, seguindo o número de Malhas possíveis e Malhas necessárias para a resolução, com apresentação de fórmula;

2. De seguida são mostradas as equações das Malhas, identificadas por número, pelos elementos que é composta e pelo seu sentido arbitrado automaticamente;

3. Posteriormente, no caso da análise em Corrente Contínua, o sistema de equações surge com as Correntes de malha como únicas incógnitas, tendo substituído nas equações elaboradas anteriormente os valores conhecidos de componentes. Em circuitos em Corrente Alternada, este ponto não se executa devido à complexidade da operação.

4. Por fim, em Corrente Contínua, são revelados os resultados das Correntes de Malha

No processo de geração da *netlists* pelo QUCS, este atribui nomes por *default* a todos os Nós existentes, com exceção do Nó real "gnd". Estes pontos de ligação são designados por "_netX", onde "X" representa um número de identificação que é incrementado sempre que é detetado um novo Nó virtual. O utilizador ao não identificar manualmente os Nós do circuito no QUCS, o resultado da análise poderá ser confuso quando é exibida a informação do sentido das Correntes, pois esta tem por base um elemento do primeiro ramo da malha e as extremidades desse mesmo ramo ajudam a estabelecer o sentido.****** Posto isto, é recomendado ao utilizador identificar no circuito todos os Nós reais, atribuindo-lhes um nome, para facilitar a leitura do resultado obtido pela ferramenta. Esta prática exige que o aluno saiba indicar corretamente os Nós reais de qualquer circuito, já que este é um conhecimento fundamental num estágio de introdução ao estudo de circuitos elétricos.

A aplicação contém ainda uma página de instruções para facilitar a adaptação do utilizador à mesma, explicando com clareza como proceder à deteção de Nós e da massa no QUCS, bem como produzir o ficheiro de texto da *netlist* e também como solucionar potenciais problemas causado pelo *browser*.

De forma a facilitar a aprendizagem na manipulação da ferramenta, esta incluí um conjunto de circuitos exemplo divididos por Corrente Continua e Corrente Alternada e por nível de complexidade, com as *netlists* criadas e com uma imagem do respetivo circuito.

4.2. ALGORITMO DE ANÁLISE

A estruturação do algoritmo foi inicialmente inspirada pelo trabalho já desenvolvido para a análise MTN, já que parte das primeiras operações são idênticas em ambos os métodos. Assim, foi possível adaptar e melhorar algumas rotinas já elaboradas O algoritmo de análise desenvolvido pode ser dividido em seis etapas:

- Filtragem e armazenamento de informação da *netlist* numa matriz;
- Remoção de aparelhos de medição;
- Deteção de erros na informação tratada;
- Definição de Nós reais e Ramos;
- Geração de combinações de Malhas;
- Validação e seleção de Malhas;

A estruturação por estágios originou uma maior fluidez na implementação por permitir otimizar a forma de organização e manipulação da informação, planear o código para evitar falhas numa próxima etapa e preparar o formato de certas variáveis na conclusão de outra fase para uma configuração adequada a usar seguidamente. No próximo subcapítulo serão expostos em pormenor todos os passos do algoritmo de análise. A Figura 14 representa o fluxograma de funcionamento do mesmo.

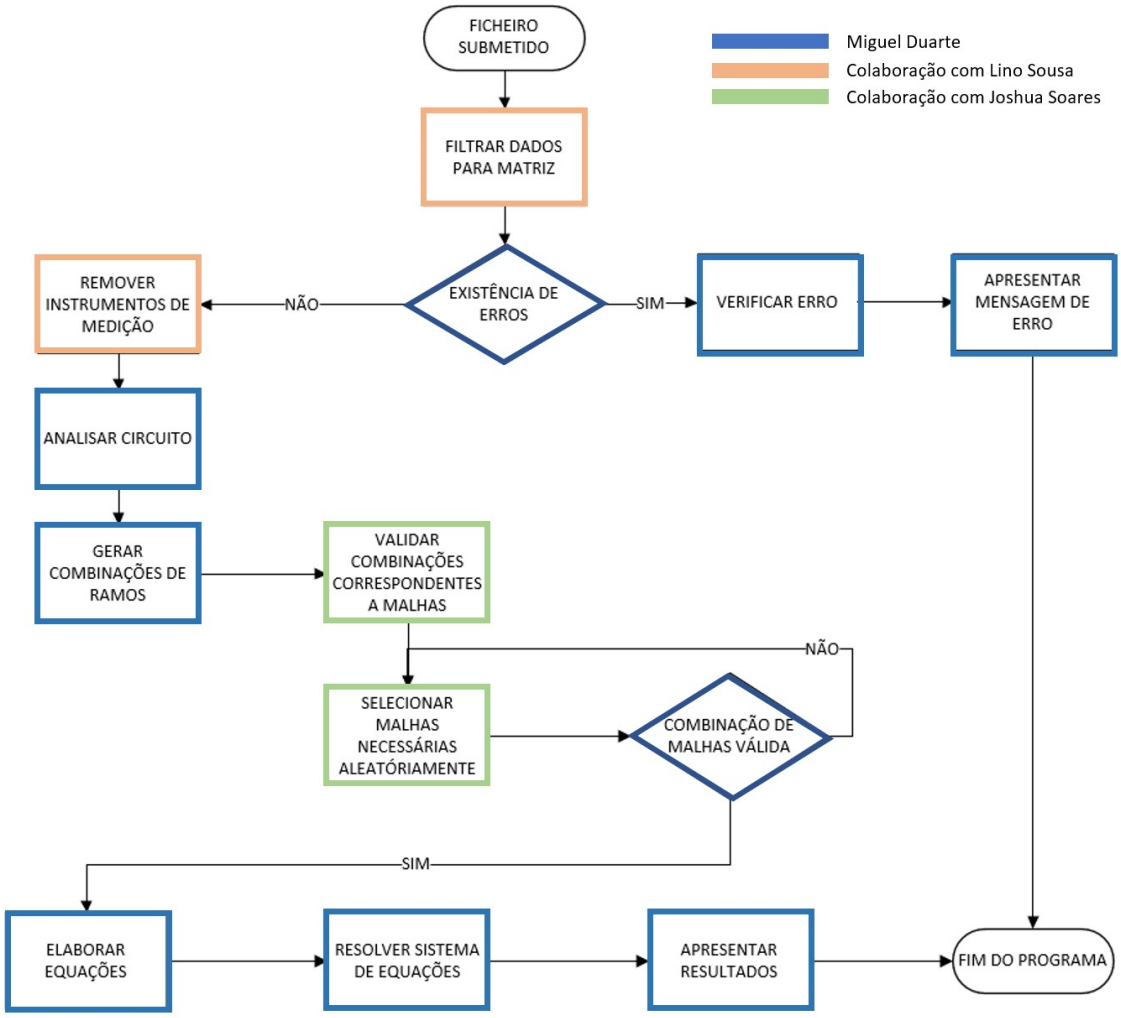


Figura 14 Fluxograma da aplicação U=RISOLVE para o MCM

Todas as contribuições no desenvolvimento de todo o módulo de análise MCM estão discriminadas na Figura 15.

Figura 15 Processos ordenados com menção das contribuições

4.2.1. FILTRAGEM E ARMAZENAMENTO DE INFORMAÇÃO

O primeiro passo de todo o algoritmo passa pela aquisição de todos os dados relevantes constantes na *netlist* gerada pelo QUCS [24]. Foi importante entender o formato em que esta informação é apresentada de maneira a perceber como pode ser extraída. A aplicação começa por armazenar numa *string* todos os caracteres da *netlist* submetida, e nesta irá procurar por todos os componentes do circuito, os Nós virtuais de cada um e os

seus respetivos valores. Na Figura 16 é apresentado um exemplo com a matriz gerada a partir *netlist*.

```
# Qucs 0.0.19  C:/Users/PC/.../example.sch

R:R2 C A R="6 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
Idc:I1 C B I="6 A"
R:R5 _net0 C R="2 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
Vdc:V1 _net0 gnd U="6 V"
R:R3 gnd A R="6 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
R:R1 B A R="2 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
R:R4 gnd B R="8 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
```

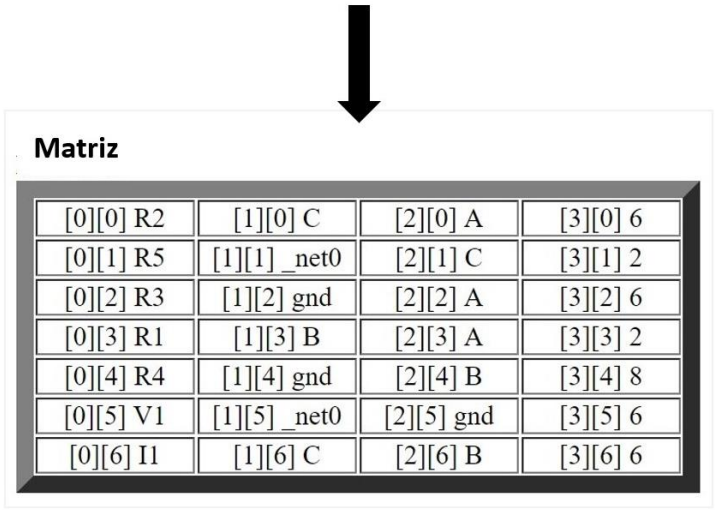


Figura 16 Filtragem de informação da *netlist* para uma matriz

O algoritmo serve-se da forma clara como o QUCS cria as *netlists*, que caracteriza os componentes seguindo sempre o mesmo padrão. No caso dos nomes dos componentes, o *script* procura pela sequência "R:Ri" para definir uma resistência com índice "i", seguidamente procura e armazena os Nós virtuais do mesmo, isto é, o ponto de ligação de inicio e fim da resistência. Por fim, guarda o valor da resistência. O mesmo se passa para os restantes componentes, procurando pelas seguintes sequências de caracteres:

- "C:C" para condensadores;
- "L:L" para bobines;
- "Vdc:V" para Fontes de Tensão DC;

- “Vac:V” para Fontes de Tensão AC;
- “Idc:I” para Fontes de Corrente DC;
- “Iac:I” para Fontes de Corrente AC;
- “IPro:” para amperímetros.

Os voltímetros podem ser igualmente encontrados por uma sequência de carateres, porém a sua aquisição seria inútil neste sentido, pois em nada interferem na estrutura do circuito.

De seguida, é necessário verificar se os valores armazenados estão na escala correta. Neste momento, o algoritmo procura ainda a presença de prefixos do Sistema Internacional (SI) após ler o valor do componente. Estes prefixos são aceites pelo QUCS na hora de definir valores, como referenciado na Figura 17.

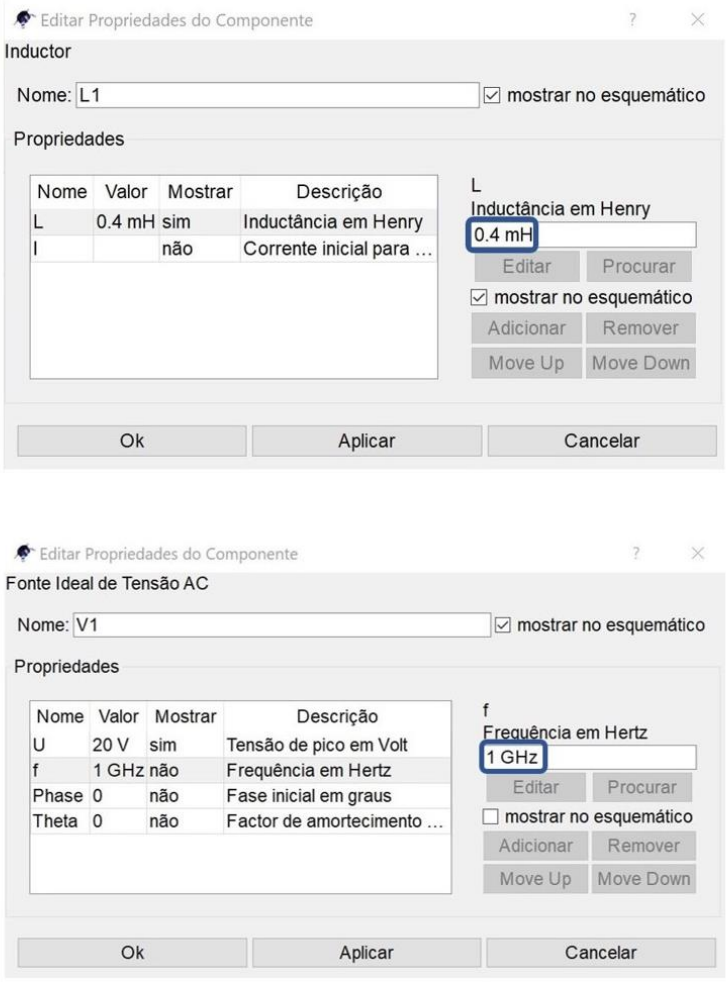


Figura 17 Exemplo de definição de valores no QUCS

Finalmente, o software faz a conversão dos valores para a unidade SI, multiplicando por 10 elevado ao valor do prefixo. Este passo revela-se fundamental nesta fase para simplificar a manipulação das equações na parte final do algoritmo.

4.2.2. MÉTODO DE AQUISIÇÃO DE NÓS E RAMOS

A definição dos Nós e dos Ramos com base na matriz estabelecida é a etapa basilar de todo o processo de análise. Adquiridos estes dados, será possível avançar para a composição de Malhas sem necessitar recorrer ao esquemático gráfico. De seguida, representada na Figura 19, encontra-se a matriz de dados obtidos pela *netlist* associada ao circuito exemplo exposto na Figura 18.

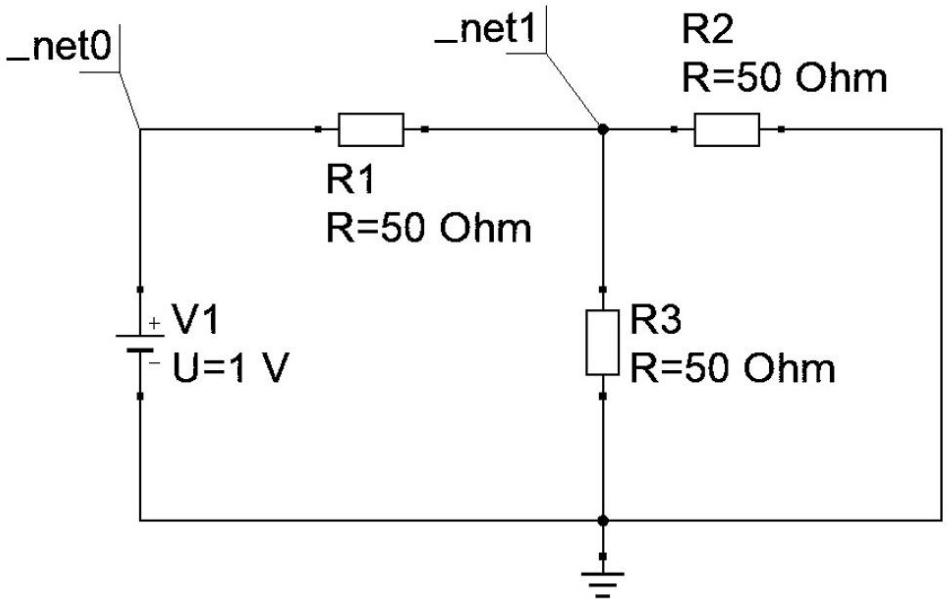


Figura 18 Exemplo de circuito simples elaborado no QUCS

[0][0] R1	[1][0] _net0	[2][0] _net1	[3][0] 50
[0][1] R2	[1][1] gnd	[2][1] _net1	[3][1] 50
[0][2] R3	[1][2] _net1	[2][2] gnd	[3][2] 50
[0][3] V1	[1][3] _net0	[2][3] gnd	[3][3] 1

Figura 19 Matriz com valores obtidos a partir da *netlist* do circuito da Figura 17

Esta matriz define univocamente o circuito exemplo. Com as características filtradas e organizadas, outras informações podem estar subjacentes, como é o exemplo dos Nós reais e dos Ramos do circuito. Segue-se o detalhe da aquisição destes elementos, tendo como exemplo a matriz da Figura 18:

- **Nós virtuais** – Um Nó virtual é o ponto intermédio que liga apenas dois componentes e é criado automaticamente pelo simulador a partir do momento em que se interligam dois elementos. Assim sendo, este só poderá estar mencionado duas vezes na matriz. Neste caso, só o ponto "_net0" se define como um Nó virtual;

- **Nós reais** –Um Nó real é o ponto em que se ligam pelo menos 3 componentes. Em suma, os Nós reais devem constar três ou mais vezes na matriz. Este é o caso dos pontos "_net1" (liga R1, R2 e R3) e "gnd" (liga R2, R3 e V1). Este método encontra-se representado na forma de fluxograma na Figura 20.

- **Ramos** – Um Ramo de um circuito é um percurso entre dois Nós reais consecutivos. Somente após a definição dos Nós reais do circuito é que é possível calcular os Ramos. Na matriz, define-se como ponto de partida um elemento que esteja ligado a um Nó real e analisa-se o caminho que este deve percorrer pelo outro Nó, até encontrar outro Nó real. No exemplo, começando por V1, verifica-se que este está ligado a um dos Nós reais, portanto, testará se o componente seguido do seu Nó virtual conecta com um Nó real, neste caso R1, que conecta com o Nó real "_net1". Os outros dois Ramos ficam simples de se definir visto que cada um contem apenas um elemento, ou seja, ambos os pontos de ligação desses componentes na matriz correspondem a Nós reais. Esta lógica está simplificada na forma de fluxograma na Figura 21.

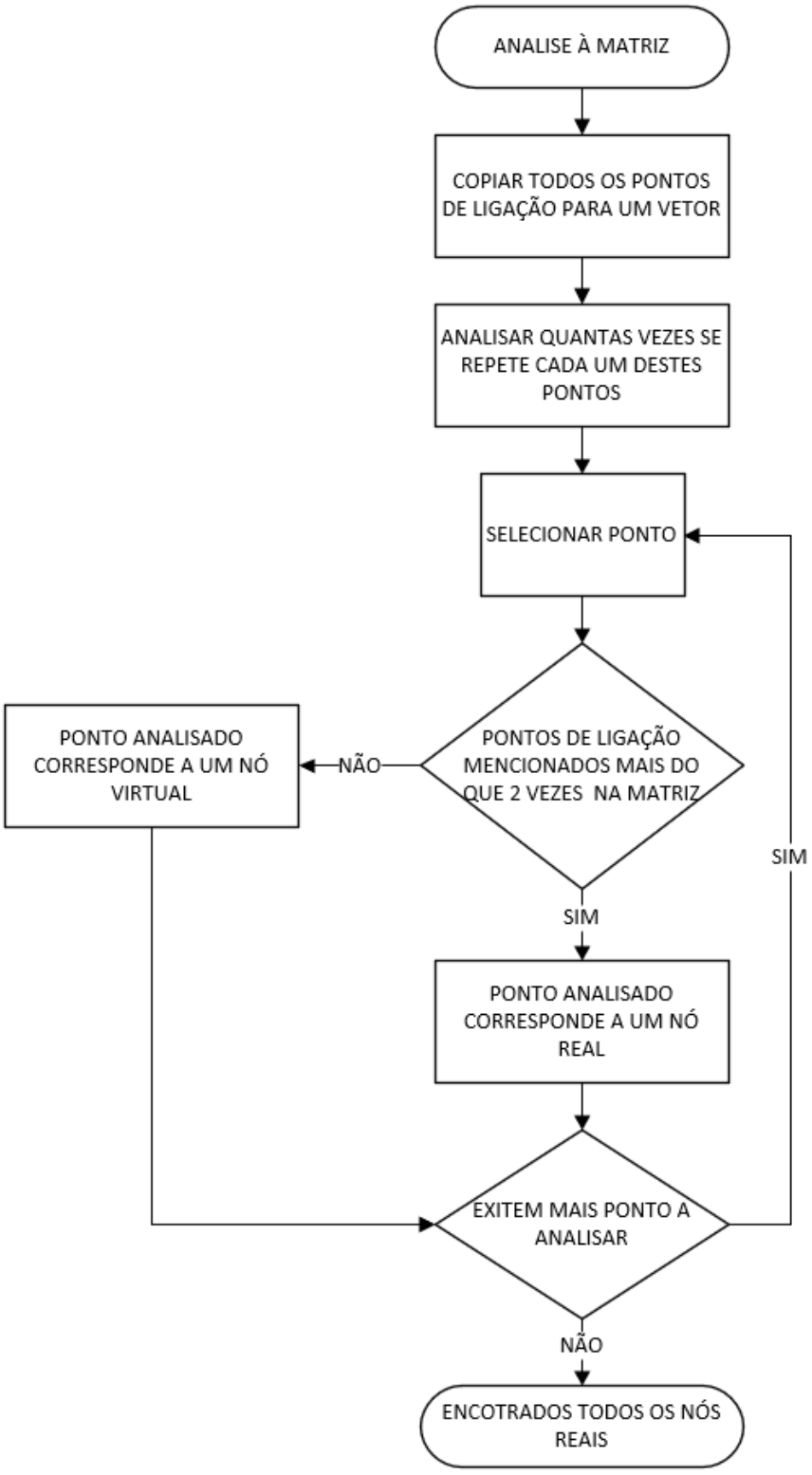


Figura 20 Fluxograma do método de aquisição de Nós reais a partir da matriz

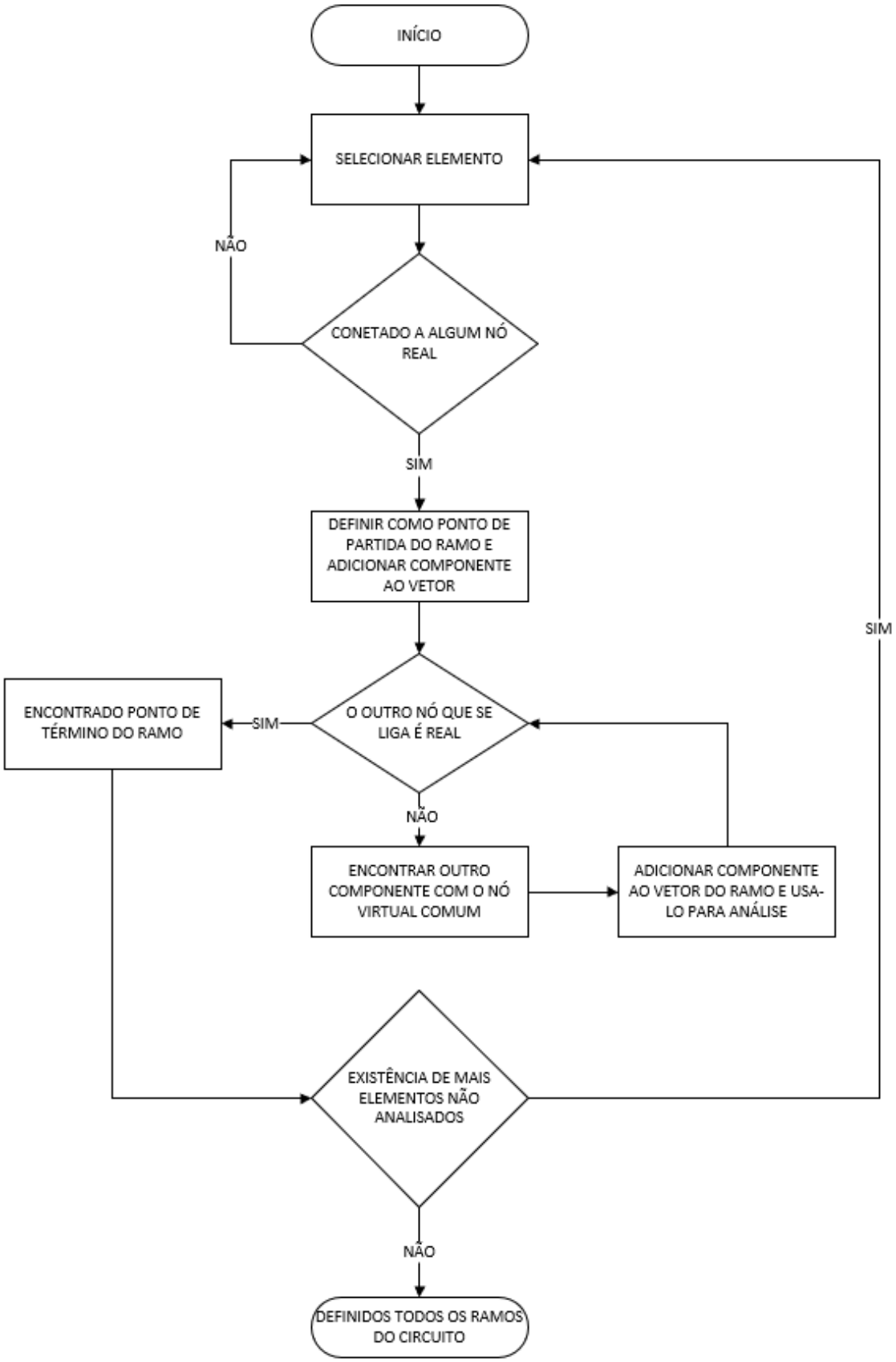


Figura 21 Fluxograma do método de aquisição de Ramos a partir dos Nós

4.2.3. REMOÇÃO DE APARELHOS DE MEDIÇÃO

De modo a evitar defeitos na análise do circuito, é essencial retirar da matriz os aparelhos de medição, tendo em conta que estes são usados frequentemente em simulações.

No que diz respeito aos voltímetros, estes não são problema visto que não alteram qualquer propriedade no circuito: Por serem ligados em paralelo, basta apenas não se armazenar qualquer informação sobre estes, como foi anteriormente mencionado.

Relativamente aos amperímetros, devido ao facto destes serem ligados em série, irão criar obrigatoriamente um Nó virtual. Ou seja, se os amperímetros fossem tratados da mesma forma que os voltímetros, o circuito ficaria em aberto no ramo em que o amperímetro estava presente. A solução para esta adversidade passa por armazenar os dados dos amperímetros e shuntar os seus Nós. Segue-se o fluxograma da Figura 22 para explicar o método usado.

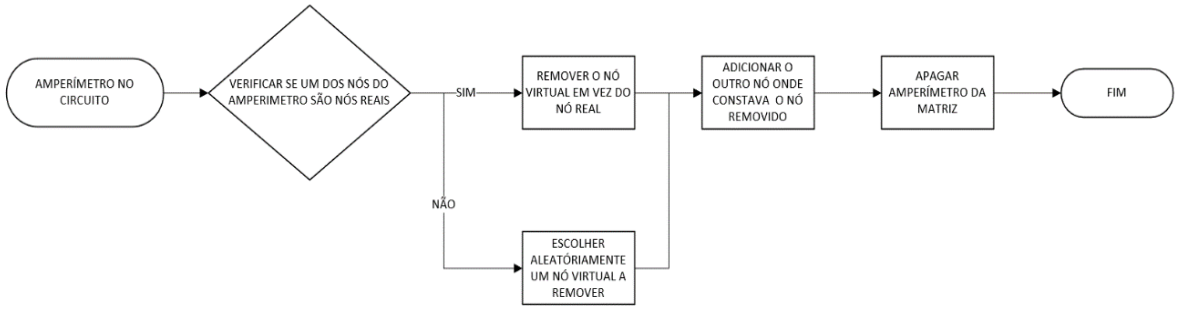


Figura 22 Fluxograma do método de remoção de amperímetros

A grande vantagem deste recurso constata-se pelo motivo do utilizador não necessitar de apagar os instrumentos de medição da simulação, conferindo uma maior flexibilidade à aplicação.

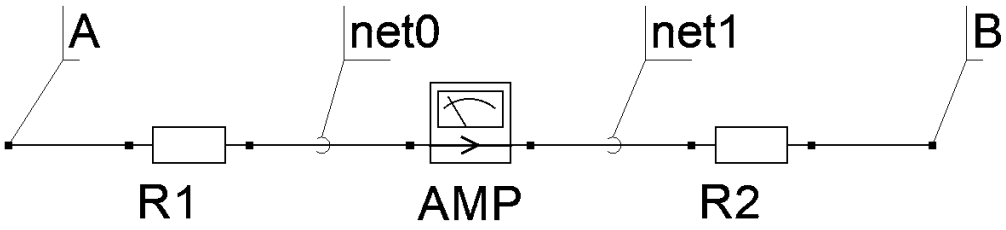


Figura 23 Exemplo de ligação com amperímetro no QUCS

4.2.4. DETEÇÃO DE ERROS

Com o intuito de proporcionar uma análise objetiva e de fácil compreensão é essencial estabelecer um conjunto de regras ao utilizador no momento em que está a projetar o circuito elétrico no QUCS. Neste conjunto de imposições englobam:

- Identificar todos os Nós reais com uma letra ou legenda à escolha;
- Identificar o Nó de referência com o símbolo de *ground* existente no QUCS.

Esta necessidade obriga o utilizador a saber identificar e referenciar os Nós reais de um circuito corretamente, que é um tópico de elevada importância no início do estudo da EE.

Esta etapa foi implementada novamente com recurso à matriz de dados construída no momento da filtragem da informação proveniente da *netlist*. No final da verificação, caso existam alguns destes erros, estes são apresentados ao utilizador como instrução de correção ao circuito. A exposição dos erros ao utilizador é realizada através de uma rotina do JavaScript designada por "alert()". Esta função, no momento em que é invocada, mostra uma caixa de alerta com uma determinada mensagem. Na aplicação, a mensagem corresponde a uma string dinâmica que no final de cada verificação vai acrescentando o texto referente ao erro que foi detetado. Este alerta contém a lista de orientações a seguir pelo utilizador de maneira a saber como agir na situação de erro, tal como mostra a Figura 24.

Figura 24 Exemplo de alerta num caso de erro de identificação

4.2.5. GERAÇÃO DE COMBINAÇÕES DE MALHAS

Esta ideia da geração de combinações de Malhas é resultado da necessidade de munir o algoritmo com um método de concebimento sólido e infalível. Só assim é que a aplicação consegue garantir resultados sempre que é solicitada, analisando todas as hipóteses de Malha quando possível.

A ideia passa por criar combinações sem repetições de todos os Ramos do circuito, sendo que a cada Ramo é atribuído um número de identificação. Na criação de Malhas, é necessário perceber que estas devem ser compostas de, no mínimo, dois Ramos, com os seus dois Nós reais compatíveis, e o número máximo de Ramos corresponde ao número de Nós reais do circuito.

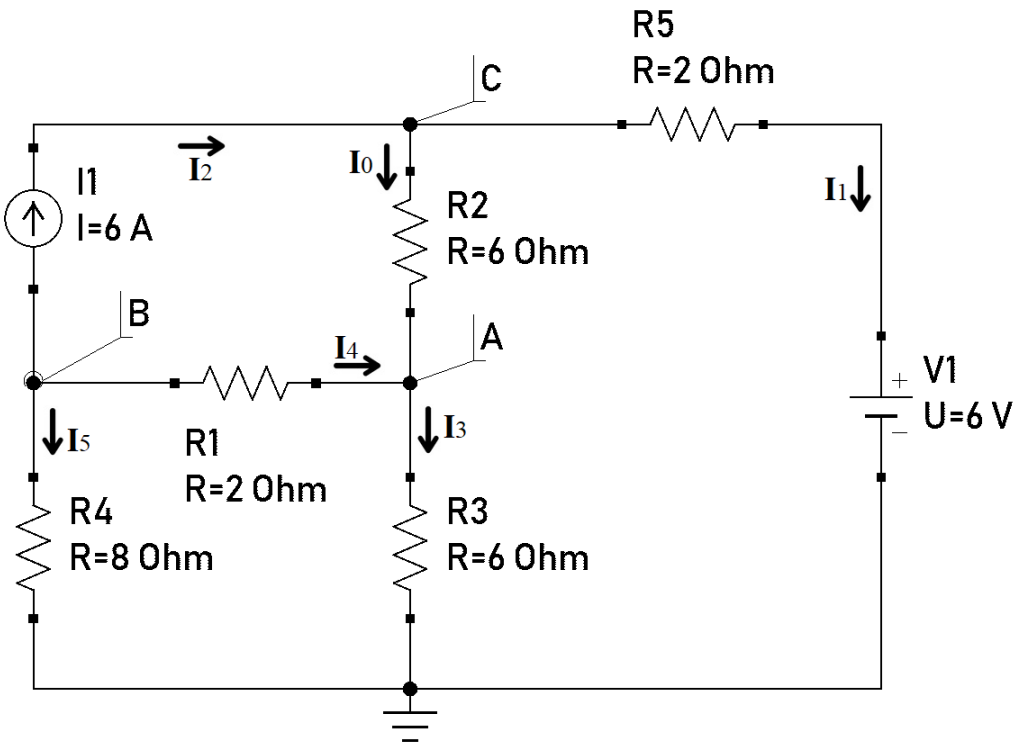


Figura 25 Circuito exemplo com 4 Nós, 6 Ramos e 1 Fonte de Corrente

Servindo de exemplo o circuito da Figura 25, composto por 4 Nós, 6 Ramos e 1 Fonte de Corrente, seria possível criar no máximo combinações de 4 Ramos. Segue-se exemplo de parte das combinações geradas apresentadas na Figura 25.

970	4	1	3	2
971	4	1	3	3
972	4	1	3	4
973	4	1	3	5
974	4	1	4	0
975	4	1	4	1
976	4	1	4	2
977	4	1	4	3
978	4	1	4	4
979	4	1	4	5
980	4	1	5	0
981	4	1	5	1
982	4	1	5	2
983	4	1	5	3
984	4	1	5	4
985	4	1	5	5
986	4	2	0	0
987	4	2	0	1
988	4	2	0	2
989	4	2	0	3
990	4	2	0	4
991	4	2	0	5
992	4	2	1	0
993	4	2	1	1
994	4	2	1	2
995	4	2	1	3
996	4	2	1	4
997	4	2	1	5
998	4	2	2	0
999	4	2	2	1

▶ Array(1346)

Figura 26 Exemplo de combinações de Ramos sem repetições

Para o circuito exemplo, foram encontradas 1346 combinações diferentes de Ramos que serão analisadas e filtradas na fase seguinte.

O algoritmo usado nesta fase, baseia-se no algoritmo geral para JavaScript das combinações sem repetições [21].

Numa fase de depuração, verificou-se que o facto do algoritmo criar um número elevado de combinações, quanto maior o número de Nós reais e Ramos que constitui, poderia exceder o limite da *stack* do *browser*, impedindo-o de prosseguir a análise. Nesse momento, foram implantadas novas rotinas para simplificar o processo e torná-lo mais leve para o processador. As novas rotinas implementadas passam pelos seguintes pontos:

- No caso de possíveis Malhas com 2 ou 3 Ramos, em vez de criar combinações sem repetição, são calculadas permutações, pois neste caso a ordem da disposição não é relevante, como sugere a Figura 27;

- Se o circuito apresentar mais do que 6 Nós e 6 Ramos, o algoritmo apenas gera as combinações de todos os Ramos sem repetição em conjuntos máximos de 6 Ramos. Neste cenário, o método limita-se a analisar as Malhas possíveis de construir, excluindo possíveis Malhas com 7 ou mais Ramos.

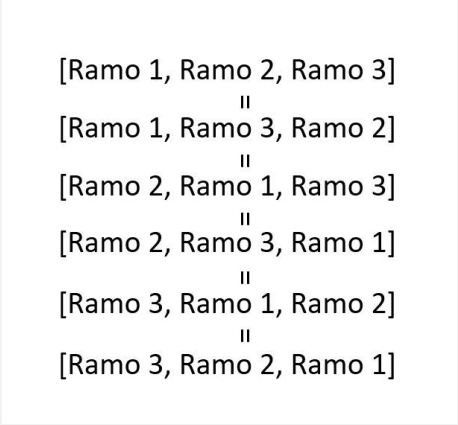


Figura 27 Exemplo de combinações com 3 Ramos correspondentes à mesma Malha

Estas permutações e combinações são armazenadas num vetor que será posteriormente testado, um a um, na etapa seguinte.

4.2.6. VALIDAÇÃO E SELEÇÃO DE MALHAS

No desenvolvimento desta fase, foram descobertas noções importantes e bastante curiosas no que diz respeito à composição e geração de Malhas para a análise MCM. Na abordagem ao problema, definem-se os parâmetros cruciais à validação de uma malha. Esta tem de ser constituída por no mínimo dois Ramos, quando estes partilham dos mesmos dois Nós reais, e no máximo por "N" Ramos, sendo "N" o número de Nós reais do circuito. Se se tentasse criar uma malha com "N+1" Ramos, a falsa malha iria obrigatoriamente repetir um Nó real, deixando de representar um ciclo fechado sem repetição. Este último dado, apesar de simples, é interessante por não ser constante na definição trivial de uma Malha elétrica.

Seguido da elaboração de todas as combinações de Ramos, é preciso aferir quais delimitam as Malhas possíveis do circuito. A validação de Malhas recai num ciclo contínuo de análise a cada combinação produzida, analisando em primeiro lugar o seu tamanho, isto é, a quantidade de Ramos que contém essa combinação. Posto isto, o

software valida as Malhas por ordem de tamanho, começando pelas Malhas com dois Ramos até Malhas com seis Ramos.

Na validação, o exemplo de maior simplicidade dá-se na ocasião em que a combinação é constituída por somente dois Ramos. Neste cenário o algoritmo tem unicamente de comparar se os dois Nós reais dos dois Ramos integrantes da malha são os mesmos, independentemente da ordem. Como exibido na Figura 28, os dois Ramos, que são compostos por uma resistência cada, partilham dos mesmos Nós reais.

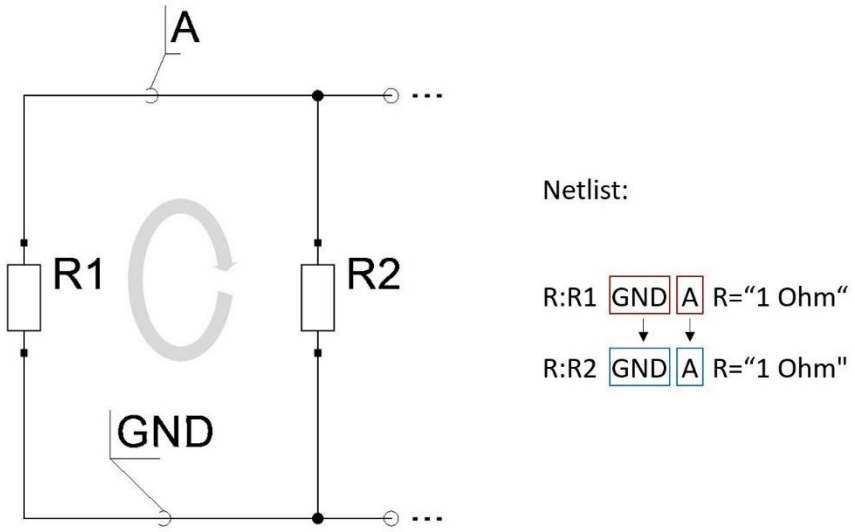


Figura 28 Exemplo de malha com dois Ramos

Após validação das Malhas mais descomplicadas, se o circuito possuir mais do que dois Nós reais, o algoritmo passa para a validação de Malhas com três e/ou mais Ramos. Neste processo, o algoritmo compara os Nós do primeiro ramo com os do segundo ramo, mas só prossegue na validação se um dos Nós reais for comum aos dois Ramos. Isto porque, se não corresponderem a nenhum dos Nós reais, significa que estes Ramos não se ligam, e, caso correspondam os dois Nós reais, significa que correspondem a uma malha já validada antecedentemente. O *script* ao encontrar este primeiro Nó real comum, guarda-o num vetor para registar o caminho percorrido da malha, sendo que antes deste fica registado o outro Nó real do primeiro ramo, portanto, aquele que ainda não foi correspondido. A partir daqui a aplicação percorre o resto da malha a validar, comparando o segundo ramo com o terceiro, e assim sucessivamente. Quando todas as correspondências entre os Nós reais dos Ramos intermédios, e se estiver a tratar do último ramo, o algoritmo precisa de confirmar se o primeiro Nó real armazenado no

vetor de caminho corresponde ao Nó real do último ramo. Para uma melhor compreensão da abordagem, o fluxograma representado na Figura 29 explica a ordem de todos os passos necessários, com um exemplo.

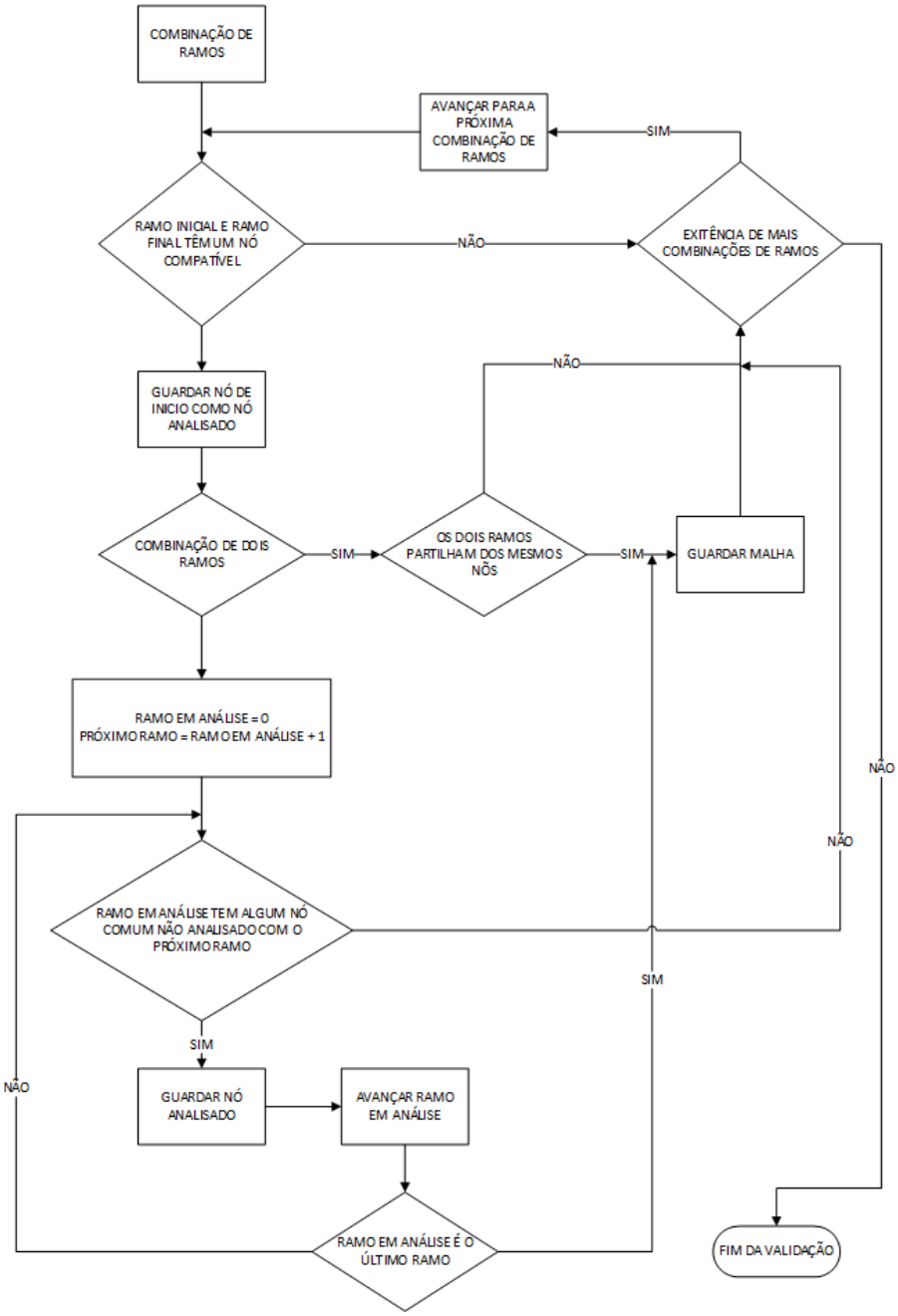


Figura 29 Fluxograma do algoritmo de validação de Malhas

Depois de validar todas as Malhas de todas as combinações de Ramos geradas, é essencial um processo de filtragem para eliminar Malhas duplicadas. O *script* apaga todas as Malhas que se repetem, por outras palavras, Malhas com os mesmos Ramos e

com sentido contrário. Desta forma, consegue-se a partir deste momento informar a aplicação o número máximo de Malhas elegíveis para o circuito submetido. É de notar que este valor encontrado não é calculável por nenhuma fórmula matemática, visto que dois circuitos com o mesmo número de Nós reais e Ramos pode ter o número máximo de Malhas possíveis diferente.

Posteriormente à validação de todas as Malhas possíveis no circuito, é fundamental eleger as necessárias para a resolução do método. Neste procedimento, estabeleceram-se algumas regras de modo a obter a combinação de Malhas mais adequada à análise. Sempre que é selecionada uma malha, esta é armazenada num vetor de Malhas selecionadas e excluída do vetor de Malhas possíveis.

Na hipótese da existência de Fontes de Corrente no circuito, o programa escolhe uma malha para cada uma, sendo que em cada uma destas só deve constar a própria como Fonte de Corrente. Após verificadas as Malhas possíveis para este caso, é elegida a mais simples, com menor número de Ramos. Escolhidas e armazenadas todas as Malhas para cada Fonte de Corrente, o algoritmo exclui da lista de Malhas possíveis todas as que contêm Ramos com Fontes de Corrente.

De seguida, o *script* começa por selecionar Malhas sem Fontes de Corrente. Caso estas não existam no circuito, é nesta fase que a processo de seleção se inicia. A escolha de Malhas é feita dentro um ciclo que apenas acaba quando todas as condições forem cumpridas. A triagem das Malhas realiza-se aleatoriamente a partir do vetor de Malhas possíveis. Quando selecionadas, segue-se uma lista de condições obrigatórias:

- Todos os Ramos do circuito devem estar incluídos em pelo menos numa malha;
- Todas as Malhas devem ter pelo menos um ramo em comum com outra malha;
- O número total de Malhas selecionadas deve ser igual ao número de Malhas necessárias.

Ao falhar em alguma das condições, o programa executa uma função de *shuffle* no vetor de Malhas possíveis, volta a escolher aleatoriamente e testa novamente as condições. Assim que todas as condições são reunidas, o vetor de Malhas selecionadas fica

completo, podendo o *script* avançar para a próxima etapa. A Figura 30 contempla o fluxograma da metodologia de seleção de Malhas.

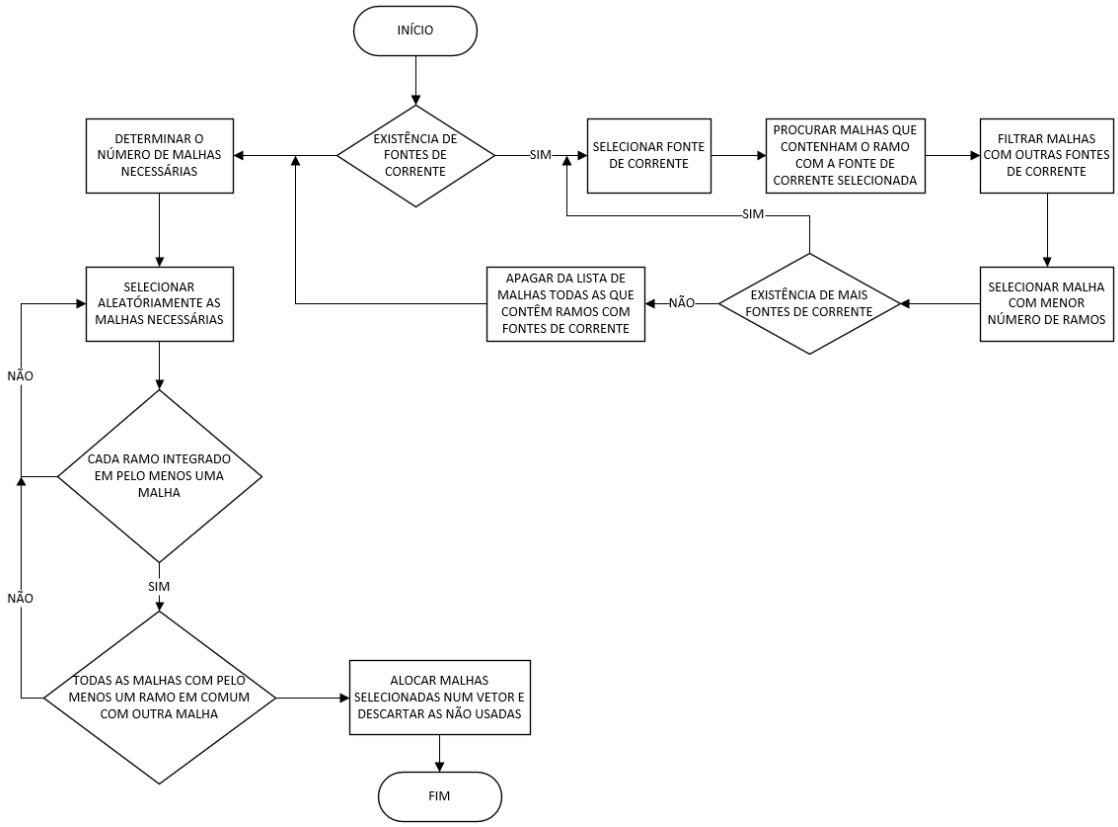


Figura 30 Fluxograma da metodologia de seleção de Malhas

4.3. CONSTRUÇÃO DAS EQUAÇÕES DE MALHA

Gerar as equações em código obriga a um elevado grau de clareza e objetividade. Com o intuito de melhorar o processo de interpretação, os resultados apresentados são organizados com um nível crescente de profundidade, começando por definir os componentes integrantes da Malha, o nome da Corrente fictícia e o seu sentido. Na definição da equação, são analisados todos os componentes constituintes e relacioná-los entre as Malhas com Ramos comuns.

Primeiramente, é fundamental criar as Correntes fictícias de cada Malha, usando o número de malha como índice de identificação. Em seguida, procede-se à definição do sentido das Correntes. Neste ponto, após várias abordagens quanto à arbitrariedade da seleção, implementou-se o seguinte processo de validação:

- Correntes nas Malhas com Fontes de Corrente têm o mesmo sentido que as Fontes;

- Correntes em Malhas sem Fontes de Corrente têm sentido aleatório;

- Sentido das Correntes é definido componente a componente ao invés de ramo a ramo;

- Criação de vetor com o caminho da Corrente, descartando Nós virtuais "_net" das ligações intermédias entre componentes do mesmo ramo para não confundir o utilizador;

É importante ter em mente que, no último passo da validação exposta acima, refere-se a ligações entre componentes do mesmo Ramo e não de ligações entre Ramos, visto que estas são necessárias constar no vetor de caminho para posterior utilização no processo de definição de equações

Na Figura 32 é exibido o produto na consola de desenvolvimento, o resultado dos vetores que definem as Correntes das Malhas do circuito da Figura 31.

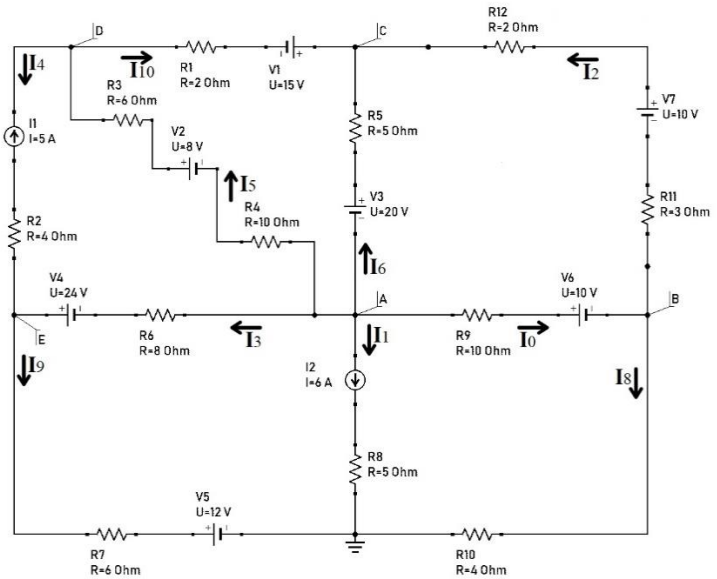


Figura 31 Circuito exemplo

Correntes de Malha script.js:3051

script.js:3052

(index)	Value
0	"i0"
1	"i1"
2	"i2"
3	"i3"
4	"i4"

▶ Array(5)

Caminho da Corrente sem "_net" script.js:3053

script.js:3054

(index)	0	1	2	3	4	5
0	"A"	"E"	"D"	"_net2"		
1	"E"	"A"	"gnd"			
2	"A"	"B"	"_net8"	"C"		
3	"D"	"_net2"	"A"	"C"		
4	"gnd"	"B"	"_net8"	"C"	"A"	"E"

▶ Array(5)

Componentes de cada Malha script.js:3055

script.js:3056

(index)	0	1	2	3	4	5	6	7	8	9
0	"R6"	"V4"	"R2"	"I1"	"R3"	"V2"	"R4"			
1	"V5"	"R7"	"V4"	"R6"	"I2"	"R8"				
2	"R9"	"V6"	"R11"	"V7"	"R12"	"R5"	"V3"			
3	"R3"	"V2"	"R4"	"V3"	"R5"	"V1"	"R1"			
4	"R10"	"R11"	"V7"	"R12"	"R5"	"V3"	"R6"	"V4"	"R7"	"V5"

▶ Array(5)

Figura 32 Exemplo da estruturação de dados das Correntes

Posteriormente ao tratamento das Correntes das Malhas, é necessário definir o sentido das Fontes de Tensão para cada Malha. Para o efeito, compara-se o caminho da Corrente da Malha que contém a Fonte de Tensão e, caso os Nós virtuais negativo e positivo da Fonte estejam seguidos no vetor, respetivamente, então considera-se a Fonte de Tensão positiva por estar em sentido com a Malha. Caso a verificação anterior falhar, significa que a Fonte está em sentido contrário.

Na hipótese de estarem incluídas mais do que uma Fonte de Tensão na mesma Malha, estas são reunidas numa *string* com o sinal de soma ou subtração, dependo das suas orientações em relação à Malha que as engloba. No final, estas são armazenadas num vetor auxiliar.

De seguida, procede-se para a metodologia de associação de resistências, condensadores e bobinas às Correntes fictícias. O processo complica-se quando os componentes integram mais do que uma Malha, acabando por obrigar a comparar os sentidos das Correntes de Malha entre as Malhas. Para tal foi preciso seguir algumas tarefas em série, retratadas no fluxograma da Figura 33.

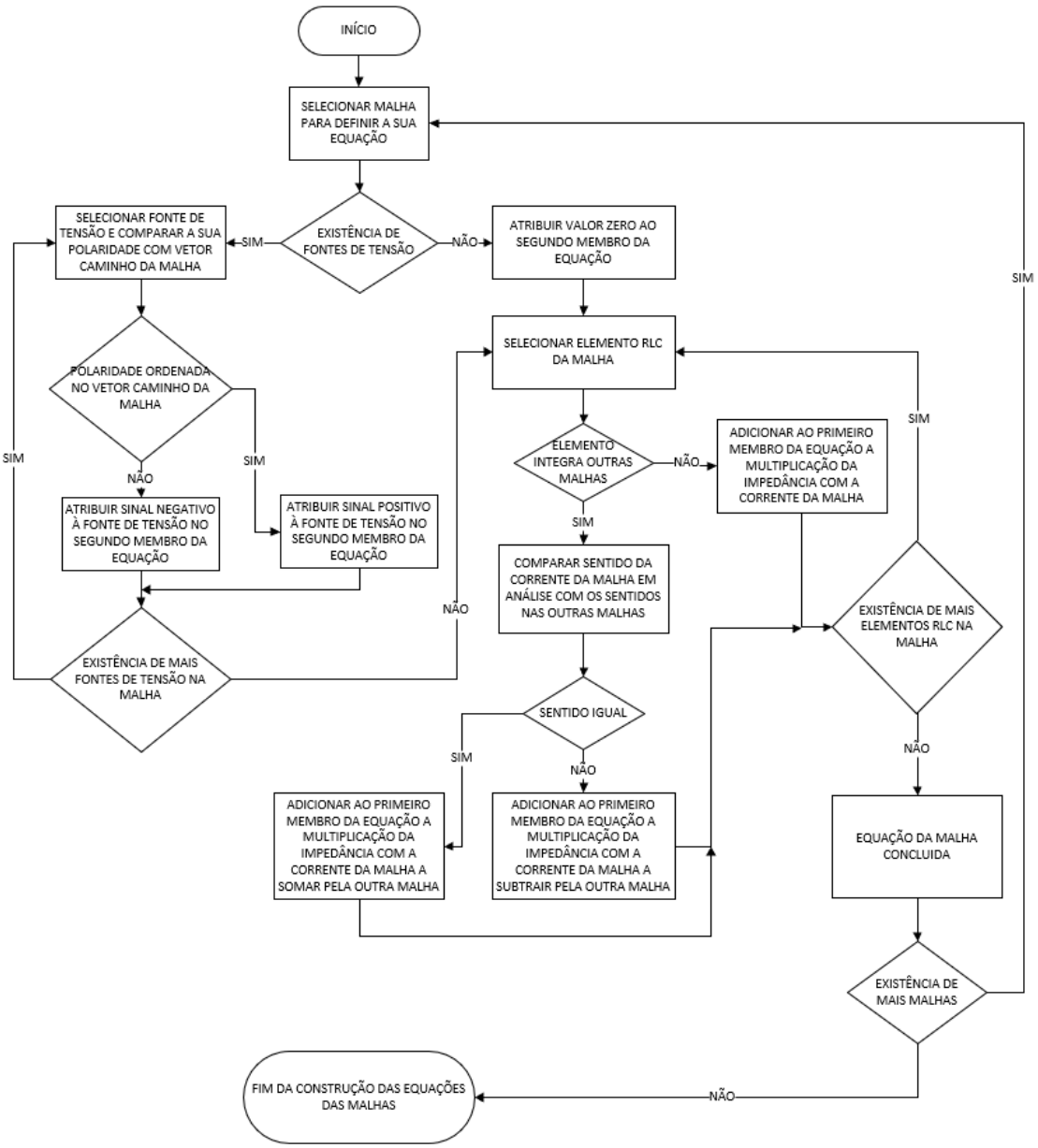


Figura 33 Fluxograma de método de construção de equações de Malha

Finalmente, reúnem-se todos os dados recolhidos das operações mencionadas anteriormente, igualando na *string* a parte da associação de resistências, condensadores e bobinas às Correntes fictícias com as forças eletromotrizes. A Figura 34 retrata um exemplo de uma equação de Malha.

Malha 3: I_{33}

Composta por R10,V6,R9,R6,V4,R7,V5 - Sentido de gnd → B com início em R10

$$R6 \cdot (I_{33} - 1) + R7 \cdot (I_{33} - 6) + R9 \cdot (-I_{22} + I_{33}) + R10 \cdot I_{33} = -V5 + V4 + V6$$

Figura 34 Exemplo de equação de Malha

4.4. RESOLUÇÃO DE SISTEMA DE EQUAÇÕES

Posteriormente à implementação do método de elaboração das equações das Malhas, considerou-se mandatório adicionar um novo objetivo ao projeto proposto, que previa somente a apresentação de equações. Com a finalidade de tornar a aplicação ainda mais prestável, desenvolveu-se um método de cálculo para os sistemas de equações de malha.

Depois de alguma pesquisa, percebeu-se que a ferramenta apropriada para a situação seria a biblioteca *Nerdamer* [17]. Esta é provavelmente a melhor e mais completa biblioteca disponível no momento para *JavaScript*. Através da quantidade de documentação disponível, foi possível desenvolver o processo de preparação das *strings* para posterior adaptação com o *solver*. Alguns exemplos de operações com a biblioteca são representados na Figura 35.

```
var sol = nerdamer.solveEquations('x^3+8=x^2+6','x');
console.log(sol.toString());
//1+i,-i+1,-1
```

```
var e = nerdamer.solveEquations('x^2+4-y', 'y');
console.log(e[0].text());
//4+x^2
```

```
var sol = nerdamer.solveEquations('x^2+8+y=x+6','x');
console.log(sol.toString());
//0.5*((-4*y-7)^0.5+1),0.5*(-(-4*y-7)^0.5+1)
```

```
var sol = nerdamer.solveEquations('cos(x)+cos(3*x)=1','x');
console.log(sol.toString());
//5.7981235959208695,0.4850617112587174
```

Figura 35 Exemplos de manipulação da biblioteca *Nerdamer* [18]

Durante a definição das equações das Malhas, o algoritmo aproveita e copia todas as *strings* geradas, substituindo o nome dos componentes pelos seus valores constantes na matriz inicial. De seguida, trata-se de dividir a *string* do sistema de equações por múltiplas *strings* correspondentes a cada equação de Malha.

Uma ferramenta fulcral desta biblioteca foi o facto de esta permitir definir o número imaginário na notação comum em EE, aceitando desta forma as variáveis de Corrente de Malha "i_xx" onde "xx" corresponde ao ID da sua Malha.

```
nerdamer.set('IMAGINARY', 'j');
x = nerdamer('sqrt(-1)');
console.log(x.toString());
```

Figura 36 Exemplo de operação com *Nerdamer*

Para realizar a resolução dos sistemas de equações, o *Nerdamer*, após a chamada da função "nerdamer.solveEquations()", retorna o vetor com os valores das Correntes das Malhas.

Após as sucessivas tentativas, sem sucesso, na resolução de equações com números complexos, no caso da análise de circuitos em Corrente Alternada, colaborando com a equipa de desenvolvedores da biblioteca, optou-se por incluir esta operação apenas na análise de circuitos em Corrente Contínua.

4.5. APRESENTAÇÃO DE EQUAÇÕES EM LATEX

Com a intenção de oferecer a melhor solução de apresentação possível, foi necessário ajustar a forma como se exporia os resultados da análise. Descobriu-se a biblioteca *KaTeX* [19], que permite adaptar o formato de escrita *LaTeX* [20], através de implementação *JavaScript*, para páginas HTML.

Este passo foi conseguido graças à funcionalidade "nerdamer.toTeX()" da ferramenta *Nerdamer*, que transforma qualquer *string* de formato de texto simples em formato *LaTeX*.

Na Figura 37, está representada uma comparação entre o *output* obtido em formato de texto e em configuração *LaTeX* [25].

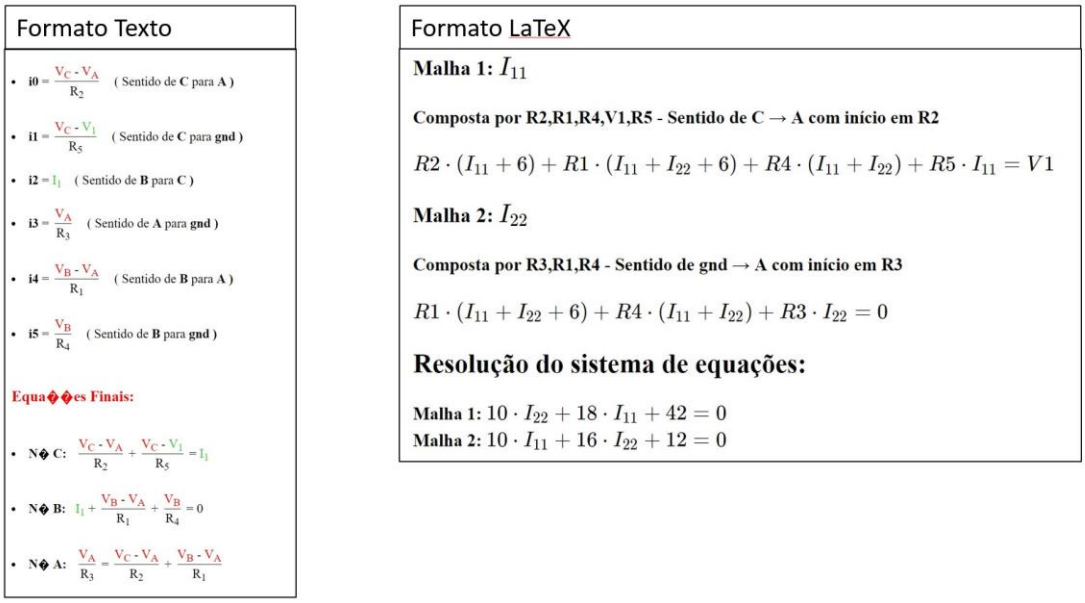


Figura 37 Comparação de *output* em formato de texto e LaTeX

A principal vantagem no uso desta ferramenta apoia-se nos factos de permitir uma renderização síncrona na página HTML de resultados, não necessitar de dependências (outras bibliotecas) e ser eficiente a processar *strings* formatadas em *LaTeX* para a devida apresentação gráfica.

4.6. ADAPTAÇÃO À VERSÃO MODIFICADA DO QUCS

Sabendo da existência da implementação de novas funcionalidades no QUCS por parte de estudantes do ISEP, foi imprescindível acertar o algoritmo de análise. No intuito de proporcionar uma utilização da ferramenta de forma descontraída, sem que o utilizador se tenha de preocupar em perceber qual a versão do QUCS, é indispensável a compatibilidade da aplicação entre ambas as versões.

A Figura 38 retrata o processo de criação da resistência interna de uma Fonte de Tensão, funcionalidade que não está disponível por defeito no simulador em questão. Para contornar esta questão, foi preciso estudar a forma como esta resistência influenciava a *netlist* produzida.

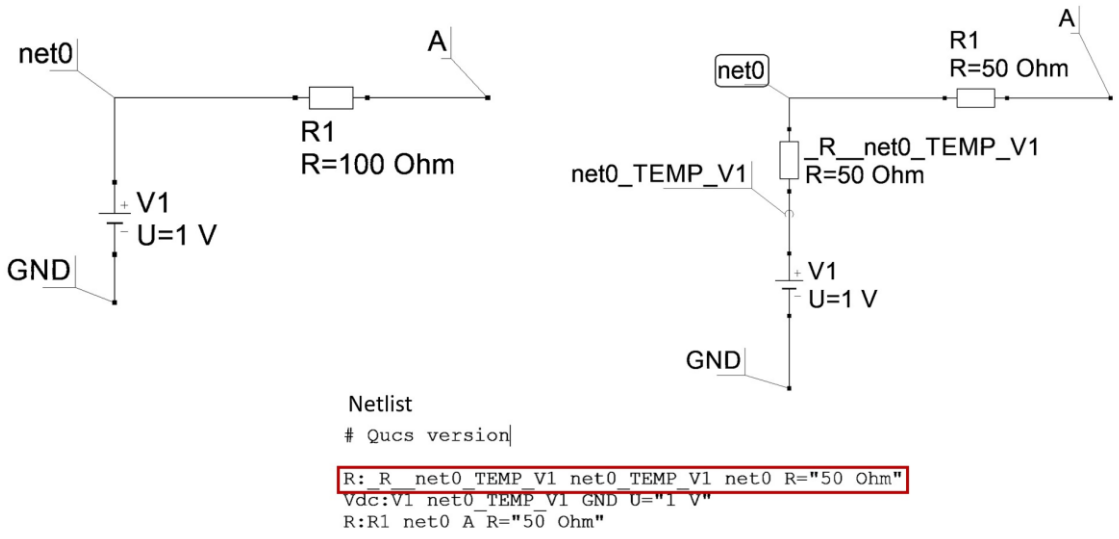


Figura 38 Exemplo da netlist gerada na versão modificada

As resistências internas das Fontes são caracterizadas de forma diferente, para posterior diferenciação na análise interna do QUCS. Captando o exemplo da Figura 38, o

processo da criação da resistência interna "_R_net0_TEMP_V1" dá-se pela seguinte ordem:

- O código "_R_" é usado para diferençar as resistências internas das Fontes;
- O pedaço "_net0_" refere o Nó original intermédio entre a Fonte e o componente que a segue, no seu Nó positivo;
- O código "TEMP" confirma o Nó criado pela resistência interna, neste caso, "_net0_TEMP";
- O código "_V1" refere-se à Fonte a que resistência interna pertence.

O algoritmo desenvolvido foi munido de uma técnica de filtragem capaz de resolver o problema pela sua raiz. Na aquisição do primeiro Nó das Fontes, que corresponde ao Nó positivo, se se detetar a sequência "_TEMP", este automaticamente guarda apenas as coordenadas lidas entre o primeiro *underscore* e o segundo. Usando mais uma vez o exemplo acima mencionado, na obtenção da coordenada, como é detetada a sequência "_TEMP", o algoritmo guarda exclusivamente a informação "_net0".

5. EXEMPLOS DE RESULTADOS DA APLICAÇÃO

Este capítulo é dedicado à demonstração da aplicação, de forma a explorar ao máximo as suas capacidades, com a intenção de comprovar o seu bom funcionamento. Os circuitos aqui estudados serão parte complementar na aplicação, não só para que numa fase inicial possa atrair os estudantes para a sua utilização, mas também para que a aplicação U=RISOLVE seja vista como uma ferramenta de autoaprendizagem de uso simples. Para tal, torna-se imperial fornecer aos alunos um ponto de partida. Assim, todos os circuitos facultados são acompanhados do seu esquemático do QUCS, a *netlist* do circuito e uma imagem do circuito.

5.1. CASOS-EXEMPLO EM CORRENTE CONTÍNUA

Os circuitos apresentados foram ordenados por ordem de complexidade, tanto de dimensão como de resolução, oferecendo desta forma um método de estudo sistematizado. O primeiro circuito encontra-se representado na Figura 39 e o resultado obtido através da plataforma U=RISOLVE na Figura 40.

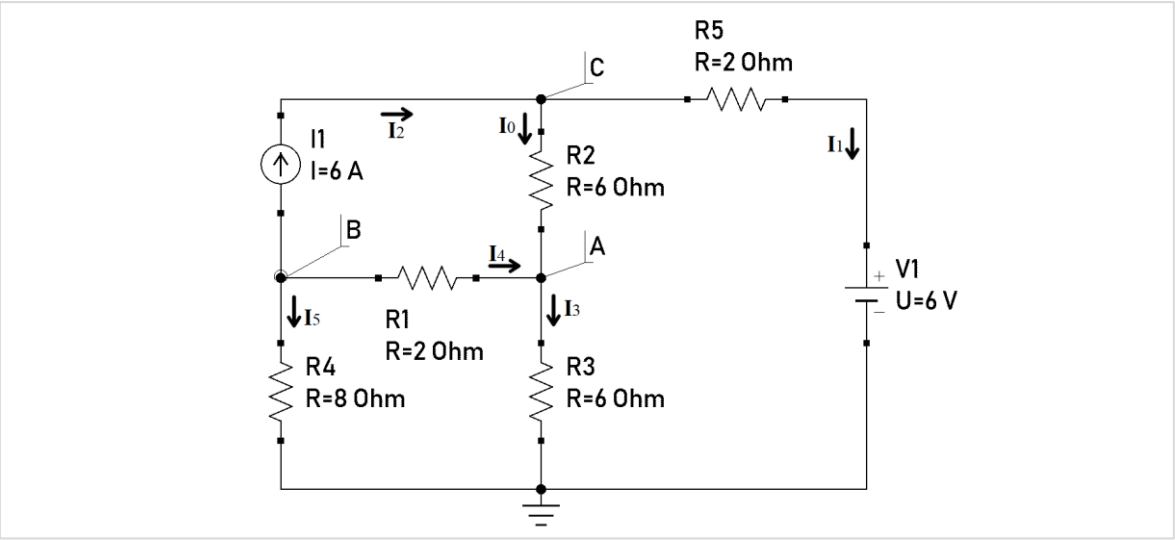


Figura 39 Circuito de nível básico em Corrente Continua

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 6 | Nós (N): 4 | Fontes de Corrente (FC): 1

Existem 7 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 6 - (4 - 1) - 1 = 2 Malhas Independentes;**
- **FC = 1 Malha Auxiliar.**

Equações das Malhas:

Malha 0 (Auxiliar): I_{00}

Composta por R2,R1,I1 - Sentido da Fonte de Corrente I1

I1 $= I_{00} = 6\,\text{A}$

Malha 1: I_{11}

Composta por R2,R3,V1,R5 - Sentido de C → A com início em R2

$$R3 \cdot (-I_{22} + I_{11}) + R2 \cdot (I_{11} + 6) + R5 \cdot I_{11} = V1$$

Malha 2: I_{22}

Composta por R3,R1,R4 - Sentido de gnd → A com início em R3

$$R3 \cdot (-I_{11} + I_{22}) + R1 \cdot (I_{22} + 6) + R4 \cdot I_{22} = 0$$

Sistema de equações:

Malha 1: $-6 \cdot I_{22} + 14 \cdot I_{11} + 30 = 0$
Malha 2: $-6 \cdot I_{11} + 16 \cdot I_{22} + 12 = 0$

Resultados das Correntes de Malha:

$I_{11} = -2,936\,\text{A}$
$I_{22} = -1,851\,\text{A}$

Figura 40 Solução MCM do circuito de nível básico em Corrente Continua

Na Figura 41, encontra-se representado o segundo circuito em Corrente Continua, com nível de complexidade intermédio. A solução pela análise MCM para o respetivo circuito está demonstrada na Figura 42.

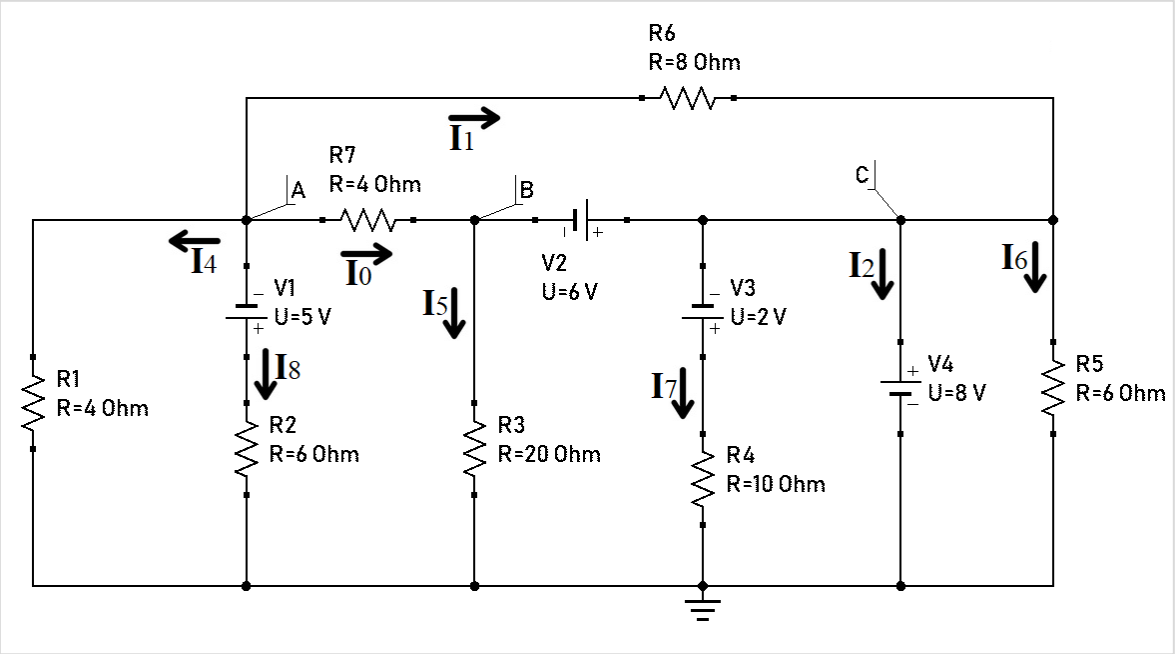


Figura 41 Circuito de nível intermédio em Corrente Continua

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 9 | Nós (N): 4 | Fontes de Corrente (FC): 0

Existem 27 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 9 - (4 - 1) - 0 = 6 Malhas Independentes.**

Equações das Malhas:

Malha 0: I_{00}

Composta por R7,R6,V2 - Sentido de B → A com início em R7

$$R7 \cdot (-I_{22} + I_{00}) + R6 \cdot (I_{00} + I_{55}) = -V2$$

Malha 1: I_{11}

Composta por R5,R4,V3 - Sentido de C → gnd com início em R5

$$R5 \cdot (I_{11} + I_{44} + I_{55}) + R4 \cdot I_{11} = -V3$$

Malha 2: I_{22}

Composta por R1,R7,R3 - Sentido de gnd → A com início em R1

$$R7 \cdot (-I_{00} + I_{22}) + R1 \cdot (-I_{33} + I_{22} + I_{55}) + R3 \cdot I_{22} = 0$$

Malha 3: I_{33}

Composta por R1,R2,V1 - Sentido de A → gnd com início em R1

$$R1 \cdot (-I_{22} - I_{55} + I_{33}) + R2 \cdot I_{33} = -V1$$

Malha 4: I_{44}

Composta por R5,V4 - Sentido de C → gnd com início em R5

$$R5 \cdot (I_{11} + I_{44} + I_{55}) = V4$$

Malha 5: I_{55}

Composta por R1,R6,R5 - Sentido de gnd → A com início em R1

$$R1 \cdot (-I_{33} + I_{22} + I_{55}) + R6 \cdot (I_{00} + I_{55}) + R5 \cdot (I_{11} + I_{44} + I_{55}) = 0$$

Sistema de equações:

Malha 0: $-4 \cdot I_{22} + 12 \cdot I_{00} + 8 \cdot I_{55} + 6 = 0$
Malha 1: $16 \cdot I_{11} + 6 \cdot I_{44} + 6 \cdot I_{55} + 2 = 0$
Malha 2: $-4 \cdot I_{00} - 4 \cdot I_{33} + 28 \cdot I_{22} + 4 \cdot I_{55} = 0$
Malha 3: $-4 \cdot I_{22} - 4 \cdot I_{55} + 10 \cdot I_{33} + 5 = 0$
Malha 4: $6 \cdot I_{11} + 6 \cdot I_{44} + 6 \cdot I_{55} - 8 = 0$
Malha 5: $-4 \cdot I_{33} + 18 \cdot I_{55} + 4 \cdot I_{22} + 6 \cdot I_{11} + 6 \cdot I_{44} + 8 \cdot I_{00} = 0$

Resultados das Correntes de Malha:

$I_{00} = 3,458\,\text{A}$
$I_{11} = -1,000\,\text{A}$
$I_{22} = 0,800\,\text{A}$
$I_{33} = -2,395\,\text{A}$
$I_{44} = 7,870\,\text{A}$
$I_{55} = -5,537\,\text{A}$

Figura 42 Solução MCM do circuito de nível intermédio em Corrente Continua

Para finalizar a demonstração da aplicação para circuitos em Corrente Contínua, apresenta-se o circuito da Figura 43, correspondente ao nível avançado, juntamente com a solução resultante da análise MCM representada na Figura 44.

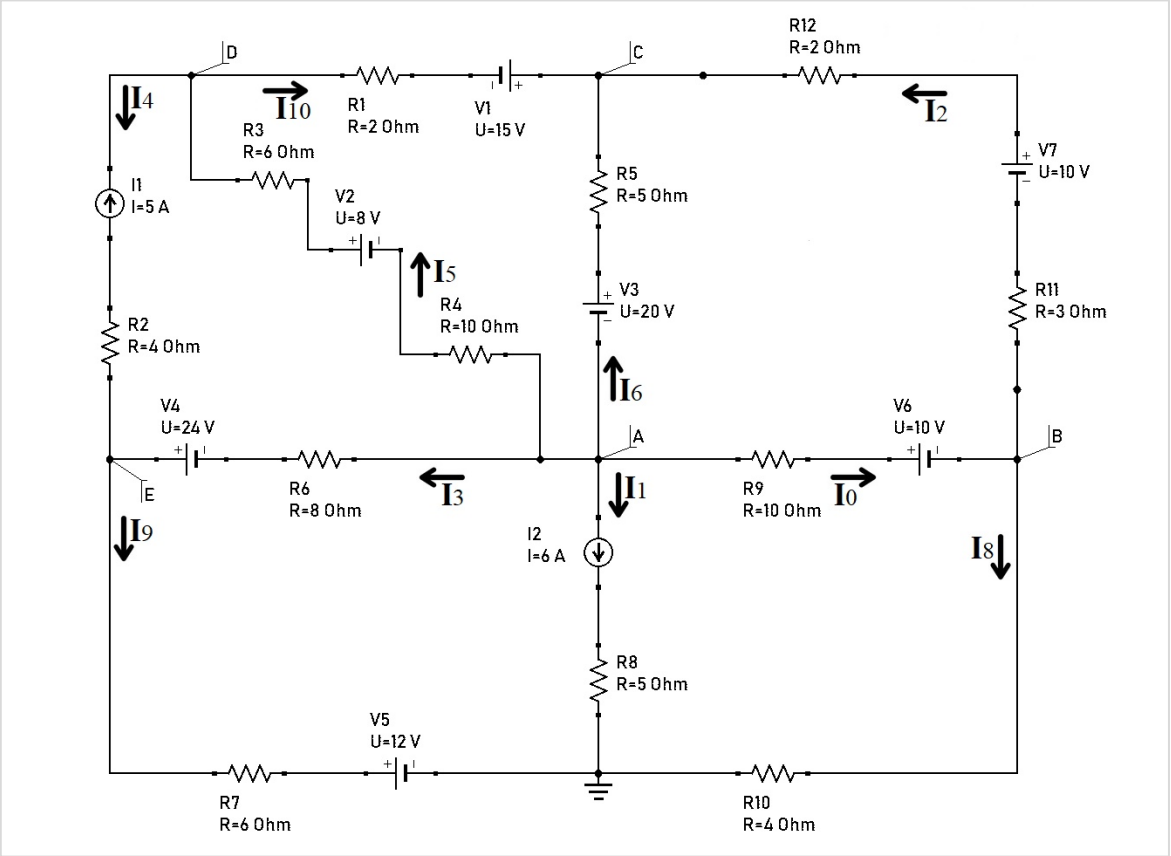


Figura 43 Circuito de nível avançado em Corrente Continua

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 10 | Nós (N): 6 | Fontes de Corrente (FC): 2

Existem 16 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 10 - (6 - 1) - 2 = 3 Malhas Independentes;**
- **FC = 2 Malhas Auxiliares.**

Equações das Malhas:

Malha 0 (Auxiliar): I_{00}

Composta por R6,V4,R2,I1,R3,V2,R4 - Sentido da Fonte de Corrente I1

I1 $= I_{00} = 5\ \text{A}$

Malha 1 (Auxiliar): I_{11}

Composta por V5,R7,V4,R6,I2,R8 - Sentido da Fonte de Corrente I2

I2 $= I_{11} = 6\ \text{A}$

Malha 2: I_{22}

Composta por R9,V6,R11,V7,R12,V1,R1,R3,V2,R4 - Sentido de A → B com início em R9

$$R3 \cdot (I_{22} + I_{44} + 5) + R4 \cdot (I_{22} + I_{44} + 5) + R11 \cdot (I_{22} + I_{33}) + R12 \cdot (I_{22} + I_{33}) + R1 \cdot (I_{22} + I_{44}) + R9 \cdot I_{22} = -V1 - V2 - V6 + V7$$

Malha 3: I_{33}

Composta por R10,R11,V7,R12,R5,V3,R6,V4,R7,V5 - Sentido de gnd → B com início em R10

$$R6 \cdot (I_{33} - 1) + R7 \cdot (I_{33} - 6) + R5 \cdot (-I_{44} + I_{33}) + R11 \cdot (I_{22} + I_{33}) + R12 \cdot (I_{22} + I_{33}) + R10 \cdot I_{33} = -V3 - V5 + V4 + V7$$

Malha 4: I_{44}

Composta por R3,V2,R4,V3,R5,V1,R1 - Sentido de D → A com início em R4

$$R5 \cdot (-I_{33} + I_{44}) + R3 \cdot (I_{22} + I_{44} + 5) + R4 \cdot (I_{22} + I_{44} + 5) + R1 \cdot (I_{22} + I_{44}) = -V1 - V2 + V3$$

Sistema de equações:

Malha 2: $18 \cdot I_{44} + 33 \cdot I_{22} + 5 \cdot I_{33} + 103 = 0$
Malha 3: $-5 \cdot I_{44} + 28 \cdot I_{33} + 5 \cdot I_{22} - 46 = 0$
Malha 4: $-5 \cdot I_{33} + 18 \cdot I_{22} + 23 \cdot I_{44} + 83 = 0$

Resultados das Correntes de Malha:

$I_{22} = -2,964\ \text{A}$
$I_{33} = 2,020\ \text{A}$
$I_{44} = -0,850\ \text{A}$

Figura 44 Solução MCM do circuito nível avançado em Corrente Continua

5.2. CASOS-EXEMPLO EM CORRENTE ALTERNADA

Agora em Corrente Alternada, os circuitos foram novamente dispostos por ordem de complexidade, lembrando que nestes, o processo de resolução das Correntes de Malha foi excluído. O primeiro circuito encontra-se representado na Figura 45 e o resultado obtido através da plataforma U=RISOLVE na Figura 46.

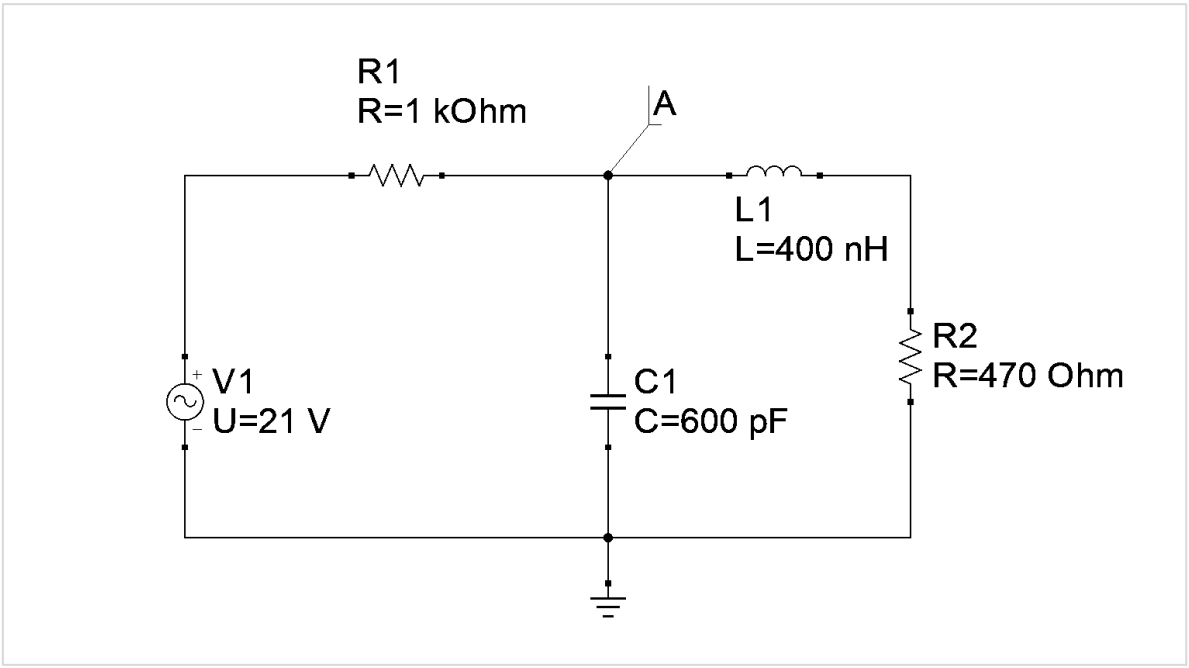


Figura 45 Circuito de nível básico em Corrente Alternada

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 3 | Nós (N): 2 | Fontes de Corrente (FC): 0

Existem 3 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 3 - (2 - 1) - 0 = 2 Malhas Independentes.**

Equações das Malhas:

Malha 0: I_{00}

Composta por C1,R2,L1 - Sentido de A → gnd com início em C1

$$\underline{Z}_{C1} \cdot (I_{00} + I_{11}) + \underline{Z}_{L1} \cdot I_{00} + R2 \cdot I_{00} = 0$$

Malha 1: I_{11}

Composta por C1,V1,R1 - Sentido de A → gnd com início em C1

$$\underline{Z}_{C1} \cdot (I_{00} + I_{11}) + R1 \cdot I_{11} = V1$$

Figura 46 Solução MCM do circuito de nível básico em Corrente Alternada

Na Figura 47, encontra-se representado o segundo circuito em Corrente Alternada, com nível de complexidade intermédio. A solução pela análise MCM para o respetivo circuito está demonstrada na Figura 48.

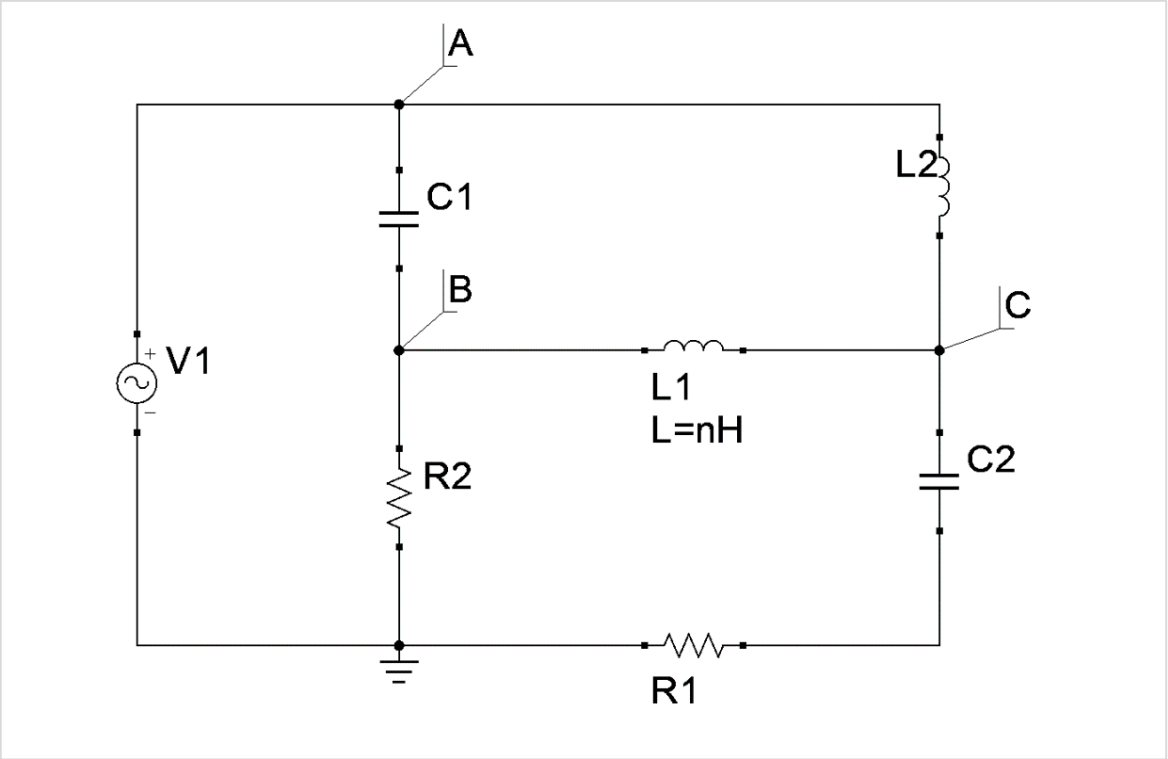


Figura 47 Circuito de nível intermédio em Corrente Alternada

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 6 | Nós (N): 4 | Fontes de Corrente (FC): 0

Existem 7 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 6 - (4 - 1) - 0 = 3 Malhas Independentes.**

Equações das Malhas:

Malha 0: I_{00}

Composta por C1,L1,L2 - Sentido de A → B com início em C1

$$\underline{Z}_{C1} \cdot (-I_{11} + I_{00}) + \underline{Z}_{L2} \cdot (I_{00} + I_{22}) + \underline{Z}_{L1} \cdot I_{00} = 0$$

Malha 1: I_{11}

Composta por R2,C1,V1 - Sentido de gnd → B com início em R2

$$\underline{Z}_{C1} \cdot (-I_{00} + I_{11}) + R2 \cdot I_{11} = -V1$$

Malha 2: I_{22}

Composta por R1,C2,L2,V1 - Sentido de gnd → C com início em C2

$$\underline{Z}_{L2} \cdot (I_{00} + I_{22}) + \underline{Z}_{C2} \cdot I_{22} + R1 \cdot I_{22} = -V1$$

Figura 48 Solução MCM do circuito de nível intermédio em Corrente Continua

Para concluir, apresenta-se o circuito da Figura 49, correspondente ao nível avançado para a análise em Corrente Alternada, juntamente com a solução resultante da análise MCM representada na figura 50.

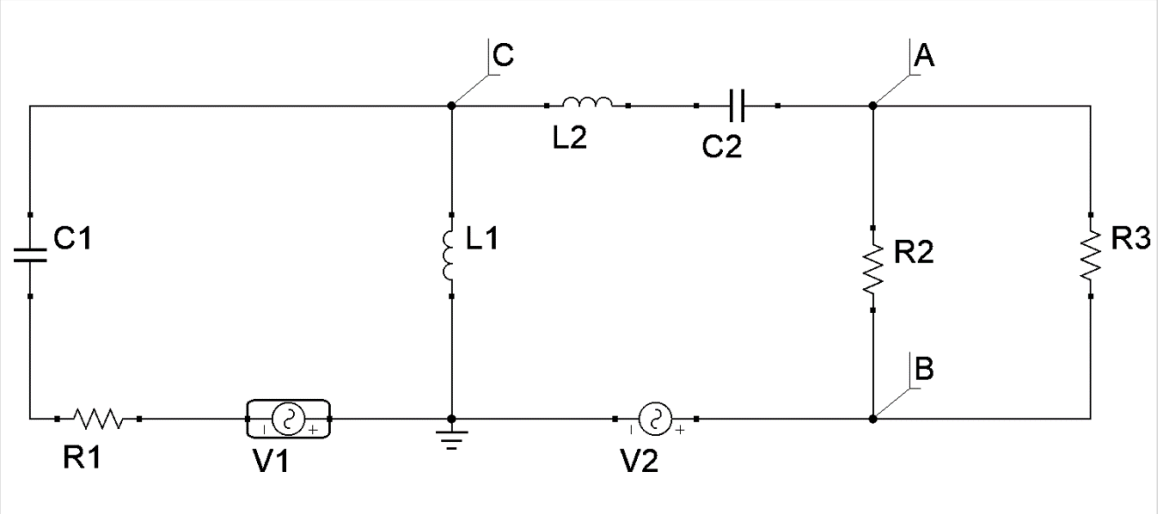


Figura 49 Circuito de nível avançado em Corrente Alternada

Determinação das variáveis fundamentais do circuito (para o MCM):

Ramos (R): 6 | Nós (N): 4 | Fontes de Corrente (FC): 0

Existem 6 combinações de Malhas possíveis no circuito.

São necessárias:

- **R - (N - 1) - FC = 6 - (4 - 1) - 0 = 3 Malhas Independentes.**

Equações das Malhas:

Malha 0: I_{00}

Composta por L1,V1,R1,C1 - Sentido de C → gnd com início em L1

$$\underline{Z}_{L1} \cdot (I_{00} + I_{22}) + \underline{Z}_{C1} \cdot I_{00} + R1 \cdot I_{00} = -V1$$

Malha 1: I_{11}

Composta por R2,R3 - Sentido de A → B com início em R2

$$R2 \cdot (-I_{22} + I_{11}) + R3 \cdot I_{11} = 0$$

Malha 2: I_{22}

Composta por R2,C2,L2,L1,V2 - Sentido de B → A com início em R2

$$R2 \cdot (-I_{11} + I_{22}) + \underline{Z}_{L1} \cdot (I_{00} + I_{22}) + \underline{Z}_{C2} \cdot I_{22} + \underline{Z}_{L2} \cdot I_{22} = V2$$

Figura 50 Solução MCM do circuito nível avançado em Corrente Alternada

A aplicação U=RISOLVE inclui este conjunto de circuitos exemplo, permitindo ao utilizador usufruir da ferramenta com maior facilidade, tendo disponível exemplos práticos para entender como deve proceder para produzir as suas próprias *netlists*.

Através dos mesmos exemplos de *netlists* facultadas, estes poderão ser testados tanto pelo MCM como pelo MTN, valorizando assim o *software* desenvolvido atribuindo-lhe uma maior versatilidade.

6. CONCLUSÕES

Ao longo do desenvolvimento deste projeto, pela sua complexidade, tornou-se obrigatório esboçar uma planificação adequada das etapas a elaborar, de modo a alcançar o resultado esperado cumprindo com os objetivos propostos.

Sendo o QUCS o simulador de circuitos elétricos de eleição dos estudantes de Engenharia Eletrotécnica no ISEP, este programa foi desenvolvido numa Corrente de desenvolvimento contínuo por parte de alunos da Licenciatura em Engenharia Eletrotécnica e Computadores [1][2][3][4], que visa introduzir novas funcionalidades ao *software* QUCS, para auxiliar os alunos no seu processo de aprendizagem e beneficiar o seu ensino. Assim sendo, a Aplicação para Geração da Metodologia/Equações do Método das Correntes nas Malhas, com suporte do QUCS, integra a plataforma U=RISOLVE que contempla a Aplicação Metodologia/Equações do Método das Tensões nos Nós [4], englobando também ferramentas de aquisição de dados adaptadas à versão alterada e melhorada do QUCS. Desta forma, a aplicação permite receber tanto ficheiros de texto *netlist* gerados pela versão oficial como pela versão transformada do QUCS.

Com fundamento na pesquisa aprofundada relativa à produção do ficheiro de *netlist* por parte do QUCS, foi possível melhorar a técnica de aquisição de dados, desenvolvendo novas formas de filtrar a informação necessária, como a obtenção de valores de

componentes com prefixos do SI e a filtragem de resistência internas das Fontes de Tensão. Uma potencial evolução ao projeto elaborado passaria por adaptar as técnicas de análise e aquisição a outros simuladores existentes, estudando a forma como estes criam as suas *netlists*.

O desenvolvimento de uma aplicação capaz de escrever as equações das Malhas obrigou a otimizar a informação adquirida na análise pelo método para poder converter este conhecimento num algoritmo prático e implementável. A grande dificuldade passou por decifrar qual a melhor abordagem à sua implementação. Após variadas tentativas de conceção do algoritmo indicado para o efeito, surgiu a ideia de montar combinações de Ramos, de dois até ao número de Nós do circuito, de forma a obter um vetor com todas as possibilidades de ligação entre os Ramos. De seguida, estas combinações são sujeitas a um processo de validação e seleção automática.

A última fase do desenvolvimento baseou-se na necessidade de entregar a resolução ao utilizador da forma mais descritiva e apelativa, com os resultados das equações propostas. Para isto, fez-se uso das bibliotecas *Nerdamer* [17] e *KaTeX* [18], para permitir a resolução dos eventuais sistemas de equações construídos pelo algoritmo e possibilitar a representação de todas as equações e operações em formato *LaTeX* [19], respetivamente.

Os principais objetivos deste projeto foram cumpridos com sucesso, tendo em conta que, o *output* inicialmente previsto, considerava somente a apresentação das equações das Correntes de Malha. No entanto, para além deste feito, a ferramenta cumpre um propósito extra ao mostrar o resultado para o sistema de equações definido, poupando o utilizador do trabalho de verificação de resultados.

O programa foi implementado com a finalidade de o partilhar na plataforma de hospedagem de código-fonte GitHub [16], de maneira a permitir o seu aproveitamento noutros projetos relacionados com o tema, permitindo outros desenvolvedores servirem-se de rotinas testadas de aquisição de dados a partir de ficheiros *netlist*, algoritmos depurados de análise e de exibição de resultados e, especialmente, do método de geração de Malhas e respetiva validação das mesmas.

Aliado ao facto de não existir nenhuma ferramenta que defina Malhas e conceba as suas equações automaticamente, este projeto revelou-se bastante gratificante no ponto de vista de desenvolvedor, por se refletir como uma ferramenta útil e única.

Durante a implementação do algoritmo, foi necessário estudar e investigar formas de contornar o problema da definição automática de Malhas a partir apenas de um ficheiro de texto. Nesta pesquisa, descobriu-se que o problema em questão figurava um problema definido na teoria da complexidade computacional designado NP-*hard* [20] (NP-difícil ou NP-complexo). Uma eventual otimização ao processo desenvolvido na geração de Malhas, passaria por aplicar soluções estudadas na teoria dos grafos, ao escrever um algoritmo *Spanning Tree Protocol* (STP), basando-se no algoritmo de busca em largura *Breadth-First Search* (BFS).

Por fim, mas não menos importante, este projeto permitiu desenvolver diferentes capacidades de natureza técnica, pela análise de circuitos, manipulação aprofundada do simulador QUCS e programação *web*, despertando um especial interesse pela linguagem de programação JavaScript, que representa no momento a maior comunidade de programação mundial, como de natureza pessoal, no desafio que representou a sua coordenação, planeamento e execução.

Referências Documentais

[1] MACEDO, Pedro – *Análise Comparativa e Melhoramento de Simuladores de Circuitos Elétricos*. Dissertação de Licenciatura em Engenharia Eletrotécnica e Computadores orientada pelo Eng.º Mário Alves e apresentada no Instituto Superior de Engenharia do Instituto Politécnico do Porto, em 2015.

[2] MARTINS, Nelson – *Desenvolvimento de Módulos para o Qucs*. Dissertação de Licenciatura em Engenharia Eletrotécnica e Computadores orientada pelo Eng.º Mário Alves e apresentada no Instituto Superior de Engenharia do Instituto Politécnico do Porto, em 2016.

[3] SILVA, Alberto – *Design, Implementation and Integration of new Functionalities in the QUCS Simulator*. Dissertação de Licenciatura em Engenharia Eletrotécnica e Computadores orientada pelos Eng. Mário Alves e Francisco Pereira e apresentada no Instituto Superior de Engenharia do Instituto Politécnico do Porto, em 2017.

[4] SOUSA, Lino – *U=RISOLVE Aplicativo de Análise de Circuitos Elétricos Simulados no QUCS*. Dissertação de Licenciatura em Engenharia Eletrotécnica e Computadores orientada pelo Eng.º Mário Alves e Francisco Pereira e apresentada no Instituto Superior de Engenharia do Instituto Politécnico do Porto, em 2018.

[5] EASYEDA – EasyEDA, [visitado em agosto de 2019]; https:// www.easyeda.com/

[6] CIRCUITLAB – CircuitLab, [visitado em agosto de 2019]; https://www.circuitlab.com/

[7] EASYEDA – EasyEDA, [visitado em agosto de 2019]; https:// www.easyeda.com/

[8] EVERYCIRCUIT – EveryCircuit, [visitado em agosto de 2019]; http:// www.everycircuit.com/

[9] GITHUB – KTechLab, [visitado em agosto de 2019]; https:// www.github.com/ktechlab/ktechlab/

[10] ANALOG DEVICES – LTspice, [visitado em agosto de 2019]; https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

[11] NATIONAL INSTRUMENTS – Multisim, [visitado em agosto de 2019]; https://www.ni.com/pt-pt/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim.html

[12] GITHUB – Oregano, [visitado em agosto de 2019]; https://www.github.com/drahnr/oregano/

[13] POWERSIMTECH – PSIM, [visitado em agosto de 2019]; https://www.powersimtech.com/products/psim/

[14] CADENCE – PSpice, [visitado em agosto de 2019]; https://www.pspice.com/

[15] SOURCEFORGE – QUCS, [visitado em agosto de 2019]; http://www.qucs.sourceforge.net/

[16] GITHUB – GitHub, [visitado em agosto de 2019]; https://www.github.com/

[17] NERDAMER – Nerdamer, [visitado em agosto de 2019]; https://www.nerdamer.com/

[18] GITHUB – Nerdamer, [visitado em agosto de 2019]; https://www.github.com/jiggzson/nerdamer

[19] KATEX – KaTeX, [visitado em agosto de 2019]; https://www.katex.org/

[20] LATEX – LaTeX, [visitado em agosto de 2019]; https://www.latex-project.org/

[21] HOCHBAUM, Dorit - *Approximation Algorithms for NP-Hard Problems*, 1ª Edição, pp. 8-10, 1996

[22] ROSETTACODE - Permutations with repetitions, https://www.rosettacode.org/wiki/Permutations_with_repetitions

[23] VIANA, Ana; PEREIRA, Francisco; ALVES, Mário – FEELE, Métodos de Análise e Simplificação de Circuitos Elétricos (Parte 2), slides 4, 9-16, 2015.

[24] SOURCEFORGE – QUCS, *A Tutorial, Qucs Simulation of SPICE Netlists*, [visitado em setembro de 2019]; http://qucs.sourceforge.net/docs/tutorial/spicetoqucs.pdf/

[25] KATEX – API KaTeX, [visitado em agosto de 2019]; https://www.katex.org/docs/api.html