UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

LUÍS FERNANDO CAMPOS DE CARVALHO COSTA

Simulador de Circuitos Lineares RLC usando Java

Guaratinguetá
2018

Luís Fernando Campos de Carvalho Costa

Simulador de Circuitos Lineares RLC usando Java

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estatual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica .

Orientadora: Profª Drª. Paloma Maria Silva Rocha Rizol

Guaratinguetá
2018

Costa, Luís Fernando Campos de Carvalho Costa
C837s Simulador de circuitos elétricos usando Java / Luís Fernando Campos de Carvalho Costa – Guaratinguetá, 2018.
65 f : il.
Bibliografia: f. 65

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2018.
Orientadora: Profª. Drª. Paloma Maria Silva Rocha Rizol

1. Circuitos elétricos - Análise. 2. Java (Linguagem de programação de computador). 3. Cálculos numéricos. I. Título.

CDU 621.3.049

Luciana Máximo
Bibliotecária CRB-8/3595

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

LUÍS FERNANDO CAMPOS DE CARVALHO COSTA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"**GRADUADO ENGENHARIA ELÉTRICA**"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM NOME DO CURSO

Profª. Drª. PALOMA M. S. ROCHA RIZOL
Coordenadora

BANCA EXAMINADORA:

Profª. Drª. PALOMA MARIA SILVA ROCHA RIZOL
Orientador/UNESP-FEG

Prof. Dr. DURVAL LUIZ SILVA RICCIULLI
UNESP-FEG

Engª DANIELA ROSA RIBEIRO
MEMBRO EXTERNO

Dezembro de 2018

AGRADECIMENTOS

Agradeço à minha família, especialmente minha mãe e minha irmã, pelo suporte ao longo desses anos.

Agradeço ao Corpo Docente da Universidade, que permitiu que eu tivesse uma ótima educação, que eu fizesse intercâmbio no exterior e me qualificou para trabalhar numa instituição honrada e servir ao meu país.

E também agradeço à minha segunda família, a República Amoribunda, pela amizade, pela convivência e aprendizado durante a minha formação.

“Pambola ngai, nzambe pambola ngai

Abençõe-me senhor, abençõe-me “

(Maître Gims)

RESUMO

Este trabalho de conclusão de curso apresenta o desenvolvimento de um aplicativo em Java construído para a resolução de circuitos elétricos lineares. Estes circuitos são compostos por resistências, indutores e capacitores, alimentados por fontes de tensão constante, fontes de tensão senoidal, fontes de correntes constantes e fontes de corrente senoidais. O programa desenvolvido se fundamenta em conceitos aprendidos durante o Curso de Engenharia Elétrica, entre eles, a Análise Nodal por Inspeção, Transformadas de *Laplace*, o método de *Newton-Raphson*, *Durand-Kerner*, Eliminação de *Gauss-Jordan*, Expansão em frações parciais, entre outros. Tais conceitos são relativamente simples quando aplicados isoladamente, mas geram cálculos complicados e grandes na resolução de circuitos. Por isso, esta ferramenta auxilia para obter tais resoluções. O programa possui uma interface simples, similar a simuladores tais como o PSpice. O tempo de execução obtido para os circuitos é da ordem de segundos e o programa fornece todas as tensões e correntes de todos os componentes. Assim, o programa se mostrou como uma ferramenta útil para auxiliar no aprendizado da Teoria de Circuitos Elétricos.

PALAVRAS-CHAVE: Análise de circuitos. Métodos numéricos. Java.

ABSTRACT

This Final Paper details the development of a Java application Linear Circuit Analysis. This circuit is composed of resistors, inductor and capacitor with sources of tension or the current constant or sinusoidal. The developed software is based on concepts learned on Electric Engeener Course, like Nodal Analysis Techniques, Laplace Transform, Newton's Method, Durand-Kerner Method, Gauss Elimination, Partial Fractions Decomposition, among others. This concept is usually simple, but when combined, this concept gives big and complicated calculations and that is why this tool will be useful to the analysis. The goal was the development of a software whose interface was simples, similar to softwares like PSPice. The runtime to the analysis normally is less than a minute and the application provides the tensions and currents of all components. Thus, the software has proved as a tool to learn of Circuit Analysys.
KEYWORDS: circuit analysis. numerical methods. java.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathcal{L}\{f(t)\}$	Transformada de *Laplace* da função $f(t)$
Ω	Símbolo da Unidade Ohm
δ	Função Delta de Dirac

SUMÁRIO

1 INTRODUÇÃO

1.1 *SOFTWARES* SIMILARES

Existe outros *softwares* que realizam a função de auxiliar estudantes a resolver circuitos elétricos, como o *MultiSIM*, *Solve Elec* e *MatLab*, mas esses programas tem algumas diferenças e limitações na resolução de Circuitos Elétricos.

1.1.1 *MultiSIM*

O *MultiSIM* permite simular circuitos analógicos e digitais (NATIONAL INSTRUMENTS CORPORATION, 2009), possui uma interface simplificada e sua uma versão de avaliação disponível para uso por 30 dias.

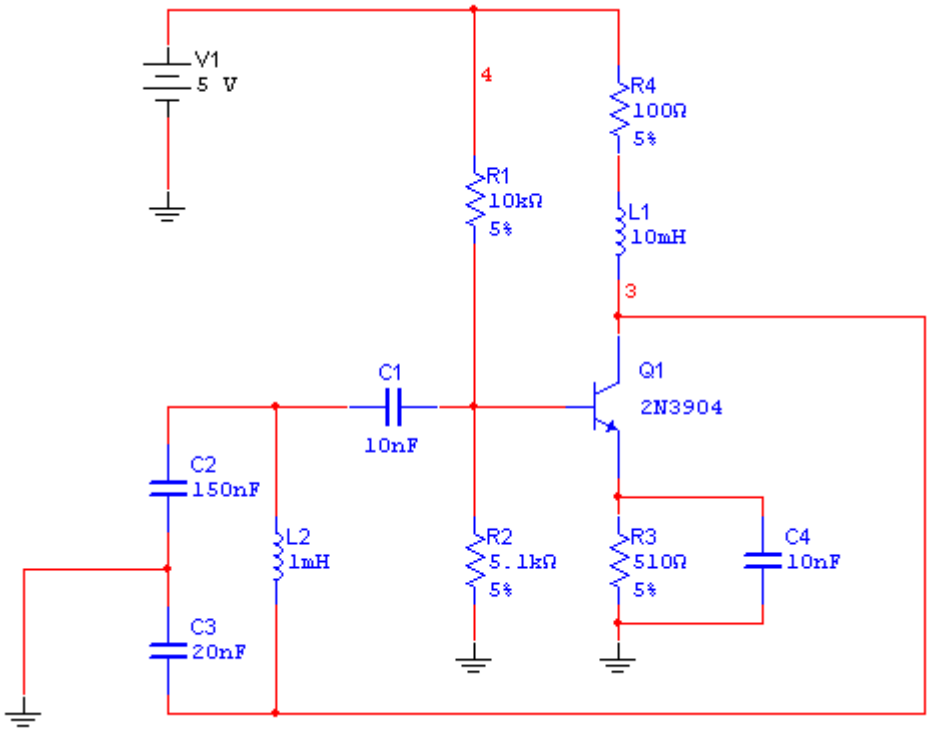
A Figura 1 mostra um circuito simulado do *MultiSIM*, que apesar de permitir o uso de mais componentes como o amplificador de tensão, a saída do programa é mostrada como um gráfico e é calculada utilizando métodos iterativos, portanto, se o passo de cada iteração for grande, a saída do circuito pode não corresponder a solução correta ou se o passo for pequeno, o programa irá demorar muito tempo para simular.

1.1.2 *Solve Elec*

O *Solve Elec* é um software gratuito (LOGICIELS POUR LA PHYSIQUE, 2018) que permite simular circuitos em DC ou AC, onde nos circuitos DC é possível inserir resistores, diodos, transistores, fontes de tensão, fontes de corrente e amplificadores de tensão, porém não é possível capacitores e indutores e no modo AC é possível inserir circuitos RLC, com fontes de tensão, fontes de corrente e amplificadores de tensão.

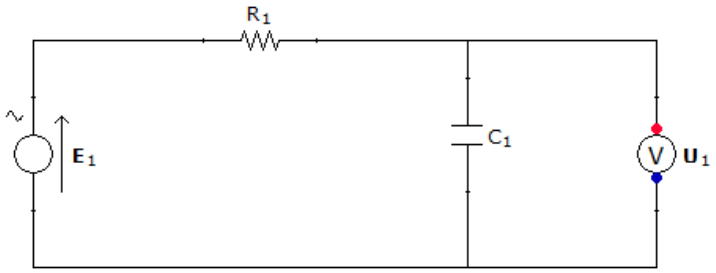
A saída do circuito pode ser expressa pelo valor da tensão no componente e se a simulação for AC, o programa também fornece a fase da tensão ou gráfico. Na simulação AC mostrada pela Figura 2 e a resposta mostrada na Figura 3, a saída é a tensão RMS de saída e a fase. No modo AC, o programa não permite alterar a tensão inicial no capacitor, colocar fontes DC e se o circuito tivesse indutor não seria possível alterar a corrente inicial.

Figura 1 – Circuito Simulado no *MultiSIM*



fonte: (NATIONAL INSTRUMENTS CORPORATION, 2009).

Figura 2 – Circuito Simulado no *Solve Elec*

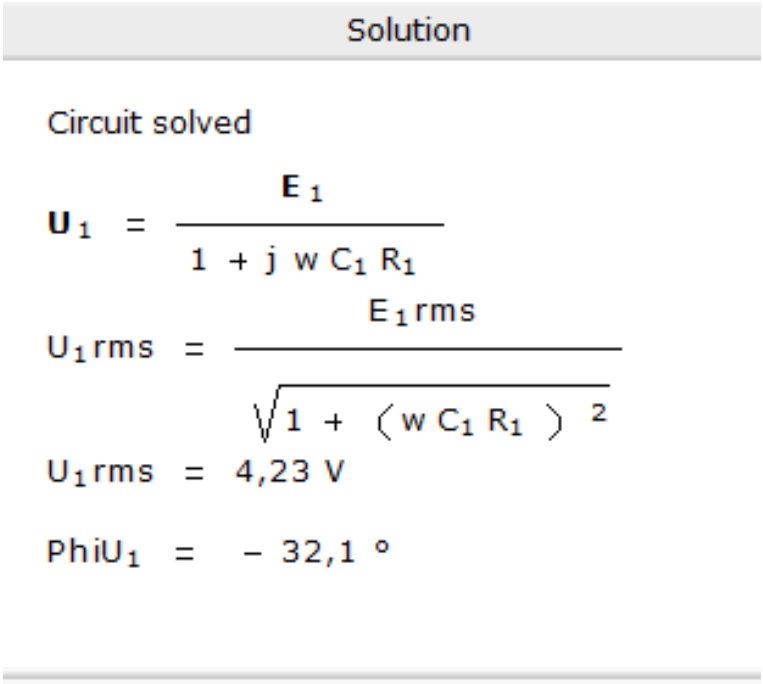


Fonte: Produção do próprio autor.

1.1.3 *MATLAB*

A plataforma *MATLAB*, apesar de não ser um simulador de circuitos elétricos (THE MATHWORKS, INC, 2018), é uma ferramenta muito útil para a resolução de circuitos, pois realiza operações matemáticas envolvendo matrizes com facilidade, mesmo com o uso de variáveis simbólicas, sendo assim, o *MATLAB* pode ser utilizado para inverter a matriz de admitâncias e ainda possui ferramentas para fazer a Transformada Inversa de *Laplace*. Portanto, a ferramenta pode oferece a solução literal do circuito elétrico.

Figura 3 – Resolução do Circuito Simulado no *Solve Elec*



Fonte: Produção do próprio autor.

No entanto, a ferramenta não faz a análise nodal ou a análise de malhas, logo, o usuário deve conhecer a teoria de circuitos elétricos antes de utilizar o *MATLAB* para esse fim.

1.2 MOTIVAÇÃO

A motivação desse trabalho foi desenvolver uma ferramenta mais adequada ao uso de alunos de Engenharia Elétrica, que simule circuitos elétricos, capaz de trabalhar tanto com circuitos alimentados por fontes DC e AC ao mesmo tempo, capacitores e indutores carregados e forneça a expressão da tensão e da corrente em função do tempo e ainda seja um *softtware* livre, ao contrário do *MultiSIM* e do *MATLAB* que possui versões de avaliação limitadas.

1.3 OBJETIVO DO TRABALHO

Este trabalho tem por objetivo criar um aplicativo que visa simulador circuitos elétricos lineares compostos por resistência, capacitores, indutores, fontes de tensão e fontes de correntes, usando o Método da Análise Nodal por Inspeção e Resolução de Equações Diferenciais - Integrais usando Transformadas de *Laplace*.

1.4 PLANO DE TRABALHO

O Trabalho foi dividido em 5 capítulos, onde no 1º capítulo, é apresentada a Introdução, comparando o Aplicativo com Softwares Similares, a Motivação e o Objetivo do Trabalho.

No 2º capítulo, é apresentado o Embasamento Teórico para o desenvolvimento do trabalho com a revisão da teoria de circuitos elétricos, transformadas de *Laplace* e algoritmos para fatorar polinômios e resolução de sistemas lineares.

No 3º capítulo, é apresentado o Desenvolvimento do trabalho, onde a Teorema da Conversão de Fontes foi adaptado para a solução de circuitos RLC e a descrição das classes que formam o aplicativo, como a classe Frac, responsável por realizar operações matemáticas envolvendo frações de polinômios, as classes que tratam os componentes que são adicionados no aplicativo e finalmente as classes que formam a interface gráfica do aplicativo.

2 EMBASAMENTO TEÓRICO

Nesse capítulo, a Análise Nodal por Inspeção é apresentada, método implementado computacionalmente para a resolução de circuitos, o método de resolução de equações diferenciais lineares de circuitos elétricos via Transformadas de *Laplace* e algoritmos computacionais para fatorar polinômios e algoritmo para a solução de sistemas de equações lineares.

2.1 TEORIA DE CIRCUITOS ELÉTRICOS

2.1.1 Análise nodal por inspeção

De acordo com as definições da teoria de circuitos (BOYLESTAD, 2012),tem-se:

- Ramo: Caminho constituído por um ou mais bipolos em série.
- Nó: Junção entre 2 ou mais ramos.
- Malha: Caminho fechado composto por ramos.

2.1.1.1 Lei de *Kirchhoff* para as correntes

A soma algébrica das correntes em um nó é zero, e pode ser observada através da equação (2.1).

$$\sum_{k=1}^{n} I_k = 0 \tag{2.1}$$

Para o circuito mostrado na Figura 4 , a equação para a corrente é dada pela Equação 2.2:

$$I_1 + I_2 + (-I_3) + I_4 - I_5 = 0 \tag{2.2}$$

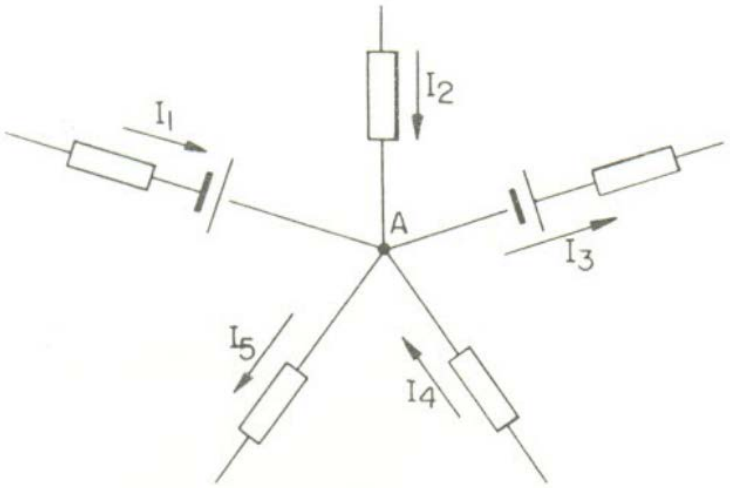
A Figura 4 apresenta um exemplo para a aplicação da Lei de *Kirchoff* para as correntes.

2.1.1.2 Resolução de um circuito

A Figura 5 mostra um circuito a ser resolvido:

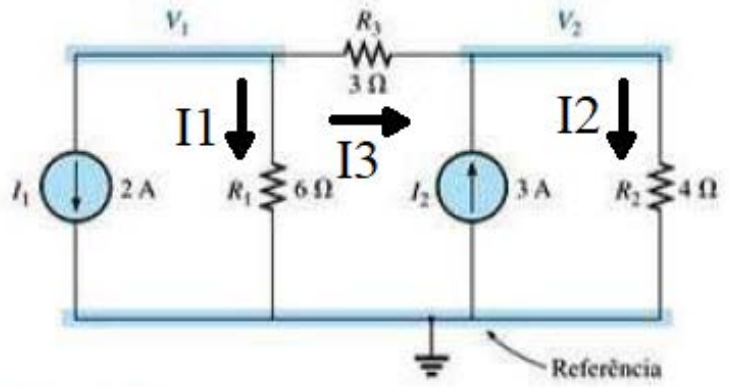
Resolvendo o circuito da Figura 5 pela Lei de *Kirchhoff* para as correntes, obtém-se:

Figura 4 – Exemplo de aplicação da Lei de *Kirchhoff*



Fonte: Produção do próprio autor.

Figura 5 – Exemplo de circuito



Fonte: (BOYLESTAD, 2012)

Nó 1:

$$2 = I_1 + I_3$$
$$2 = \frac{V_1}{6} + \frac{V_1 - V_2}{3} \tag{2.3}$$

Nó 2:

$$3 = I_2 - I_3$$
$$3 = \frac{V_2}{4} - \frac{V_1 - V_2}{3} \tag{2.4}$$

Escrevendo essas equações na forma matricial:

$$\begin{vmatrix} \frac{1}{6}+\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3}+\frac{1}{4} \end{vmatrix} \begin{Vmatrix} V_1 \\ V_2 \end{vmatrix} = \begin{vmatrix} 2 \\ 4 \end{vmatrix} \tag{2.5}$$

Ao resolver o sistema linear, serão encontradas todas as tensões nos nós e as correntes das fontes de tensão. Com esses valores, é possível encontrar rapidamente todas as tensões e correntes de todos os componentes.

2.1.2 Algoritmo para Análise Computacional

A seguir, será apresentado o algoritmo para a análise nodal computacional.

Para um circuito linear com n nós e m fontes de tensão, a análise nodal por inspeção gera um sistema linear representado pela equação (2.6), na qual A é a matriz de admitâncias nodais, com $n+m-1$ linhas e $n+m-1$ colunas. B é a matriz resposta, com $n+m-1$ linhas e uma coluna e é composta pelas correntes de entradas nos nós e as tensões das fontes de tensão. X é a matriz incógnita, com as mesmas dimensões da matriz resposta e composta pelas tensões nos nós e as correntes das fontes de tensão.

$$A * X = B \tag{2.6}$$

A matriz incógnita é a solução do sistema linear da equação (2.6) e será encontrada ao inverter a matriz A e a multiplicando pela matriz B, como mostra e equação (2.7).

$$X = A^{-1} * B \tag{2.7}$$

2.1.2.1 Algoritmo para a montagem da Matriz A

1. Escolher um nó para ser a referência. Para a aplicação, será escolhido o nó "0".
2. Ordenar crescentemente os outros nós.
3. Os elementos $A_{i,j}$ da Matriz A entre as linhas 1 e $n-1$ e colunas 1 e $n-1$ serão preenchidos pela seguinte regra:

- Se $i = j$

  $A_{i,i}$ é a somatória de todas as admitâncias que possuem o nó i como um dos nós.

- Se $i \neq j$

  $A_{i,j}$ é o oposto da somatória de todas as admitâncias que possuem os nós i e j como nós.

4. Ordenar alfabeticamente todas as fontes de tensão.

5. Os elementos $A_{i,j}$ da Matriz A entre as linhas 1 e $n-1$e colunas n e $n+m-1$ serão preenchidos pela seguinte regra:

   - Se a fonte k possuir como nó 1 o nó i

     $A_{i,(k+n-1)} = -1$

   - Se a fonte k possuir como nó 2 o nó i

     $A_{i,(k+n-1)} = 1$

6. Os elementos $A_{i,j}$ da Matriz A entre as linhas n e $n+m-1$ e colunas 1 e n-1 serão preenchidos pela seguinte regra:

   - Se a fonte k possuir como nó 1 o nó i

     $A_{(k+n-1),i} = 1$

   - Se a fonte k possuir como nó 2 o nó i

     $A_{(k+n-1),i} = -1$

7. Os elementos não preenchidos da Matriz pelas regras anteriores serão iguais a 0.

2.1.2.2 Algoritmo para a montagem da Matriz B

A Matriz B é uma matriz-coluna ou um vetor

1. Os elementos B_i da Matriz B entre as linhas 1 e n-1 é o somatório das correntes que entram no nó i menos o somatório das correntes que saem no nó i. Em outras palavras, é o somatório das fontes de corrente que possuem nó 1 igual ao nó i menos o somatório das fontes de corrente que possuem nó 2 igual ao nó i.

2. Os elementos B_i da Matriz B entre as linhas n e $n+m-1$ é igual ao valor das fontes de tensão, organizados alfabeticamente.

2.1.2.3 Resolução do Sistema Linear

Para a resolução do sistema da equação (2.7) será utilizado o algoritmo de Eliminação de *Gauss-Jordan* que será detalhado na seção 2.4.

A seguir será apresentado como as Transformadas de Laplace são utilizadas para a resolução de circuitos RLC.

2.2 TRANSFORMADAS DE LAPLACE

O método da Análise por Inspeção Nodal inicialmente foi desenvolvido para simular circuitos que possuem apenas fontes de tensão, fontes de corrente e resistências. Resistência definida pela relação linear entre tensão e corrente.

A corrente no capacitor é função da derivada da tensão (ZILL, 2001) , como mostra a equação 2.9 e a corrente no indutor é função da integral da tensão como mostra a equação 2.13. Portanto, é necessário resolver sistemas de equações diferencias-integrais para obter as tensões nos nós.

$$i = C\frac{dv_C}{dt} \tag{2.8}$$

$$i = C\frac{d(v_1 - v_2)}{dt} \tag{2.9}$$

$$v_L = L\frac{di_L}{dt} \tag{2.10}$$

$$(v_1 - v_2) = L\frac{di}{dt} \tag{2.11}$$

$$di = \frac{1}{L}(v_1 - v_2)dt \tag{2.12}$$

$$i = \frac{1}{L}\int (v_1 - v_2)dt \tag{2.13}$$

Como as equações introduzidas em 2.9 e 2.13 são equações diferenciais-integrais lineares com coeficientes constantes, na qual o método das Transformadas de *Laplace* será usado para resolver as equações. As integrais e diferenciais são transformadas em funções algébricas que

são mais fáceis de se resolver. As Equações 2.17 e 2.22 mostram as Transformadas de *Laplace* da Função Derivada e da Função Integral.

$$\mathcal{L}\{f'(t)\} = \int_0^\infty f'(t)e^{-st}dt \tag{2.14}$$

$$\mathcal{L}\{f'(t)\} = f(t)e^{-st}|_0^\infty + \int_0^\infty f(t)s\ e^{-st}dt \tag{2.15}$$

$$\mathcal{L}\{f'(t)\} = \lim_{x\to\infty} f(t)e^{-st} - \lim_{x\to 0} f(t)e^{-st} + s\int_0^\infty f(t)e^{-st}dt \tag{2.16}$$

$$\mathcal{L}\{f'(t)\} = sF(s) - f(0) \tag{2.17}$$

$$f^{-1}(t) = \int f(t)dt \tag{2.18}$$

$$\mathcal{L}\{f^{-1}(t)\} = \int_0^\infty f^{-1}(t)e^{-st}dt \tag{2.19}$$

$$\mathcal{L}\{f^{-1}(t)\} = f^{-1}(t)\frac{-1}{s}e^{-st}|_0^\infty + \int_0^\infty f(t)e^{-st}dt \tag{2.20}$$

$$\mathcal{L}\{f^{-1}(t)\} = \lim_{x\to\infty} -f^{-1}(t)e^{-st} + \lim_{x\to 0)} f^{-1}(t)e^{-st} + \frac{1}{s}\int_0^\infty f(t)e^{-st}dt \tag{2.21}$$

$$\mathcal{L}\{f^{-1}(t)\} = \frac{f^{-1}(0)}{s} + \frac{F(s)}{s} \tag{2.22}$$

As Equações 2.25 e 2.28 mostram as conversões nas equações diferenciais da tensão e corrente dos indutores e capacitores para as Transformadas de *Laplace* no formato $I = A * V$ para a utilização no método de Análises Nodais.

$$i = \frac{1}{L}\int (v_1 - v_2)dt \tag{2.23}$$

$$I(s) = \frac{1}{L}\left[\frac{V_1(s) - V_2(s)}{s}\right] + \frac{i_0}{s} \tag{2.24}$$

$$I(s) - \frac{i_0}{s} = \frac{1}{sL}\left[V_1(s) - V_2(s)\right] \tag{2.25}$$

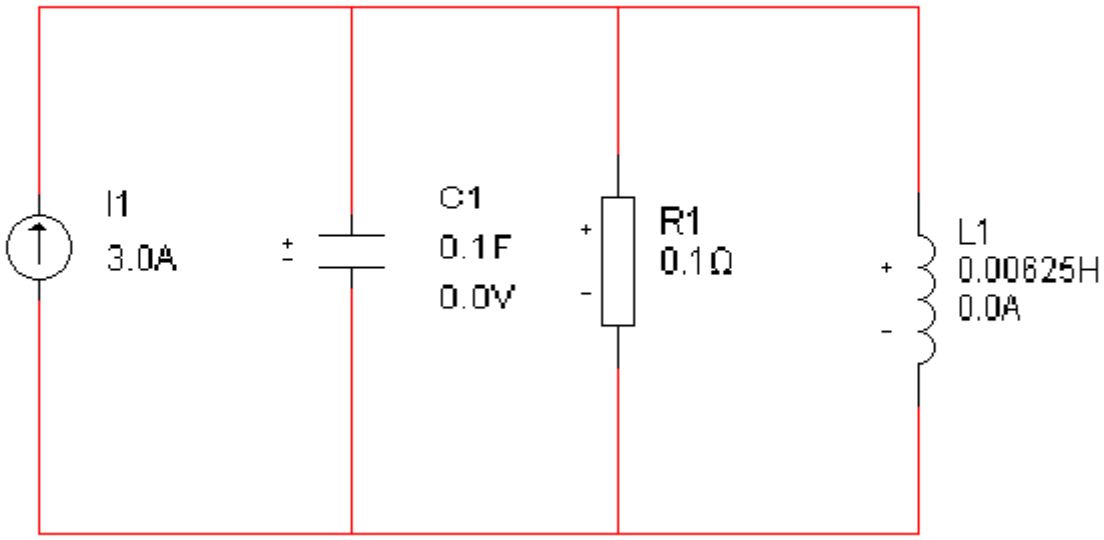
$$i = C\frac{dv}{dt} \tag{2.26}$$

$$I(s) = C\left[s(V_1(s) - V_2(s)) - v_0\right] \tag{2.27}$$

$$I(s) + C * v_0 = sC\left[V_1(s) - V_2(s)\right] \tag{2.28}$$

O circuito da Figura 6 será resolvido pela Equação 2.36 usando o Método de Análise Nodal e o Método das Transformadas de *Laplace*.

Figura 6 – Circuito a ser resolvido pelo Método das Transformadas de *Laplace*



Fonte: Produção do próprio autor.

$$I_1 = i_c + i_r + i_l \tag{2.29}$$

$$I = C_1\frac{v(t)}{dt} + \frac{v(t)}{R_1} + \frac{1}{L_1}\int v(t)dt \tag{2.30}$$

$$\frac{I}{s} = C_1 \left[sV(s) - v(0)\right] + \frac{V(s)}{R} + \frac{1}{L}\left[\frac{V(s)}{s}\right] + \frac{i_0}{s} \tag{2.31}$$

$$\frac{I_1}{s} + C_1 v(0) - \frac{i_0}{s} = V(s)\left[sC_1 + \frac{1}{R_1} + \frac{1}{sL_1}\right] \tag{2.32}$$

Aplicando $I_1 = 3A$, $C_1 = 0,1F$, $R_1 = 0,1\Omega$, $L_1 = 6,25mH$ e condições iniciais nulas:

$$\frac{3}{s} = V(s)\left[0,1s + 10 + \frac{160}{s}\right] 3 = V(s)\left[0,1s^2 + 10s + 160\right] \tag{2.33}$$

$$V(s) = \frac{3}{0,1s^2 + 10s + 160} \tag{2.34}$$

$$V(s) = \frac{0,5}{s + 20} - \frac{0,5}{s + 80} \tag{2.35}$$

$$v(t) = 0,5e^{-20t} - 0,5e^{-80t}\ V \tag{2.36}$$

A Tabela 1 fornece as transformadas de *Laplace* dos valores típicos de corrente e tensão em circuitos RLC com fontes de tensão constante ou senoidais.

A Tabela 1 e as Equações 2.25 e 2.28 mostram que as Transformadas de *Laplace* para as funções exponenciais, senoidais e constante, além das equações diferenciais lineares, podem ser expressas como uma razão entre 2 polinômios em função de s.

Por isso, foi criado a classe Frac, detalhada em 3.2 cujos atributos são 2 vetores, representando o numerador e o denominador da fração. A classe Frac define as operações de soma, subtração, multiplicação e fatoração de vetores. A partir dessas operações, a classe Frac realiza as operações de soma, subtração, multiplicação, divisão e simplificação de frações, que são as operações matemáticas usadas na Análise Nodal por Inspeção.

2.3 TEOREMAS E ALGORITMOS PARA FATORAR POLINÔMIOS

2.3.1 Teorema Fundamental da Álgebra

O Teorema Fundamental da Álgebra (WEISSTEIN, 2018c) afirma que o polinômio $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_mx^m$ tem m raízes complexas, algumas delas podendo ser

Tabela 1 – Tabela das Transformadas de *Laplace* Usuais

f(t)	F(s)
δ	1
$u(t)$	$\frac{1}{s}$
$e^{-at}u(t)$	$\frac{1}{s+a}$
$t^n e^{-at}u(t)$	$\frac{1}{(n+1)!\ (s+a)^{n+1}}$
$\sin(\omega t+\phi)u(t)$	$\frac{s\ sin(\phi)+\omega\ cos(\phi)}{s^2+\omega^2}$
$e^{-at}[A\cos(bt)+B\sin(bt)]u(t)$	$\frac{1}{2}\left(\frac{A+jB}{s+a+jb}+\frac{A-jB}{s+a-jb}\right)$
$t^n e^{-at}[A\cos(bt)+B\sin(bt)]u(t)$	$\frac{n!}{2}\left(\frac{A+jB}{(s+a+jb)^{n+1}}+\frac{A-jB}{(s+a-jb)^{n+1}}\right)$

Fonte: (WOLFRAM ALPHA LLC, 2018).

repetidas. Ou seja, o polinômio pode ser decomposto em fatores de grau 1: $P(x) = a_m(x - r_1)(x - r_1)(x - r_2)(x - r_3)...(x - r_m)$.

2.3.2 Método de Newton-Raphson

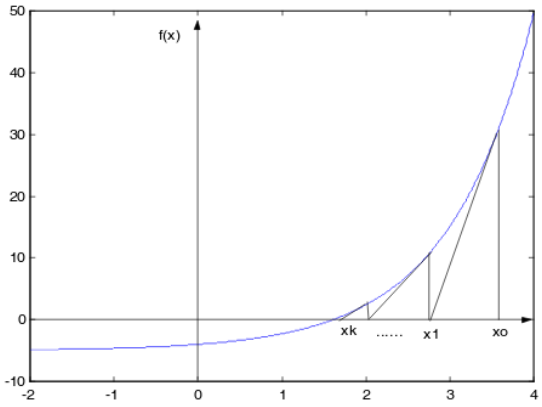
O método consiste em encontrar uma raiz do polinômio a partir do valor inicial x_0, pela aproximação definida na equação (2.37).

$$x_k + 1 = x_{k-1} - \frac{f(x_k)}{f'(x_k)} \tag{2.37}$$

A Figura 7 mostra a evolução do processo iterativo da Equação 2.37).

Observa-se a partir da Figura 7 que para $x_0 \approx 3,5$ e $p(x_0) \approx 30$. Como a função é positiva e

Figura 7 – Exemplo de aplicação do método de Newton



Fonte: Produção do próprio autor.

a sua derivada também positiva, a próxima iteração x_1 será menor do que x_0, sendo que $x_1 \approx 2,7$ e $p(x_1) \approx 10$. Aos poucos, o polinômio se aproxima do eixo das abscissas, ou seja, encontra uma raiz. Os desafios para a aplicação computacional é encontrar com maior precisão possível e menor número de operações matemáticas, o valor do polinômio $p(x)$ e de sua derivada $p'(x)$.

O método de *Newton-Raphson* (CAMPAGNOLO, 2012) é muito eficiente para calcular raízes de polinômios de raízes reais sem raízes múltiplas.Para um polinômio de grau 5 e com raiz r_1 de multiplicidade 3, tem-se que:

$$P(x) = (x-1)^3(x-2)(x-3) \tag{2.38}$$

$$P'(x) = (x-1)^2(5x^2 - 22x + 23) \tag{2.39}$$

$$\frac{P(x)}{P'(x)} = \frac{(x-1)(x-2)(x-3)}{(5x^2 - 22x + 23)} \tag{2.40}$$

Na vizinhança de $x = 1$, tem-se que:

$$x_{i+1} = x_1 + \frac{(x-1)(x-2)(x-3)}{(5x^2 - 22x + 23)} \tag{2.41}$$

$$\lim_{x \to 1} x_{i+1} = x_1 + \frac{(x-1)(x-2)(x-3)}{(5x^2 - 22x + 23)} \tag{2.42}$$

$$\lim_{dx \to 0} x_{i+1} = x_1 + \frac{dx(x-2)(x-3)}{(5x^2 - 22x + 23)} \tag{2.43}$$

A equação 2.43 mostra que em torno de $x = 1$, a convergência do Método de *Newton-Raphson* é linear, logo, o método precisará de mais iterações e demorará mais tempo para convergir. Por isso, o método será adaptado para atender ao programa.

2.3.3 Método de Durand-Kerner

Este método implementa o algoritmo da Equação para calcular todas as raízes do polinômio simultaneamente.

$$x_i^{k+1} = x_i^k - \frac{p(x_i^{(k)})}{\prod\limits_{j=1 i \neq 1} (x_i^k - x_k^k)} \tag{2.44}$$

O método traduzido para a Linguagem Java (MATTAR, 2006) permite o cálculo de raízes complexas e de multiplicidade até 2. Para polinômios com multiplicidade maior que 3, a saída é calcular a derivada do polinômio, pois como é demonstrada pela Equação (2.47), se o polinômio tiver raízes múltiplas, uma das raízes da derivada também é raiz do polinômio.

$$P(x) = (x - r_1)^n Q(x) \tag{2.45}$$

$$P'(x) = (x - r_1)^{n-1} Q(x) + (x - r_1)^n Q(x) \tag{2.46}$$

$$P'(x) = (x - r_1)^{n-1} (Q(x) + (x - r_1) Q'(x)) \tag{2.47}$$

2.4 ELIMINAÇÃO DE GAUSS-JORDAN

Esse algoritmo (WEISSTEIN, 2018a) é utilizado para inverter a matriz A para resolver o sistema de equações lineares $A * X = B$. Normalmente, o algoritmo inverte matrizes de números reais, no entanto, nesse trabalho, também foi utilizado para inverter matrizes de frações polinomiais, desde que as operações de adição, subtração, multiplicação e divisão sejam definidas

para frações polinomiais.

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}^{-1} \tag{2.48}$$

1. Colocar a matriz identidade à direita da matriz a ser invertida

$$\left[\begin{array}{ccc|ccc} 2 & 1 & -1 & 1 & 0 & 0 \\ -3 & -1 & 2 & 0 & 1 & 0 \\ -2 & 1 & 2 & 0 & 0 & 1 \end{array}\right] \tag{2.49}$$

2. 1ª iteração i=1 Adotar o elemento p_{ii} como pivô

   $pivô = 2$

3. Dividir a linha i pelo pivô

$$\left[\begin{array}{ccc|ccc} 1 & .5 & -.5 & .5 & 0 & 0 \\ -3 & -1 & 2 & 0 & 1 & 0 \\ -2 & 1 & 2 & 0 & 0 & 1 \end{array}\right] \quad L1_{i+1} = L1_i/2 \tag{2.50}$$

4. Anular os elementos da coluna i que não pertence à diagonal principal:

$$\left[\begin{array}{ccc|ccc} 1 & 0.5 & -0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 1.5 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 1 \end{array}\right] \begin{array}{l} \\ L2_{i+1} = L2_i + 3 * L1i \\ L3_{i+1} = L3_i + 2 * L1i \end{array} \tag{2.51}$$

5. Repetir o passo 2, com i=2 até i=n.

   $pivô = 0.5$

$$\left[\begin{array}{ccc|ccc} 1 & 0.5 & -0.5 & 0.5 & 0 & 0 \\ 0 & 1 & 1 & 3 & 2 & 0 \\ 0 & 2 & 1 & 1 & 0 & 1 \end{array}\right] \quad L2_{i+1} = L2i/0.5 \tag{2.52}$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 1 & 1 & 3 & 2 & 0 \\ 0 & 0 & -1 & -5 & -4 & 1 \end{array}\right] \begin{array}{l} L1_{i+1} = L1_i - .5 * L2i \\ \\ L3_{i+1} = L3_i - 2 * L1i \end{array} \tag{2.53}$$

$piv\hat{o} = -1$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 1 & 1 & 3 & 2 & 0 \\ 0 & 0 & 1 & 5 & 4 & -1 \end{array}\right] \quad L3i + 1 = L3i/(-1) \tag{2.54}$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 4 & 3 & -1 \\ 0 & 1 & 0 & -2 & -2 & 1 \\ 0 & 0 & 1 & 5 & 4 & -1 \end{array}\right] \begin{array}{l} L1_{i+1} = L1_i + L3i \\ L2_{i+1} = L2_i - L3i \\ \\ \end{array} \tag{2.55}$$

6. A matriz que sobrou a direita é a matriz inversa de $p1$

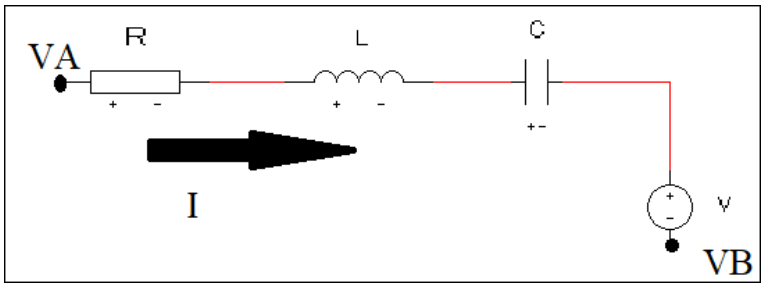
3 DESENVOLVIMENTO

Neste capítulo, são apresentados a Teoria de Conversão de Fontes adaptada para a resolução de circuitos elétricos via Transformada de *Laplace* e os algoritmos em Java para realizar operações com frações polinomiais e criar a interface para a simulação de circuitos elétricos.

3.1 TEOREMA DA CONVERSÃO DE FONTES

Essa adaptação do Teorema de Conversão de Fontes visa reduzir o número de equações e consequentemente o tamanho das matrizes que são geradas pelo Método de Inspeção Nodal. Os ramos contendo Resistores, Indutores ou Capacitores, alimentados ou não por fontes de tensão como o exemplo da Figura 8 podem ser unidos e transformados em fontes de corrente em paralelo com a impedância equivalente de acordo com a Equação 3.4.

Figura 8 – Circuito RLC série com Fonte de Tensão



Fonte: Produção do próprio autor.

$$V_A - RI - L[sI - i(0)] - \frac{1}{sC}I - \frac{v(0)}{s} - V - V_B = 0 \tag{3.1}$$

$$V_A - V_B - \left[-Li(0) + \frac{v(0)}{s} + V\right] = \left[R + sL + \frac{1}{sC}\right] I \tag{3.2}$$

$$\frac{V_A - V_B - \left[-Li(0) + \frac{v(0)}{s} + V\right]}{\left[R + sL + \frac{1}{sC}\right]} = I \tag{3.3}$$

$$\frac{V_A - V_B}{\left[R + sL + \frac{1}{sC}\right]} = I + \frac{\left[-Li(0) + \frac{v(0)}{s} + V\right]}{\left[R + sL + \frac{1}{sC}\right]} \tag{3.4}$$

$$\frac{V_A - V_B}{R} = I \tag{3.5}$$

A Equação 3.4 é simular à lei de Ohm mostrada na Equação 3.5, assim o método da inspeção nodal pode ser usado para resolver circuitos RLC, basta substituir R pela impedância equivalente do ramo $\left[R + sL + \frac{1}{sC}\right]$ e adicionar no valor da corrente o fator a direita de I na Equação 3.4 que corresponde a corrente equivalente vindo da transformação da fonte de tensão e das condições iniciais dos capacitores e indutores.

3.2 CLASSE FRAC

A classe Frac realiza as operações matemáticas envolvendo frações de polinômios e seus atributos são dois vetores representando os polinômios que definem o numerador e o denominador da fração. Para simplificar a implementação da classe, inicialmente, foram desenvolvidos os algoritmos de soma, subtração, multiplicação e fatoração de polinômios.

O Quadro 1 mostra o diagrama UML da classe Frac.

3.2.1 Atributos da Classe Frac

A expressão (3.6) define um objeto *p* da classe *Frac*:

$$Frac\ p = \frac{num}{den} = \frac{a_0 + a_1 s + a_2 s^2 + a_3 s^3 + ... + a_{m-1} s^{m-1} + a_m s^m}{b_0 + b_1 s + b_2 s^2 + b_3 s^3 + ... + b_{n-1} s^{n-1} + b_n s^n} \tag{3.6}$$

Cada coeficiente a_m do numerador de p é associado a um elemento do vetor num[m], assim como cada coeficiente b_n do denominador é associado a um elemento do vetor den[n]. As dimensões dos vetores num e den são $m+1$ e $n+1$, respectivamente. A classe $BigDecimal$ foi usada para representar os coeficientes da fração, pois permite que o programador use uma quantidade de algarismos significativos maior do que $double$ e tenha controle sobre o arredondamento. Sendo assim, cada operação matemática é calculada com maior exatidão possível, minimizando os erros que as funções iterativas fornecem.

Quadro 1 – Diagrama UML da classe Frac

Frac
-BigDecimal[] num -BigDecimal[] den
-sum(BigDecimal[] v1, BigDecimal v2): BigDecimal[] -menos(BigDecimal[] v1, BigDecimal v2): BigDecimal[] -mult(BigDecimal[] v1, BigDecimal[] v2): BigDecimal[] -mult(BigComplexo[] c1, BigComplexo[] c2): BigComplexo[] -nDerivada(BigDecimal[] V, int n): BigDecimal[] +roots(BigDecimal[] p): BigDecimal[][] +simplificar(): void +add(Frac p): Frac +sub(Frac p): Frac +prod(Frac p): Frac +div(Frac p): Frac +inversa(Frac[][] p1): Frac[][] +multm(Frac[][] p1, Frac[][]p2): Frac[][] +toString(): String +fparcial(): Frac[] +invLap(): String

3.2.2 Métodos de Manipulação de Vetores

Os métodos a seguir (DEITEL; DEITEL, 2017) realizam as operações de soma, subtração e multiplicação entre 2 vetores, podendo ser tanto do numerador ou do denominador de uma fração polinomial, além do método para calcular a derivada de um polinômio.

As operações serão definidas para $v1$ e $v2$ definidos pela expressão (3.7).

$$\begin{cases} v_1 = v_{10} + v_{11}s + v_{12}s^2 + v_{13}s^3 + ... + v_{1m}s^m \\ v_2 = v_{20} + v_{21}s + v_{22}s^2 + v_{23}s^3 + ... + v_{2n}s^n \end{cases} \tag{3.7}$$

3.2.2.1 Método sum(BigDecimal[] v1, BigDecimal[] v2)

O algoritmo para calcular a soma V entre v_1 e v_2:

- Criar um vetor V com o maior tamanho entre v_1 e v_2.
- Copiar os valores de v_1 no vetor V.

- Adicionar no vetor V todos os valores de v_2

3.2.2.2 Método menos(BigDecimal v1[], BigDecimal v2[])

O algoritmo para calcular a subtração V entre v_1 e v_2:

- Criar um vetor V com o maior tamanho entre v_1 e v_2.
- Copiar os valores de v_1 no vetor V.
- Subtrair do vetor V todos os valores de v_2

3.2.2.3 Método mult(BigDecimal v1[], BigDecimal v2[])

O algoritmo para calcular o produto V entre v_1 e v_2: Esse algoritmo serve para multiplicar vetores de números reais ou vetores de números complexos.

- Criar um vetor V cujo tamanhos é as soma do tamanho de v_1 e v_2 menos 1.
- Aplicar o somatório da seguir:

$$V(s) = \sum_{i=0}^{N} \sum_{j=0}^{M} v_1[i] v_2[j] s^{i+j} \tag{3.8}$$

3.2.2.4 Método nDerivada(BigDecimal[] V, int n)

A derivada de ordem n do vetor V é definida pela Equação 3.10.

$$V = v_0 + v_1 s + v_2 s^2 + v_3 s^3 + ... + v_m s^m \tag{3.9}$$

$$V^n = \frac{n!}{0!} v_n + \frac{(n+1)!}{1!} v_{n+1} s + \frac{(n+2)!}{2!} v_{n+2} s^2 + ... + \frac{m!}{(m-n)!} v_m s^{m-n} \tag{3.10}$$

3.2.2.5 +roots(BigDecimal p): BigDecimal[][]

Este método fatora o polinômio $p(x)$ em vários binômios, se a raiz for um número real, ou em trinômio irredutíveis, no caso de raízes complexas. Para isso, o método implementa o método de *Durand-Kerner* descrito em 2.3.3 seguindo o algoritmo:

- Encontrar uma raiz real ou complexa pelo método de *Durand-Kerner*
- Realizar Dividir o polinômio $p(x)$ pelo fator encontrado até encontrar todas as raízes.

- Caso alguma raiz demore para ser encontrada, provavelmente, o polinômio possui raízes complexas, então deve-se calcular todas as raízes da derivada e verificar qual delas também é raiz de $p(x)$
- Se nenhuma das raízes da derivada for raiz do polinômio, repetir o primeiro passo e aumentar o tempo para o cálculo da raiz.

3.2.3 Métodos de Manipulação de Frações Polinomiais

Métodos que realizam operações matemáticas entre a fração que invoca o método ($*this$) e o polinômio $p1$.

3.2.3.1 Método add(Frac p)

Esse método realiza a soma entre das frações X e Y, com mostra a Equação 3.13

$$X = \frac{A}{B} \tag{3.11}$$

$$Y = \frac{C}{D} \tag{3.12}$$

$$X + Y = \frac{A}{B} + \frac{C}{D} = \frac{AD + BC}{CD} \tag{3.13}$$

A Equação 3.15 apresenta um exemplo de aplicação do algoritmo descrita em 3.13.

$$\frac{1}{s+1} + \frac{1}{s+2} = \frac{(s+2) + (s+1)}{(s+2)(s+1)} \tag{3.14}$$

$$\frac{1}{s+1} + \frac{1}{s+2} = \frac{2s+3}{s^2+3s+2} \tag{3.15}$$

Se o numerador e o denominador da fração resultante possuir fatores em comum, a fração deve ser simplificada, usando o método simplificar, descrito em 3.2.3.5, como mostra o exemplo da Equação 3.20.

$$\frac{1}{(s+1)(s+2)} + \frac{1}{(s+1)(s+3)} \tag{3.16}$$

$$= \frac{(s+1)(s+3)+(s+1)(s+2)}{(s+1)(s+2)(s+1)(s+3)} \tag{3.17}$$

$$= \frac{2s^2+7s+5}{s^4+7s^3+17s^2+17s+6} \tag{3.18}$$

$$= \frac{(s+1)(2s+5)}{(s+1)^2(s+2)(s+3)} \tag{3.19}$$

$$= \frac{2s+5}{(s+1)(s+2)(s+3)} \tag{3.20}$$

3.2.3.2 Método sub(Frac p1)

A subtração entre a fração X e Y é definida pela soma entre X e a fração oposta de Y .

3.2.3.3 Método prod(Frac p)

Algoritmo para calcular o produto p entre X e Y

- Fatorar os numeradores e denominadores de X e Y
- Criar 2 vetores $vNum$ e $vDen$ para abrigar os fatores de X e Y.
- Comparar os fatores de $vNum$ e $vDen$. Eliminar os fatores em comum, caso existam.
- Multiplicar os fatores de $vNum$ e $vDen$ para voltar a fração para a forma expandida.

Exemplo:

$$\frac{s^2+3s+2}{s^2+9s+20}\frac{s^2+6s+8}{s^2+4s+3} \tag{3.21}$$

$$= \frac{(s+2)(s+1)}{(s+5)(s+4)}\frac{(s+2)(s+4)}{(s+1)(s+3)} \tag{3.22}$$

$$= \frac{(s+1)(s+2)^2(s+4)}{(s+1)(s+3)(s+4)(s+5)} \tag{3.23}$$

$$= \frac{(s+2)^2}{(s+3)(s+5)} \tag{3.24}$$

$$= \frac{(s^2 + 4s + 4)}{(s^2 + 8s + 15)} \tag{3.25}$$

3.2.3.4 Método div(Frac p)

Algoritmo para calcular a divisão $pDiv$ entre X e Y

- Criar a fração Z correspondendo ao inverso de Y.
- Multiplicar X por Z.

3.2.3.5 +simplificar(): void

Método que simplifica as frações de polinômios, sempre que possível. A equação (3.26) mostra um caso em que a ordem do polinômio pode ser reduzida, pois o numerador e o denominador possui fatores comuns.

$$\frac{s^2 + 3s + 2}{s^3 + 5s^2 + 8s + 4} = \frac{(s + 1)(s + 2)}{(s + 1)(s + 2)^2} = \frac{1}{s + 2} \tag{3.26}$$

O algoritmo desse método segue a lógica mostrada na equação (3.26).

- Fatorar o tanto o numerador e o denominador usando o método roots();
- Comparar os polinômios das listas dos numeradores e dos denominadores.
- Caso tenha polinômios em comum, retirar esses polinômios da lista.
- Multiplicar os polinômios do numerador e denominador que sobrarem.

3.2.4 Métodos para Manipular Matrizes de Frações de Polinômios

Como o objetivo desse programa é resolver circuitos elétricos, é preciso criar uma função que inverta a matriz de admitâncias e depois a multiplique pela matriz das correntes nodais.

3.2.4.1 inversa(Frac[][] p1): Frac[][]

Esse método utiliza o método conhecido como Eliminação de *Gauss-Jordan* descrito em 2.4 e adaptado para ser usado com frações polinomiais.

3.2.4.2 multm(Frac[][] p1, Frac[][]p2): Frac[][]

O produto p entre duas matrizes de frações $p1$ e $p2$ é definido pela Equação (3.27) (WEISSTEIN, 2018b).

$$p_{ij} = \sum_{k=1}^{k=n} p1_{ik} * p2_{kj} \tag{3.27}$$

3.2.5 Métodos para Exibição de Frações

3.2.5.1 +toString(): String

O método toString exibe a fração $(*this)$ da maneira mais amigável para o usuário de acordo com a fórmula (3.28).

$$(*this.toString()) = \frac{num[n-1] + num[n-2] + ... + num[1] + num[0]}{den[m-1] + den[m-2] + ... + den[1] + den[0]} \tag{3.28}$$

onde n é o tamanho do numerador e
m é o tamanho do denominador

3.2.5.2 +fparcial(): Frac[]

O método fparcial realizada a decomposição em frações parciais como o Exemplo da Equação 3.30

$$\frac{s^4 + 6s^3 + 15s^2 + 19s + 10}{s^3 + 3s^2 + 4s + 2} = \frac{A}{s+1} + \frac{B + Cs}{s^2 + 2s + 2} + D + Es \tag{3.29}$$

$$= \frac{A[s^2 + 2s + 2] + (B + Cs)[s + 1] + (D + Es)[s^3 + 3s^2 + 4s + 2]}{s^3 + 3s^2 + 4s + 2} \tag{3.30}$$

Comparando o numerador:

$$s^4+6s^3+15s^2+19s+10 = A[s^2+2s+2]+(B+Cs)[s+1]+(D+Es)[s^3+3s^2+4s+2] \tag{3.31}$$

$$= Es^4+(D+3E)s^3+(A+C+3D+4E)s^2+(2A+B+C+4D+2E)s+2A+B+2D \quad (3.32)$$

Comparando termo-a-termo do polinômio:

$$\begin{cases} E = 1 \\ 6 = D + 3E \\ 15 = A + C + 3D + 4E \\ 19 = 2A + B + C + 4D + 2E \\ 10 = 2A + B + 2D \end{cases} \quad (3.33)$$

Ao resolver o sistema para as variáveis A, B, C ,D e E:

$$\begin{cases} A = 1 \\ B = 2 \\ C = 1 \\ D = 3 \\ E = 1; \end{cases} \quad (3.34)$$

Assim a fração inicial decomposta é:

$$\frac{1}{s+1} + \frac{2+s}{s^2+2s+2} + 3 + s \quad (3.35)$$

3.2.5.3 +invLap(): String

O método invLap() realiza a Transformada Inversa de *Laplace*. Esse método é acionado para exibir as tensões e correntes do circuito em função do tempo.

O método aproveita a decomposição em frações parciais feita pelo método fparcial() e para cada fração decomposta existe uma equivalente na Tabela 1. A Transformada Inversa da Equação 3.35 é mostrada na Equação 3.37.

$$\mathcal{L}^{-1}\left\{\frac{1}{s+1} + \frac{2+s}{s^2+2s+2} + 3 + s\right\} \quad (3.36)$$

$$= e^{-t} + e^{-t}[\sin(t) + \cos(t)] + 3\delta + \delta' \quad (3.37)$$

3.3 COMPONENTE

As classe Componente e as suas sub-classes (Resistor, Indutor, Capacitor, VDC, VAC, IDC e IAC) descrevem cada componente do circuito.

Cada Componente possui um Nome, normalmente V1, R1, I3, dependendo do componente, uma posição no Circuito, uma Impedância ou Amplitude de Tensão ou Amplitude de Corrente e informações sobre os nós em que o componente está conectado.

O Quadro 2 mostra o Diagrama UML da Classe Componente.

Quadro 2 – Diagrama UML da Classe Componente

Componente
+String nome +int largura +int altura +Color cor +int posicao +Frac lap +Point point1 +Point point2 +String no1 +String no2 +Frac corrente +Frac tensao
+Componente
+rotate: void +setNo: void +setComponente: void +setNome: void +toString: String +tcString: String

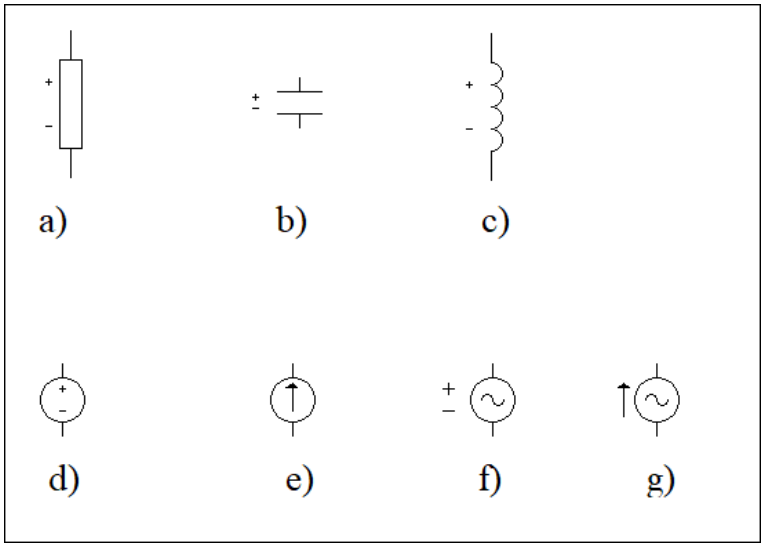
A Figura 9 mostra os Componentes que podem ser utilizados neste Simulador.

A classe Interface permite movimentar os componentes, girar, alterar o valor dos componentes, além de conectá-los para montar o circuito a ser iniciado.

3.3.1 Classe Resistor

Essa classe descrita pelo Quadro 3 recebe o valor da resistência, além dos atributos da super-classe que definem a posição do Resistor no circuito. A impedância do componente é

Figura 9 – Lista de Componentes



a) Resistor b) Capacitor c) Indutor
d) Fonte de Tensão Contínua e) Fonte de Corrente Contínua
f) Fonte de Tensão Senoidal g) Fonte de Corrente Senoidal.
Fonte: Produção do próprio autor.

armazenada na variável $Frac\ lap$, de acordo com a Lei de Ohm descrita na Equação 3.38.

$$lap = Resistencia \tag{3.38}$$

Quadro 3 – Diagrama UML da Classe Resistor

Resistor
BigDecimal resistencia
+Resistor(Point p)
+setComponente: void +toString: String

3.3.2 Classe Capacitor

Esse classe implementa o diagrama do Quadro 4 que tem os atributos da super-classe, mais 2 atributos, definindo a capacitância e a tensão inicial do componente. A impedância do capacitor é calculada de acordo com a Equação 2.28 e armazenada na na variável $Frac\ lap$, de acordo com a Equação 3.42

$$I(s) + C * v_0 = sC\left[V_1(s) - V_2(s)\right] \tag{3.39}$$

$$[V_1(s) - V_2(s)] = \frac{I(s) + C * v_0}{sC} \tag{3.40}$$

$$[V_1(s) - V_2(s)] = \frac{I(s)}{sC} + \frac{v_0}{s} \tag{3.41}$$

$$lap = \frac{1}{sC} \tag{3.42}$$

Quadro 4 – Diagrama UML da Classe Capacitor

Capacitor
BigDecimal capacitancia # BigDecimal t_ inicial
+Capacitor(Point p)
+setComponente: void +toString: String

3.3.3 Classe Indutor

Esse classe implementa o diagrama do Quadro 5 que tem os atributos da super-classe, mais 2 atributos, definindo a indutância e a corrente inicial do componente. A impedância do indutor é calculada de acordo com a Equação 2.25 e armazenada na variável $Frac\,lap$, de acordo com a Equação 3.46

$$I(s) + \frac{i_0}{s} = \frac{1}{sL}\left[V_1(s) - V_2(s)\right] \tag{3.43}$$

$$[V_1(s) - V_2(s)] = \left[I(s) + \frac{i_0}{s}\right]\{sL\} \tag{3.44}$$

$$[V_1(s) - V_2(s)] = I(s)\{sL\} + L\,i_0 \tag{3.45}$$

$$lap = sL \tag{3.46}$$

Quadro 5 – Diagrama UML da Classe Indutor

Indutor
BigDecimal indutancia # BigDecimal c_ inicial
+Indutor(Point p)
+setComponente: void +toString: String

3.3.4 Classes VDC e IDC

A Fonte de Tensão Continua e a Fonte de Corrente Contínua são descritas pelo diagrama do Quadro 6 e do Quadro 7 e tem o valor de sua amplitude armazenada na variável $Frac\,lap$, de acordo com a Equação 3.48 e com a Equação 3.50, onde $u(t)$ é a função degrau.

$$V(t) = Vu(t) \tag{3.47}$$

$$V(s) = lap = \frac{V}{s} \tag{3.48}$$

$$I(t) = Iu(t) \tag{3.49}$$

$$lap = I(s) = \frac{I}{s} \tag{3.50}$$

Quadro 6 – Diagrama UML da Classe VDC

VDC
BigDecimal amp_ tensao
+VDC(Point p)
+setComponente: void +toString: String

Quadro 7 – Diagrama UML da Classe IDC

IDC
BigDecimal amp_ corrente
+IDC(Point p)
+setComponente: void +toString: String

3.3.5 Classes VAC e IAC

A Fonte de Tensão Senoidal e a Fonte de Corrente Senoidal são descritas pelo diagrama do Quadro 8 e do Quadro 9.

A variável $Frac\ lap$ é função da amplitude, frequência e fase do componente de acordo com a Equação 3.52, onde ω é a frequência angular, ϕ é a fase e $u(t)$ é a função degrau.

$$Amplitude(t) = Asin(\omega t + \phi)u(t) \tag{3.51}$$

$$lap = A\frac{s\ sin(\phi) + \omega\ cos(\phi)}{s^2 + \omega^2} \tag{3.52}$$

Quadro 8 – Diagrama UML da Classe VAC

VAC
BigDecimal amp_ tensao # BigDecimal frequencia # BigDecimal fase
+VAC(Point p)
+setComponente: void +toString: String

Quadro 9 – Diagrama UML da Classe IAC

IAC
BigDecimal amp_ corrente # BigDecimal frequencia # BigDecimal fase
+VAC(Point p)
+setComponente: void +toString: String

3.3.6 Classe Linha

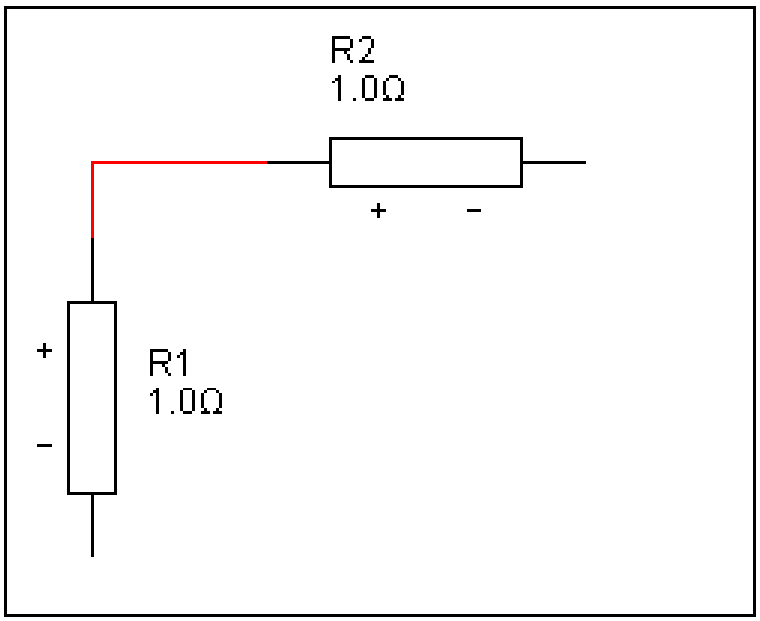
Esta classe não é uma sub-classe de Componente, mas está diretamente relacionada, pois armazena as coordenadas das conexões entre os componentes, como mostra a Figura 10, onde 2 resistores são conectados pela linha em vermelho.

A Classe Linha armazena a posição inicial e a final que são usadas para desenhar a linha. Como um componente está na posição vertical e outro na horizontal, foi necessário armaze-

nar também uma posição intermediária para a linha formar um ângulo reto, como em outros simuladores de circuitos.

O Quadro 10 mostra o Diagrama UML da Classe Linha.

Figura 10 – Resistências em série



Fonte: Produção do próprio autor.

Quadro 10 – Diagrama UML da Classe Linha

Linha
int pos # int x[] # int y[] # String nome
+Linha
+add: void +toString: String

3.3.7 Classe ListaC

A Classe ListaC armazena a lista dos Componentes do circuito em ArrayList<Componente> l e implementa a Análise de Inspeção Nodal descrita em 2.1.2 e a Teoria de Conversão de Fontes descrita em 3.1. O Quadro 11 mostra o Diagrama UML da Classe ListaC.

Os métodos a seguir são invocados pela classe Interface, para montar as equações do circuito e resolver as equações.

- +listarNos: Identifica a quantidade de nós do circuito.

Quadro 11 – Diagrama UML da Classe ListaC

ListaC
-ArrayList<Componente> l # ArrayList<String> nos # Frac resp[] # Frac adm[] # Frac incog[] # Frac inv[]
+ListaC()
+listarNos: void +matrizAdmitancias: void +matrizResp: void +matrizIncog: void +calculaT: void +calculaI: void +imprimir: void

- +MatrizAdmitancias: Monta a matriz de acordo com o algoritmo descrito em 2.1.2.1.
- +MatrizResp: Monta a matriz Resposta de acordo com o algoritmo descrito em 2.1.2.2.
- +MatrizIncog: Utiliza o algoritmo de Eliminação de *Gauss-Jordan* para inverter a matriz A e depois multiplica a Matriz Inversa pela Matriz B, obtendo assim a Matriz das Incógnitas que possui as tensões nos nós e as correntes das fontes de tensão que não foram convertidas.
- +calculaT: Identifica os 2 nós em que o Ramo está situado e realiza a subtração da tensão dos nós.
- +calculaI: Cada componente tem a forma de cálculo de corrente diferente:
  - Fonte de Corrente: O valor da Corrente é o próprio valor nominal.
  - Fonte de Tensão: Identificar qual incógnita representa a corrente da fonte.
  - Ramo RLC + Fonte de Tensão: Utiliza a Equação 3.4, onde a Tensão do Ramo foi calculada por CalculaT, as condições iniciais dos componentes e as impedâncias já são conhecidas logo, a corrente pode ser isolada.

    E a tensão de cada componente pode ser calculada:
    - Resistor: Lei de Ohm.
    - Capacitor: A Equação 3.53;

$$[V_1(s) - V_2(s)] = \frac{I(s)}{sC} + \frac{v_0}{s} \tag{3.53}$$

* Indutor: A Equação 3.54;

$$[V_1(s) - V_2(s)] = I(s)\{sL\} + L\, i_0 \tag{3.54}$$

3.4 CLASSE INTERFACE

A classe Interface descreve a interface gráfica do aplicativo. A classe armazena e modifica a lista de componentes adicionados, responde a ações vindas do *Mouse*, Teclado e do Menu para modificar o circuito e implementa os métodos para desenhar os componentes na Tela.

Ao executar o programa, abre-se uma janela com um PaintPanel branco, para desenhar o circuito. O Quadro 12 mostra o Diagrama UML da Classe Interface.

Quadro 12 – Diagrama UML da Classe Interface

Interface
-ListaC[] list -ArrayList<Linha> linha[] den
+Interface()
-class ActionPerformed -class KeyHandler -class MouseClickHandler -class ResistorFrame -class IndutorFrame -class CapacitorFrame -class VDCFrame -class IDCFrame -class VACFrame -class IACFrame -class ResultadosFrame -class PaintPanel

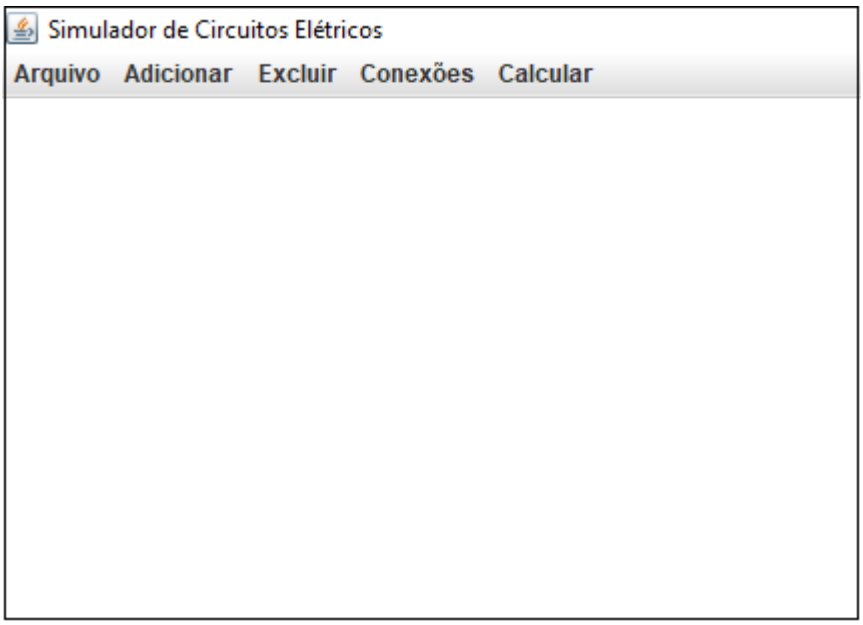
A Figura 11 apresenta um exemplo para a aplicação da Lei de Kirchoff para as correntes.

As classes privadas *ActionPerformed*, *KeyHandler* e *MouseClickHandler* executam rotinas respondendo a ação do teclado, cliques ou arrasto do mouse para permitir o desenho do circuito e realizar o cálculo das tensões e correntes nos componentes.

3.4.1 Private class ActionPerformed

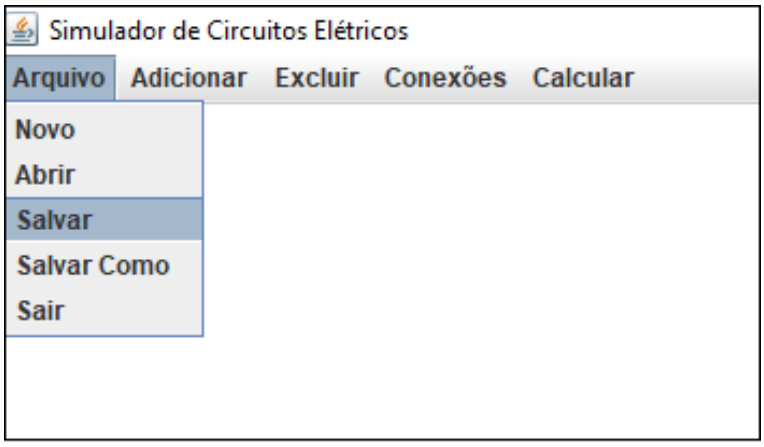
A classe *ActionPerformed* responde às ações originadas do menu. O menu Arquivo mostrado pela Figura 12 possui 5 Menus-Itens.

Figura 11 – Interface do aplicativo



Fonte: Produção do próprio autor.

Figura 12 – Menu Arquivo



Fonte: Produção do próprio autor.

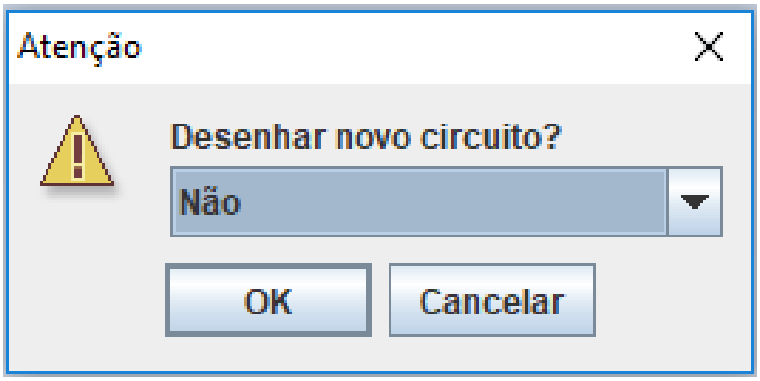
- Novo: Apaga todos os componentes e conexões do circuito, se o circuito não for salvo, aparece um *JOptionPane*, representado pela Figura 13, perguntando para se o usuário deseja salvar o circuito.

- Abrir: Abre um *JFileChooser*, representado pela Figura 14 , pedindo para abrir um arquivo tipo circuito já desenhado anteriormente.

- Salvar: Salva o Circuito. Caso não exista nenhum arquivo aberto, comporta da mesma maneira que Salvar Como.

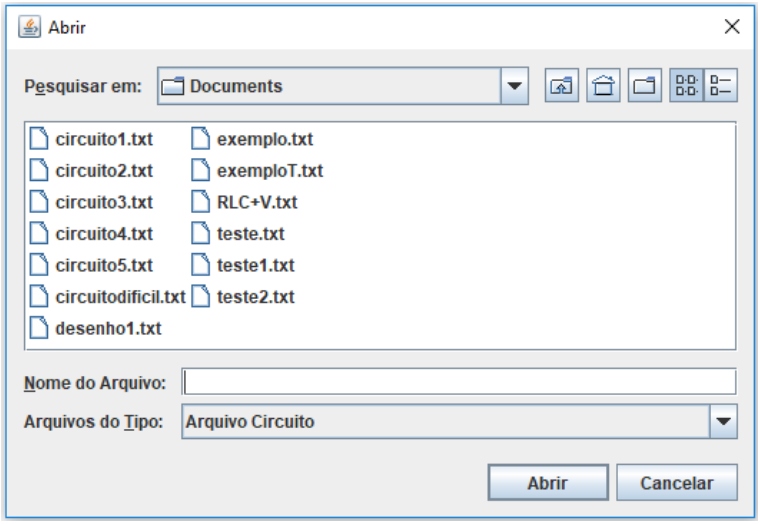
- Salvar Como: Abre um *JFileChooser*, representado pela Figura 15, para o usuário criar

Figura 13 – Novo Circuito



Fonte: Produção do próprio autor.

Figura 14 – Abrir Arquivo



Fonte: Produção do próprio autor.

um arquivo e salvar os dados do circuito.

- Sair: Abre um *JOptionPane*, representado pela Figura 16, perguntando se o usuário deseja salvar o circuito antes de fechar o programa.

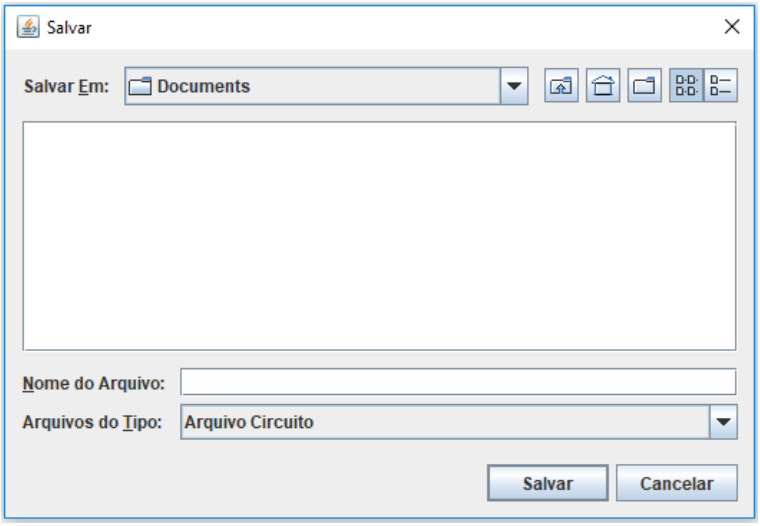
O menu Adicionar Componente mostrado pela Figura 17 muda cursor para Cursor.CROSSHAIR_ CURSOR e adiciona o componente no lugar do PaintPanel que o usuário clicar.

O menu Excluir mostrado pela Figura 18 possui apenas um Menu-Item: Componente, que muda o cursor para Cursor.E_ RESIZE_ CURSOR e apaga o componente e todas as suas conexões.

O menu Conexões mostrado pela Figura 19 possui 3 Menu-Itens:

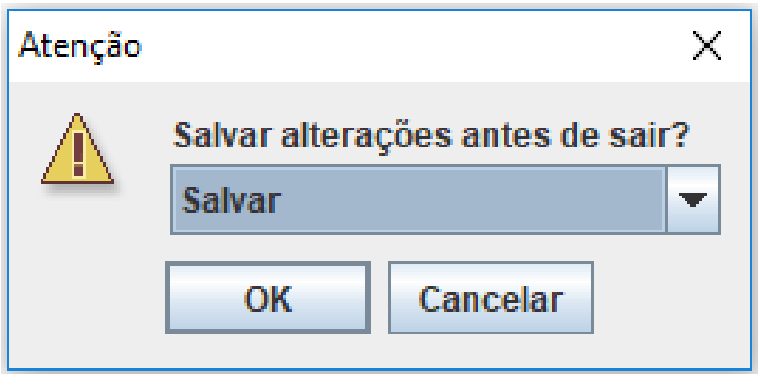
- Desenhar: Cria conexões entre os componentes. Cada linha deve começar e terminar em um componente, senão é excluída. Para fazer que a linha tenha trechos formando

Figura 15 – Salvar Como



Fonte: Produção do próprio autor.

Figura 16 – Sair



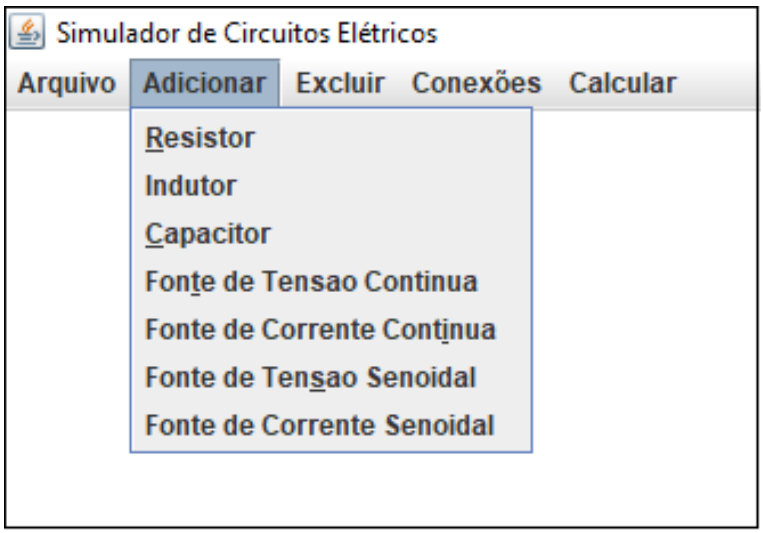
Fonte: Produção do próprio autor.

ângulos retos, basta clicar em pontos intermediários, que o programa tentará encontrar o melhor caminho. Para facilitar o desenho, nesse modo aparece pequenas circunferências vermelhas indicando a região em que o programá interpreta como próximo ao nó. A Figura 20 mostra a ligação em andamento entre dois resistores. Depois de adicionar os componentes, o usuário clicou na extremidade de um componente, depois clicou em um ponto intermediário, para a conexões forme uma linha reta e depois irá clicar na extremidade do outro componente, assim irá indicar que a linha terminar nesse local.

- Apagar Conexões de 1 componente: Funciona como Excluir Componente, porém esse Item apaga apenas as conexões.
- Apagar Todas as Conexões.

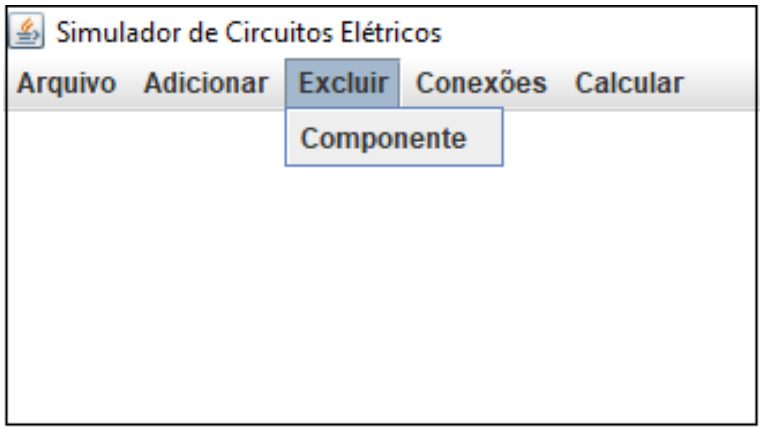
O último Menu: Calcular, mostrado pela Figura 21, identifica se todos os componentes foram

Figura 17 – Adicionar



Fonte: Produção do próprio autor.

Figura 18 – Excluir



Fonte: Produção do próprio autor.

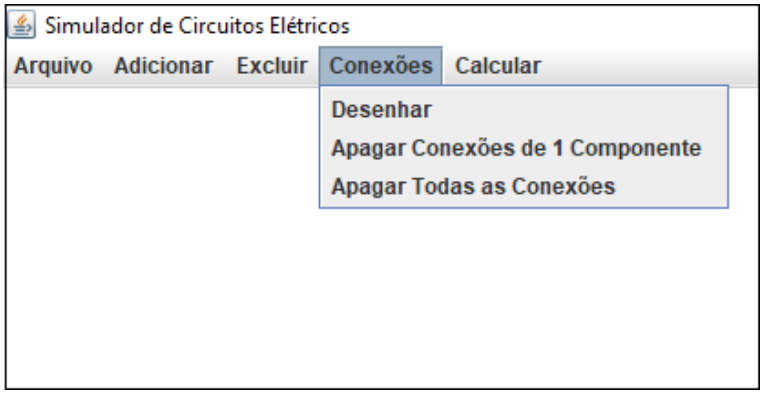
iniciados e tem conexões com outros componentes e o circuito não tiver problemas, o método chama os métodos da Classe ListaC e depois abre uma janela com uma Lista de Tensões e Correntes do Circuito.

3.4.2 Private Class KeyHandler

A Classe KeyHandler apresenta atalhos para facilitar o desenho. Todas as teclas, com exceção da tecla ESC, realizam uma função descrita da classe ActionPerformed.

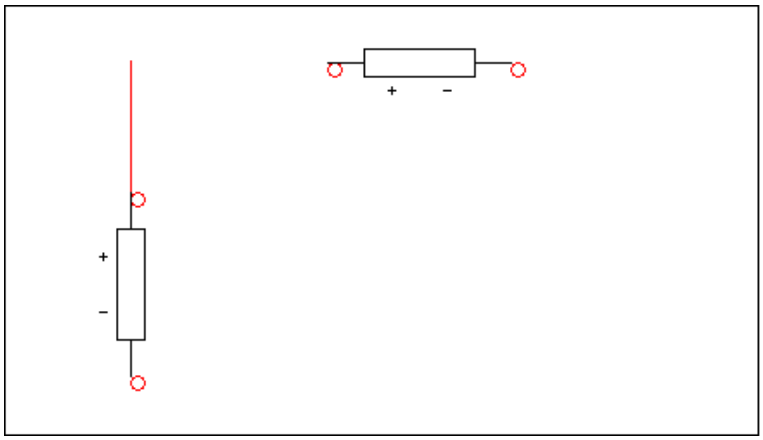
- R - Adiciona um Resistor
- L - Adiciona um Indutor

Figura 19 – Conexões



Fonte: Produção do próprio autor.

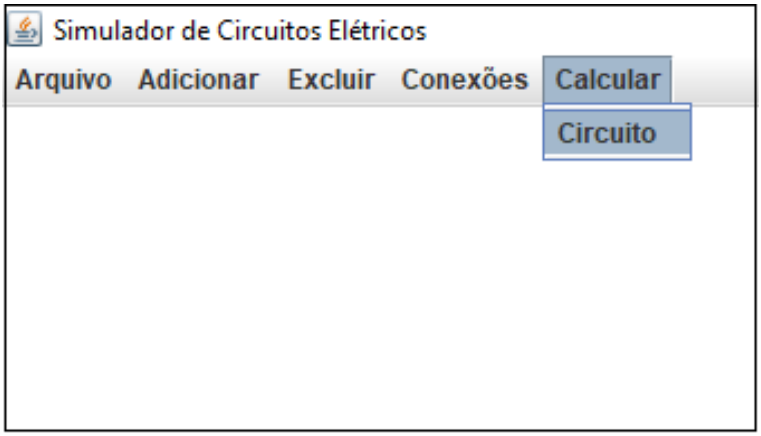
Figura 20 – Conectando Componentes



Fonte: Produção do próprio autor.

- C - Adiciona um Capacitor
- T - Adiciona uma Fonte de Tensão Contínua
- I - Adiciona uma Fonte de Corrente Contínua
- S - Adiciona uma Fonte de Tensão Senoidal
- G - Adiciona uma Fonte de Corrente Senoidal
- E - Exclui um componente
- A - Apaga as conexões de um componente
- W - Cria conexões entre componentes
- ESC - Cancela a operação atual

Figura 21 – Calcular Circuito



Fonte: Produção do próprio autor.

3.4.3 Private Class MouseClickHandler

A Classe MouseClickHandler tem uma ação diferente para cada tipo de ação com o mouse.

- MouseClicked: Se clicar em cima de um componente, muda a cor do componente para verde.

  Se clicar duas vezes no componente, abre uma Janela para alterar o valor do componente.

  Se clicar com o botão direito do mouse, gira o componente no sentido horário.

  Se clicar após um evento de ActionPerformed ou KeyHandler, adiciona, remove o componente ou realiza conexões de acordo com o evento anterior.

- Mouse Pressed e Mouse Dragged: Realiza operações de arrasto do componente. Essa operação é realizada apenas se o componente não tiver nenhuma conexão.

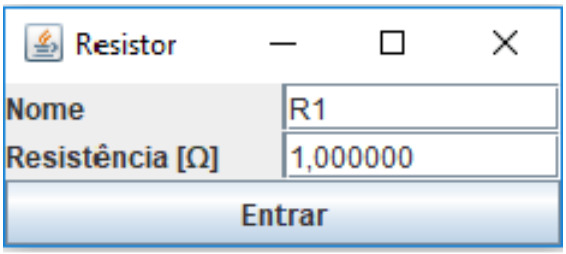
3.4.4 Classes ResistorFrame, IndutorFrame e CapacitorFrame

As classes Classes ResistorFrame, IndutorFrame e CapacitorFrame são chamadas ao clicar 2 vezes em um componente e abre uma janela para o preenchimento dos atributos das classes Resistor, Indutor e Capacitor. A Figura 22, a Figura 23 e a Figura 24 mostram a Interface das Classes ResistorFrame, IndutorFrame e CapacitorFrame, respectivamente.

3.4.5 Classes VDCFrame, IDCFrame, VACFrame e IACFrame

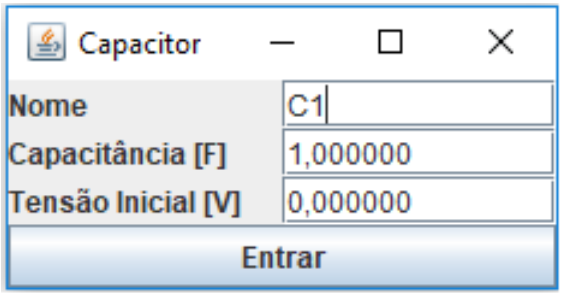
As Classes VDCFrame, IDCFrame, VACFrame e IACFrame também são abertas para editar as atributos das classes VDC, IDC, VAC e IAC, respectivamente. A Figura 25, a Figura 26,

Figura 22 – Interface da Classe ResistorFrame



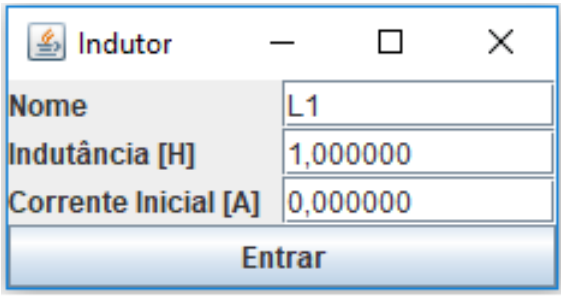
Fonte: Produção do próprio autor.

Figura 23 – Interface da Classe CapacitorFrame



Fonte: Produção do próprio autor.

Figura 24 – Interface da Classe IndutorFrame



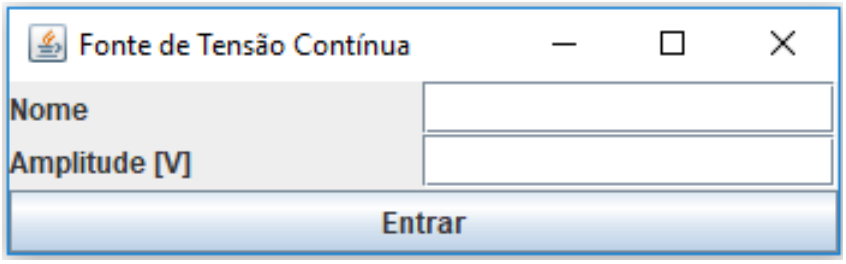
Fonte: Produção do próprio autor.

a Figura 27 e a Figura 28 mostram a Interface das Classes ResistorFrame, IndutorFrame e CapacitorFrame, respectivamente.

3.4.6 Classe ResultadosFrame

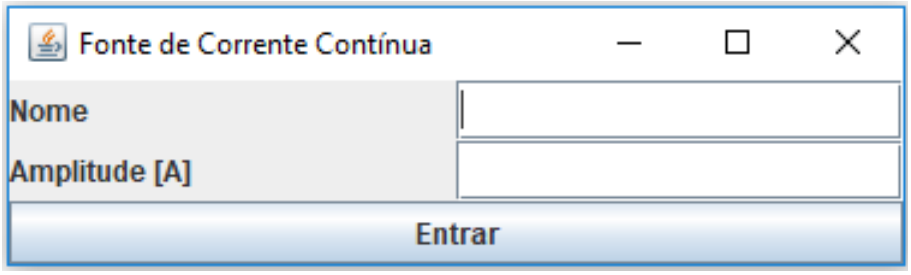
Essa classe quando ativada abre uma janela contendo as tensões e correntes de todos os componentes.

Figura 25 – Interface da Classe VDCFrame



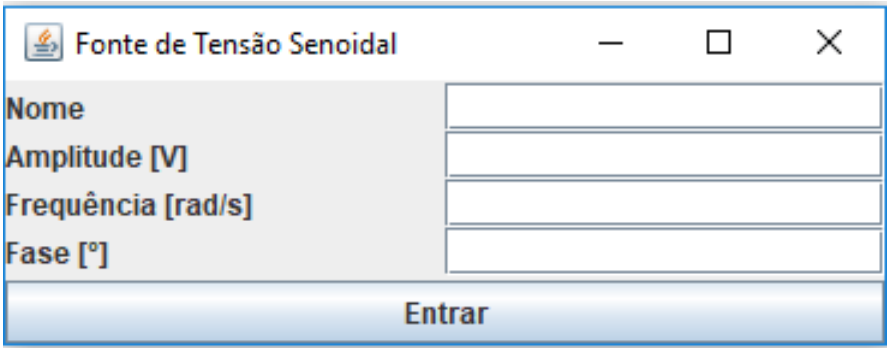
Fonte: Produção do próprio autor.

Figura 26 – Interface da Classe IDCFrame



Fonte: Produção do próprio autor.

Figura 27 – Interface da Classe VACFrame

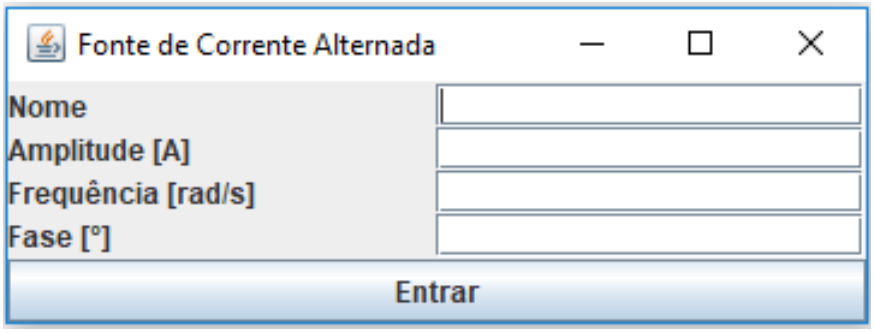


Fonte: Produção do próprio autor.

3.4.7 Classe PaintPanel

A classe PaintPanel possui apenas um método paintComponente que atualiza o desenho sempre que o aplicativo recebe um evento. O método paintComponente percorre as listas de linha e de componentes e com base na posição, que é um dos atributos da lista, desenha cada item. O Quadro 13 mostra parcialmente o código implementado. Se o componente for um Resistor, o método desenha um retângulo na posição do aplicativo, se o componente for um Indutor, o método desenha os 4 semi-circulos que definem o componente. Cada componente é desenhado diferentemente para representar o componente como a Figura 9.

Figura 28 – Interface da Classe IACFrame



Fonte: Produção do próprio autor.

Quadro 13 – Implementação parcial do método paintComponente

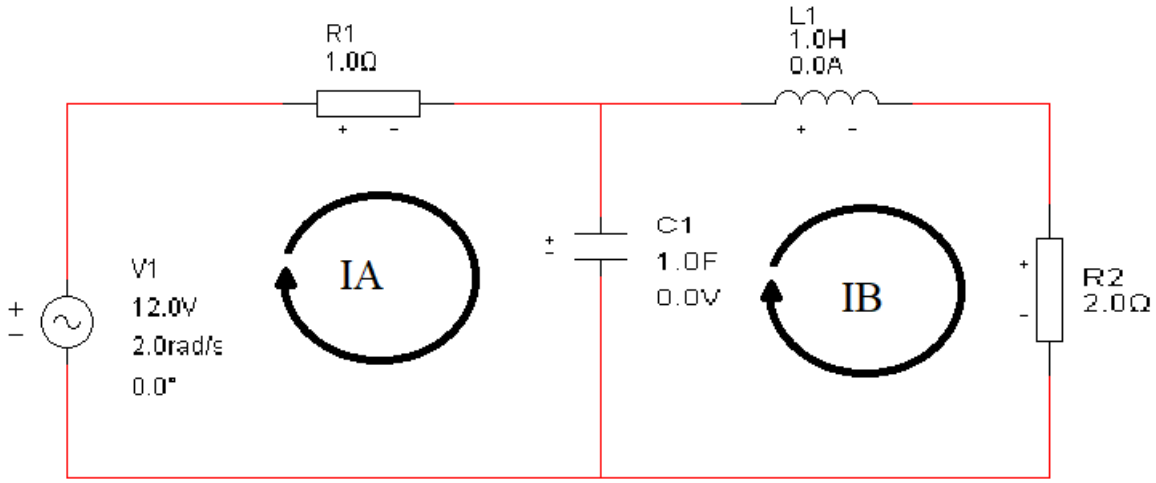
```
if(list.l.get(i) instanceof Resistor)
{

g.drawRect(list.l.get(i).point.x, list.l.get(i).point.y,
list.l.get(i).largura, list.l.get(i).altura);
}
else if(list.l.get(i) instanceof Indutor)
{
if(list.l.get(i).posicao==0)
 {
 g.drawArc(list.l.get(i).point.x, list.l.get(i).point.y,
list.l.get(i).largura, list.l.get(i).largura, 90,-180);
 g.drawArc(list.l.get(i).point.x, list.l.get(i).point.y+list.l.get(i).largura,
list.l.get(i).largura, list.l.get(i).largura, 90,-180);
 g.drawArc(list.l.get(i).point.x, list.l.get(i).point.y+2*list.l.get(i).largura,
list.l.get(i).largura, list.l.get(i).largura, 90,-180);
 g.drawArc(list.l.get(i).point.x, list.l.get(i).point.y+3*list.l.get(i).largura,
list.l.get(i).largura, list.l.get(i).largura, 90,-180);
 }
} }
```

4 RESULTADOS

Os circuitos a seguir foram adaptados do Livro (IRWIN; NELMS, 2015) com as equações matriciais escritas manualmente e resolvidas com auxílio da calculadora TI-Nspire CX CAS e site Wolfram Alpha (WOLFRAM ALPHA LLC, 2018)

O circuito da Figura 29 apresenta 2 malhas, uma Fonte de Tensão Senoidal, Resistores, Indutor e Capacitor e foi resolvido pelo Métodos das Tensões nas Malhas como mostram as Matrizes da Equação 4.1. A resolução do circuito via aplicativo é mostrada no Quadro 14, onde a corrente I_A é igual a corrente da Fonte V_1 e da Resistência R_1 e a corrente I_B é igual a corrente no Indutor L_1 e da Resistência R_2.

Figura 29 – Circuito 1



Fonte: Produção do próprio autor.

$$\begin{vmatrix} 1+\frac{1}{s} & -\frac{1}{s} \\ -\frac{1}{s} & \frac{1}{s}+s+2 \end{vmatrix} \begin{vmatrix} I_A \\ I_B \end{vmatrix} = \begin{vmatrix} \frac{24}{s^2+4} \\ 0 \end{vmatrix} \tag{4.1a}$$

$$\begin{vmatrix} I_A(s) \\ I_B(s) \end{vmatrix} = \begin{vmatrix} \frac{24(s^2+2s+1)}{(s^2+4)(s^2+3s+3)} \\ \frac{24}{(s^2+4)(s^2+3s+3)} \end{vmatrix} \tag{4.1b}$$

$$\begin{vmatrix} I_A(t) \\ \\ \\ I_B(t) \end{vmatrix} = \begin{vmatrix} 8,7567567\sin(2t) + 4,54054054cos(2t) \\ +e^{-1,5t}\left[-0,374497\sin(0,866025t) + 4,54054\cos(0,866025t)\right]\ A \\ \\ -0,32432432\sin(2t) - 1,945945\cos(2t) \\ +e^{-1,5t}\left[4,11947\sin(0,866025t) + 1,94594\cos(0,866025t)\right]\ A \end{vmatrix} \tag{4.1c}$$

Quadro 14 – Resolução do Circuito 1 via Aplicativo

Nome: V1
Amplitude da Tensão: 12.0 V
Frequência: 2.0 rad/s
Fase: 0.0 °
Tensão: [12.0sin(2.0t)] V
Corrente: [4.54054054054054cos(2.0t)+8.756756756756756sin(2.0t)]
+ exp (-1.5t)[-4.54054054054054cos(0.8660254037844386t)
- 0.3744974719067843sin(0.8660254037844386t)] A

Nome: R1
Resistência: 1.0 Ω
Tensão: [4.54054054054054cos(2.0t)+8.756756756756756sin(2.0t)]
+ exp (-1.5t)[-4.5405405405405cos(0.8660254037844386t)
- 0.374497471906784sin(0.8660254037844386t)] V
Corrente: [4.54054054054054cos(2.0t) + 8.756756756756756sin(2.0t)]
+ exp (-1.5t)[-4.5405405405405cos(0.8660254037844386t)
- 0.374497471906784sin(0.8660254037844386t)] A

Nome: L1
Indutância: 1.0 H
Corrente inicial: 0.0 A
Tensão: [-0.6486486486486487cos(2.0t)+ 3.891891891891892sin(2.0t)]
+ exp (-1.5t)[0.648648648648648cos(0.8660254037844386t)
- 7.8644469100424sin(0.8660254037844386t)] V
Corrente: [-1.945945945945946cos(2.0t) - 0.32432432432432434sin(2.0t)]
+ exp (-1.5t)[1.94594594594594cos(0.8660254037844386t)
+ 4.11947219097462sin(0.8660254037844386t)] A

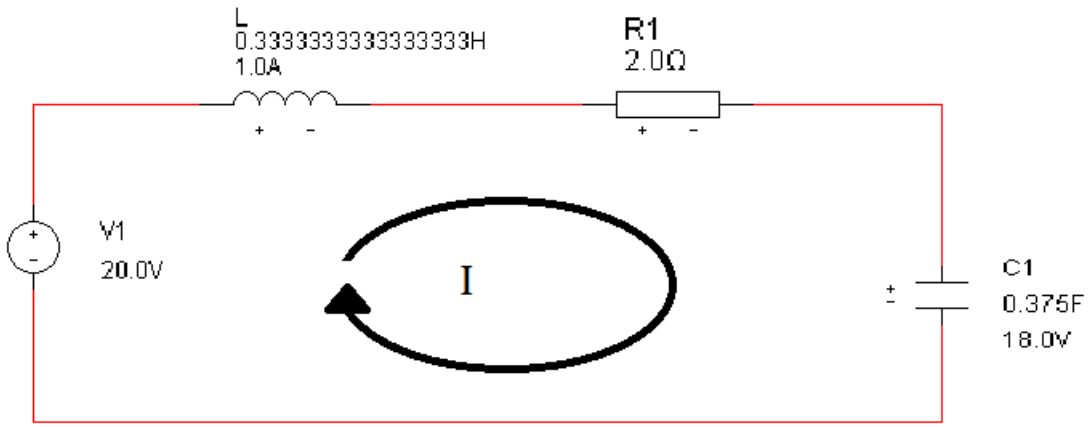
Nome: C1
Capacitância: 1.0 F
Tensão inicial: 0.0 V
Tensão: [-4.54054054054054cos(2.0t) + 3.2432432432432434sin(2.0t)]
+ exp (-1.5t)[4.5405405405405cos(0.8660254037844386t)
+ 0.374497471906784sin(0.8660254037844386t)] V
Corrente: [6.486486486486487cos(2.0t) + 9.08108108108108sin(2.0t)]
+ exp (-1.5t)[-6.48648648648648cos(0.8660254037844386t)
- 4.49396966288141sin(0.8660254037844386t)] A

Nome: R2
Resistência: 2.0 Ω
Tensão: [-3.891891891891892cos(2.0t)-0.6486486486486487sin(2.0t)]
+ exp (-1.5t)[3.89189189189189cos(0.8660254037844386t)
+ 8.23894438194925sin(0.8660254037844386t)] V
Corrente: [-1.945945945945946cos(2.0t)-0.32432432432432434sin(2.0t)]
+ exp (-1.5t)[1.94594594594594cos(0.8660254037844386t)
+ 4.11947219097462sin(0.8660254037844386t)] A

Tempo de execução: 18.78 s

O circuito da Figura 30 apresenta apenas 1 malha, uma Fonte de Tensão Contínua, Resistor, Indutor com Corrente Inicial e Capacitor Carregado e foi resolvido pelo Métodos das Tensões nas Malhas como mostra a Equação 4.5. A resolução do circuito via aplicativo é mostrada no Quadro 15, onde a corrente I é igual a corrente de Malha e de todos os componentes.

Figura 30 – Circuito 2



Fonte: Produção do próprio autor.

$$20 = \frac{1}{3}\frac{di(t)}{dt} + 2i(t) + \frac{8}{3}\int i(t)dt \tag{4.2}$$

$$\frac{20}{s} = \frac{1}{3}[sI(s) - i(0)] + 2I(s) + \frac{8}{3}\frac{I(s)}{s} + \frac{v(0)}{s} \tag{4.3}$$

$$I(s) = \frac{s+6}{s^2+6s+8} \tag{4.4}$$

$$I(t) = 2e^{-2t} - e^{-4}\ A \tag{4.5}$$

Quadro 15 – Resolução do Circuito 2 via Aplicativo

Nome: V1
Tensão: 20.0 V
Tensão: 20.0 V
Corrente: 2.0 exp (-2.0t)-1.0 exp (-4.0t) A

Nome: R1
Resistência: 2.0 Ω
Tensão: 4.0 exp (-2.0t)-2.0 exp (-4.0t) V
Corrente: 2.0 exp (-2.0t)-1.0 exp (-4.0t) A

Nome: C1
Capacitância: 0.375 F
Tensão inicial: 18.0 V
Tensão: 20.0-2.6666666666666665 exp (-2.0t) + 0.6666666666666666 exp (-4.0t) V
Corrente: 2.0 exp (-2.0t)-1.0 exp (-4.0t) A

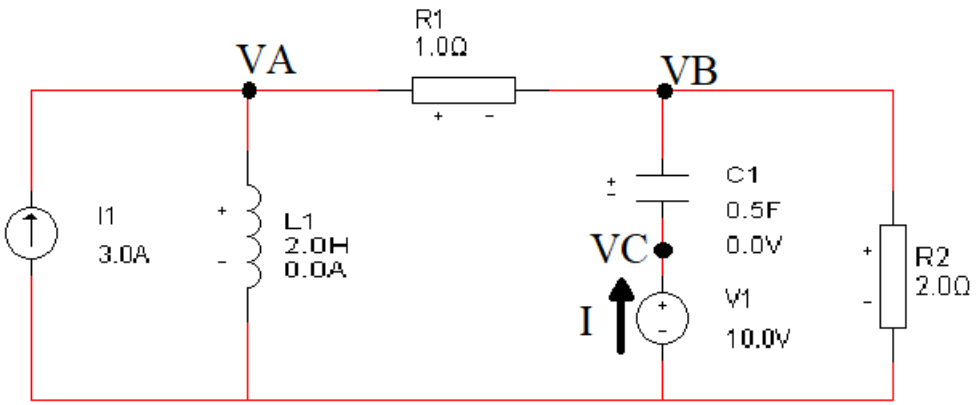
Nome: L
Indutância: 0.3333333333333333 H
Corrente inicial: 1.0 A
Tensão: -1.3333333333333333 exp (-2.0t) + 1.3333333333333333 exp (-4.0t) V
Corrente: 2.0 exp (-2.0t)-1.0 exp (-4.0t) A

Tempo de execução: 1.881 s

O circuito da Figura 31 foi resolvido utilizando o Método da Inspeção Nodal pelas Equações 4.6. O resultado do aplicativo do Quadro 16 foi o mesmo do teórico, onde as Tensões V_A,V_B e V_C são as tensões da fonte de corrente contínua $I1$, do resistor $R2$ e da fonte de tensão continua $V1$ e como existe uma fonte de tensão, foi gerada a variável I para representar a corrente da fonte.

$$\begin{vmatrix} \frac{1}{2s}+1 & -1 & 0 & 0 \\ -1 & 1+\frac{1}{2}+0,5s & -0,5s & 0 \\ 0 & -0,5s & 0,5s & -1 \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} V_A \\ V_B \\ V_C \\ I \end{vmatrix} = \begin{vmatrix} \frac{3}{s} \\ 0 \\ 0 \\ \frac{10}{s} \end{vmatrix} \tag{4.6a}$$

Figura 31 – Circuito 3



Fonte: Produção do próprio autor.

$$\begin{vmatrix} V_A(s) \\ V_B(s) \\ \\ V_C(s) \\ I(s) \end{vmatrix} = \begin{vmatrix} \dfrac{13s+9}{s^2+1,5s+1,5} \\ \dfrac{10s+11}{s^2+1,5s+1,5} \\ \\ \dfrac{10}{s} \\ \dfrac{2s+7,5}{s^2+1,5s+1,5} \end{vmatrix} \tag{4.6b}$$

$$\begin{vmatrix} V_A(t) \\ V_B(t) \\ \\ V_C(t) \\ I(t) \end{vmatrix} = \begin{vmatrix} e^{-0,75t}\left[13\cos(0,96824t) + 0,7745967\sin(0,96824t)\right]\ V \\ e^{-0,75t}\left[2,61784\cos(0,96824t) + 10\sin(0,96824t)\right]\ V; \\ \\ 10\ V \\ e^{-0,75t}\left[1\cos(0,96824t) + 3,09839\sin(0,96824t)\right]\ A \end{vmatrix} \tag{4.6c}$$

Quadro 16 – Resolução do Circuito 3 via Aplicativo

Nome: I1
Corrente: 3.0 A
Tensão: e ^ (-0.75t) [13.0 cos(0.9682458365518543t)
-0.7745966692414834 sin(0.9682458365518543t)] V
Corrente: 3.0 A

Nome: L1
Indutância: 2.0 H
Corrente inicial: 0.0 A
Tensão: exp (-0.75t)[13.0 cos(0.9682458365518543t)
- 0.7745966692414834 sin(0.9682458365518543t)] V
Corrente: 3.0+ exp (-0.75t)[-3.0 cos(0.9682458365518543t)
+ 4.389381125701739 sin(0.9682458365518543t)] A

Nome: R1
Resistência: 1.0 Ω
Tensão: exp (-0.75t)[3.0 cos(0.9682458365518543t)
- 4.389381125701739 sin(0.9682458365518543t)] V
Corrente: exp (-0.75t)[3.0 cos(0.9682458365518543t)
- 4.389381125701739 sin(0.9682458365518543t)] A

Nome: C1
Capacitância: 0.5 F
Tensão inicial: 0.0 V
Tensão: -10.0+ exp (-0.75t) [10.0 cos(0.9682458365518543t)
+ 3.614784456460255 sin(0.9682458365518543t)] V
Corrente: exp (-0.75t) [-2.0 cos(0.9682458365518543t)
- 6.19677335393186 sin(0.9682458365518543t)] A

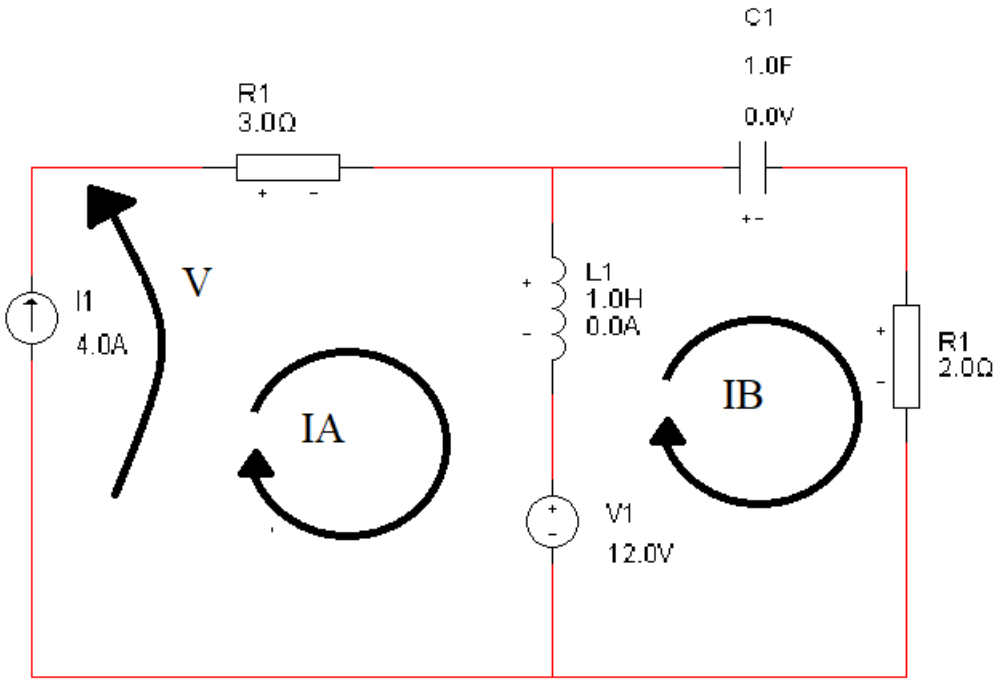
Nome: V1
Tensão: 10.0 V
Tensão: 10.0 V
Corrente: e ^ (-0.75t) [2.0 cos(0.9682458365518543t)
+ 6.196773353931867 sin(0.9682458365518543t)] A

Nome: R2
Resistência: 2.0 Ω
Tensão: exp (-0.75t) [10.0 cos(0.9682458365518543t)
+ 3.6147844564602556 sin(0.9682458365518543t)] V
Corrente: exp(-0.75t) [5.0 cos(0.9682458365518543t)
+ 1.8073922282301278 sin(0.9682458365518543t)] A

Tempo de execução: 2.028 s

O circuito da Figura 32 apresenta uma fonte de corrente continua e fonte de tensão continua e possui ressonância, por isso aparece nas Equações de I_B e V aparece um fator multiplicado por t na Equação 4.7 e a Resolução do Aplicativo aparece no Quadro 17.

Figura 32 – Circuito 4



Fonte: Produção do próprio autor.

$$\begin{vmatrix} 3+s & -s & -1 \\ -s & s+\frac{1}{s}+2 & 0 \\ 1 & 0 & 0 \end{vmatrix} \begin{vmatrix} I_A \\ I_B \\ V \end{vmatrix} = \begin{vmatrix} \frac{-12}{s} \\ \frac{12}{s} \\ \frac{4}{s} \end{vmatrix} \tag{4.7a}$$

$$\begin{vmatrix} I_A(s) \\ I_B(s) \\ V(s) \end{vmatrix} = \begin{vmatrix} \frac{4}{s} \\ \frac{4}{s+1}+\frac{8}{(s+1)^2} \\ \frac{-4}{s+1}+\frac{8}{(s+1)^2}+\frac{24}{s} \end{vmatrix} \tag{4.7b}$$

$$\begin{vmatrix} I_A(t) \\ \\ I_B(t) \\ \\ V(t) \end{vmatrix} = \begin{vmatrix} 4\ A \\ \\ 4e^{-t} + 8te^{-t}\ A \\ \\ -4e^{-t} + 8te^{-t} + 24\ V \end{vmatrix} \tag{4.7c}$$

Quadro 17 – Resolução do Circuito 4 via Aplicativo

Nome: V1
Tensão: 12.0 V
Tensão: 12.0 V
Corrente: -4.0 + 8.0 t exp(-1.0t) + 4.0 exp(-1.0t) A

Nome: I1
Corrente: 4.0 A
Tensão: 24.0 + 8.0 t exp(-1.0t)-4.0 exp(-1.0t) V
Corrente: 4.0 A

Nome: C1
Capacitância: 1.0 F
Tensão inicial: 0.0 V
Tensão: 12.0-8.0 t exp(-1.0t)-12.0 exp(-1.0t) V
Corrente: 8.0 t exp(-1.0t) + 4.0 exp(-1.0t) A
Nome: L1
Indutância: 1.0 H
Corrente inicial: 0.0 A
Tensão: 8.0 t exp(-1.0t)-4.0 exp(-1.0t) V
Corrente: 4.0-8.0 t exp(-1.0t)-4.0 exp(-1.0t) A

Nome: R1
Resistência: 3.0 Ω
Tensão: -12.0 V
Corrente: -4.0 A

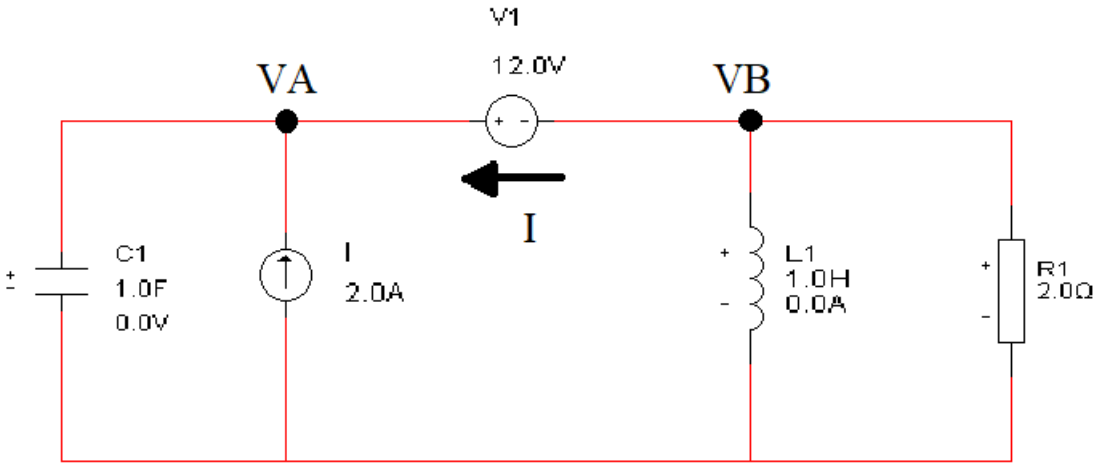
Nome: R2
Resistência: 2.0 Ω
Tensão: 16.0 t exp(-1.0t) + 8.0 exp(-1.0t) V
Corrente: 8.0 t exp(-1.0t) + 4.0 exp(-1.0t) A

Tempo de execução: 4.409 s

O circuito da Figura 33 possui uma fonte de tensão contínua que não pôde ser convertida para fonte de corrente, então a corrente da fonte é mais uma variável pelo método de Inspeção

Nodal da Equação 4.8 e a Resolução via Aplicativo está no Quadro 18.

Figura 33 – Circuito 5



Fonte: Produção do próprio autor.

$$\begin{vmatrix} s & 0 & -1 \\ 0 & \frac{1}{s}+\frac{1}{2} & 1 \\ 1 & -1 & 0 \end{vmatrix} \begin{vmatrix} V_A \\ V_B \\ I \end{vmatrix} = \begin{vmatrix} \frac{2}{s} \\ 0 \\ \frac{12}{s} \end{vmatrix} \tag{4.8a}$$

$$\begin{vmatrix} V_A(s) \\ V_B(s) \\ I(s) \end{vmatrix} = \begin{vmatrix} \frac{12}{s}+\frac{-24s+4}{2s^2+s+2} \\ \frac{-24s+4}{2s^2+s+2} \\ \frac{-2}{s}+\frac{24s+16}{2s^2+s+2} \end{vmatrix} \tag{4.8b}$$

$$\begin{vmatrix} V_A(t) \\ V_B(t) \\ I(t) \end{vmatrix} = \begin{vmatrix} e^{-0,25t}\left[-12\cos(0,96824t)+5,163978\sin(0,96824t)\right]+12\ V \\ e^{-0,25t}\left[-12\cos(0,96824t)+5,163978\sin(0,96824t)\right]\ V \\ e^{-0,25t}\left[3,09839\cos(0,96824t)+12\sin(0,96824t)\right]\ A \end{vmatrix} \tag{4.8c}$$

Quadro 18 – Resolução do Circuito 5 via Aplicativo

Nome: C1
Capacitância: 1.0 F
Tensão inicial: 0.0 V
Tensão: 12.0+ exp(-0.25t)[-12.0cos(0.9682458365518543t)
+ 5.163977794943222sin(0.9682458365518543t)] V
Corrente: exp(-0.25t)[8.0cos(0.9682458365518543t)
+ 10.327955589886445sin(0.9682458365518543t)] A

Nome: I
Corrente: 2.0 A
Tensão: 12.0+ exp(-0.25t)[-12.0cos(0.9682458365518543t)
+ 5.163977794943222sin(0.9682458365518543t)] V
Corrente: 2.0 A

Nome: V1
Tensão: 12.0 V
Tensão: 12.0 V
Corrente: -2.0+ exp(-0.25t)[8.0cos(0.9682458365518543t)
+ 10.327955589886445sin(0.9682458365518543t)] A

Nome: L1
Indutância: 1.0 H
Corrente inicial: 0.0 A
Tensão: exp(-0.25t)[-12.0cos(0.9682458365518543t)
+ 5.163977794943222sin(0.9682458365518543t)] V
Corrente: 2.0+ exp(-0.25t)[-2.0cos(0.9682458365518543t)
- 12.909944487358056sin(0.9682458365518543t)] A

Nome: R1
Resistência: 2.0 Ω
Tensão: exp(-0.25t)[-12.0cos(0.9682458365518543t)
+ 5.163977794943222sin(0.9682458365518543t)] V
Corrente: exp(-0.25t)[-6.0cos(0.9682458365518543t)
+ 2.581988897471611sin(0.9682458365518543t)] A

Tempo de execução: 2.477 s

5 CONCLUSÃO

Uma das dificuldades do Curso de Engenharia Elétrica é resolver Circuitos Elétricos trabalhosos sem ter acesso à resposta correta. A fim de solucionar tal problema, buscou-se desenvolver, no presente trabalho, um programa que resolvesse complexos exercícios. Para isso, foi necessário revisar e aprofundar os conceitos aprendidos durante o Curso, desde o pré-Cálculo, onde foi trabalhado as quatro operações: soma, subtração, multiplicação e divisão envolvendo frações de polinômios, além de fatoração de polinômios utilizando métodos mais avançados que o conteúdo do curso de Cálculo Numérico; a Análise Nodal por Inspeção foi ampliada para permitir o uso de capacitores e indutores, e criou-se um algoritmo para converter fontes de tensão em série com algum componente RLC em fontes de corrente em paralelo com uma admitância.

Durante o curso, o conceito de conversão de fontes foi estudado exclusivamente para a Análise de Regime Permanente de Circuitos Lineares, mas fez-se necessário mostrar, neste trabalho, que a conversão também pode ser usada em circuitos RLC, com Indutores e/ou Capacitores Carregados.

A fim de proporcionar ao usuário do programa uma melhor compreensão do que está sendo calculado, desenvolveu-se uma interface gráfica em Java a qual permite o desenho de circuitos elétricos com facilidade.

Os resultados obtidos mostraram que o aplicativo atende sua principal tarefa que é a de resolver, em poucos segundos, circuitos complexos que levariam pelo menos quinze minutos se resolvidos manualmente, isto considerando que o usuário tem bom conhecimento do assunto e faz uso de uma calculadora gráfica. Para alunos em fase de aprendizagem e portando somente calculadoras científicas as quais não trabalham com números complexos, este tempo é muito maior e a resolução de circuitos elétricos se torna um trabalho árduo.

Assim, o programa oferece suporte e auxílio àqueles que desejam assegurar-se da correta resolução de exercícios e poder corrigi-los para melhor aproveitamento de seus estudos.

REFERÊNCIAS

BOYLESTAD, R. L. **Introdução à Análise de circuitos**. São Paulo: Pearson Prentice Hall, 2012.

CAMPAGNOLO, J. M. **Resolução Numérica de Equações Não-Lineares**. 2012. Disponível em: <http://www.labspot.ufsc.br/~campagno/numerico/Aula_6.pdf>. Acesso em: 03 fev. 2018.

DEITEL, P.; DEITEL, H. **Java: como programar**. São Paulo: Pearson Education do Brasil, 2017. Acesso em: 15 fev. 2018.

IRWIN, J. D.; NELMS, R. M. **Basic engineering circuit analysis**. New Jersey: John Wiley and Sons, 2015. Acesso em: 30 out. 2018.

LOGICIELS POUR LA PHYSIQUE. **Solve Elec**. 2018. Disponível em: <http://www.physicsbox.com/indexsolveelec2en.html>. Acesso em: 26 nov. 2018.

MATTAR, R. J. **A uni-variant polynomial with rational coefficients**. 2006. Disponível em: <https://github.com/miraclefoxx/bigdecimal-math/blob/master/src/main/java/io/github/miraclefoxx/math/RatPoly.java>. Acesso em: 7 set. 2018.

NATIONAL INSTRUMENTS CORPORATION. **NI Multisim User Manual**. 2009. Disponível em: <http://www.ni.com/pdf/manuals/374483d.pdf>. Acesso em: 26 nov. 2018.

THE MATHWORKS, INC. **MATLAB Product Description**. 2018. Disponível em: <https://www.mathworks.com/help/matlab/learn_matlab/product-description.html>. Acesso em: 26 nov. 2018.

WEISSTEIN, E. W. **Gaussian Elimination**. 2018. Disponível em: <http://mathworld.wolfram.com/GaussianElimination.html>. Acesso em: 13 mai. 2018.

WEISSTEIN, E. W. **Matrix Multiplication**. 2018. Disponível em: <http://mathworld.wolfram.com/MatrixMultiplication.html>. Acesso em: 13 mai. 2018.

WEISSTEIN, E. W. **Polynomial Roots**. 2018. Disponível em: <http://mathworld.wolfram.com/PolynomialRoots.html>. Acesso em: 13 mai. 2018.

WOLFRAM ALPHA LLC. **Wolfram Alpha**. 2018. Disponível em: <https://www.wolframalpha.com>. Acesso em: 03 abr. 2018.

ZILL, D. G. **Equações Diferenciais**. São Paulo: Pearson Makron Books, 2001.