UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UMA ESTRATÉGIA DE INTERAÇÃO NA *WEB* PARA A ANÁLISE DE SISTEMAS ELÉTRICOS DE POTÊNCIA

MÁRCIO AUGUSTO TAMASHIRO

ORIENTADOR: PROF. GERALDO CAIXETA GUIMARÃES, PH.D.

Uberlândia - MG
Outubro/2016

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UMA ESTRATÉGIA DE INTERAÇÃO NA *WEB* PARA A ANÁLISE DE SISTEMAS ELÉTRICOS DE POTÊNCIA

MÁRCIO AUGUSTO TAMASHIRO

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de Doutor em Ciências, área de concentração: Dinâmica de Sistemas Elétricos
Orientador: Profº Geraldo Caixeta Guimarães, Ph.D.

Uberlândia - MG
Outubro/2016

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

T153e
2016

Tamashiro, Márcio Augusto, 1974-

Uma estratégia de interação na Web para a análise de sistemas elétricos de potência / Márcio Augusto Tamashiro. - 2016.

127 f. : il.

Orientador: Geraldo Caixeta Guimarães.

Tese (doutorado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.

Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Sistemas de energia elétrica - Teses. 3. Software - Desenvolvimento - Teses. 4. Simulação (Computadores) - Teses. I. Guimarães, Geraldo Caixeta, 1954- II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

UMA ESTRATÉGIA DE INTERAÇÃO NA *WEB* PARA A ANÁLISE DE SISTEMAS ELÉTRICOS DE POTÊNCIA

Márcio Augusto Tamashiro

"Esta tese foi julgada adequada para obtenção do título de Doutor em Ciências, área de concentração em *Dinâmica de Sistemas Elétricos*, e aprovada pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Uberlândia"

_____

Prof. Geraldo Caixeta Guimarães, Ph.D.

Orientador

_____

Prof. Darizon Alves de Andrade, Ph.D.

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Profº. Geraldo Caixeta Guimarães, Ph.D. –UFU (Presidente)

Profº. Adélio José de Moraes, Dr. – UFU

Profº. David Calhau Jorge, Dr. – UFTM

Profº. Edgard Afonso Lamounier Júnior, Ph.D. – UFU

Profº. Sérgio Batista da Silva, Dr. – IFG

Dedico este trabalho a minha esposa Ludiane e aos meus filhos, Luís Felipe e Eduardo,, minhas fontes permanentes de carinho, amor e inspirações

AGRADECIMENTOS

Acima de tudo a Deus por ter me conduzido do início ao fim desta tese e por sempre estar presente, principalmente nas horas em que mais precisei.

Ao professor Geraldo Caixeta Guimarães, meu sincero agradecimento pela orientação, apoio e confiança depositada, que foram imprescindíveis para que este trabalho pudesse ser concluído com êxito.

Aos amigos do Núcleo de Dinâmica de Sistemas Elétricos da UFU, Anderson, Antônio Manoel, André Roger, Daniel, Glauber, Larissa, Leonardo, Jaqueline, Marcelo Pansani, Roberta e Thales pelo convívio e apoio ao longo desses últimos quatro anos.

À secretária da pós-graduação Cinara Fagundes pela sua atenção e disponibilidade no atendimento de solicitações e esclarecimentos de dúvidas.

À Universidade Federal de Uberlândia, pelo apoio institucional, ao Programa de Pós-graduação em Engenharia Elétrica pelo suporte técnico, e à CAPES, pelo apoio financeiro.

Ao IFTO por permitir e incentivar a capacitação de docentes e principalmente aos colegas da Coordenação da Área Indústria do Campus Palmas.

Tudo posso naquele que me fortalece

Filipenses 4:13

RESUMO

UMA ESTRATÉGIA DE INTERAÇÃO NA *WEB* PARA A ANÁLISE DE SISTEMAS ELÉTRICOS DE POTÊNCIA

Atualmente, muitos trabalhos técnicos e científicos importantes só são possíveis com o auxílio de computadores e de programas específicos. Na Engenharia Elétrica esses recursos são utilizados em estudos estáticos e dinâmicos de sistemas elétricos de potência, os quais dão suporte, por exemplo, ao planejamento e a operação da rede elétrica realizados pelas empresas do setor. Em função dessa importância é grande a quantidade de programas disponíveis, comerciais ou não. As aplicações comerciais são conhecidas por serem computacionalmente eficientes, bem como pela quantidade de recursos oferecidos, mas apesar disso, não são adequadas para fins educacionais ou para a realização de pesquisas. Um dos principais motivos apontados é porque o código fonte não é fornecido, e assim não podem ser estudados ou adaptados conforme a necessidade. Por isso muitos usuários principalmente do meio acadêmico preferem criar suas próprias aplicações, sendo a maioria delas desenvolvidas no MATLAB ou escritas nas linguagens de programação FORTRAN e C++. Esses programas são disponibilizados como aplicações *desktop* destinadas geralmente a somente um tipo de estudo e com uma interface pouco amigável. Entretanto, existem algumas opções com características mais atrativas como a existência de uma interface gráfica com o usuário, e número de recursos computacionais próximos àqueles encontrados nas aplicações comercias. Na literatura há ainda propostas de aplicações *web* cuja principal vantagem é o acesso remoto e simultâneo por qualquer computador. No geral, as aplicações existentes não disponibilizam recursos de colaboração em tempo real, e não permitem a interoperabilidade com outras aplicações. Nesse contexto, esta tese focou na investigação da implementação de uma aplicação *web* para a análise de sistemas elétricos, explorando esses dois aspectos supracitados. Para isso alguns programas similares, sem fins comerciais e com código fonte disponível, foram investigados. E também foram selecionadas e apresentadas as ferramentas computacionais necessárias ao desenvolvimento da aplicação. As investigações e as implementações computacionais realizadas, bem como os resultados obtidos são devidamente apresentados e analisados ao final deste trabalho.

Palavras-chave: programa para a análise de sistemas elétricos, aplicação *web*, colaboração em tempo real.

ABSTRACT

AN INTERACTION STRATEGY ON THE WEB FOR ELECTRICAL POWER SYSTEMS ANALYSIS

Currently, many important technical and scientific works are only possible with the aid of computers and specific programs. In Electrical Engineering, these resources are both used in static and dynamic studies of electric power systems, which give support, for example, to the planning, and operation of the grid performed by companies in the sector. Because of this, there is a large number of commercial or non-commercial programs available. Commercial applications are known to be computationally efficient as well as the amount of offered resources; nevertheless, they are not suitable for educational purposes or for conducting research. One of the main reasons pointed out is because they are not open source, and thus they cannot be studied or adapted as needed. Thus, many users, quite often in academia, prefer to create their own applications, most of them written in MATLAB, FORTRAN and C ++ programming languages. These programs are provided as desktop applications usually designed to only one type of study and without user-friendly interface. However, there are a few options with more attractive features such as the existence of a graphical user interface, and number of computational resources close to those found in commercial applications. There are also proposals in the literature for web applications whose main advantage is the remote and simultaneous access by any computer. Generally, the existing applications do not make available real-time collaboration features, and no interoperability with other applications. In this context, this thesis focused on the research of the implementation of a web application for analysis of electrical power systems, exploring these two aspects above mentioned. For that, some similar non-commercial programs and open-source were deeply investigated. In addition, the computational tools necessary to develop the application were selected and presented. The investigations and computational implementation performed here, as well as the results are properly presented and analyzed at the end of this work.

Keywords: power systems analysis software, web application, real-time collaboration.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX - *Asynchronous JavaScript and* XML

ANAREDE - Análise de Redes Elétricas

ANATEM - Análise de Transitórios Eletromecânicos

ATP - *Alternative Transients Program*

BCSSD - Biblioteca de Classes para Simulação de Sistemas Dinâmicos

CDF - *Common Data Format*

CEPEL - Centro de Pesquisas de Energia Elétrica

CGI - *Common Gateway Interface*

CIM - *Common Information Model*

CSS - *Cascading Style Sheet*

DBM - *Data Base Modularization*

EMTP - *Electromagnetic Transients Program*

EMTP-RV - *Electromagnetic Transients Program-Restructed Version*

EPRI - *Electrical Power Research Institute*

FACTS - *Flexible AC Transmission System*

FASEE - Ferramenta de Análise de Sistemas de Energia Elétrica

FORTRAN - *Formula Translator*

HTML - *HyperText Markup Language*

HVDC - *High-Voltage Direct Current*

IEEE - *Institute of Electrical and Electronic Engineers*

InterPSS - *Internet technology based Power System Simulator*

JSON - *JavaScript Object Notation*

LAMP - Linux+Apache+MySQL+PHP

MAPE - *Mean Absolute Percentage Error*

MATLAB - *MATrix LABoratory*

MVC - *Model-View-Controller*

NDSE - Núcleo de Dinâmica de Sistemas Elétricos

NWS - NDSE WEB *Simulator*

PHP - *Hypertext Preprocessor*

PSASP - *Power System Analysis Software Package*

PSAT - *Power System Analysis Toolbox*

REST - *REpresentational State Transfer*

RIA - *Rich Internet Applications*

SVG - *Scalable Vector Graphics*

TACS - *Transient Analysis of Control Systems*

TNA - *Transient Network Analyzer*

UML - *Unified Modeling Language*

VSC - *Voltage Source Converter*

WAMP - Windows+Apache+MySQL+PHP

XML - *eXtensible Markup Language*

PUBLICAÇÕES

TAMASHIRO, M. A.; GUIMARÃES, G. C.; RODRIGUES, A. R.; SILVA, A. M. B.; CAIXETA, D. A.; MONTEIRO, R. V. A.. *Features of Present Computer Tools for Load Flow Calculation and Experience Acquired with Improvement of UFUFlow Program*. Revista IEEE América Latina, v. 14, pp. 704-712, 2016.

TAMASHIRO, M. A.; GUIMARÃES, G. C.; RODRIGUES, A. R.; MONTEIRO, R. V. A.; OLIVEIRA, T. L.; SILVA, L. R. C.. *Comparative Study of TACS/DBM and MODELS of ATP-EMTP Applied to Power Systems Computer Simulation*. Revista IEEE América Latina, v. 14, pp. 1737-1744, 2016.

TAMASHIRO, M. A.; PERES, L. M.; PICCINI, A. R.; REZENDE, J. O.; GUIMARAES, G. C.. *Avaliação de Programas Gratuitos para Análise de Estabilidade Transitória em Sistemas Elétricos de Potência*. In: V Simpósio Brasileiro de Sistemas Elétricos - SBSE2014, 2014, Foz do Iguaçu. Anais do SBSE 2014, 2014.

TAMASHIRO, M. A.; PICCINI, A. R.; RODRIGUES, A. R.; GUIMARAES, G. C.. *Características de Simulação dos Programas PSAT e TransUFU para Análise de Estabilidade Transitória de Sistemas Elétricos de Potência*. In: XI Conferência de Estudos em Engenharia Elétrica (CEEL), 2013, Uberlândia. Anais da XI CEEL, 2013.

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

Os computadores tornaram-se as principais ferramentas para estudos científicos importantes nas mais diversas áreas do conhecimento. Na Engenharia Elétrica, eles podem ser utilizados tanto na análise estática como na análise dinâmica dos sistemas elétricos de potência. Até o surgimento dos computadores digitais, ocorrido durante a década de 50, o cálculo do fluxo de potência e a análise de estabilidade, por exemplo, eram realizados com o auxílio dos Analisadores de Redes (STAGG e EL-ABIAD, 1968). Esses dispositivos nada mais eram do que computadores analógicos especiais desenvolvidos especificamente para a análise de um sistema elétrico em escala reduzida (ELGERD, 1971).

As velocidades de processamento e a capacidade de armazenamento dos primeiros computadores digitais limitavam e dificultavam a realização de estudos em sistemas de grande porte, mas isso foi superado rapidamente (BOSE, 1998). Desde então muitos programas para a análise de sistemas elétricos têm sido desenvolvidos. Eles são escritos, tradicionalmente, em linguagens procedimentais como FORTRAN, PASCAL e C, mas C++ e JAVA, que são linguagens orientadas a objetos também têm sido utilizadas (SOMAN, KHAPARDE e PANDIT, 2002).

Segundo Zhou (1996) o potencial do paradigma de programação orientado a objetos para o desenvolvimento e aprimoramento dessas ferramentas computacionais é bem reconhecido. Esse paradigma permite o desenvolvimento de programas grandes, que podem ser combinados, adaptados ou ampliados mais facilmente, e possibilita ainda a reutilização de código já existente para a construção de novas aplicações (NEYER, WU e IMHOF, 1990).

Atualmente, é grande a quantidade de programas comerciais para a análise de sistemas elétricos, vários deles estão listados no *site* OpenElectrical (2016) e são conhecidos pela eficiência computacional. Apesar disso, segundo Milano (2005), eles podem ser inadequados para fins educacionais ou para a realização de pesquisas. Alguns dos motivos estão relacionados seguramente às restrições de uso e de distribuição impostas pelos seus desenvolvedores, e também ao não fornecimento do código fonte para estudo ou modificação.

Assim, tanto no âmbito acadêmico e científico, é comum o desenvolvimento de programas próprios, num esforço considerável para proporcionar a estudantes e pesquisadores ferramentas computacionais focadas nos mais diferentes aspectos da análise de sistemas elétricos (MILANO, VANFRETTI e MORATAYA, 2008). E com isso, é frequente a produção de trabalhos científicos relevantes sobre o assunto, de teses a artigos publicados em periódicos ou congressos importantes.

1.2 Motivações

Os trabalhos que seguem essa temática geralmente abordam a implementação de um ou mais recursos para a análise de sistemas elétricos considerando alguns dos seguintes aspectos:

a) Uso da programação orientada a objetos;
b) Uso da modelagem computacional orientada a objetos;
c) Uso de padrões de projeto para o desenvolvimento dos programas;
d) Integração de aplicações;
e) Construção de interfaces gráficas;
f) Implementações realizadas no MATLAB ou em diferentes linguagens de programação.

Muitos deles são tratados em conjunto e utilizados na construção de aplicações *desktop* ou aplicaçõs *web*, sendo que as dos primeiro grupo são a maioria. No caso das aplicações *web* observa-se a produção de interfaces gráficas pouco interativas e sem recursos de colaboração. E ainda não utilizam nenhuma estratégia de integração com outras aplicações. No entanto, com a tecnologia disponível atualmente é possível construir programas com todos esses recursos e ainda com aparência semelhante a de uma aplicação *desktop*. Há diversos

exemplos disso na *internet*, que vão desde serviços de *email*, editores de texto e de planilhas eletrônicas, e também simuladores de circuitos eletrônicos.

As vantagens das aplicações *web* em relação às *desktop* são muitas, entre as principais podem ser citadas estas:

a) Distribuição mundial e atualizações imediatas;
b) Não há necessidade de instalação;
c) Possibilitam o uso em diversos dispositivos ou sistemas operacionais;
d) Possibilitam o uso simultâneo por mais de um usuário;
e) Possibilitam a interoperabilidade com outras aplicações pela *internet*.

Para usufruir de todas essas vantagens de forma satisfatória é necessário o conhecimento e o uso correto de mais de uma tecnologia, mas muitas das informações necessárias não são suficientes ou mesmo não são tratadas nos trabalhos publicados até o momento, pelo menos entre aqueles consultados para o desenvolvimento desta tese.

1.3 Objetivos

1.3.1 Objetivo geral

O principal objetivo desta tese é contribuir com o aprimoramento de aplicações *web* para a análise de sistemas elétricos, por meio da investigação, apresentação e o uso de tecnologias para *web*, e que serão utilizadas na construção de uma aplicação de demonstração com interface interativa e recursos de colaboração em tempo real, e que permita ainda a integração com outras aplicações, mantendo um desempenho computacional satisfatório.

1.3.2 Objetivos específicos

Outros objetivos necessários estão aqui elencados:

a) Investigar o estado da arte sobre o assunto por meio da pesquisa e a seleção de publicações relevantes, de modo a identificar conhecimentos já estabelecidos e utilizá-los como base para o desenvolvimento da aplicação;

b) Investigar e analisar os principais programas disponíveis para utilizá-los como referência, com prioridade aos gratuitos e que também disponibilizem o seu código fonte, de modo a identificar, melhorar ou agregar funcionalidades;
c) Investigar e utilizar códigos fontes disponíveis para facilitar a implementação correta do fluxo de carga e a análise de estabilidade transitória para demonstração e testes da aplicação;
d) Investigar e apresentar as principais tecnologias para *web* necessárias para a construção da aplicação;
e) Realizar testes computacionais da aplicação para a verificação e produção de resultados confiáveis, e o correto funcionamento dos recursos implementados.

1.4 Organização da Tese

Esta tese foi organizada em sete capítulos descritos resumidamente a seguir.

Capítulo 1 – Introdução: É apresentada uma contextualização do assunto tratado, as motivações, os objetivos e a organização desta tese.

Capítulo 2 – Revisão da Literatura: É realizado neste capítulo uma revisão da literatura e uma análise das principais informações contidas nas publicações selecionadas, no intuito de fornecer suporte a esta tese.

Capítulo 3 – Programas para a Análise de Sistemas Elétricos: São apresentados os principais programas comerciais e sem fins comerciais disponíveis para a análise de sistemas elétricos. As principais características desses programas são expostas e analisadas, com ênfase nos programas sem fins comerciais e com o código fonte disponível para estudo e/ou modificação.

Capítulo 4 – Tecnologias Utilizadas na Aplicação: O capítulo apresenta informações sobre as diversas tecnologias utilizadas na implementação da aplicação, com o objetivo de facilitar a compreensão dos seus diversos aspectos construtivos.

Capítulo 5 – Apresentação da Aplicação: O capítulo apresenta a aplicação e as implementações realizadas. Detalhes de sua arquitetura e alguns trechos de códigos importantes são devidamente expostos e explicados.

Capítulo 6 – Testes Computacionais e Análise dos Resultados: O capítulo apresenta diversos testes propostos para a validação da aplicação implementada nesta tese. São realizadas comparações com os resultados obtidos em outros programas já consolidados no meio acadêmico e científico.

Capítulo 7 – Conclusões: O capítulo finaliza esta tese e estabelece diversas ponderações sobre os resultados alcançados à luz dos objetivos inicialmente elencados, ressaltando as principais contribuições. São sugeridos diversas linhas de trabalho para estudos futuros para que esta tese tenha continuidade e possa gerar outros similares.

Capítulo 2

REVISÃO DA LITERATURA

2.1 Considerações Iniciais

A quantidade de publicações sobre aplicações computacionais para a análise de sistemas elétricos é grande. Desse modo, para dar suporte a esta tese foi necessário selecionar os trabalhos mais relevantes vinculados a esse assunto.

A revisão da literatura realizada, a partir da década de 90, levou em consideração estas condições:

a) Divisão das publicações em dois grupos: Aplicações *desktop* e Aplicações *web*, para facilitar a exposição e a análise de seus conteúdos;
b) Divisão do primeiro grupo em sete sub-grupos, de modo a agregar trabalhos que abordem aspectos similares;

Cabe ressaltar que o processo de divulgação e acesso ao conhecimento constitui-se numa ação contínua e dinâmica, o que pode resultar na omissão involuntária de uma ou outra publicação científica também importante sobre o tema.

2.2 Aplicações *desktop*

2.2.1 Uso da programação orientada a objetos

Neyer, Wu e Imhof (1990) demonstram as vantagens da programação orientada a objetos aplicada à análise de sistemas elétricos. É dado destaque para a flexibilidade da

utilização desse paradigma de desenvolvimento, a reutilização de código já implementado, a redução do esforço de desenvolvimento e a manutenção de grandes programas. Os autores apresentam ainda os conceitos básicos desse paradigma de programação, além de aplicarem a modelagem orientada a objetos ao problema do fluxo de potência e realizarem a implementação computacional da versão polar do método numérico de Newton-Raphson para a solução das equações do problema. Por fim, eles relatam ainda que o programa desenvolvido na linguagem C++ foi somente 1,5 vezes mais lento do que uma implementação equivalente feita em FORTRAN.

Flinn e Dugan (1992) descrevem um banco de dados orientado a objetos capaz de suportar diversas aplicações, tais como análise harmônica, fluxo de potência, curto-circuito e estabilidade transitória. Os autores destacam a necessidade de um banco de dados único capaz de suportar essas e outras aplicações e relatam dificuldades na utilização do banco de dados do tipo relacional para modelar estruturas complexas como as encontradas em sistemas elétricos. No estudo, diversas estruturas de dados principais são apresentadas, bem como as técnicas de programação típicas para acesso ao banco de dados.

Hakavik e Holen (1994) analisam o modelamento dos sistemas elétricos e operações com matrizes esparsas utilizando orientação a objetos e a linguagem C++. Os autores propõem uma nova estrutura hierárquica de classes para a modelagem computacional dos elementos dos sistemas elétricos e avaliam a viabilidade da utilização desse paradigma de programação para a manipulações de matrizes esparsas. Além disso, eles descrevem como bibliotecas existentes de rotinas escritas em FORTRAN para manipulação desse tipo de matrizes, podem ser encapsuladas em um programa orientado a objetos, mostrando ainda que a programação orientada a objetos não causa necessariamente aumento do tempo de processamento.

Phillips et al. (1996) apresentam um *framework*[1] orientada a objetos, implementada na linguagem C++, destinada ao modelamento, à análise e ao controle de sistemas elétricos. Esse framework tem como núcleo um banco de dados orientado a objetos. Os autores desenvolvem uma aplicação para o cálculo do fluxo de potência com o objetivo de testar essa estrutura de programa, relatando um desempenho computacional menor dessa aplicação quando comparada a um programa escrito de forma tradicional. Mas, eles afirmam que a utilização do

1 É um conjunto de códigos devidamente escritos, organizados e associados para a solução de um problema específico, formando uma base sólida que auxilia e facilita o desenvolvimento de programas.

banco de dados orientado a objetos apresentou desempenho de 36% a 173% superior a um banco de dados relacional tradicional.

Zhou (1996) apresenta a proposta de um modelo orientado a objetos, implementado na linguagem C++, aplicado na simulação de sistemas elétricos. O autor realiza uma breve revisão dos conceitos e características da linguagem C++ e implementa um programa para cálculo do fluxo de potência em corrente contínua e outro em corrente alternada. Ele relata que esses programas são mais eficientes do que equivalentes implementados em outras linguagens de programação, como FORTRAN e PASCAL, apresentando também a aplicação dos conceitos desse paradigma de programação para a manipulação de matrizes esparsas e a solução de sistemas lineares de grande porte.

Zhu e Lubkeman. (1997) defendem a utilização da abordagem orientada a objetos para o desenvolvimento de aplicações destinadas à simulação de sistemas elétricos, sendo apresentado, em detalhes, um sistema para análise de sistemas de distribuição. Os autores afirmam que esse sistema, modelado utilizando como base os conceitos reais do sistema, é capaz de suportar uma grande variedade de aplicações.

Handschin et al. (1998) aplicam a programação orientada a objetos em um problema de planejamento do sistema de transmissão em um ambiente desregulamentado, modelando tanto os elementos físicos da rede elétrica quanto o seu mercado. O estudo propõe também que os métodos de análise do sistema, que por sua vez contêm os algoritmos numéricos propriamente ditos, sejam separados da representação dos elementos do sistema, modelados como objetos de aplicação.

Fuerte-Esquivel et al. (1998) utilizam a programação orientada a objetos para a implementação do cálculo do fluxo de potência considerando a modelagem de dispositivos FACTS[2]. Neste trabalho também é relatado um desempenho computacional compatível com implementações equivalentes em FORTRAN. As razões desses resultados são atribuídas a boas práticas de programação e a não utilização purista da filosofia desse paradigma de programação na resolução de sistemas lineares de grande porte. Os autores apresentam resultados de desempenho do código orientado a objetos implementado em linguagem C++, da ordem de 17% mais lento em relação à linguagem FORTRAN.

2 Conceito introduido em 1988 por Hingorani, pesquisador do EPRI, e associado à capacidade do controle direto do fluxo de potência no nível da transmissão de energia elétrica por meio do emprego de dispositivos da eletrônica de potência.

2.2.2 Uso da modelagem computacional orientada a objetos

Pandit et al. (2000) apresentam uma proposta de modelagem orientada a objetos para o sistema elétrico, realizando a divisão do modelo dos elementos físicos e suas aplicações. Na modelagem dos elementos físicos, eles relatam que esta é dependente da aplicação, ou seja, as características dos objetos que modelam elementos físicos de um sistema elétrico dependem em parte do tipo de aplicação que se deseja executar.

Manzoni (2005) desenvolve um modelo hierárquico de classes para a representação de sistemas elétricos, denominada de Ferramenta de Análise de Sistemas de Energia Elétrica (FASEE). Essa estrutura é aplicada especificamente para o problema da simulação dinâmica, onde são propostas uma classe Matriz e técnicas para a solução eficiente de sistemas lineares esparsos de grande porte, envolvendo matrizes com elementos complexos. São informados ainda que esse modelo apresenta bom desempenho computacional em função das práticas de programação utilizadas e que contemplam os requisitos de eficiência computacional para a resolução de sistemas lineares de grande porte, sendo uma alternativa adequada para o desenvolvimento de programas de grande porte para sistemas elétricos, mesmo quando os requisitos de desempenho computacional são críticos.

Marinho (2008) amplia os recursos da ferramenta FASEE para acomodar a simulação do comportamento dinâmico dos sistemas elétricos em condições desbalanceadas. Assim ele flexibiliza a construção de modelos trifásicos mais acurados e utilizados no cálculo do fluxo de potência e na análise da estabilidade de curta ou longa duração.

Sena (2013) apresenta a proposta de um *framework* desenvolvida na linguagem C++, e que utiliza uma metodologia flexível e orientada a objetos para o projeto e a implementação de programas destinados ao estudo dinâmico de sistemas elétricos de grande porte. Ele aplica diversos conceitos importantes do paradigma de programação orientada a objetos, além de fazer um grande uso de conceitos e ferramentas de Engenharia de *Software* para descrever em detalhes o *framework* denominado de Biblioteca de Classes para Simulação de Sistemas Dinâmicos (BCSSD).

2.2.3 Uso de padrões de projeto

Zhu e Jossman (1999) apresentam os conceitos de padrões de projeto (*design patterns*) na modelagem de sistemas elétricos. Os autores enfatizam a dificuldade de se chegar a uma estrutura ótima para a modelagem do sistema, a qual atenderia uma ampla gama de

aplicações, mostrando ainda alguns conceitos de padrões de projeto, assim como suas aplicações na modelagem orientada a objetos de sistemas elétricos.

Agostini (2002) desenvolve um *framework* orientada a objetos para aplicações gerais em sistemas elétricos utilizando uma estrutura hierárquica baseada nos conceitos de elementos estruturais e elementos de composição. Os elementos estruturais representam os dispositivos físicos do sistema e os elementos de composição representam o conjunto de equipamentos que compõem um elemento físico (para uma unidade de geração, por exemplo, os elementos de composição compreendem a máquina síncrona e seu conjunto de controladores). São aplicados diversos conceitos importantes do paradigma de programação orientada a objetos, além de fazer um grande uso de metodologias e ferramentas de Engenharia de *Software*, e de padrões de projeto.

DiPaolo et al. (2010), Bibin et al. (2012), e Weijiang, Weimei e Yongfeng (2012) apresentam diferentes padrões de projetos aplicados no desenvolvimento de programas para a análise de sistemas elétricos. Os autores da primeira referência apresentam novos aspectos do *framework* BCSSD relativos ao uso dos padrões *Factory* e *Builder*. O *Factory* também está presente na segunda referência junto com os padrões *Adapter*, *Strategy* e o *Singleton*. E na terceira publicação são vistos os padrões *Model-View-Controller* (MVC), *Composite*, *Bridge*, *Decorator*, *Command* e *Abstract Factory* utilizados no desenvolvimento do programa denominado de PSASP.

2.2.4 Integração de aplicações

Becker et al. (2000) discutem o problema da integração entre as aplicações. Para resolver isso apresentam o *Common Information Model* (CIM), um conjunto de classes para a representação de sistemas elétricos, resultado do trabalho desenvolvido por uma força tarefa criada no *Electrical Power Research Institute* (EPRI). O CIM utiliza a linguagem *Unified Modeling Language* (UML) para a sua documentação.

Milano, Zhou e Hou (2009) propõem um padrão aberto, baseado em XML, para o armazenamento e o intercâmbio dos dados de simulação de sistemas elétricos. Eles apontam os motivos e as vantagens em relação as limitações do IEEE e aos principais formatos de dados proprietários existentes, além de apresentarem e discutirem em detalhes as características e a estrutura desse padrão proposta.

2.2.5 Construção de interfaces gráficas

Foley et al. (1993) discutem a implementação de interfaces gráficas com programação orientada a objetos para aplicação na análise de sistemas elétricos. Os autores demonstram uma implementação na linguagem C capaz de funcionar em plataformas de *hardware* e sistemas operacionais diferentes, ou seja, realizaram a implementação de uma interface gráfica independente de plataforma. Eles apresentam também uma estrutura hierárquica de classes direcionada para o desenvolvimento de interfaces gráficas.

Ong e Gooi (1999) discutem um protótipo de programa que implementa um ambiente interativo para ensino à distância e inclui um simulador *web* para o cálculo do fluxo de potência e uma interface gráfica, independente e flexível, feita na linguagem JAVA. Os autores utilizam um sistema com seis barras, dois geradores em condições leves, médios e de pico de carga para ilustrar os princípios da operação do sistema de potência e controle. Além disso, eles mencionam algumas limitações existentes no pacote e as melhorias futuras a serem implementadas.

Kober, Manzoni e Lemos (2003) apresentam uma descrição do projeto e implementação de uma interface gráfica e sua integração a um *framework* para sistemas elétricos. As duas aplicações foram desenvolvidas na linguagem C++.

Dzafic (2007) apresenta um *framework* orientado a objetos para a análise de sistemas elétricos com enfoque para a sua interface gráfica. É utilizado a linguagem C++, alguns padrões de projeto e diagramas da UML.

2.2.6 Programação na linguagem MATLAB ou outras

Schaffner (2002) descreve um programa interativo para visualização do fluxo de potência com fins educacionais e implementado no MATLAB. O programa desenvolvido possui exemplos pré-definidos que podem ser executados em qualquer navegador da Internet sem a necessidade de nenhuma instalação especial.

Milano (2005) descreve uma ferramenta de código aberto para o MATLAB e OCTAVE, e voltado para a análise de sistemas elétricos de pequeno e médio porte, denominada PSAT. A ferramenta inclui cálculo de fluxo de potência, fluxo de potência continuado, fluxo de potência ótimo, análise de estabilidade a pequenas perturbações, e simulação no domínio do tempo. Tem disponível vários modelos estáticos e dinâmicos, incluindo cargas não convencionais, máquinas síncronas e assíncronas, reguladores e dispositivos FACTS. Possui interface gráfica com editor de diagramas unifilares baseado no

Simulink As características básicas, algoritmos e uma variedade de estudos de caso são apresentados neste artigo para ilustrar as capacidades da ferramenta apresentada e sua adequação para fins educacionais e de pesquisa.

Zhou e Zhou (2007) apresentam uma visão geral de um programa de código aberto, denominado InterPSS, para a simulação de sistemas elétricos e desenvolvido na linguagem JAVA. Eles discutem sua arquitetura construtiva e processo de desenvolvimento, que permite ao usuário avançado criar módulos adicionais para aumentar suas funcionalidades e/ou alterar o seu comportamento.

Zimmerman, Murillo-Sánchez e Thomas (2011) apresentam uma toolbox para o MATLAB, de código aberto, denominado MATPOWER, e voltado para o cálculo do fluxo de potência em sistemas de grande porte. A ferramenta possui arquitetura extensível para o cálculo do fluxo de potência ótimo.

Cole e Belmans (2011) apresentam outra toolbox para o MATLAB, de código aberto denominada de MATDYN, e voltada para a análise de estabilidade transitória e simulação no domínio do tempo de sistemas elétricos. Os autores informam que essa ferramenta permite a definição de modelos definidos pelo usuário e métodos de integração personalizados.

Milano (2013) apresenta uma nova ferramenta para análise de sistemas elétricos, denominada DOME, inteiramente baseado em linguagem de script PYTHON, bem como em bibliotecas de domínio público escritas na linguagem C e FORTRAN. O artigo discute as características que tornam a linguagem PYTHON um instrumento adequado para pesquisa, simulações numéricas grandes e educação. Em seguida, o trabalho descreve a arquitetura do programa desenvolvido e fornece uma variedade de exemplos para mostrar as características avançadas e o desempenho da ferramenta desenvolvida.

2.3 Aplicações *web*

Leou e Gaing (2002) descrevem uma aplicação *web* para análise de fluxo de potência. A aplicação utiliza de forma integrada as linguagens HTML, ASP e JAVA. Eles comentam de forma positiva o potencial e as vantagens da abordagem *web* como plataforma para o desenvolvimento de simulações de sistemas de potência.

Chen e Lu (2002) apresentam um pacote de programas para a simulação de sistemas elétricos totalmente baseado na *web*. É utilizado um ambiente em *cluster*, a fim de manter e

compartilhar diversas rotinas de simulação existentes já implementadas em plataformas diferentes. Os autores apoiam-se fortemente em tecnologias avançadas de computação distribuída e também na utilização do conceito de MVC, além de comentarem de forma positiva o potencial e as vantagens da abordagem web como plataforma para o desenvolvimento e a implantação de simulações de sistemas elétricos complexos.

Zimmerman e Thomas (2004) descrevem uma aplicação *web* para a simulação do mercado de energia elétrica denominada PowerWEB. Eles informam que pode ser utilizada como uma ferramenta de pesquisa econômica experimental para avaliar a economia e a confiabilidade dos impactos em um dado modelo de mercado, bem como treinar e educar os alunos, profissionais da indústria, além de nortear decisões políticas.

Lee e Chu (2005) apresentam um modelo de fluxo de potência de um conversor do tipo fonte de tensão (VSC) para um sistema de transmissão de Corrente Contínua em Alta Tensão (HVDC), considerando em detalhes, diferentes modos de controle de compensação reativa nos seus terminais. O modelo proposto é implementado na linguagem de programação C no padrão *Common Gateway Interface* (CGI). A interface com o usuário, desenvolvida em JAVA, é realizada por meio de uma página web onde os usuários entram com os dados do sistema elétrico, configurações, objetivos de controle, parâmetros e modos de compensação reativa de HVDC. O cálculo do fluxo de potência é realizado no servidor e os resultados são retornados para o usuário também na forma de uma página *web*.

Abdel-Aker, El-Nemr e Abdel-Galil (2010) apresentam uma aplicação *web* projetada, segundo os autores, para fornecer uma topologia flexível, confiável e extensível para simulação de sistemas elétricos. A aplicação mantém características das aplicações *desktop* e faz uso da tecnologia ActiveX e classes Microsoft Foundation (MFC), ambas da Microsoft. A interface gráfica é implementada usando HTML e Java Script associado ao controle ActiveX.

Malarvizhi e Veena (2011) apresentam uma aplicação *web* para estudos de fluxo de potência. O ambiente proposto faz uso de Matlab Builder JA para a implementação de módulos MATLAB como componentes JAVA.

Agamah e Ekonomou (2015) descrevem uma biblioteca na linguagem PHP para o desenvolvimento de aplicações web para a análise de sistemas elétricos. Eles apontam as vantagens das linguagens de *script* como PHP para a implementação de programas dessa natureza e apresentam a arquitetura tradicional de três camadas utilizadas em soluções similares, bem como as razões para essa escolha. Nesse tipo de arquitetura as linguagens de script no lado do servidor somente intermediam a comunicação de clientes com o servidor da aplicação responsável pelos cálculos matemáticos implementados em linguagens tradicionais

como C, C++, C#, JAVA, etc. Mas, os autores mencionam a evolução e as vantagens atuais das linguagens de script, dando destaque ao PHP, para a construção dessas soluções numa arquitetura de apenas duas camadas.

2.4 Considerações Finais

Em geral, os trabalhos tratam de aplicações específicas para a análise computacional de sistemas elétricos, principalmente o cálculo do fluxo de potência, e que são implementadas na sua grande maioria em C++ e MATLAB, nessa ordem. Mas outras linguagens como JAVA, PYTHON, ASP e PHP também têm sido utilizadas.

É possível observar que os autores, em sua maioria, utilizam tecnologias proprietárias e/ou ambientes de programação comerciais, como o caso da linguagem ASP de propriedade da Microsoft, JAVA da Oracle, e o MATLAB da MathWorks. Além disso, muitos deles não disponibilizam o código fonte, ou o arquivo de instalação, ou mesmo o arquivo executável dos programas desenvolvidos. Tudo isso leva as seguintes situações e dificuldades:

a) Restrições de uso e de distribuição;
b) Necessidade da instalação de outros programas ou complementos (*plugins*) para o correto funcionamento da aplicação;
c) Necessidade do uso de uma infraestrutura mais elaborada, como aquela necessária para o funcionamento de aplicações *web* na linguagem JAVA;
d) Impossibilidade ou dificuldades para a realização de testes por terceiros;
e) Impedimento ou dificuldades para o recebimento de contribuições, seja para a realização de correções ou a adição de melhorias.

A análise das publicações do início da década de 90 aponta como principais objetivos:

a) Demonstrar a viabilidade do paradigma de programação orientado a objetos aplicado na simulação e análise de sistemas elétricos;
b) Realizar comparações entre programas escritos em linguagens orientadas a objetos e linguagens procedimentais, avaliando basicamente dois aspectos, tempo de processamento e consumo de memória.

E a partir do final da década de 90 os trabalhos começaram a utilizar recursos mais avançados do paradigma de programação orientada a objetos. Alguns poucos passaram a dar mais ênfase na modelagem orientada a objetos e na utilização de padrões de projeto com a

finalidade de produzir códigos computacionais mais eficientes e flexíveis. Nesses trabalhos, que não constituem uma maioria, percebe-se um maior uso de metodologias e de ferramentas da Engenharia de *Software*.

A integração de aplicações é abordada pelo menos em dois trabalhos, um propondo um conjunto de classes padronizadas para a representação dos diversos elemento do sistema elétrico (BECKER et al., 2000) e outro sobre a padronização de dados (MILANO, ZHOU e HOU, 2009). Isso leva à discussão de um assunto maior que é a interoperabilidade desses programas. Esse termo diz respeito a existência de uma comunicação transparente de uma aplicação com outras, ou mesmo com outros sistemas[3].

O desenvolvimento dessas aplicações no ambiente *web* ao invés do *desktop* tem encontrado condições favoráveis para isso. Como sustentado em (AGAMAH e EKONOMOU, 2015), as linguagens para *web*, que na maioria são linguagens de script podem muito bem ser utilizadas no desenvolvimento de programas para a análise de sistemas elétricos.

Essas linguagens eram utilizadas mais para o desenvolvimento de páginas *web* e consideradas inadequadas para outros fins. Seguindo essa corrente, os trabalhos sobre aplicações *web* utilizam essas linguagens de forma secundária. No geral, como mencionado em (AGAMAH e EKONOMOU, 2015), o cálculo matemático necessário é realizado no servidor por programas implementados em linguagens tradicionais, não específicas para *web*. E a interface com o usuário muitas vezes necessita de pequenos programas complementares para torná-la mais interativa, ou simplesmente não dispõe de objetos gráficos manipuláveis, semelhantes àqueles disponíveis na maioria das aplicações *desktop* comerciais.

No próximo capítulo serão apresentados e analisados os principais programas disponíveis, tanto comerciais como aquelas sem fins comerciais.

3 Sistemas aqui diz respeito, basicamente, a qualquer dispositivo eletrônico, *smartphone* ou *tablet* por exemplo, com sistema operacional diferente daquele onde a aplicação em questão foi gerada.

Capítulo 3

PROGRAMAS PARA A ANÁLISE DE SISTEMAS ELÉTRICOS

3.1 Considerações Iniciais

Este capítulo apresenta alguns programas para a análise de sistemas elétricos divididos em dois grupos, comerciais e sem fins comerciais. Entre os comerciais estão os seguintes programas: ANAREDE, CYME, DigSilent PowerFactory, EMTP-RV, PowerWorld, PSS/E e o SimPowerSystems. E entre os gratuitos estão: ATP, InterPSS, PSAT, TransUFU, UFUFlow, MATPOWER e o MatDyn.

A maioria desses programas tem um ou mais recursos de referência e por isso possuem uma grande quantidade de usuários, sendo facilmente encontrados por ferramentas de busca na internet. Outros têm sido amplamente utilizados no âmbito acadêmico, para o ensino e/ou a realização de pesquisas. Muitos encontram-se numa lista disponível no site OpenElectrical (2016) ou mencionados no levantamento bibliográfico desta tese.

Entre as opções sem fins comerciais estão programas desenvolvidos por uma comunidade fechada e com alguma limitação ou condição de uso, mas há aqueles também sem nenhuma restrição e que ainda disponibilizam o código fonte. Convém mencionar também a existência de versões gratuitas de alguns programas comerciais, mas com certas restrições, principalmente limitações de recursos.

Os programas comerciais são apresentados neste capítulo para conhecimento e também para possibilitar a realização de comparações com aqueles sem fins comerciais.

Os programas sem fins comerciais, principalmente os de código aberto, foram utilizados como referência para o desenvolvimento da aplicação desta tese, de modo a

aproveitar conhecimentos comuns já estabelecidos e acessíveis. Assim, eles foram devidamente estudados e analisados, com atenção especial aos detalhes de funcionamento, como o tratamento, organização e processamento dos dados, e os métodos numéricos, algoritmos e lógica de programação utilizados.

3.2 Programas Comerciais

3.2.1 ANAREDE

É um programa desenvolvido pelo CEPEL (Centro e Pesquisas de Energia Elétrica) que contém um conjunto de aplicações integradas destinadas à realização de estudos nas áreas de operação e planejamento de sistemas elétricos de potência. Inclui fluxo de potência, análise de contingências, análise de sensibilidade de tensão e fluxo, redespacho de potência ativa e fluxo de potência continuado (ANAREDE, 2016).

O programa dispõe de interface gráfica amigável, modelo de curva de carga, modelo de bancos de capacitores/reatores chaveados para controle de tensão, modelos de equipamentos equivalentes e individualizados, algoritmo para verificação de conflito de controles e facilidades para estudos de recomposição do sistema elétrico.

A interface gráfica possibilita a representação visual de todos os elementos da rede elétrica. Permite ainda a identificação por cores dos níveis de tensão da rede elétrica representada de forma gráfica. E ainda, por meio de filtros de visualização, permite a identificação de violações de limites de tensão, geração de potência reativa e carregamento de circuitos de corrente alternada.

Para o cálculo do fluxo de potência, utiliza os métodos de Newton-Raphson e o Desacoplado rápido. A análise do desempenho dos algoritmos e as adaptações do programa às características particulares dos sistemas brasileiros foram realizadas através de projetos conjuntos com diversas empresas brasileiras do setor de energia elétrica, tanto públicas como privadas. Diversos arquivos de dados do sistema elétrico brasileiro encontram-se disponíveis para *download* em *sites* dessas empresas.

A versão atual do programa pode ser executada em qualquer microcomputador com sistema operacional Windows 2000 ou superior. O CEPEL, após a avaliação do pedido,

fornece gratuitamente versões acadêmicas exclusivamente para fins educacionais. A versão acadêmica possui limite de utilização para apenas 120 barras.

3.2.2 CYME

Comercializado desde 1986 é um programa modular destinado à análise de sistemas de transmissão e elétricos industriais (CYME, 2016). O seu módulo principal é o CYMFLOW, utilizado para o cálculo do fluxo de potência resolvido pelos métodos numéricos de Newton-Raphson, Desacoplado Rápido ou Gauss-Seidel. Possui também o módulo denominado CYME que realiza a análise trifásica do sistema elétrico utilizando técnicas de esparsidade. Fornece ainda o CYMDIST com funções necessárias para o planejamento, operação e análise de redes de distribuição.

Possui modelos detalhados de máquinas de indução e síncronas, cogeração, dispositivos FACTS e linhas HVDC com retificadores e inversores. Tem como requisitos de *hardware* um computador Pentium IV com 1MB de memória RAM e 1GB de espaço disponível no disco rígido. Roda em ambiente Microsoft Windows XP, Vista, 7, Microsoft Server 2003 ou 2008.

3.2.3 DigSilent PowerFactory

Este programa tem como foco sistemas de transmissão, de distribuição e elétricos industriais. Estudos de sistemas elétricos de grande porte, interligados e com a presença de geração distribuída, como solar fotovoltaica e eólica, linhas HVDC e dispositivos FACTS podem ser realizados (DigSilent, 2016).

Possui recursos para análise de fluxo de potência, tanto monofásica, bifásica ou trifásica em conjunto ou separado, de curto-circuito, de harmônicos, de estabilidade e de transitórios eletromagnéticos no domínio do tempo, de autovalor, de contingência, de confiabilidade, de adequação da geração, de proteção, cálculo de arco elétrico, simulação de partida estática de motor, fluxo de potência ótimo, otimização de redes de distribuição e estimativa de estados.

Dispõe de interface gráfica para a criação, edição e visualização do sistema em estudo. Para o fluxo de potência utiliza como método principal o de Newton-Raphson. Possui como requisitos processador de 2GHz ou superior, 1GB de espaço no disco rígido, de 0,5 a 8GB de RAM, sistema operacional Windows, 2000, XP, Vista ou 7 (de 32 ou 64 bits).

O desenvolvedor fornece uma licença especial individual e gratuita, denominada PowerFAcetory4Thesis (PF4T), para ser utilizada em trabalhos de TCC, Mestrado ou Doutorado. As funcionalidades dessa versão, bem como o limite de barras e duração da licença são definidas nos processos individuais de aprovação necessária para o seu fornecimento.

3.2.4 EMTP-RV

É formado por um conjunto de aplicações para a simulação de transitórios eletromagnéticos, eletromecânicos e sistemas de controle em sistemas elétricos de potência polifásicos. Foi desenvolvido em 1998 como resultado final do projeto de reestruturação e modernização do programa EMTP96 (EMTP-RV, 2016).

Seus recursos típicos são voltados para o planejamento de sistemas elétricos, fluxo de potência trifásico, linhas HVDC, geração eólica, coordenação de isolação, estudos de descargas atmosféricas, chaveamento de surtos, ferroressonância, partida de motores, modelamento de linhas e cabos elétricos, dispositivos FACTS e projeto de sistemas de controle.

O programa inclui a interface gráfica denominada EMTPWorks e o ScopeView utilizado para a visualização dos resultados gerados. Possui uma grande biblioteca de modelos elétricos e eletrônicos, dispositivos usados em redes elétricas de distribuição, e permite ainda ao usuário criar seus próprios dispositivos. Tem como requisitos um processador Pentium III de 800MHz ou melhor, 500MB disponível no disco rígido, mínimo de 256MB, sistema operacional Windows XP, Vista, 7 ou 8. Possui versão acadêmica com preço diferenciado, mas não gratuita.

3.2.5 PowerWorld

Este programa permite realizar a análise de despacho econômico, análise econômica da transferência entre áreas, cálculo do fator de distribuição de potência, estudo das curvas PV e QV, análise de faltas, análise de curto circuito e de contingência.

Realiza de forma eficiente o cálculo do fluxo de potência para sistemas de até 100.000 barras, e tem como métodos principais os de Newton-Raphson e o Desacoplado rápido (PowerWorld, 2016).

O programa permite ao usuário criar e visualizar graficamente o sistema elétrico, fornece diagramas unifilares animados e interativos. Além disso, possui ferramentas para a

utilização de Sistema de Informações Geográficas (GIS), e recurso para a visualização de alguns parâmetros elétricos, como níveis de tensão e potência, numa escala de cores.

Disponibiliza também recursos adicionais para a determinação da máxima capacidade de transferência de potência possível entre áreas, recurso para utilização de computação distribuída, avaliação do risco decorrente da corrente de indução geomagnética, solução do fluxo de potência ótimo, análise de estabilidade transitória e de tensão, interação com programas externos como Microsoft Excel e ambientes integrados de programação como o Borland Delphi, Microsoft Visual C++ e Microsoft Visual Basic.

Roda em sistema operacional Microsoft Windows, 2003/XP e nas versões mais recentes de 32 ou 64 bits. O desenvolvedor disponibiliza uma versão educacional limitado ao uso de até 12 barras.

3.2.6 PSS/E

Foi adquirido pela SIEMENS em 2005, sendo que sua primeira versão data de 1976. É considerado o primeiro programa da área com interface gráfica. Permite a realização de uma série de estudos como o cálculo do fluxo de potência, simulação dinâmica de pequenas e grandes perturbações, fluxo de potência ótimo, análise de contingência, análise harmônica, curvas PV e QV, faltas balanceadas e desbalanceadas, dentre outros (PSS/E, 2016).

Contém modelos de um grande conjunto de elementos do sistema elétrico como geradores síncronos, assíncronos, estabilizadores, compensadores, linhas HVDC, dispositivos FACTS, reguladores de velocidade, sistemas de proteção, cargas, modelos de excitação e unidades eólicas genéricas.

Permite ainda que o usuário crie seus próprios modelos de controle por meio de rotinas elaboradas em linguagens, tais como FORTRAN e, PYTHON, ou pela construção na forma de diagramas de blocos.

Possui uma versão acadêmica gratuita limitada a 50 barras. Roda em ambiente Microsoft Windows XP, Vista ou 7.

3.2.7 SimPowerSystems

É um programa oferecido na forma de uma *toolbox* para o MATLAB. Fornece bibliotecas de componentes e recursos para análise, modelagem e simulação de sistemas elétricos de potência (SimPowerSystems, 2016). A interface gráfica segue os padrões da ferramenta SIMULINK do próprio MATLAB.

As bibliotecas oferecem modelos de componentes da rede elétrica, incluindo máquinas trifásicas, motores elétricos e componentes para aplicações tais como dispositivos FACTS e sistemas de energia renovável. Entre os principais tipos de análise que podem ser realizadas pelo programa estão a análise harmônica, cálculo da distorção harmônica total (THD), fluxo de potência e estabilidade transitória.

Modelos mecânicos, hidráulicos, pneumáticos e da *toolbox* Simscape podem ser utilizados em conjunto com os modelos elétricos. O SimPowerSystems suporta ainda a geração de código na linguagem C.

3.3 Programas sem Fins Comerciais

3.3.1 ATP

O ATP, também conhecido como ATP-EMTP, é escrito na linguagem de programação FORTRAN e foi concebido em 1984 por iniciativa dos Doutores W. Scott Meyer e Tsu- Huei Liu, e desde então tem sido desenvolvido por meio das contribuições de grupos de usuários oficiais localizados pelo mundo (ATP-EMTP, 2016). Esse programa consiste de uma versão *desktop* adaptada de um dos primeiros programas para simulação computacional de transitórios eletromagnéticos, o EMTP, desenvolvido por Herman W. Dommel na década de 60.

Diferentemente do EMTP, o ATP não é de domínio público, entretanto, o seu licenciamento, por meio de seus grupos de usuários, é disponibilizado gratuitamente para qualquer pessoa no mundo que não tenha participado voluntariamente na venda ou tentativa de comercialização de qualquer programa de transitórios eletromagnéticos.

Tem como foco a simulação de fenômenos transitórios eletromagnéticos em sistemas elétricos, permitindo a simulação de redes complexas, além de sistemas de controle com estruturas arbitrárias. Possibilita a realização de estudos de transitórios eletromagnéticos, estabilidade transitória, sobretensão, descargas atmosféricas, modelagem de máquinas elétricas, partida de motores, ocorrência de harmônicos, curto-circuito etc. Possui capacidade de modelagem ampla, permitindo a representação de não-linearidades, elementos com parâmetros concentrados, elementos com parâmetros distribuídos, chaves, transformadores, reatores, etc.

O programa utiliza o Método Trapezoidal Implícito de integração numérica para resolver as equações diferenciais dos componentes do sistema no domínio do tempo. Utiliza técnicas de esparsidade e de fatoração triangular otimizada de matrizes, e tem disponível ainda recursos adicionais como a função DBM, as rotinas: TACS e a MODELS, uma linguagem de simulação.

A DBM permite a criação de bibliotecas de modelos, expandindo e melhorando o uso do programa em várias aplicações. Um ou mais componentes do programa podem ser agrupados num único módulo, o qual passa a ser visto pelo ATP como se fosse um modelo único.

A TACS é um recurso do ATP utilizado para a análise de sistemas de controle no domínio do tempo. Foi originalmente desenvolvido para a simulação do controle de conversores em linhas HVDC. Permite realizar o controle do sistema elétrico através do comando de operação de chaves, ou estabelecendo valores de fontes de tensão ou de corrente, e ainda pela criação de elementos não lineares.

Já a MODELS, desenvolvida por Laurent Dubé, também autor da TACS, é uma linguagem descritiva de uso geral utilizada na representação e estudo de sistemas variantes no tempo.

A MODELS, bem como a TACS, permite que o usuário do ATP especifique e modifique os valores de parâmetros numéricos e/ou analógicos, para controlar a operação de componentes elétricos do sistema simulado, além da criação de modelos elétricos. Entretanto, a MODELS apresenta uma série de vantagens em relação à TACS, pois simplifica o trabalho de representação de sistemas complexos através da decomposição em módulos independentes e do uso de algoritmos de forma mais generalizada.

A entrada de dados e a configuração dos parâmetros da simulação podem ser feitas por meio da utilização de uma ferramenta gráfica denominada de ATPDraw. Já para a plotagem de gráficos existem outros programas acessórios, sendo comumente utilizado o PlotXY.

3.3.2 MatDyn

É um programa para ser utilizado no MATLAB voltado para a análise dinâmica de sistemas elétricos. Os valores das tensões e potências do sistema em regime permanente são obtidos de outro programa, o MATPOWER, e por isso possui um padrão para a entrada de dados semelhante a este (MatDyn, 2016).

Possibilita realizar simulações com faltas trifásicas, desligamento e religamento de linhas, aumento ou rejeição de cargas do sistema. Implementa modelos para o gerador,

sistema de excitação e reguladores. No caso do primeiro utiliza o modelo clássico e o de 4º ordem. Para os demais são utilizados modelos básicos do IEEE. Apesar de disponibilizar poucos modelos permite a inclusão de novos por meio da escrita das suas equações diferenciais e algébricas nos seus arquivos devidamente separados para cada tipo.

Dispõe de três métodos numéricos bem conhecidos para a solução das equações diferenciais do sistema: Método de Euler modificado conhecido também como Método Trapezoidal Implícito, Runge-Kuntta e Runge-Kuntta Fehlberg.

3.3.3 MATPOWER

É desenvolvido para o MATLAB e OCTAVE, este similar ao primeiro, mas gratuito. É utilizado para a resolução do fluxo de potência e fluxo de potência ótimo (MATPOWER, 2016). Possui implementado as formulações para o cálculo do fluxo de potência considerando modelos dos elementos do sistema elétrico tanto em corrente alternada como em corrente contínua.

A entrada de dados é por meio da escrita de um arquivo de texto contendo valores dispostos na forma de matrizes do MATLAB. As colunas dessas matrizes são similares as do formato IEEE CDF[4].

Para o cálculo do fluxo de potência, considerando a formulação em corrente alternada, utiliza como padrão o método de Newton-Raphson, mas tem disponível também o método de Gauss-Seidel e o Desacoplado rápido.

3.3.4 InterPSS

Este programa de código fonte aberto é escrito na linguagem JAVA e utiliza tecnologias da internet. Possui como foco o projeto, a análise e a simulação de sistemas elétricos. Sua arquitetura aberta e modular, ainda em desenvolvimento, permitirá o uso e a conexão de componentes desenvolvidos por terceiros para aumentar as suas funcionalidades (InterPSS, 2016). Realiza atualmente os seguintes tipos de estudos: fluxo de potência, análise de curto-circuito, de estabilidade transitória, da capacidade de transferência e sensibilidade, e da transferência de potência entre áreas. Os principais métodos disponíveis para o cálculo do fluxo de potência são o de Newton-Raphson e o Desacoplado rápido.

[4] É um formato comum de dados para o fluxo de potência definido por um grupo de trabalho do IEEE no intuito de padronizar essas informações e permitir o uso em programas diferentes

O programa possui duas versões, uma para instalação em computadores pessoais, a *Desktop edition*, e outra para utilização pela internet, *Cloud editon*. A versão *desktop* possui um editor gráfico para a criação do diagrama unifilar do sistema elétrico. Fornece ainda diversos exemplos prontos. A outra versão faz uso de uma planilha eletrônica elaborada no Google *Spreadsheet* e hospedada no Google *Drive*. Para utilização desta versão é necessário possuir uma conta de usuário no Google.

Na versão *Cloud editon* toda a simulação é realizada por um "motor" (*engine*) de simulação hospedado na internet. Esse "motor" de simulação escrito em Java é o mesmo utilizado pela versão *desktop*, mas de uso local.

O arquivo padrão da versão *Cloud editon* é baseado em XML, sendo ainda capaz de ler o formato IEEE CDF, e também os arquivos de dados dos programas PSS/E e PowerWorld.

3.3.5 PSAT

É outro programa para MATLAB que pode ser utilizado na análise e simulação de sistemas elétricos de pequeno a médio porte, e para isso disponibiliza recursos para o cálculo do fluxo de potência, fluxo de potência continuado, fluxo de potência ótimo e análise de estabilidade de pequenos sinais (PSAT, 2016). Possui uma versão sem interface gráfica para o programa OCTAVE.

A interface gráfica e suas bibliotecas na versão para o MATLAB são baseadas no SIMULINK. Suas principais características são: análise no domínio do tempo e da frequência, diversos modelos, conversão de arquivos de dados a partir de vários formatos, como TXT, Excel e arquivos LaTeX. A primeira versão deste programa foi lançada em 2002, mas só foi apresentado à comunidade científica por meio de um artigo publicado em 2005.

O PSAT permite realizar simulações com a inclusão de cargas não convencionais, máquinas síncronas e assíncronas, reguladores e dispositivos FACTS. O seu núcleo de funcionamento é um algoritmo de fluxo de potência, que também é responsável pela inicialização das variáveis de estado. Assim que o fluxo de potência é calculado, o usuário pode realizar as simulações para análise dinâmica e/ou estática do sistema modelado. Disponibiliza ainda dois métodos numéricos implícitos, o de Euler regressivo e o trapezoidal.

Nas versões mais recentes do PSAT, estão presentes o modelamento de turbinas eólicas, a possibilidade de conversão dos arquivos de dados em outros formatos de saída, como arquivo de texto, planilha eletrônica, e LaTeX.

3.3.6 TransUFU

O programa TransUFU, desenvolvido em FORTRAN, foi criado com a finalidade didática e de apoio a estudos quanto à resposta dinâmica de um sistema elétrico de potência face a distúrbios decorrentes de chaveamentos e faltas.

Desde a sua criação vem sendo utilizado no âmbito da FEELT/UFU (Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia) em disciplinas dos cursos de graduação e pós-graduação, e também para o desenvolvimento de trabalhos e projetos de pesquisa.

A versão atual limita-se a sistemas de pequeno e médio porte. Implementações futuras, em desenvolvimento, deverão possibilitar o trabalho com sistemas maiores.

O programa permite a realização de simulações limitadas as seguintes quantidades de componentes (GUIMARÃES, 2000a):

a) 20 máquinas síncronas (geradores);
b) 30 motores de indução;
c) 20 compensadores estáticos;
d) 50 barras ou nós do sistema elétrico;
e) 100 ramos;
f) 20 linhas chaveadas (incluindo as usadas para rejeição).

Este programa, assim como a maioria dos programas apresentados neste capítulo, utiliza o Método Trapezoidal Implícito para a realização de integrações numéricas. Devido à estabilidade desse método numérico, a precisão dos resultados não fica muito sensível à escolha do valor do passo de integração. Esta vantagem geralmente não acontece com outros métodos de integração como o de Runge Kutta (DOMMEL, 1972).

Disponibiliza 4 modelos diferentes para a máquina síncrona, sendo que para cada máquina do sistema o programa escolhe, automaticamente, o modelo mais conveniente, de acordo com o conjunto de dados fornecidos pelo usuário. Com o modelo mais detalhado é possível envolver o efeito de saliência transitória e subtransitória, saturação e reguladores de tensão e velocidade. Pode-se ainda usar diferentes representações de reguladores para diversas máquinas, por exemplo, reguladores de velocidade para geradores térmicos e hidráulicos.

Os motores de indução são representados por um detalhado modelo dinâmico, incluindo o efeito transitório da posição das barras do rotor ou mesmo se ele é do tipo gaiola de esquilo dupla. Todas as cargas, com exceção das relacionadas aos motores de indução, são representadas por modelos estáticos como impedância constante, corrente constante, potência

constante ou uma combinação destas. E as linhas de transmissão e transformadores com *taps* (derivações) fora do nominal são representados por seus circuitos equivalentes na sequência positiva.

A rejeição ou restauração de carga pode ser feita com base na frequência média ou na frequência da barra de geração utilizada como referência simulando, assim, um esquema de alívio automático de carga. A rejeição não deliberada de carga também é possível, pois o programa permite representação de relés de tensão (subtensão e sobretensão).

O programa é disponibilizado na forma de um arquivo executável, com cerca de 700 kB. Não possui interface gráfica, sendo que toda entrada e saída de dados são feitas por meio de arquivos no formato texto. Ao ser executado, por linha de comando, o programa solicita o nome do arquivo que contém os dados de entrada da simulação. Em seguida solicita o nome do arquivo de saída que será gerado contendo um relatório completo da simulação.

O TransUFU gera ainda alguns arquivos temporários e outros com a extensão ADF. Cada um desses arquivos possui dados para a geração de gráficos de diversas variáveis do modelo simulado. A impressão de qualquer um dos gráficos é realizada por meio do programa PlotXY desenvolvido originalmente para o ATP-EMTP.

3.3.7 UFUFlow

O UFUFlow é um programa escrito em FORTRAN e utilizado para o cálculo do fluxo de potência. Sua primeira versão foi gerada em 1985, utilizando um ambiente de programação comercial, o Microsoft Fortran PowerStation, que foi descontinuado pelo desenvolvedor.

Assim como o TransUFU é também utilizado no âmbito da Faculdade de Engenharia Elétrica (FEELT) da UFU em disciplinas dos cursos de graduação e pós-graduação, e também para o desenvolvimento de trabalhos e projetos de pesquisa.

Para fins de estudos e de melhorias, diversas modificações já foram realizadas em seu código fonte, mas sem que houvesse um controle de versões. A versão disponível trabalha com alguns limites para a simulação do sistema elétrico (GUIMARÃES, 2000b):

a) Uso máximo de 40 barras ou nós;

b) Uso máximo de 40 ramos, incluindo as linhas de transmissão e os transformadores.

Para o processamento dos dados de entrada, o UFUFlow considera o sistema trifásico equilibrado concentrando sua análise em apenas uma fase e fazendo uso do método de Newton-Raphson.

Os dados de entrada, assim como os parâmetros da simulação e do sistema, devem ser escritos em um arquivo do tipo texto, rigorosamente tabulado e organizado numa estrutura

denominada de "cartões", sendo no total de três tipos. O primeiro cartão contém os dados gerais do sistema e comandos do programa. O segundo cartão contém os dados das barras do sistema elétrico. Os dados de ramos (linhas e transformadores) do sistema são informados no terceiro cartão.

Os resultados gerados na forma de um relatório, e armazenados em um arquivo do tipo texto, são aproveitados por um programa de análise de estabilidade, denominado TransUFU, também utilizado frequentemente por estudantes e pesquisadores da FEELT/UFU.

O programa executável do UFUFlow tem cerca de 255kB e roda somente em ambiente Windows, no seu *prompt* de comando. O manual é bastante didático e fornece um exemplo de entrada de dados para simulação.

3.4 Considerações Finais

De modo geral a maioria das aplicações apresentadas neste capítulo são do tipo *desktop*. A única exceção é o InterPSS que possui uma versão para *web*, mas sem a possiblidade do uso de uma interface gráfica interativa. Possuem ainda em comum a implementação de dois dos principais métodos numéricos, o Método de Newton-Raphson aplicado ao problema do fluxo de potência, e o Método de Euler Modificado ou Trapezoidal Implícito para a solução das equações diferenciais por meio da realização de integrações numéricas. Pode-se constatar ainda que nenhuma das aplicações apresentadas e analisadas neste capítulo possuem recursos para colaboração em tempo real e interoperabilidade com outros sistemas.

As aplicações comerciais possuem diversos pontos em comum a saber:

a) Todas são aplicações *desktop* e dependem do sistema operacional Windows;
b) Restrição de acesso ao código fonte, o que impossibilita estudá-lo e modificá-lo;
c) Possuem interfaces gráficas interativas que facilitam a entrada dos dados e a visualização dos resultados;
d) Permitem a realização de muitos tipos de estudos, como o cálculo do fluxo de potência, fluxo de potência ótimo, curto-circuito, estabilidade transitória, estabilidade a pequenas e grandes perturbações, entre outros;
e) Disponibilizam diversos modelos dos elementos do sistema elétrico e permitem ainda a criação de novos.

Já no caso das aplicações gratuitas registram-se os seguintes apontamentos:

a) O ATP-EMTP possui recursos que mais se aproximam daqueles disponíveis nos programas comerciais, tanto em quantidade como em qualidade, mas com a vantagem de ser gratuito. No entanto, os desenvolvedores não disponibilizam o seu código fonte de forma pública;
b) A MODELS no ATP-EMTP exige o aprendizado de uma nova linguagem de programação, o que pode dificultar sua utilização ou mesmo limitar suas potencialidades. Existe ainda o problema dessa linguagem não evoluir para se adaptar à solução de problemas futuros, já que ela e o próprio ATP estão restritos a um grupo fechado de desenvolvedores;
c) Não foi possível localizar o código fonte do InterPSS em suas páginas oficiais na internet;
d) Os códigos fontes do PSAT, MATPOWER, MatDyn estão disponíveis em suas páginas oficiais na internet;
e) Os códigos fontes do UFUFlow e TransUFU estão disponíveis para estudantes e pesquisadores no âmbito da Universidade Federal de Uberlândia;
f) Com exceção do InterPSS e do PSAT todos os demais foram escritos utilizando o paradigma de programação procedimental;
g) O PSAT utiliza o paradigma de programação orientado a objetos num formato padrão do MATLAB que não é muito usual e dificulta assim o seu entendimento;
h) Com exceção do InterPSS escrito na linguagem JAVA, os demais ou foram escritos em FORTRAN ou na linguagem de *script* do MATLAB;
i) Programas gratuitos e de código aberto como o MatDyn, MATPOWER e o PSAT necessitam de um programa comercial para que possam ser utilizados, nesse caso o MATLAB;
j) Os programas para MATLAB utilizam muito de suas funções disponíveis, principalmente as matemáticas, o que dificulta a reescrita dos mesmos em outras linguagens de programação;
k) MatDyn, MATPOWER, UFUFlow e TransUFU não dispõem de interface gráfica e a entrada de dados é feita pela escrita de um arquivo no formato texto, sendo que para os dois últimos é necessário ainda seguir uma tabulação rígida;
l) MATPOWER e PSAT possuem versões para o OCTAVE;
m) A versão do PSAT para o OCTAVE não dispõe de interface gráfica;

n) MatDyn, MATPOWER, UFUFlow e TransUFU são os que oferecem menos recursos em relação aos demais, mas são adequados para fins acadêmicos e podem muito bem servir de referência para a construção de novas aplicações;

o) Os códigos fontes do MatDyn e do UFUFlow são os de mais fácil compreensão, em função da documentação existente, do número reduzido de linhas de código, e também pelo fato de que o código foi escrito de forma bastante organizada e simples;

p) MatDyn depende do MATPOWER para o cálculo do fluxo de potência;

q) TransUFU depende do UFUFlow, ou de qualquer outro programa de fluxo de potência.

No próximo capítulo serão apresentadas as tecnologias utilizadas no desenvolvimento da aplicação *web* foco desta tese.

Capítulo 4

TECNOLOGIAS UTILIZADAS NA APLICAÇÃO

4.1 Considerações Iniciais

A implementação tradicional da maioria das aplicações *desktop* consiste basicamente na escrita de um código fonte e na realização do processo de compilação que gera um arquivo executável (Figura 4.1). Esse arquivo é instalado em um computador escolhido pelo usuário, sendo o mesmo específico para cada sistema operacional existente.

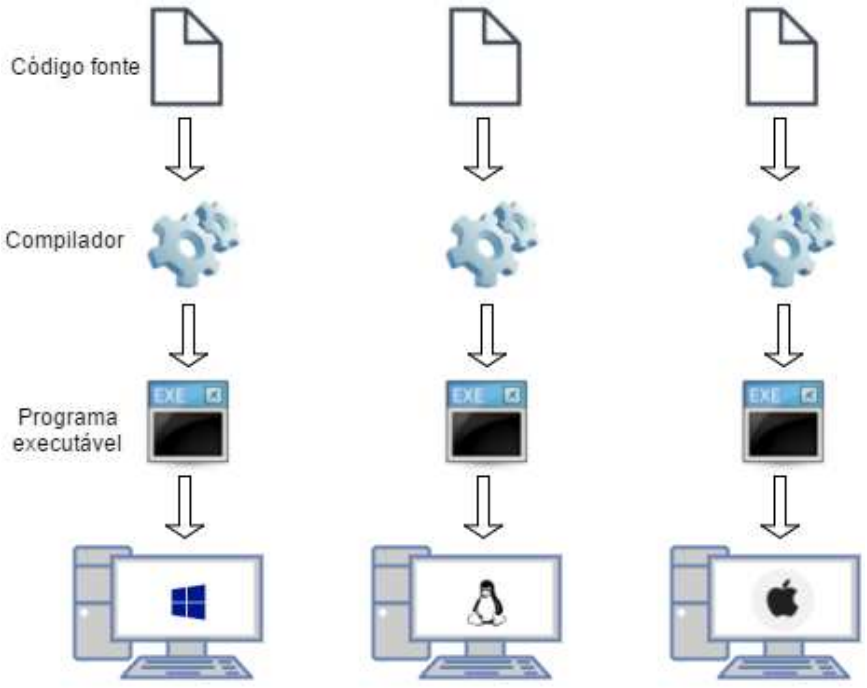


Figura 4.1 – Implementação de uma aplicação *desktop* na forma tradicional

Fonte: Elaborado pelo Autor

Outra forma disponível é a escrita de um único código fonte e a utilização de um compilador capaz de gerar arquivos executáveis para os três principais sistemas operacionais da atualidade: Windows, Linux e MacOS (Figura 4.2).

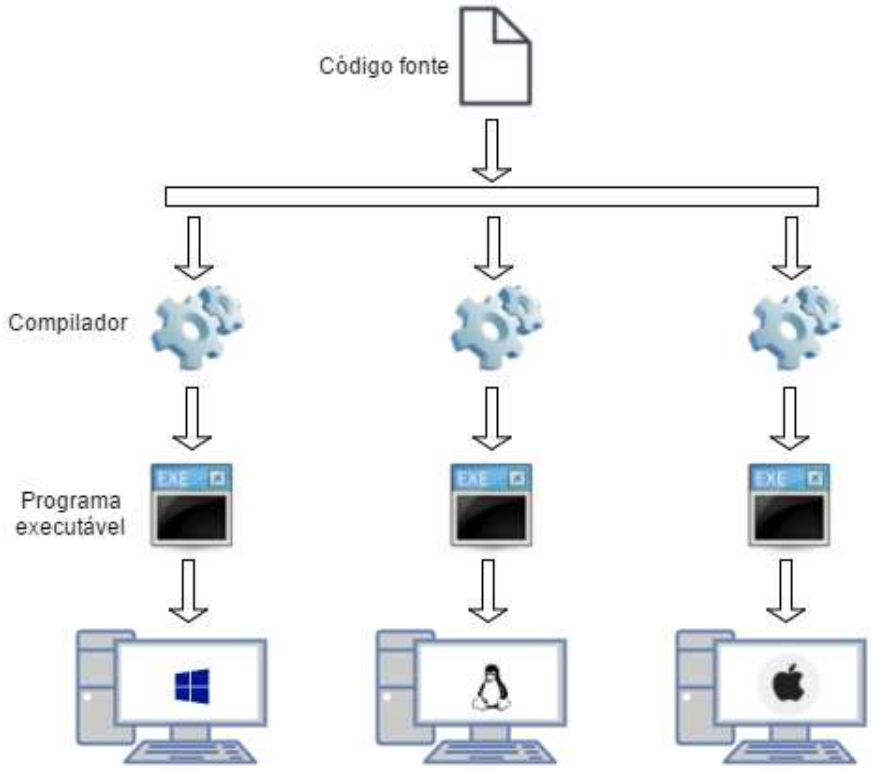


Figura 4.2 – Outra forma disponível para a implementação de uma aplicação *desktop*

Fonte: Elaborado pelo Autor

Independentemente da forma de implementação, toda aplicação *desktop* tem seu uso restrito ao computador onde o arquivo executável é instalado. Desse modo, qualquer informação manipulada na aplicação para que possa ser utilizado em outros computadores, deve ser disponibilizada na forma de seu arquivo padrão. Para isso é necessário o fornecimento de uma cópia desse arquivo, mas isso dificulta o seu controle de versões e impede o acesso e o uso simultâneo das informações por outros usuários.

No caso das aplicações *web* o código fonte da aplicação, escrita basicamente em HTML e CSS, fica alojado em um único computador, o servidor, e assim fica disponível simultaneamente para qualquer usuário e dispositivo com acesso à internet, independentemente do sistema operacional (Figura 4.3). Qualquer modificação na aplicação, para fins de correções ou adição de recursos, ficará disponível a todos os seus usuários por meio da atualização dos códigos fontes localizados no servidor. As informações manipuladas na aplicação também podem ser salvas somente no servidor na forma de seu arquivo padrão, o que facilita o acesso e permite um melhor controle de versões.

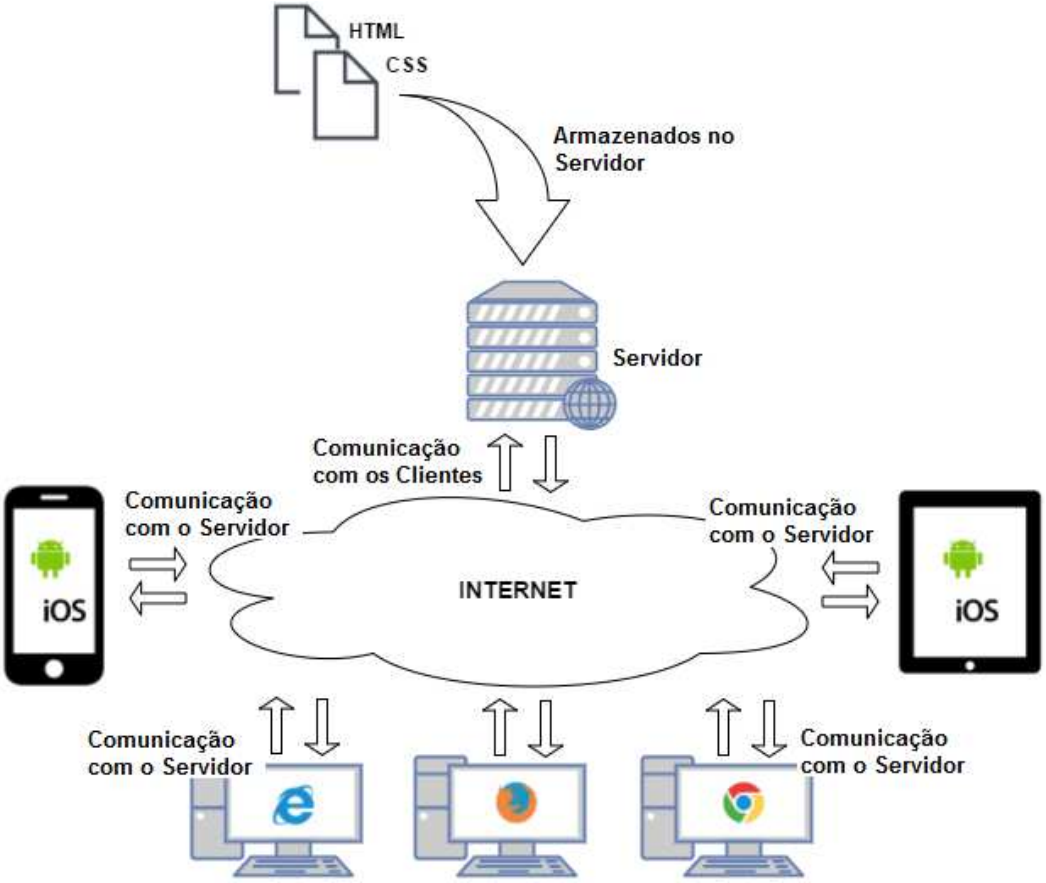


Figura 4.3 – Formas de acesso a uma aplicação *web*

Fonte: Elaborado pelo Autor

Com o objetivo de facilitar a compreensão da estrutura e do funcionamento da aplicação desta tese, que segue a arquitetura cliente-servidor, são apresentadas neste capítulo as principais tecnologias para *web* utilizadas no seu desenvolvimento.

4.2 Tecnologias *web*

Qualquer aplicação *web*, desde a mais simples até a mais sofisticada, opera basicamente sobre uma arquitetura denominada cliente-servidor, ilustrada na Figura 4.4. Como exemplos de aplicações podem ser citadas as que geram uma simples página *web*, aquelas que realizam o processamento dos dados de um formulário, e as que mantém em operação serviços de correio eletrônico.

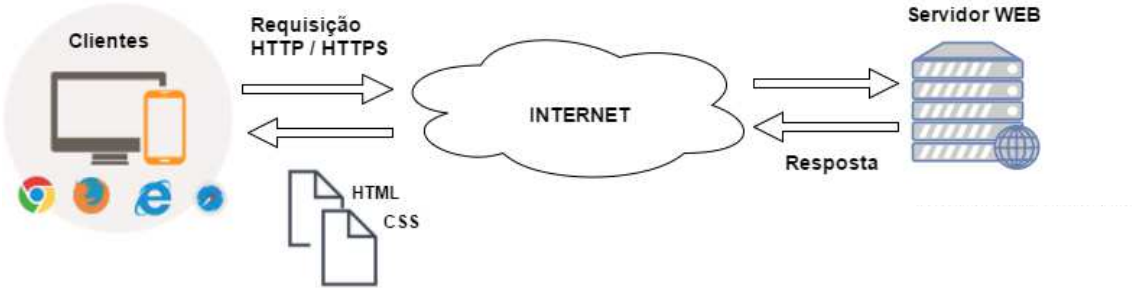


Figura 4.4 – Arquitetura básica cliente-servidor

Fonte: Elaborado pelo Autor

Nessa arquitetura um cliente por meio de um programa denominado *browser* (navegador *web* ou somente navegador) conecta-se a um servidor remoto. O cliente pode então tanto solicitar, por meio do protocolo HTTP ou HTTPS, apenas o acesso às informações de um documento, acessar informações processadas pelo servidor ou ainda enviar dados para processamento no servidor.

Na primeira situação, o servidor simplesmente procura pelo documento em seu sistema de arquivos e o envia de volta ao navegador do cliente para visualização. Esse documento escrito na linguagem HTML contém instruções textuais que permitem, após processamento no navegador, a exibição de uma página *web*. Nesta situação as informações que serão visualizadas já estão previamente definidas no código HTML contido no documento. Estilos visuais podem ainda ser adicionados com o uso conjunto de CSS.

Na segunda e terceira situação, programas armazenados no servidor é que se encarregam de realizar o processamento solicitado pelo cliente (Figura 4.5). Os programas podem estar disponíveis na forma de executáveis ou arquivos contendo códigos escritos em linguagens de *script*, como o PHP. Esta última forma é a mais usual, mas o modo de devolução do resultado obtido para o cliente em ambos os casos é o mesmo. É gerado no servidor, de forma dinâmica, e enviado para o cliente um documento escrito em HTML ou outro formato de dados como XML ou JSON.

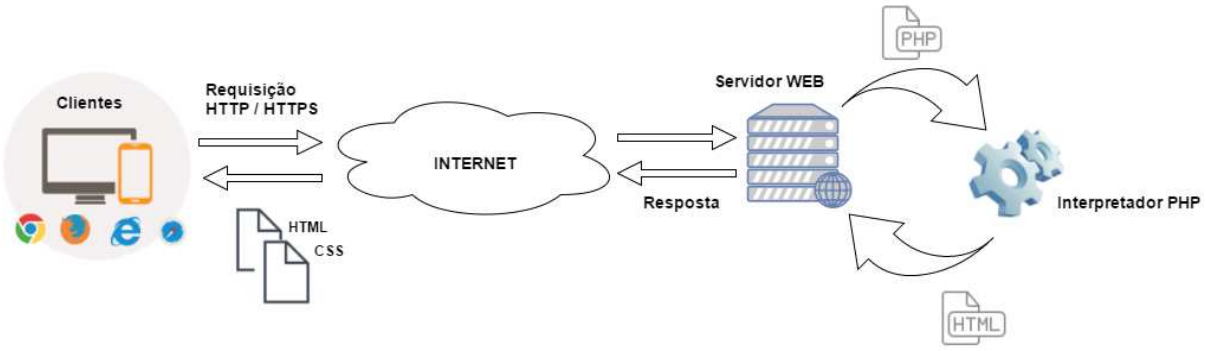


Figura 4.5 – Execução de uma aplicação PHP no servidor

Fonte: Elaborado pelo Autor

Observa-se então uma clara divisão de responsabilidades operando de forma complementares. No lado cliente temos a interpretação de códigos HTML pelo navegador, e no servidor o processamento dos dados enviados ou requisitados pelo cliente. Mas outras linguagens, principalmente JavaScript, possibilitam realizar no lado cliente processamentos que antes só eram realizados no servidor.

O conhecimento dessas linguagens e de outras tecnologias associadas contribuem para o desenvolvimento mais eficiente de uma aplicação *web*. Apesar dessa diversidade, esses códigos escritos separadamente para o cliente e para o servidor permite que a aplicação seja construída por partes, de forma modular. Isso torna a aplicação flexível e possibilita uma rápida e prática substituição de qualquer uma de suas partes, seja para a utilização de uma nova tecnologia ou a realização de correções e melhorias. E permite ainda que desenvolvedores especialistas em cada tecnologia possam contribuir separadamente no desenvolvimento da aplicação.

4.2.1 HTML

A HTML é uma linguagem de marcação utilizada para declarar a estrutura de uma página *web* que pode conter texto, imagem, vídeo, áudio, etc (MILETTO e BERTAGNOLLI, 2014). Ela define um formato de representação que permite independência de plataforma. Ou seja, o conteúdo de um documento HTML é exibido da mesma forma em diferentes navegadores, sistemas operacionais ou dispositivos.

A diferença básica entre um texto normal digitado num editor qualquer e um HTML é a presença de marcações denominadas de *tags* (etiquetas). Uma *tag* é uma palavra específica escrita entre os sinais de "menor que" (<) e "maior que" (>). De um modo geral as *tags* aparecem em pares, uma indicando o início e a outra indicando o fim da marcação. A presença dessas é que definem um elemento HTML como pode ser visto no Quadro 4.1.

Quadro 4.1 – Elemento HTML padrão

<tag> conteúdo ou elemento(s) afetado(s) pela *tag* **</tag>**

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

Algumas *tags*, no entanto, não necessitam de uma marca para fechamento. As *tags* podem ainda conter atributos, para por exemplo, permitir uma identificação única do elemento HTML ou informar uma cor de fundo da área de um determinado conteúdo.

A estrutura básica de um documento HTML segue o padrão mostrado no Quadro 4.2.

Quadro 4.2 – Modelo da estrutura básica de um documento HTML

```
<html>
<head>
<title> Título </title>
</head>
<body>
        <div id="principal">Conteúdo da página<div>
</body>
</html>
```

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

As funções de cada uma dessas *tags* estão informadas na Tabela 4.1.

Tabela 4.1 – Funções de algumas *tags* HTML

TAG	Função
html	Marca o início e o fim da página *web*, informando ao navegador que o texto contido no documento está escrito em HTML.
head	Marca o início e o fim do cabeçalho, a área onde serão descritos os cabeçalhos e o título da página. Pode ser utilizado, ainda, para declarar scripts em JavaScript ou definir formatações CSS.
title	Marca o início e o fim do título da página, que é apresentado na barra superior da janela do navegador.
body	Marca o início e o fim do corpo da página, que contem imagens, textos, títulos, *links*, etc.
div	Define uma divisão ou uma seção no documento HTML, sendo muito utilizado em conjunto com o CSS. No exemplo da estrutura básica mostrada no Quadro 4.2 a div contém um identificador nomeado "principal".

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

4.2.2 CSS

Folhas de estilo em cascata ou CSS possibilitam criar estilos personalizados para títulos, listas, imagens, etc (MILETTO e BERTAGNOLLI, 2014). Pode-se citar como exemplo de estilos, a definição de cores, fontes, bordas, alinhamentos, entre outras características vinculadas à aparência das páginas *web*.

Para criar um estilo é preciso definir uma regra CSS. Uma regra CSS é composta por duas partes: o seletor e a declaração. A declaração compreende uma propriedade e um valor como pode ser visto no Quadro 4.3.

Quadro 4.3 – Forma padrão de uma regra CSS

```
seletor { propriedade: valor }
```

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

O seletor é o alvo da regra CSS, sendo geralmente o nome de uma *tag* HTML. A declaração, além de determinar os parâmetros de estilização, compreende a propriedade que define qual característica do elemento alvo será estilizada e o valor.

Esta regra, **p { background-color: yellow }**, por exemplo, é aplicada à *tag* "p" e define a cor de fundo amarela para o seu conteúdo. A *tag* "p", a título de informação, é utilizada para adicionar parágrafos no código HTML.

Existem praticamente três formas diferentes de se vincular folhas de estilo a um documento HTML. A mais eficiente é denominada interligação externa, pois utiliza um arquivo separado (normalmente com a extensão CSS) contendo uma ou mais regras CSS. Esse arquivo é vinculado ao documento HTML por meio do uso da *tag* "link". Esse procedimento é realizado numa forma padrão mostrada no Quadro 4.4.

Quadro 4.4 – Uso de CSS no documento HTML

```
<html>
<head>
        <title> Título </title>
        <link rel="stylesheet" type="text/css" href="estilo.css" />
</head>
<body>
        Conteúdo da página
</body>
</html>
```

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

4.2.3 JavaScript e bibliotecas de terceiros

JavaScript é uma linguagem interpretada inserida dentro de documentos HTML. Quando esse documento é carregado, o próprio navegador interpreta o *script* e realiza as operações especificadas. Embora esta tecnologia seja antiga, com o surgimento de novas técnicas, ela vem sendo amplamente utilizada para a criação e a modificação dinâmica de documentos HTML, ou mesmo nas tradicionais validações de campos de formulários (MILETTO e BERTAGNOLLI, 2014).

Há duas maneiras básicas de se introduzir um código JavaScript no HTML, ambas utilizam a *tag* "script". A mais comum segue um padrão semelhante à vinculação de CSS ao documento HTML, como mostrada no Quadro 4.5.

Quadro 4.5 – Uso de JavaScript no documento HTML

```
<html>
<head>
        <title> Título </title>
        <link rel="stylesheet" type="text/css" href="estilo.css" />
        <script src="meuscript.js"></script>
</head>
<body>
        Conteúdo da página
</body>
</html>
```

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

O arquivo "meuscript.js" é um arquivo texto comum, no qual é inserido qualquer código JavaScript. Por isso, os arquivos podem ser abertos e programados em qualquer editor de texto, assim como os documentos HTML. Nesse exemplo é informado apenas o nome do arquivo, mas caso ele esteja em um diretório local no computador, ou mesmo em um endereço na internet, é preciso escrever o caminho correto para o seu acesso.

JavaScript permite também o uso do paradigma de programação orientado a objetos, mas segue um modelo diferente de outras linguagens da mesma categoria. Ela segue um estilo de programação orientada a objetos onde não existem classes. Cada novo objeto é criado clonando-se objetos existentes que são chamados de protótipos.

Cabe ressaltar ainda que é possível encontrar, de forma gratuita, bibliotecas JavaScript de terceiros disponíveis para uso na solução dos mais diversos tipos de problemas. Isso agrega à aplicação em desenvolvimento muita qualidade e recursos bastante sofisticados, pois geralmente essas bibliotecas contam com a colaboração de uma grande comunidade de desenvolvedores experientes.

Para o desenvolvimento mais rápido e seguro da aplicação desta tese foram pesquisadas e testadas mais de oito bibliotecas JavaScript diferentes. Mas, para atender aos requisitos da aplicação cliente, e ainda considerando restrições mínimas de uso, facilidade de aprendizado, utilização e integração com outras aplicações somente seis foram selecionadas:

- jQuery
- NumericJS

- Draw2D *touch*
- DHTMLX
- jqPlot
- TogetherJS

jQuery é uma biblioteca de código aberto que simplifica o uso de JavaScript para o desenvolvimento de aplicações *web* dinâmicas de grande complexidade (BALDUINO, 2015). Permite ainda compatibilizar o uso do JavaScript em diferentes navegadores e utilizar a tecnologia AJAX de um modo mais fácil.

NumericJS **é** uma biblioteca que permite a realização de cálculos numéricos sofisticados. Fornece diversas funções para a álgebra linear tanto com números reais como complexos, matrizes esparsas, e ainda a transformada rápida de Fourier, programação linear, solução de equações diferenciais ordinárias e parciais (NumericJS, 2016). No caso da aplicação desta tese, esta biblioteca foi utilizada apenas como referência para o desenvolvimento em PHP da classe matemática destinada à manipulação e a realização de operações matemáticas que envolvam matrizes esparsas.

Draw2D *touch* é um conjunto de bibliotecas com foco no desenvolvimento de aplicações *web* que precisam lidar com diagramas ou gráficos de forma interativa e em diferentes navegadores (Draw2D touch, 2016). Possui licenças de código aberto e comercial.

DHTMLX é uma biblioteca para a construção de aplicações *web* com interfaces gráficas com aparência de *desktop* e para diferentes navegadores (DHTMLX, 2016). Assim como o Draw2D *touch* possui também licenças de código aberto e comercial. A sua arquitetura é modular e permite o uso de diversos componentes, de forma separada ou combinada. Entre os principais componentes disponíveis estão aqueles destinados à construção personalizada de layouts, barras de menu, de ferramentas, de status, formulários, etc.

jqPlot é uma biblioteca de código aberto para a construção de diversos tipos de gráficos na *web*. Necessita do jQuery para funcionamento e permite adicionar legendas, rotular os eixos do gráfico, definir cores e estilos de linhas, redimensionar e salvar como imagem a área do gráfico, possui ainda ferramenta de zoom, entre outros recursos disponíveis para configuração.

TogetherJS é uma biblioteca de código aberto que permite adicionar recursos de colaboração em tempo real a qualquer página ou aplicação *web* (TogetherJS, 2016). Permite o uso simultâneo em diferentes navegadores. Assim como as outras bibliotecas não necessita da

instalação de nenhum programa ou complemento adicional para que funcione. Entre os recursos disponíveis estão:

- Mensagem de texto (*chat*) ou áudio;
- Monitoramento da presença de novos usuários;
- Acompanhamento do movimento do ponteiro do mouse e cliques de cada usuário conectado;
- Atualização dinâmica e simultânea dos conteúdos nos navegadores de cada usuário conectado por meio da criação de mensagens personalizadas.

A inicialização do TogetherJS pode ser feita de duas maneiras diferente, logo após o carregamento da página *web* ou por um clique num botão definido pelo usuário, que deve estar vinculado a um código JavaScript para realizar o tratamento dessa ação.

Assim que a ferramenta é ativada surge na lateral direita da tela uma barra com três opções básicas:

- Personalização do usuário (basicamente definição do nome e foto) e finalização da ferramenta;
- Adição de um novo usuário – É gerado um link com uma identificação única para cada aplicação que faça uso da ferramenta. Esse link deve ser repassado a um novo usuário com quem se deseja iniciar uma atividade de colaboração em tempo real;
- Mensagem de texto (*chat*) para iniciar uma conversação com todos os usuários conectados.

Essa biblioteca utiliza um servidor público do desenvolvedor para o recebimento e retransmissão, a todos os usuários conectados a aplicação, das mensagens padrões ou personalizadas pelo usuário. Entre as mensagens padrões estão, por exemplo, o envio das posições do mouse na tela de cada usuário. Um exemplo de mensagem personalizada pelo usuário poderia ser a verificação do clique em um determinado botão existente na interface, para que o efeito disso seja exibido nas telas de todos os usuários.

Existe a possibilidade de se configurar um servidor privado, mas o disponibilizado pelo desenvolvedor é suficiente para a grande maioria das aplicações, pois as mensagens recebidas e retransmitidas são pequenas e não o sobrecarregam.

4.2.4 AJAX

O AJAX não é uma linguagem de programação, mas uma técnica de programação que possibilita enviar e receber dados de um servidor *web* de forma assíncrona (MILETTO e

BERTAGNOLLI, 2014). Essa tecnologia permite que o usuário continue a interagir com uma página *web*, enquanto uma requisição é processada no servidor. Além disso, possibilita a atualização de partes de uma página sem a necessidade de recarregar todo o documento HTML.

A adoção do AJAX traz algumas vantagens importantes. Por permitir a atualização de partes de um documento HTML, a quantidade de dados trafegados entre o cliente e o servidor, e, consequentemente, o tempo de resposta, diminui, e a performance da montagem da interface *web* para o usuário aumenta. Também é possível desenvolver componentes de interface interativos, tornando as aplicações *web* cada vez mais semelhantes às aplicações *desktop*. Essa classe de aplicações *web*, que possui uma interface similar a uma aplicação *desktop*, é denominada RIA.

AJAX utiliza JavaScript e XML para a troca de dados com o servidor. Apesar deste último fazer parte do nome desta tecnologia, outro formato tem sido mais utilizado no seu lugar, o JSON. JSON é mais leve que o XML, e é ainda um subconjunto da notação de objetos em JavaScript, sendo assim mais facilmente manipulado por esta linguagem. Uma comparação entre XML e JSON para a representação de um mesmo conjunto de informações é apresentada na Tabela 4.2

Tabela 4.2 – Comparação entre XML e JSON

XML	JSON

<?xml version="1.0" ?>
<analysis>
<type>loadflow</type>
<method>nr</method>
<tolerance>1e-3</tolerance>
</analysis>
	{
	“analysis”: {
	“type”: “loadflow”,
	“method”: ”nr”,
	“tolerance”: 1e-3
	}
	}

Fonte: Elaborado pelo Autor

4.2.5 PHP

PHP é uma linguagem de *script* de uso geral, interpretada, muito utilizada para o desenvolvimento de aplicações *web*, podendo ser mesclada dentro do código HTML (DALL'OGLIO, 2007). A linguagem surgiu em 1994 como um projeto pessoal de Rasmus Lerdorf com o intuito de controlar acessos a sua página *web*. Atualmente PHP é um projeto da Apache *Software Foundation*.

É uma linguagem bastante popular para o desenvolvimento para *web*, e as principais razões disso são (BENTO, 2015):

- PHP nasceu para a *web* e sua integração com servidores *web* é simples;
- PHP tem uma curva de aprendizado suave, comparada a outras linguagens como, por exemplo, JAVA;
- PHP é um *software* livre;
- É fácil encontrar serviço de hospedagem que oferece PHP;
- Serviços de hospedagem PHP são mais baratos do que aqueles para outras tecnologias.

PHP pode ser utilizado em conjunto com as marcações da linguagem HTML. Para a identificação, pelo servidor *web*, dos trechos que devem ser interpretados como *scripts* em PHP, é preciso utilizar a *tag* "php", como apresentado no Quadro 4.6.

Quadro 4.6 – Forma padrão para inserir código PHP no documento HTML

<?php código PHP **?>**

Fonte: MILETTO e BERTAGNOLLI, 2014 (modificado)

O código PHP é interpretado no servidor e gera, após processamento, códigos HTML, JavaScript, além de documentos PDF, XML, JSON, imagens ou textos, os quais podem ser enviados ao cliente ou simplesmente armazenados no servidor.

O PHP permite ainda o uso do paradigma de programação orientado a objetos e se destaca por oferecer suporte a um grande número de Sistemas de Gerenciamento de Banco de Dados, como dBase, Firebird/Interbase, Sybase, MySQL, PostgreSQL, entre outros.

Os recursos disponíveis na PHP são disponibilizados por meio de módulos (ou extensões), que nada mais são do que bibliotecas dinâmicas na forma de arquivos com extensão "dll", no ambiente Microsoft Windows, e "so", no Linux. A geração desses arquivos passa pelo processo de compilação do código fonte do PHP em conjunto com o código que irá implementar um novo recurso. Desse modo, novos módulos podem ser criados utilizando para isso tanto a linguagem C ou C++.

Assim como JavaScript possui muitas bibliotecas e *frameworks* gratuitas desenvolvidas por terceiros para a solução dos mais diversos tipos de problemas. Esses recursos também contam com a colaboração de uma grande comunidade de desenvolvedores experientes.

4.2.6 SERVIÇOS *WEB*

Serviços *Web* (*Web* Services) são utilizados como uma forma de integração e comunicação de aplicações, de modo que uma aplicação possa realizar uma chamada para um serviço de outra aplicação a fim de obter informações (LECHETA, 2015). Estas chamadas podem enviar e receber informações em diversos formatos, sendo que atualmente os mais populares são XML e JSON.

Uma das grandes vantagens na construção de serviços *web* é que eles permitem acessar os serviços de uma forma padronizada e independentemente de plataforma e da linguagem de programação. Por exemplo, é possível escrever um serviço *web* com a linguagem PHP, mas "consumi-lo" em qualquer outra linguagem, como JavaScript.

Existem várias formas de se criar serviços *web*, mas pode-se dizer que duas das mais conhecidas são os serviços *web* em SOAP ou REST.

REST é uma técnica de desenvolvimento de serviços *web* fortemente baseada nos métodos do protocolo HTTP, como GET, POST, PUT e DELETE. O primeiro é o mais utilizado de todos. Qualquer acesso ao conteúdo de uma página na internet é realizado com o método GET. Depois deste, POST também é muito usado, sendo bastante comum seu uso quando da submissão das informações de um formulário, principalmente quando é necessário o envio de arquivos ao servidor e um grande volume de dados.

Qualquer serviço *web* que siga o padrão REST é denominado RESTful. Este apresenta uma sintaxe mais enxuta e pode enviar e receber informações em formatos mais leves, neste caso JSON, se comparado com o SOAP que utiliza XML.

Em PHP uma maneira mais prática de se construir um serviço *web* RESTful é com o auxílio do *microframework* SLIM.

4.2.7 GOOGLE API

É um conjunto de rotinas e padrões de programação para acesso a diversos serviços *web* públicos e gratuitos do Google e que podem ser utilizados e integrados a qualquer página ou aplicação *web* (Google API, 2016). Entre alguns desses serviços cita-se: autenticação, tradução, armazenamento de arquivos, mapas etc.

Para isso são fornecidas bibliotecas em diversas linguagens, como JAVA, JavaScript, PHP e Python para a programação das aplicações clientes.

No caso da aplicação desta tese foram utilizadas a biblioteca JavaScript para acesso ao serviço de disco virtual (Google Drive).

4.2.8 APACHE

O Apache é um *software* livre que funciona como um servidor *web*, tanto no Linux como no Windows. Assim como qualquer servidor do tipo, o Apache é responsável por disponibilizar páginas *web* e todos os recursos que podem ser acessados na internet (Apache, 2016). Envio de e-mails, mensagens, compras online e diversas outras funções podem ser executadas por servidores *web* como o Apache.

A Apache *Software Foundation* é a responsável pelo projeto, além de desenvolver e trabalhar com outras tecnologias de transmissão via *web*, processamento de dados e execução de aplicativos distribuídos.

Ele permite ainda mudanças em suas características mesmo as mais internas através da utilização de módulos, o que o torna bastante flexível.

4.3 Considerações Finais

Este capítulo apresentou diversas tecnologias para *web* que foram utilizadas no desenvolvimento da aplicação desta tese. Elas podem ser divididas entre as que rodam no computador do usuário, lado cliente, e as que rodam no servidor. A Figura 4.3 apresenta a divisão das tecnologias apresentadas neste capítulo.

Tabela 4.3 – Tecnologias para *web* utilizadas

Tecnologias no Lado Cliente	Tecnologias no Lado Servidor
HTML	PHP
CSS	Serviços *web* com Slim Framework
JavaScript e bibliotecas de terceiros (JavaScript: jQuery, NumericJS, Draw2D, DHTMLX, jqPlot e TogetherJS)	Google API
AJAX	Apache

Fonte Elaborado pelo Autor

Todas essas tecnologias são gratuitas e possuem um vasto material de consulta na internet. As linguagens JavaScript e PHP são muito populares. Elas estão, respectivamente, na 1ª e 3ª colocação num ranking de 2015 feito pela empresa de consultoria Redmonk (The RedMonk Programming Language Rankings: June 2015, 2015). Esse *ranking* leva em consideração a discussão sobre linguagens de programação no *site* Stack Overflow e sua utilização no Github, um repositório de projetos de *software*. Por serem dois *sites* importantes para a comunidade de desenvolvedores, eles servem como uma boa base para análise.

Convém mencionar que a escrita de códigos em HTML, CSS, JavaScript e PHP pode ser realizado em qualquer editor de texto, como o Bloco de Notas do Microsoft Windows. Apesar dessa facilidade o mais indicado é o uso de um ambiente de programação, que permite a marcação de sintaxe, identação do código e verificação de erros de uma forma mais prática. Há muitos gratuitos e de qualidade, tendo sido escolhido o Netbeans IDE, que pode ser utilizado como ambiente de programação para diversas linguagens (NetBeans, 2016).

As tecnologias para *web* no lado cliente não precisam de nenhuma instalação, pois são utilizadas diretamente em qualquer navegador. Já as tecnologias no lado servidor precisam ser devidamente instaladas e configuradas no computador que será utilizado como servidor. Para uma instalação local, seja num computador ou numa rede local de computadores há programas de instalação gratuitos, denominados de LAMP, para o sistema operacional Linux, ou WAMP, para Windows. Onde AMP diz respeito a Apache+MySQL+PHP. Ou seja, esses programas realizam a instalação prática dessas três tecnologias, seja no Linux ou no Windows.

No próximo capítulo será apresentada a aplicação desenvolvida, com destaque para a explicação de trechos mais relevantes do seu código fonte.

Capítulo 5

APRESENTAÇÃO DA APLICAÇÃO

5.1 Considerações Iniciais

A aplicação *web* desenvolvida opera na arquitetura cliente-servidor, sendo constituída por um conjunto de aplicações integradas e com responsabilidades distintas. Ela foi batizada de NDSE *WEB* Simulator e a sigla NWS foi escolhida para designá-la de uma forma mais prática.

No servidor foram implementadas duas aplicações para a análise de sistemas elétricos, uma para o cálculo do fluxo de potência e outra para a análise de estabilidade transitória, disponibilizadas como serviços *web*. Para isso foram utilizados o Slim Framework e o formato JSON para a troca de dados.

A aplicação construída com o Slim Framework opera com um formato de arquivo JSON padrão para a entrada de dados e outro para o retorno de resultados de processamento, conforme o tipo de análise a ser realizada.

No cliente foram criadas três aplicações para a interação do usuário com as aplicações localizadas no servidor. Cada uma delas dispõe de uma quantidade limitada de recursos e diferentes níveis de complexidade para implementação.

Neste capítulo serão apresentadas as aplicações construídas, bem como fornecidos detalhes explicativos de alguns trechos de código do NWS.

5.2 Aplicações no servidor

As aplicações criadas no lado do servidor foram escritas na linguagem PHP e contém implementações dos principais algoritmos para a solução do fluxo de potência e para a análise da estabilidade transitória de um sistema elétrico genérico. O método de Newton-Raphson foi utilizado no primeiro, e o método de Euler modificado no segundo.

A aplicação para a solução do fluxo de potência teve como referência o código fonte do UFUFlow, que foi estruturado num único arquivo, escrito em FORTRAN na forma procedimental e sem o uso de matrizes esparsas. Todo o código foi reescrito em PHP utilizando orientação a objetos e inserido ainda a possiblidade de se trabalhar com matrizes esparsas.

No caso da análise de estabilidade foi utilizado como referência o código fonte do MatDyn, que foi estruturado em mais de um arquivo, escrito na linguagem de *script* do MATLAB na forma procedimental. Partes desse código foram reescritas em PHP utilizando o paradigma de programação orientado a objetos.

Essas duas aplicações foram construídas sob a forma de um pacote de classes denominado de NDSE, e subdividido em três subpacotes, Math, Models e Tools. A Figura 5.1 apresenta o diagrama de pacotes e os relacionamentos de dependência existente, identificados pela seta tracejada. O desenvolvimento do pacote matemático foi necessário, pois o PHP possui somente funções matemáticas básicas, insuficientes para os requisitos de funcionamento da aplicação.

O pacote Math agrupa classes para a manipulação e a realização de operações matemáticas com números complexos, matrizes cheias e esparsas. Já no pacote Models estão presentes classes para o modelamento matemático, por meio da escrita das equações diferenciais e algébricas, de máquinas síncronas e de reguladores de tensão. E no pacote Tools estão as classes para a solução do fluxo de potência e para a análise de estabilidade transitória, e que dependem das classes dos pacotes Math e Models.

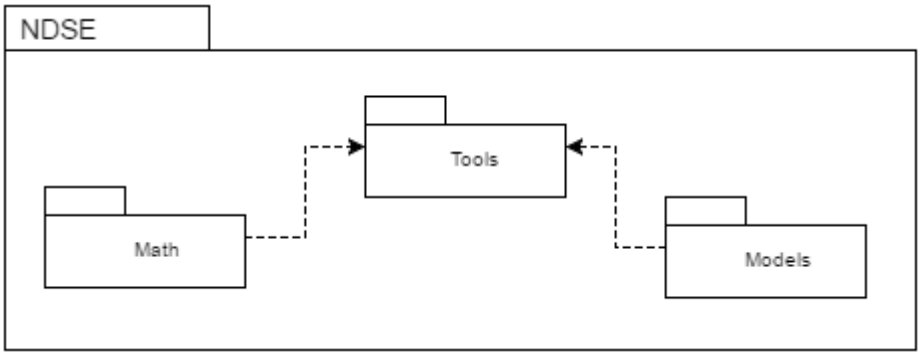


Figura 5.1 – Pacote NDSE

Fonte: Elaborado pelo autor

5.2.1 Pacote NDSE/Math

O pacote NSE/Math contém seis classes, sendo uma delas abstrata[5]. A Figura 5.2 apresenta o diagrama de classes desse pacote e os relacionamentos existentes.

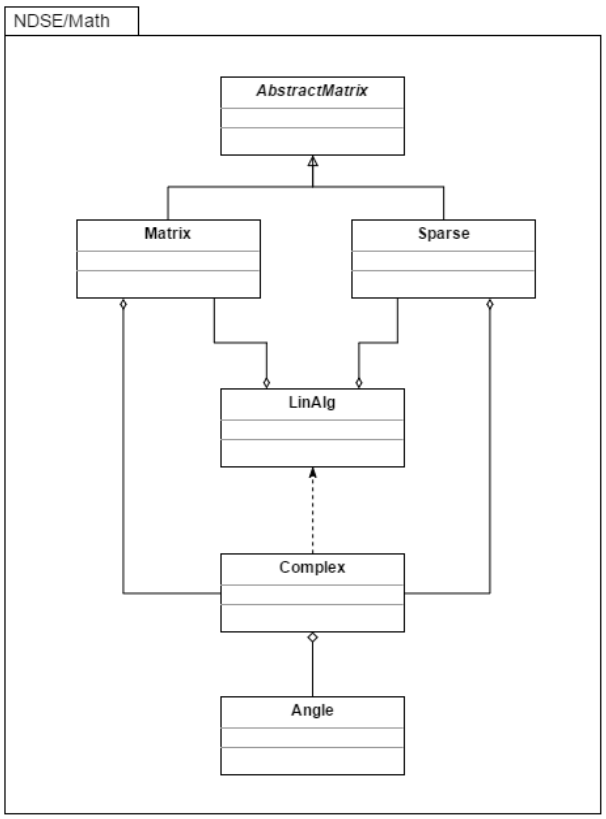


Figura 5.2 – Pacote NDSE/Math

Fonte: Elaborado pelo autor

[5] Classe abstrata é aquela que possui pelo menos um método abstrato. E método abstrato é um método que não possui implementação. Sua implementação é obrigatória apenas nas classes que herdam as propriedades e métodos (classes filhas) da classe abstrata.

5.2.1.1 Classe Angle

A função desta classe é possibilitar a criação e a manipulação de ângulos, tanto em graus como em radianos, para serem utilizados na definição de números complexos na forma polar. Na Tabela 5.1 são apresentadas as constantes, atributos e métodos da classe Angle disponíveis para uso

Tabela 5.1 – Classe *Angle*

Constantes	**Descrição**
TYPE_RAD	Possui como valor a palavra 'rad' para indicar na classe ângulos em radianos
TYPE_DEG	Possui como valor a palavra 'deg' para indicar na classe ângulos em graus
Atributos	**Descrição**
# $ang_rad	Atributo protegido utilizado para armazenar o valor do ângulo em radianos
# $ang_deg	Atributo protegido utilizado para armazenar o valor do ângulo em graus
# $type	Atributo protegido utilizado para armazenar se o valor original do ângulo é radianos ou graus
Métodos	**Descrição**
+ deg()	Método público que retorna o valor do ângulo em graus
+ rad()	Método público que retorna o valor do ângulo em radianos

Fonte Elaborado pelo Autor

5.2.1.2 Classe Complex

A função desta classe é possibilitar a criação, a manipulação e a realização de operações matemáticas com números complexos, tanto na forma retangular (ou cartesiana) como na forma polar. Ela agrega a classe Angle para a representação de números complexos na forma polar, e dispõe de métodos para devolver o valor do número complexo na forma cartesiana ou polar, independentemente da sua forma original.

Na Tabela 5.2 são apresentadas as constantes, atributos e métodos da classe Complex disponíveis para uso.

Tabela 5.2 – Classe Complex

Constantes	**Descrição**
RECTANGULAR	Possui como valor o número 1 para indicar um número complexo na forma cartesiana
POLAR	Possui como valor o número 2 para indicar um número complexo na forma polar
Atributos	**Descrição**
# $z	Atributo protegido utilizado para armazenar o número complexo na

	sua forma cartesiana. É utilizado para isso um vetor com índices 're' e 'img', respectivamente, para a parte real e para a parte imaginária
# $z_polar	Atributo protegido utilizado para armazenar o número complexo na sua forma polar. É utilizado para isso um vetor com os índices 'z_abs' e 'z_ang', respectivamente, para o seu módulo e ângulo
Métodos	**Descrição**
+ abs	Método público que retorna o módulo do número complexo
+ ang()	Método público que retorna o ângulo do número complexo
+ conj()	Método público que retorna o conjugado do número complexo
+ neg()	Método público que retorna o valor negativo do número complexo
+ inv()	Método público que retorna o inverso do número complexo
+ add($z)	Método público que soma dois números complexos, sendo um deles fornecido como argumento do método
+ sub($z)	Método público que subtrai dois números complexos, sendo um deles fornecido como argumento do método
+ multiply($z)	Método público que multiplica dois números complexos, sendo um deles fornecido como argumento do método
+ div($z)	Método público que divide dois números complexos, sendo um deles fornecido como argumento do método
+ equal($z)	Método público que compara se dois números complexos são iguais, sendo um deles fornecido como argumento do método
+ rectangular()	Método público que retorna o número complexo na sua forma cartesiana
+ polar()	Método público que retorna o número complexo na sua forma polar

Fonte Elaborado pelo Autor

5.2.1.3 Classe AbstractMatrix

A classe AbstractMatrix é do tipo abstrato e contém atributos e métodos comuns às classes concretas, Matrix e Sparse. Estas derivam da classe abstrata e herdam todos os seus atributos e métodos, implementando obrigatoriamente os seus métodos abstratos cada um a sua maneira.

A Tabela 5.3 apresenta os atributos e os métodos, sendo quatro deles abstratos, da classe AbstractMatrix.

Tabela 5.3 – Classe AbstractMatrix

Atributos	**Descrição**
# $num	Atributo protegido utilizado para armazenar num vetor o valor do número de linhas e de colunas da matriz
Métodos	**Descrição**
+ getN($name)	Método público que retorna o número de linhas ou de colunas da matriz, sendo necessário informar como argumento a palavra 'rows' ou 'cols'
+ *get*($row = [], $col = [])	Método abstrato público com dois argumentos, $row e $col, com valores padrões nulos
+ *set*($value, $row = [], $col = [])	Método abstrato público com três argumentos, $value,

	$row e $col, sendo os dois últimos com valores padrões nulos
+ *add*($matrix)	Método abstrato público com um argumento, $matrix
+ *multiply*($matrix)	Método abstrato público com um argumento, $matrix

Fonte Elaborado pelo Autor

5.2.1.4 Classe *Matrix*

A classe Matrix possui uma relação de agregação com a classe Complex e tem como função possibilitar a criação, a manipulação e a realização de operações matemáticas com matrizes reais e complexas. Na Tabela 5.4 são apresentados os atributos e os métodos da classe Matrix disponíveis para uso.

Tabela 5.4 – Classe Matrix

Atributos	**Descrição**
# $arr	Atributo protegido utilizado para armazenar os valores da matriz
# $num	Atributo protegido utilizado para armazenar num vetor o valor do número de linhas e de colunas da matriz. É utilizado para isso um vetor com os índices 'rows' e 'cols'
Métodos	**Descrição**
+ getN($name)	Método púbico que retorna o número de linhas ou de colunas da matriz, sendo necessário informar como argumento a palavra 'rows' ou 'cols'
+ get($row = [], $col = [])	Método público que retorna um elemento, ou todos os elementos de uma linha ou coluna da matriz
+ set($value, $row = [], $col = [])	Método público para atribuir um novo valor a um determinado elemento da matriz
+ add($matrix)	Método público que soma duas matrizes, sendo uma delas fornecida como argumento do método
+ multiply($matrix)	Método público que multiplica uma matriz por um escalar, ou duas matrizes, sendo um ou outro fornecido como argumento do método
+ zeros($row, $col)	Método estático público para criar uma matriz com zeros, sendo necessário para isso informar como argumentos o número de linhas e de colunas
+ equal($matrix)	Método público que compara se duas matrizes são iguais, sendo uma delas fornecida como argumento do método
+ transpose()	Método público que retorna a transposta da matriz
+ sparse()	Método público que retorna a matriz na sua forma esparsa

Fonte Elaborado pelo Autor

5.2.1.5 Classe Sparse

A classe Sparse também possui uma relação de agregação com a classe Complex e tem como função possibilitar a criação, a manipulação e a realização de operações matemáticas com matrizes esparsas reais e complexas. Na Tabela 5.5 são apresentados os atributos e os métodos da classe Sparse disponíveis para uso.

Tabela 5.5 – Classe Sparse

Atributos	Descrição
# $arr	Atributo protegido utilizado para armazenar os vetores necessários para a construção da matriz esparsa. É utilizado para isso um vetor com os índices 'i', 'j' e 'v'
# $num	Atributo protegido utilizado para armazenar num vetor o valor do número de linhas e de colunas da matriz. É utilizado para isso um vetor com os índices 'rows' e 'cols'
Métodos	**Descrição**
+ getN($name)	Método público que retorna o número de linhas ou de colunas da matriz esparsa, sendo necessário informar como argumento a palavra 'rows' ou 'cols'
+ get($row = [], $col = [])	Método público que retorna um elemento, ou todos os elementos de uma linha ou coluna da matriz esparsa
+ set($value, $row = [], $col = [])	Método público para atribuir um novo valor a um determinado elemento da matriz esparsa
+ add($matrix)	Método público que soma duas matrizes esparsas, sendo uma delas fornecida como argumento do método
+ multiply($matrix)	Método público que multiplica uma matriz esparsa por um escalar, ou duas matrizes esparsas, sendo um ou outro fornecido como argumento do método
+ full()	Método público que retorna a matriz na sua forma "cheia"

Fonte Elaborado pelo Autor

5.2.1.6 Classe LinAlg

A classe LinAlg é do tipo estático e possui uma relação de agregação com as classes Matrix e Sparse, e de dependência da classe Complex. Ela tem como função possibilitar a realização da decomposição LU de uma matriz esparsa e a solução de um sistema de equações com a presença das matrizes resultantes dessa decomposição. Na Tabela 5.6 são apresentados os atributos e os métodos da classe LinAlg disponíveis para uso.

Tabela 5.6 – Classe LinAlg

Métodos	Descrição
+ LUdecomp($sparse)	Método público que decompõe uma matriz esparsa passada como argumento do método e retorna numa

	array do PHP as matrizes L e U, também esparsas
+ LUsolver($LU, $vector)	Método público que resolve um sistema de equações lineares, sendo necessário informar como argumentos a array do PHP contendo as matrizes esparsas L e U, e o de termos independentes

Fonte Elaborado pelo Autor

5.2.2 Pacote NDSE/Models

O pacote NSE/Models contém quatro classes, sendo uma delas abstrata. A Figura 5.3 apresenta o diagrama de classes desse pacote e os relacionamentos existentes.

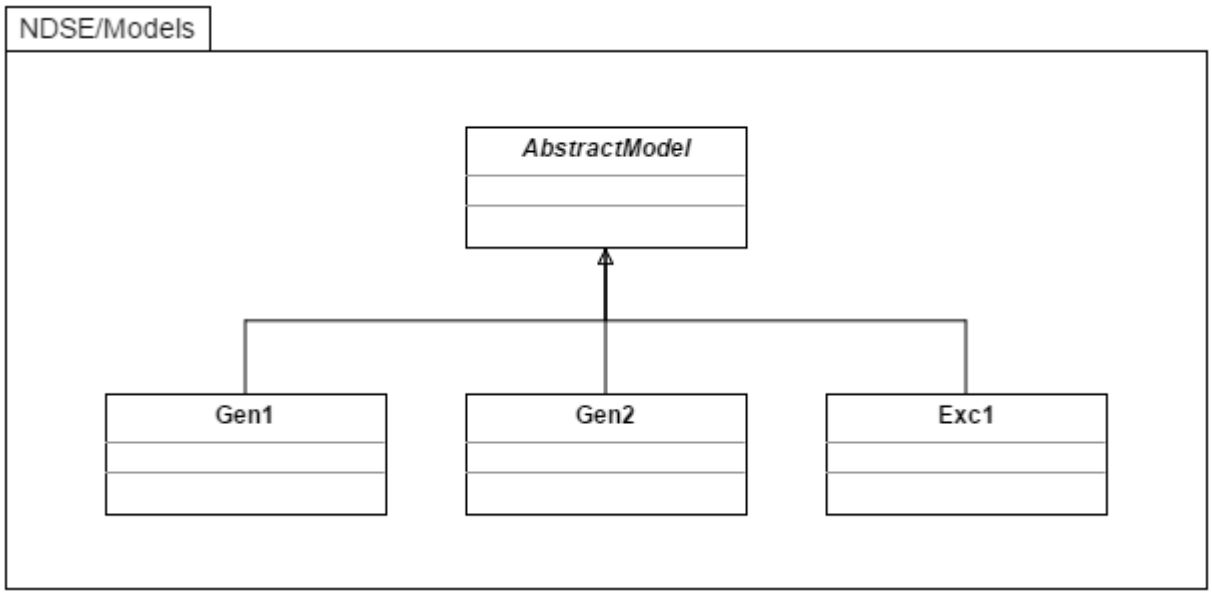


Figura 5.3 – Pacote NDSE/Models

Fonte: Elaborado pelo autor

5.2.2.1 Classe AbstractModel

A classe AbstractModel é do tipo abstrato e contém atributos e métodos comuns às classes concretas, Gen1, Gen2 e Exc1. Estas derivam da classe abstrata e herdam todos os seus atributos e métodos, implementando obrigatoriamente os seus métodos abstratos cada um a sua maneira.

A Tabela 5.7 apresenta os atributos e os métodos, sendo dois deles abstratos, da classe AbstractTools.

Tabela 5.7 – Classe AbstractModel

Atributos	Descrição
# $input	Atributo protegido utilizado para armazenar num vetor os valores de entrada do modelo

# $param	Atributo protegido utilizado para armazenar num vetor os parâmetros do modelo
# $X	Atributo protegido utilizado para armazenar num vetor os valores das variáveis de estado do modelo
# $dX	Atributo protegido utilizado para armazenar num vetor os valores dos diferenciais das variáveis de estado do modelo
# $Y	Atributo protegido utilizado para armazenar num vetor os valores das variáveis algébricas do modelo
Métodos	**Descrição**
+ get($name)	Método público que retorna o valor de uma das variáveis do modelo
+ set($name,$value)	Método público para atribuir um novo valor para uma variável de estado, para o diferencial de uma variável de estado ou para uma variável algébrica. Opera com dois argumentos, $name e $value
+ *x0*()	Método abstrato público para calcular os valores iniciais das variáveis de estado do modelo
+ *dFx*($input)	Método abstrato público para calcular os valores dos diferenciais das variáveis de estado do modelo

Fonte Elaborado pelo Autor

5.2.2.2 Classe Gen1

A classe Gen1 implementa o modelo do gerador clássico por meio das equações diferenciais 5.1 e 5.2, e a equação algébrica 5.3 (COLE, 2016). Na Tabela 5.8 são apresentados os atributos e os métodos da classe Gen1 disponíveis para uso.

$$\dot{\omega} = \frac{\pi f_0}{H}\left(-D(\omega_0 - \omega) + P_m - P_e\right) \tag{5.1}$$

$$\dot{\delta} = \omega_0 - \omega \tag{5.2}$$

$$P_e = (1/x_d) \cdot U \cdot E'_q \sin(\delta - \theta) \tag{5.3}$$

Onde:

f_0 – Frequência base do sistema (Hz);

δ – Ângulo de defasagem da posição do rotor em relação a uma referência girante na velocidade ω_0 (rad);

ω – Velocidade angular (p.u.);

ω_0 – Velocidade angular de referência na frequência base do sistema (p.u.);

D – Constante de amortecimento (p.u.);

E'_q – Tensão interna transitória do eixo em quadratura (p.u.);

H – Constante de inércia (p.u.);

Pe – Potência elétrica (p.u.);

Pm – Potência mecânica (p.u.);

U – Tensão na barra do gerador (p.u.);

θ – Ângulo da tensão U (rad);

x_d – Reatância síncrona do eixo direto (p.u.)

Tabela 5.8 – Classe Gen1

Atributos	**Descrição**
# $idv	Atributo protegido utilizado para armazenar uma array contendo todos os índices referentes a cada conjunto de variáveis do modelo
Métodos	**Descrição**
+ x0()	Método público para calcular os valores iniciais das variáveis de estado do modelo considerando iguais a zero os valores dos diferenciais nas equações de estado do modelo
+ dFx($input)	Método público para calcular os valores diferenciais das variáveis de estado do modelo. Aqui são escritas as equações diferenciais do modelo
+ Gy($input)	Método público para calcular os valores das variáveis algébricas do modelo. Aqui são escritas as equações algébricas do modelo

Fonte Elaborado pelo Autor

5.2.2.3 Classe Gen2

A classe Gen2 implementa o modelo do gerador de 4ª ordem por meio das equações diferenciais 5.4 a 5.7, e as equações algébricas 5.8 a 5.12 (COLE, 2016). Na Tabela 5.9 são apresentados os atributos e os métodos da classe Gen2 disponíveis para uso.

$$\dot{\omega} = \frac{\pi f_0}{H}\left(-D(\omega_0 - \omega) + P_m - P_e\right) \tag{5.4}$$

$$\dot{\delta} = \omega_0 - \omega \tag{5.5}$$

$$\dot{E}_q = \frac{1}{T'_d}\left(E_{fd} - E'_q + (x_d - x'_d) + I_d\right) \tag{5.6}$$

$$\dot{E}_d = \frac{1}{T'_q}\left(-E'_d - (x_q - x'_q) + I_q\right) \tag{5.7}$$

$$I_d = (v_q - E'_q)/x'_d \tag{5.8}$$

$$I_q = -(v_d - E'_d)/x'_q \tag{5.9}$$

$$P_e = E'_q I_q - E'_d I_d + (x'_d - x'_q) I_d I_q \tag{5.10}$$

$$v_d = -U\sin(\delta - \theta) \tag{5.11}$$

$$v_q = U\cos(\delta - \theta) \tag{5.12}$$

Onde:

f_0 – Frequência base do sistema (Hz);

δ – Ângulo de defasagem da posição do rotor em relação a uma referência girante na velocidade ω_0 (rad);

ω – Velocidade angular (p.u.);

ω_0 – Velocidade angular de referência na frequência base do sistema (p.u.);

D – Constante de amortecimento (p.u.);

E'_d – Tensão interna transitória do eixo direto (p.u.);

E'_q – Tensão interna transitória do eixo em quadratura (p.u.);

E_{fd} – Tensão do enrolamento de campo (p.u.);

H – Constante de inércia (p.u.);

I_d – Corrente terminal do eixo direto (p.u.);

I_q – Corrente terminal do eixo em quadratura (p.u.);

Pe – Potência elétrica (p.u.);

Pm – Potência mecânica (p.u.);

T'_d – Constante de tempo do eixo direto (s);

T'_q – Constante de tempo do eixo em quadratura (s);

U – Tensão na barra do gerador (p.u.);

θ – Ângulo da tensão U (rad);

v_d – Tensão terminal do eixo direto (p.u.);

v_q – Tensão terminal do eixo em quadratura (p.u.);

x'_d – Reatância transitória do eixo direto (p.u.);

x'_q – Reatância transitória do eixo em quadratura (p.u.);

x_d – Reatância síncrona do eixo direto (p.u.);

x_q – Reatância síncrona do eixo em quadratura (p.u.)

Tabela 5.9 – Classe Gen2

Atributos	**Descrição**
# $idv	Atributo protegido utilizado para armazenar uma array contendo todos os índices referentes a cada conjunto de variáveis do modelo
Métodos	**Descrição**
+ x0()	Método público para calcular os valores iniciais das variáveis de estado do modelo considerando iguais a zero os valores dos diferenciais nas equações de estado do modelo
+ dFx($input)	Método público para calcular os valores diferenciais das variáveis de estado do modelo. Aqui são escritas as equações diferenciais do modelo

+ Gy($input)	Método público para calcular os valores das variáveis algébricas do modelo. Aqui são escritas as equações algébricas do modelo

Fonte Elaborado pelo Autor

5.2.2.4 Classe Exc1

A classe Exc1 implementa o modelo do regulador de tensão tipo 1 do IEEE (Figura 5.4). Na Tabela 5.10 são apresentados os atributos e os métodos da classe Exc1 disponíveis para uso.

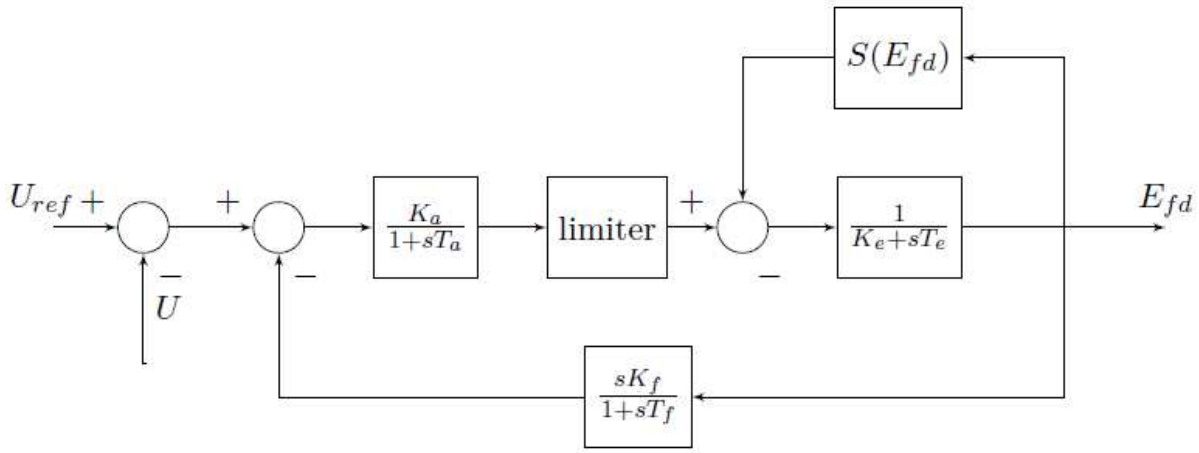


Figura 5.4 – Modelo do regulador de tensão tipo I do IEEE

Fonte: COLE, 2016

Onde:

U_{ref} – Tensão de referência em p.u.;

$\overline{U}$ – Tensão em p.u. da barra onde se encontra o gerador;

K_a e T_a – Ganho e constante de tempo em segundos do sistema que representa o regulador principal;

K_e e T_e – Ganho e constante de tempo em segundos do sistema de primeira ordem que representa a excitação;

K_f e T_f – Ganho e constantes de tempo em segundos que representam a realimentação;

limiter – Limitador da tensão de saída do regulador principal, com limites máximo (Ur_{max}) e mínimo (Ur_{min}), ambos em p.u.;

E_{fd} – Tensão de campo em p.u. do gerador síncrono;

$S(E_{fd})$ – Função da tensão de campo em p.u. que representa a saturação da excitatriz.

Tabela 5.10 – Classe Exc1

Atributos	Descrição

# $idv	Atributo protegido utilizado para armazenar uma array contendo todos os índices referentes a cada conjunto de variáveis do modelo
Métodos	**Descrição**
+ x0()	Método público para calcular os valores iniciais das variáveis de estado do modelo, considerando iguais a zero os valores dos diferenciais nas suas equações de estado
+ dFx($input)	Método público para calcular os valores diferenciais das variáveis de estado do modelo. Aqui são escritas as equações diferenciais do modelo

Fonte Elaborado pelo Autor

5.2.3 Pacote NDSE/Tools

O pacote NSE/Tools contém três classes, sendo uma delas abstrata. A Figura 5.5 apresenta o diagrama de classes desse pacote e os relacionamentos existentes.

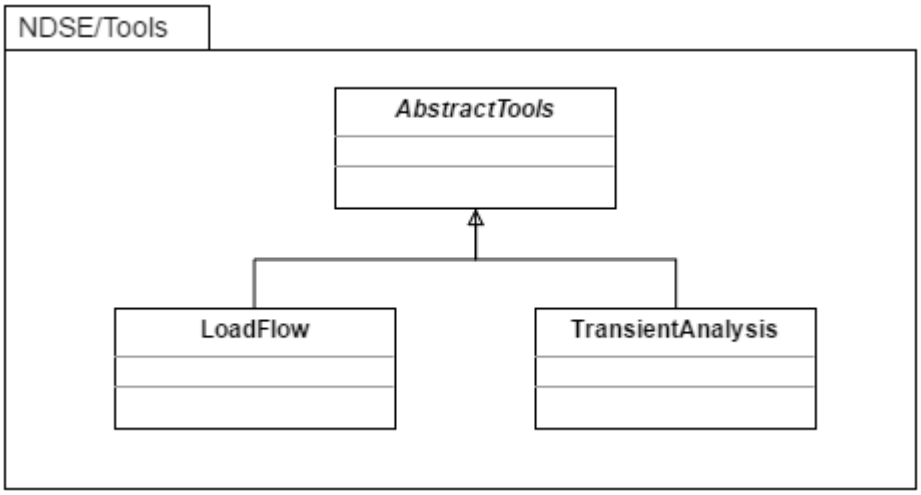


Figura 5.5 – Pacote NDSE/Tools

Fonte: Elaborado pelo autor

5.2.3.1 Classe AbstractTools

A classe AbstractTools é do tipo abstrato e contém atributos e métodos comuns às classes concretas, LoadFlow e TransientAnalysis. Estas derivam da classe abstrata e herdam todos os seus atributos e métodos, implementando obrigatoriamente os seus métodos abstratos cada um a sua maneira.

A Tabela 5.11 apresenta os atributos e os métodos, sendo um deles abstrato, da classe AbstractTools.

Tabela 5.11 – Classe AbstractTools

Atributos	Descrição

# $option	Atributo protegido utilizado para armazenar as opções do tipo de análise a ser realizada
# $data	Atributo protegido utilizado para armazenar todos os dados de entrada do tipo de análise a ser realizada
Métodos	**Descrição**
+ getOption($opt)	Método público que retorna o valor de uma opção do tipo de análise a ser realizada
+ getData($name, $row, $col)	Método público que retorna o valor de um dado de entrada do tipo de análise a ser realizada. Opera com três argumentos, $name, $row e $col
+ setData($name, $value)	Método público para atribuir um novo valor para um dado de entrada do tipo de análise a ser realizada. Opera com dois argumentos, $name e $value
+ *run*()	Método abstrato público que executa um determinado tipo de análise do sistema elétrico

Fonte Elaborado pelo Autor

5.2.3.2 Classe LoadFlow

A classe LoadFlow possui uma relação de composição com as classes Matrix, Sparse e Complex, e de dependência da classe LinAlg (Figura 5.6). Ela tem como função realizar o cálculo do fluxo de potência pelo método de Newton-Raphson. Na Tabela 5.12 são apresentados os atributos e os métodos da classe LoadFlow disponíveis para uso.

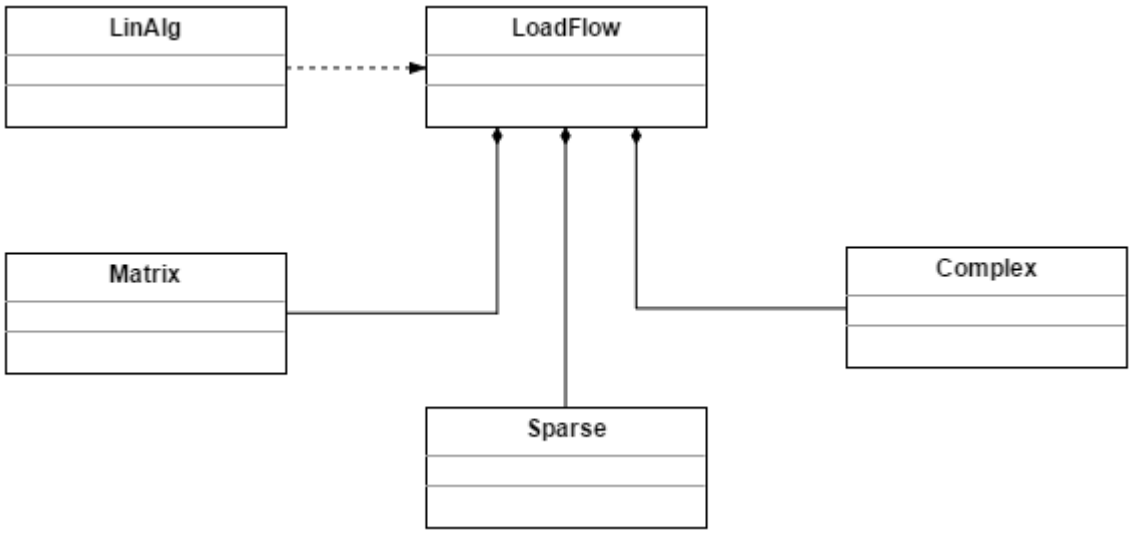


Figura 5.6 – Diagrama de classes com destaque para a classe LoadFlow

Fonte: Elaborado pelo autor

Tabela 5.12 – Classe LoadFlow

Atributos	**Descrição**

# $option	Atributo protegido utilizado para armazenar as opções padrões para a realização do cálculo do fluxo de carga. É utilizado para isso um vetor com os índices 'sbase','max_iter', 'tol' e 'qlim', com os respectivos valores padrões, 100, 10, 1e-3 e 1
# $data	Atributo protegido utilizado para armazenar principalmente os dados das barras e dos ramos
# $Ybus	Atributo protegido utilizado para armazenar a matriz de admitância na forma esparsa
Métodos	**Descrição**
+ getN($name)	Método público que retorna o número de barras, ramos, linhas ou transformadores do sistema elétrico em análise, sendo necessário informar como argumento, respectivamente, a palavra 'bus', 'branch', 'line' ou 'trafo
+ makeYbus()	Método público para construir a matriz de admitância esparsa
+ makeJ($input)	Método público que retorna a matriz Jacobiana esparsa, sendo necessário infomar como argumento uma array do PHP contendo o tamanho dessa matriz, o número de barras PV+PQ, e as tensões nas barras
+ run()	Método público que retorna o resultado do fluxo de carga no formato JSON, contendo o número de iterações, valores das barras, dos ramos e das perdas

Fonte Elaborado pelo Autor

5.2.3.3 Classe TransientAnalysis

A classe TransientAnalaysis possui uma relação de composição com as classes Matrix, Complex, LoadFlow, Gen1, Gen2 e Exc1 e de dependência da classe LinAlg (Figura 5.7). Ela tem como função realizar a análise de estabilidade transitória pelo método de Euler Modificado. Na Tabela 5.13 são apresentados os atributos e os métodos da classe TransientAnalysis disponíveis para uso.

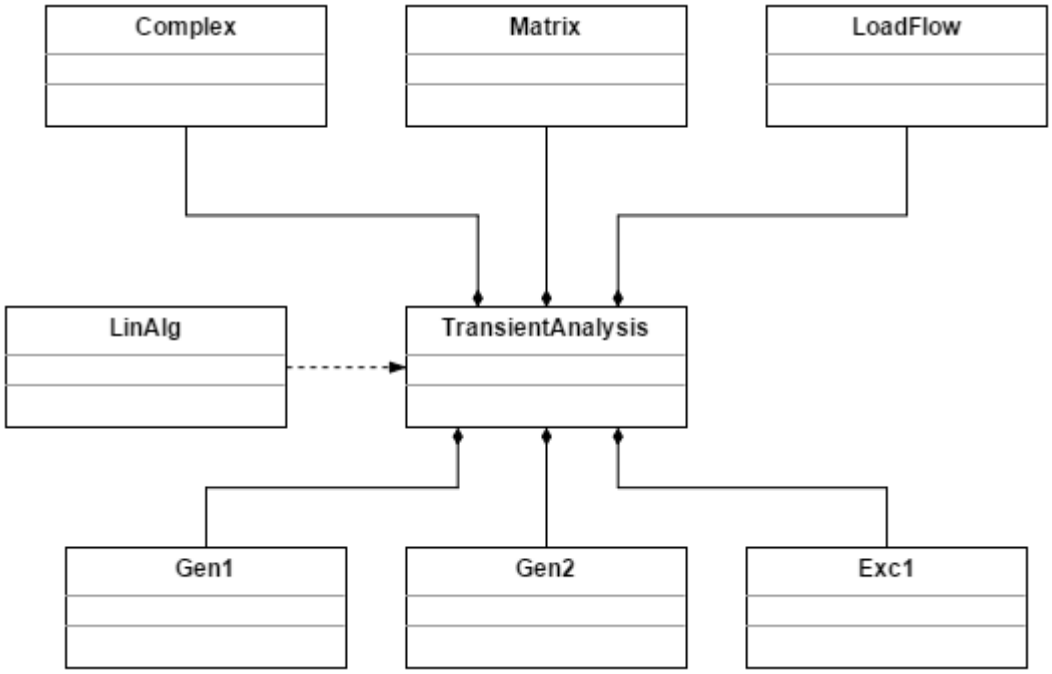


Figura 5.7 – Diagrama de classes com destaque para a classe TransientAnalysis

Fonte: Elaborado pelo autor

Tabela 5.13 – Classe TransientAnalysis

Atributos	Descrição
# $option	Atributo protegido utilizado para armazenar as opções padrões para a realização da análise de estabilidade transitória. É utilizado para isso um vetor com os índices 'freq0', 'starttime', 'stoptime' e 'stepsize', com os respectivos valores, 60, 0, 1 e 1e-3
# $data	Atributo protegido utilizado para armazenar os dados das barras, dos ramos, dos geradores, dos reguladores e dos eventos
# $Ybus	Atributo protegido utilizado para armazenar a matriz de admitância na forma esparsa
Métodos	**Descrição**
+ run()	Método público que retorna o resultado da análise de estabilidade transitória no formato JSON, contendo o número de iterações, valores das barras, dos ramos e das perdas

Fonte Elaborado pelo Autor

5.2.4 Aplicação com Slim Framework

O Slim Framework foi utilizado para implementar em PHP os serviços *web* para o cálculo do fluxo de potência e para a análise da estabilidade transitória, conforme o código apresentado no Quadro 5.1.

Esses serviços foram implementados para serem acessíveis por meio de requisições do tipo POST e pelas seguintes URL's: http://local_da_aplicação/nws/v1/loadflow para o primeiro, e http://local_da_aplicação/nws/v1/stability para o segundo.

Na URL, "local_da_aplicação" identifica o endereço IP de um servidor ou o domínio[6], por exemplo, www.ufu.br. E ainda, "nws" identifica o nome da aplicação, "v1", a sua versão, "loadflow" ou "stability" o tipo de análise.

Quadro 5.1 – Trecho de código com Slim Framework

```
$app->group('/nws/v1',function() use ($app){
        $app->post('/loadflow',function() use ($app){
            $json =  json_decode($app->request->getBody());
            $data = ['data'=>[
                                'optLF' => $json->optLF,
                                'bus' => $json->bus,
                                'branch' => $json->branch
                                ]
                    ];
            $app->response()->header("Content-Type", "application/json");
            $app->render('loadflow.php', $data, 200);
        });

        $app->post('/stability',function() use ($app){
            $json =  json_decode($app->request->getBody());
            $data = ['data'=>[
                                'optLF' => $json->optLF,
                                'optTA' => $json->optTA,
                                'bus' => $json->bus,
                                'branch' => $json->branch,
                                'gen' => $json->gen,
                                'exc' => $json->exc,
                                'gov' => $json->gov,
                                'event' => $json->event
                                ]
                    ];
            $app->response()->header("Content-Type", "application/json");
            $app->render('stability.php', $data, 200);
        });
```

[6] É um nome padrão para acesso de uma página na internet ou intranet

```
});
```

Fonte Elaborado pelo Autor

Os dados de entrada da análise em questão são recebidos pela aplicação no formato JSON e transformados no tipo array do PHP pelo uso da função json_decode(). Em seguida são repassados para um dos arquivos de processamento disponíveis: loadflow.php e stability.php (Quadro 5.2 e Quadro 5.3). Nos dois casos, é verificado se os dados não são nulos, depois é criado um objeto para o tipo de análise requisitado e no fim é executado o método run(). O resultado desse método é armazenado numa variável e seu conteúdo, no formato JSON, é devolvido para o cliente por meio da função echo do PHP.

Quadro 5.2 – Código principal do arquivo loadflow.php

```
if (!is_null($data)) {
        $lf = new LoadFlow($data);
        $lf-> makeYbus();
        $result = $lf->run();
        echo $result;
}
```

Fonte Elaborado pelo Autor

Quadro 5.3 – Código principal do arquivo stability.php

```
if (!is_null($data)) {
        $ta = new TransientAnalysis($data);
        $result = $ta->run();
        echo $result;
}
```

Fonte Elaborado pelo Autor

O arquivo JSON para a entrada de dados para o cálculo do fluxo de potência e para a análise de estabilidade transitória possui os itens especificados na Tabela 5.14 e Tabela 5.15, respectivamente.

Tabela 5.14 – Itens do arquivo JSON contendo os dados de entrada para o cálculo do fluxo e potência

Nome do Item	Especificação
"info"	Tipo de análise, neste caso assinalado pelos caracteres "lf"
"optLF"	Vetor contendo a potência base em MVA, o número máximo de iterações, valor da tolerância, e a indicação, com 1 ou 0, da verificação ou não da violação de reativos
"bus"	Array contendo os dados de cada barra dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra, tipo de barra (1 para barra de carga, 2 para barra de geração, 3 para barra de referência),

	potência ativa gerada em MVA, potência reativa gerada em MVAR, potência ativa na carga em MVA, potência reativa na carga em MVAR, valor da resistência shunt em p.u., valor da reatância shunt em p.u., módulo da tensão na barra em p.u., ângulo da tensão na barra em graus, limite de reativo máximo e mínimo em MVAR
"branch"	Array contendo os dados de cada ramo dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra de origem, número da barra de destino, resistência série m p.u., reatância série em p.u., susceptância paralela total em p.u., valor do TAP do transformador em p.u. e valor do ângulo do transformador defasador em graus. Obs.: os valores dos últimos dados serão 0, caso o ramo seja uma linha de transmissão)

Fonte Elaborado pelo Autor

Tabela 5.15 – Itens do arquivo JSON contendo os dados de entrada para a análise de estabilidade transitória

Nome do Item	Especificação
"info"	Tipo de análise, neste caso assinalado pelos caracteres "ta"
"optLF"	Vetor contendo a potência base em MVA, o número máximo de iterações, valor da tolerância, e a indicação, com 1 ou 0, da verificação ou não da violação de reativos
"optTA"	Vetor contendo a frequência base em Hz, tempo inicial da simulação em s, tempo final da simulação em s, e o valor do passo de integração
"bus"	Array contendo os dados de cada barra dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra, tipo de barra (1 para barra de carga, 2 para barra de geração, 3 para barra de referência), potência ativa gerada em MVA, potência reativa gerada em MVAR, potência ativa na carga em MVA, potência reativa na carga em MVAR, valor da resistência shunt em p.u., valor da reatância shunt em p.u., módulo da tensão na barra em p.u., ângulo da tensão na barra em graus, limite de reativo máximo e mínimo em MVAR
"branch"	Array contendo os dados de cada ramo dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra de origem, número da barra de destino, resistência série m p.u., reatância série em p.u., susceptância paralela total em p.u., valor do TAP do transformador em p.u. e valor do ângulo do transformador defasador em graus. Obs.: os valores dos últimos dados serão 0, caso o ramo seja uma linha de transmissão)
"gen"	Array contendo os dados de cada modelo de gerador disponível, dispostos em um vetor de dez posições. Para o modelo do gerador clássico o vetor deve ser preenchido da seguinte forma e nesta ordem: número da barra onde o gerador está conectado, tipo de gerador (neste caso 1), valores da constante H, D, x_d, x'_d e as demais posições preenchidas com 0. Para o modelo do gerador de 4ª ordem o vetor dever ser preenchido da seguinte forma e nesta ordem: número da barra onde o gerador está conectado, tipo de gerador (neste caso 2), valores da constante H, D, x_d, x_q, x'_d, x'_d, T'd e T'_q. Obs.: Todos valores devem ser fornecidos em p.u.
"exc"	Array contendo os dados de cada modelo de regulador de tensão

	disponível, dispostos em vetores. O vetor deve ser preenchido da seguinte forma e nesta ordem: número da barra onde o gerador está conectado, tipo de regulador (neste caso 1), valores de K_a, T_a, K_e, T_e, K_f, T_f, A_{ex}, B_{ex}, Ur_{min} Ur_{max}. Obs.: somente um modelo foi implementado nesta versão da aplicação
"gov"	Vetor contendo parâmetros do modelo do regulador de velocidade. Obs.: nenhum modelo foi implementado nesta versão da aplicação
"event"	Array contendo os dados de cada evento a ser tratado na simulação, dispostos em vetores e com as informações alocadas na seguinte ordem: tipo de evento (1 para curto-circuito trifásico), número da barra, posição no vetor do parâmetro a ser alterado, valor do instante de tempo em que o evento ocorre, e o novo valor para o parâmetro a ser alterado. Obs.: para esta versão da aplicação está disponível somente o curto-circuito trifásico em uma determinada barra do sistema

Fonte Elaborado pelo Autor

E o arquivo JSON para a disponibilização dos resultados do cálculo do fluxo de potência e da análise de estabilidade transitória possui os itens especificados na Tabela 5.16 e Tabela 5.17, respectivamente.

Tabela 5.16 – Itens do arquivo JSON contendo os resultados do cálculo do fluxo de potência

Nome do Item	Especificação
"iteration"	Número de iterações necessárias até a convergência
"bus"	Array contendo os resultados de cada barra dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra, módulo da tensão na barra em p.u., ângulo da tensão na barra em graus, potência ativa gerada em MW, potência reativa gerada em MVAR, potência ativa na carga em MW, potência reativa na carga em MVAR, limite de reativo máximo e mínimo em MVAR
"branch"	Array contendo os resultados de cada ramo dispostos em vetores e com as informações alocadas na seguinte ordem: número da barra de origem, número da barra de destino, perda de potência ativa em MW no sentido barra de origem-destino, perda de potência reativa em MVAR no sentido barra de origem-destino, perda de potência ativa em MW no sentido barra de destino-origem, perda de potência reativa em MVAR no sentido barra de destino-origem perda de potência ativa em MW na linha e perda de potência reativa em MVAR na linha
"loss"	Vetor contendo a perda total de potência ativa em MW e reativa em MVAR

Fonte Elaborado pelo Autor

Tabela 5.17 – Itens do arquivo JSON contendo os resultados da análise de estabilidade transitória

Nome do Item	Especificação
"t"	Vetor contendo todos os instantes de tempo no intervalo inicial e final da simulação
"delta"	Vetor contendo todos os valores instantâneos do ângulo delta de cada gerador no intervalo de tempo da simulação

“volt”	Vetor contendo todos os valores instantâneos da tensão na barra de cada gerador no intervalo de tempo da simulação
“omega”	Vetor contendo todos os valores instantâneos da velocidade angular de cada gerador no intervalo de tempo da simulação
“pmec”	Vetor contendo todos os valores instantâneos da potência mecânica de cada gerador no intervalo de tempo da simulação
“efd”	Vetor contendo todos os valores instantâneos da tensão de campo de cada gerador no intervalo de tempo da simulação

Fonte Elaborado pelo Autor

5.3 Aplicações clientes

As aplicações criadas para o lado cliente foram escritas utilizando HTML, CSS e a linguagem JavaScript. Três tipos de aplicações clientes foram implementadas, com diferenças relacionadas a aspectos visuais e número de recursos. Isso foi necessário para realizar os testes básicos das aplicações criadas no servidor, e também para analisar e testar as bibliotecas JavaScript selecionadas, a fim de utilizá-las de forma apropriada e integrada.

5.3.1 Cliente 1

Nesta versão de cliente foram produzidas duas interfaces semelhantes (Figura 5.8 e Figura 5.9), uma vinculada à ferramenta de fluxo de potência e outra para a análise de estabilidade transitória. Os resultados são apresentados apenas no formato JSON no lado direito da tela por meio do elemento HTML div invisível, mas identificado no código com o id “result”.

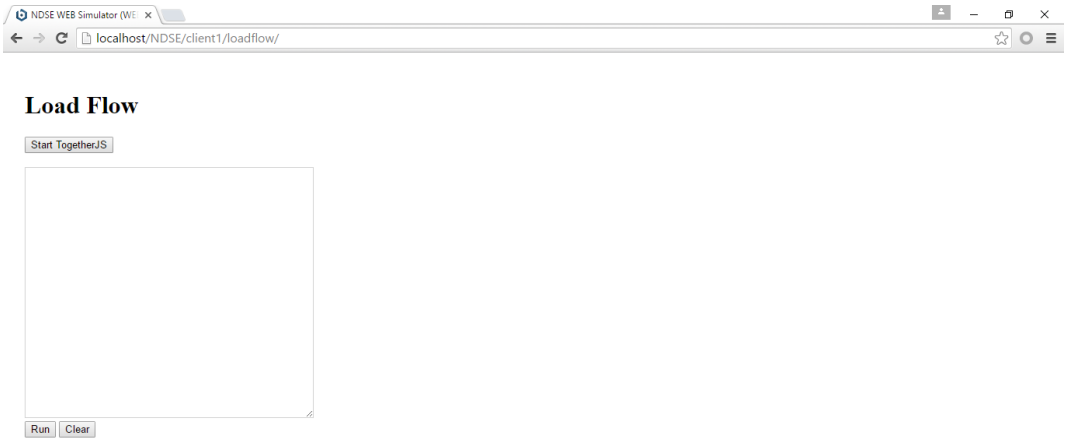


Figura 5.8 – Tela da aplicação cliente 1 para o cálculo do fluxo de potência

Fonte Elaborado pelo Autor

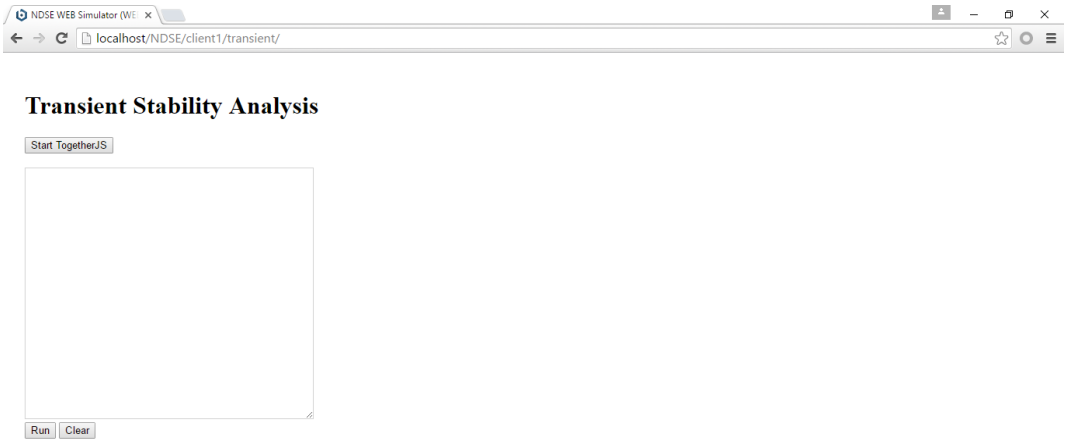


Figura 5.9 – Tela da aplicação cliente 1 para a análise de estabilidade transitória

Fonte Elaborado pelo Autor

Nessas interfaces para a execução da simulação e a entrada dos dados foram utilizadas, respectivamente, dois elementos HTML: button e textarea, sendo este identificado com o id "data".

Os botões nomeados de Start TogetherJS, Run e Clear estão vinculados por meio de código JavaScript com a inicialização da ferramenta de colaboração, à execução da simulação e a limpeza da área de texto, respectivamente. Os dois últimos botões estão ainda identificados com o id "post" e "clear", respectivamente.

No caso do botão Run, o tratamento da ação de clicar foi vinculado a um código JavaScript no padrão jQuery (Quadro 5.4 – Tratamento da ação de clicar no botão RunQuadro 5.4). Esse código executa o envio ao servidor os dados de entrada da simulação por meio de uma requisição do tipo POST utilizando AJAX.

Quadro 5.4 – Tratamento da ação de clicar no botão Run

```
$('#post').click(function () {
	$.ajax({
		type: 'POST',
		url: 'http://localhost/nws/v1/loadflow',
		data: $('#data').val(),
		contentType: 'text/plain',
		success: function (result) {
			$("#result").html("<pre>"+JSON.stringify(result, null, 2)+"</pre>");
		},
		error: function (resp) {
			$("#result").html(resp.responseText);
		}
	});
```

```
});
```

Fonte Elaborado pelo Autor

O código para executar a análise de estabilidade transitória é semelhante e está listado no Quadro 5.5.

Quadro 5.5 – Trecho do código JavaScript para "rodar" a análise de estabilidade transitória

```
$('#post').click(function () {
	$.ajax({
		type: 'POST',
		url: 'http://localhost/nws/v1/stability',
		data: $('#datalf').val(),
		contentType: 'text/plain',
		success: function (result) {
			$("#result").html("<pre>"+JSON.stringify(result, null, 2)+"</pre>");
		},
		error: function (resp) {
			$("#result").html(resp.responseText);
		}
	});
});
```

Fonte Elaborado pelo Autor

A limpeza da área de texto por meio do clique no botão Clear seguiu o padrão de código jQuery apresentado no Quadro 5.6

Quadro 5.6 – Tratamento da ação de clicar no botão Clear

```
$('#clear').click(function () {
	$('#data').val('');
	$("#result").empty();
});
```

Fonte Elaborado pelo Autor

5.3.2 Cliente 2

Nesta versão de cliente foi produzida como modelo uma interface somente para a ferramenta de fluxo de potência. Nesta interface além dos elementos HTML básicos da versão 1 são utilizadas as tabelas da biblioteca DHTMLX (Figura 5.10), onde cada uma das células pode ser editada, menos a da tabela de resultados das barras e dos ramos.

Os botões nomeados de Start TogetherJS e Run LoadFlow estão vinculados por meio de código JavaScript com a inicialização da ferramenta de colaboração e à execução da simulação, respectivamente.

No caso do botão Run LoadFlow, o tratamento da ação de clicar foi vinculado a um código JavaScript no padrão jQuery, semelhante ao código apresentado no Quadro 5.4 do cliente 1. Assim, o usuário ao clicar nesse botão inicia o envio dos dados de entrada da simulação, presentes nas tabelas, ao servidor por meio de uma requisição do tipo POST utilizando AJAX.

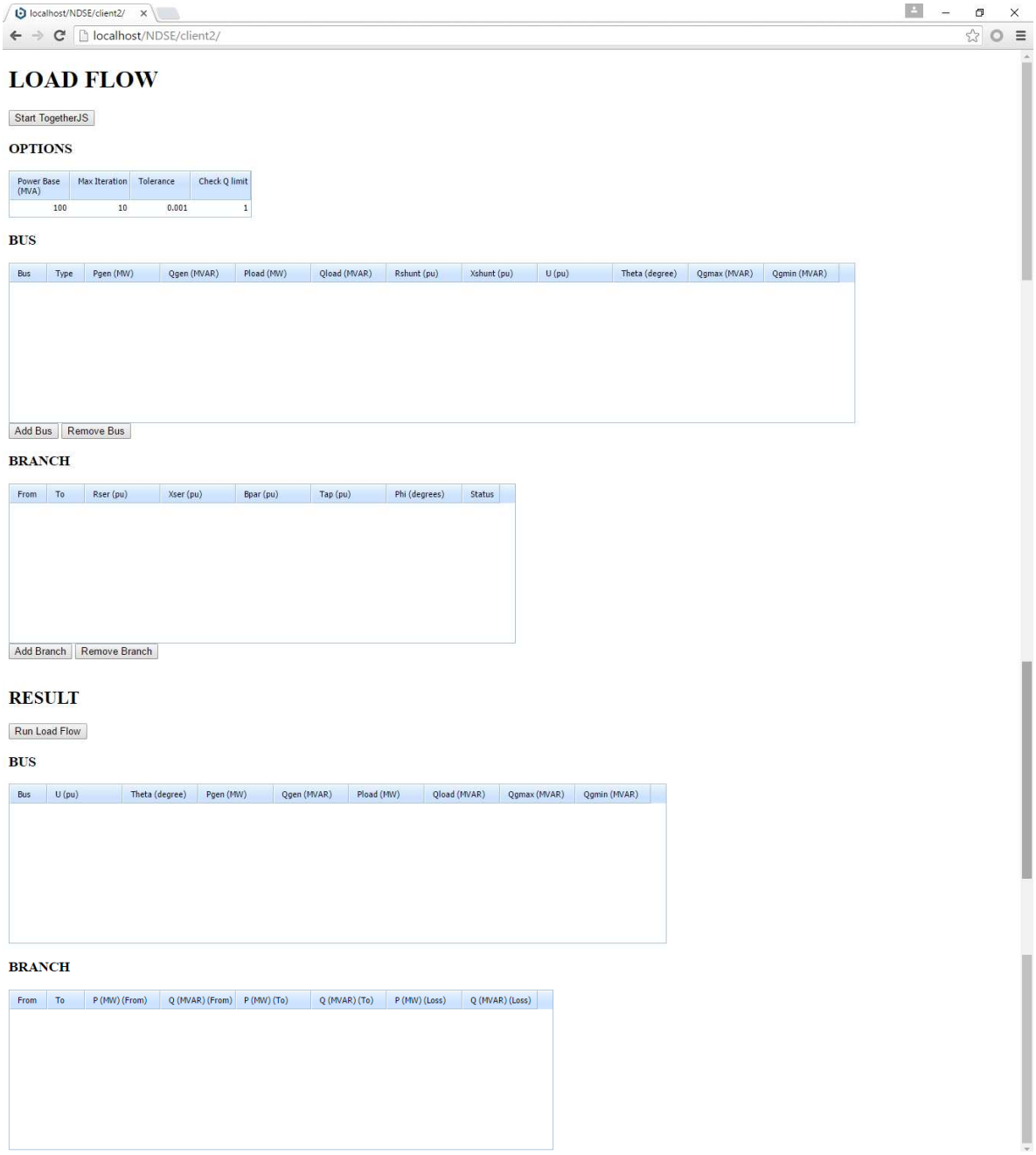


Figura 5.10 – Tela da aplicação cliente 2 para o cálculo do fluxo de potência

Fonte Elaborado pelo Autor

O código JavaScript do cliente 2, elaborado de forma procedimental, contém alguns conjuntos de funções com atribuições e implementações semelhantes, sendo por isso agrupadas da seguinte forma:

- Funções para a criação das tabelas de opções da simulação, de dados das barras e de ramos, e de resultados (Quadro 5.7);
- Funções para a adição e remoção de linhas das tabelas que contém dados de barras e de ramos (Quadro 5.8);
- Funções para mudança de opções de simulação e valores de dados das barras e de ramos (Quadro 5.9).

Quadro 5.7 – Funções para a criação de algumas tabelas

```
function createOptionGrid(name)
function createBusGrid(name)
function createBranchGrid(name)
function createBusResult(name)
```

Fonte Elaborado pelo Autor

Quadro 5.8 – Funções para a adição e remoção de linhas de algumas tabelas

```
function addBus(id,send)
function addBranch(id,send)
function removeBus(id,send)
function removeBranch(id,send)
```

Fonte Elaborado pelo Autor

Quadro 5.9 – Funções para a mudança dos valores das células de algumas tabelas

```
function changeOptionCell(row,col,value,send)
function changeBusCell(row,col,value,send)
function changeBranchCell(row,col,value,send)
```

Fonte Elaborado pelo Autor

Além dessas, há ainda funções para o carregamento e a atualização dos dados nas tabelas, e códigos escritos para a criação de mensagens personalizadas conforme especificações da biblioteca TogetherJS. Estas estão agrupadas da seguinte forma:

- Mensagens iniciais quando da conexão de novos usuários (Quadro 5.10);
- Mensagens para adição e remoção de dados de barras e de ramos (Quadro 5.11);
- Mensagens para alteração de opções de simulação e valores dos dados de barras e de ramos (Quadro 5.12).

Essas mensagens são importantes para que as alterações realizadas por um usuário sejam informadas para todos os usuários conectados. É dessa forma que todos podem visualizar o mesmo conteúdo, mesmo a distância e utilizando dispositivos diferentes.

Quadro 5.10 – Rotinas para configurações de mensagens iniciais

```
TogetherJS.hub.on('togetherjs.hello', function () {…})
TogetherJS.hub.on('init', function (msg) {…})
```

Fonte Elaborado pelo Autor

Quadro 5.11 – Rotinas para configurações de mensagens para adição e remoção dos dados de barras e ramos

```
TogetherJS.hub.on('addBus', function (msg) {…})
TogetherJS.hub.on('addBranch', function (msg) {…})
TogetherJS.hub.on('removeBus', function (msg) {…})
TogetherJS.hub.on('removeBranch', function (msg) {…})
```

Fonte Elaborado pelo Autor

Quadro 5.12 – Rotinas para configurações de mensagens para atualização de opções, dados de barras e ramos

```
TogetherJS.hub.on('changeOptionCell', function (msg) {…})
TogetherJS.hub.on('changeBusCell', function (msg) {…})
TogetherJS.hub.on('changeBranchCell', function (msg) {…})
```

Fonte Elaborado pelo Autor

5.3.3 Cliente 3

A aplicação cliente 3 foi construída para a reprodução de uma interface com aparência de *desktop*. Esta interface conta com componentes da biblioteca DHTMLX para o seu aspecto visual, e com a biblioteca Draw2D *touch* para a criação e manipulação de elementos gráficos representativos de cinco componentes básicos do sistema elétrico: barra (BUS), gerador (GEN), transformador (TRAFO), linha de transmissão (LINE) e carga (LOAD).

Essas bibliotecas foram utilizadas em conjunto com o pacote cliente NDSE escrito em JavaScript. A Figura 5.11 apresenta o diagrama de pacotes e os relacionamentos de dependência existente, identificados pela seta tracejada.

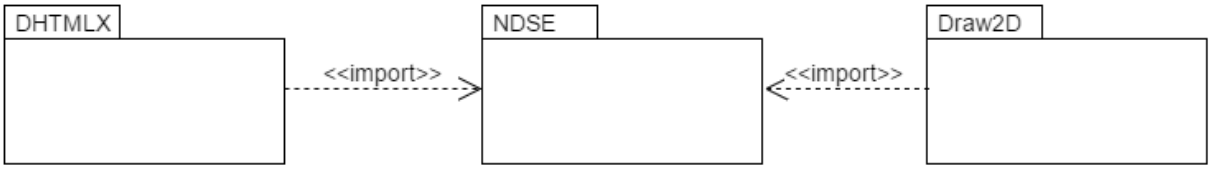


Figura 5.11 – Pacote NDSE cliente

Fonte: Elaborado pelo autor

O pacote NDSE contém cinco classes:

- application – Tem como função a inicialização da aplicação, com o carregamento do layout e dos componentes. Possui uma relação de composição com as classes layout e componet;
- layout – Tem como função a configuração do layout da aplicação. Possui uma relação de composição com as classes canvas e property;
- component – Tem como função a construção dos elementos gráficos no formato SVG[7];
- canvas – Tem como função o gerenciamento da área onde os elementos gráficos são adicionados e manipulados;
- property - Tem como função o gerenciamento das configurações de propriedades dos elementos gráficos adicionados;

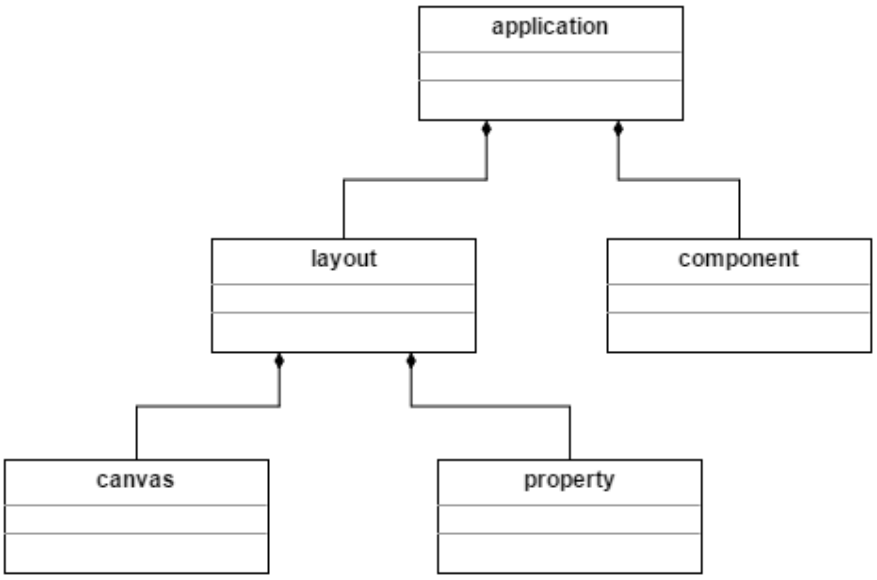


Figura 5.12 – Diagrama de classes do cliente 3

Fonte: Elaborado pelo autor

[7] É um padrão aberto para a descrição de desenhos e gráficos vetoriais na linguagem XML. Umas das principais características dos gráficos vetoriais, é que não perdem qualidade ao serem ampliados.

Os códigos escritos utilizando a biblioteca DHTMLX tornaram possível a construção de aplicação visualmente semelhante a uma aplicação *desktop*. A tela principal da aplicação e os menus disponibilizados podem ser vistos na Figura 5.13 e Figura 5.14.

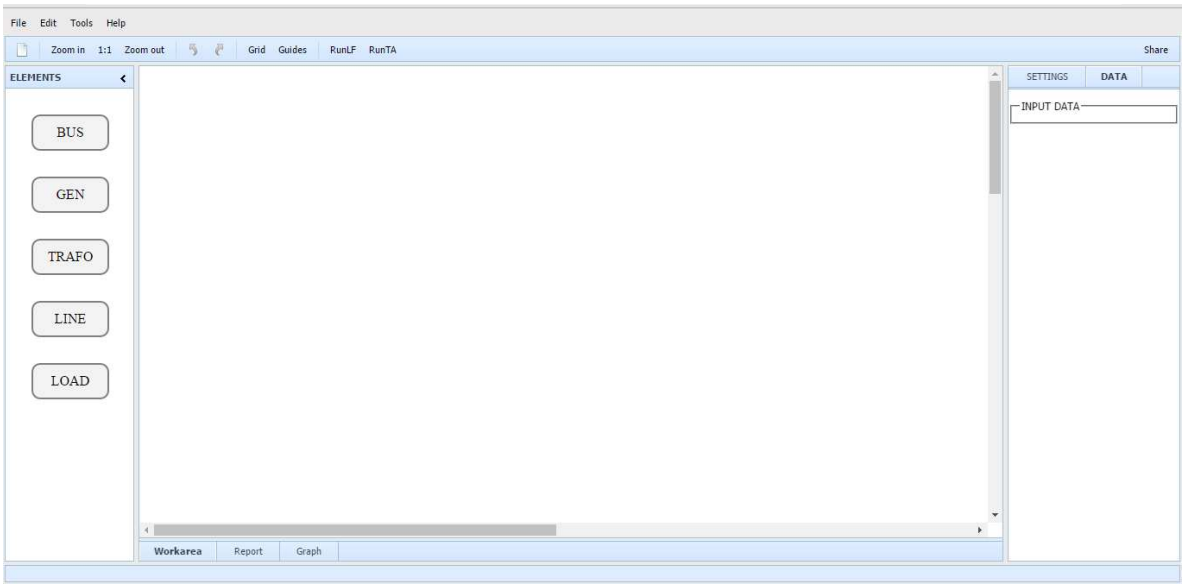


Figura 5.13 – Tela da aplicação cliente 3

Fonte: Elaborado pelo Autor

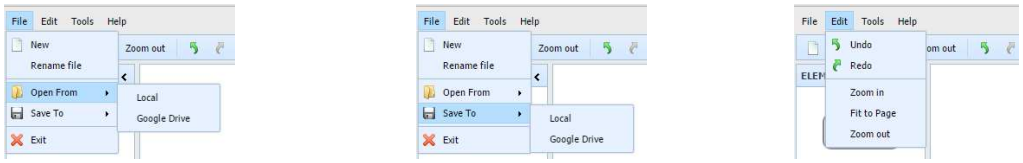


Figura 5.14 – Menus da aplicação cliente 3

Fonte: Elaborado pelo Autor

Nesta versão de cliente foi possível apenas implementar as seguintes características:

- Adição de qualquer um dos cinco elementos gráficos disponíveis por meio de "arraste e solte" (*drag and drop*);
- Ferramenta de zoom;
- Ativação de uma grade retangular de referência na área de adição dos elementos gráficos;
- Ativação de linhas guias na área de adição dos elementos gráficos;
- Armazenamento ou abertura local ou no Google Drive do arquivo JSON contendo o sistema elaborado.

5.4 Considerações Finais

Em resumo foi apresentado neste capítulo todo a concepção do NWS. Como visto, trata-se de uma aplicação *web* formada por um conjunto de pequenas aplicações, umas localizadas no servidor e outras no cliente.

As aplicações do servidor implementadas em PHP disponibilizam o cálculo do fluxo de potência e a análise de estabilidade transitória como serviços *web*. Para isso foi utilizado o Slim framework que tornou bastante simples essa implementação.

Das três aplicações clientes desenvolvidas, duas são funcionais e demonstram muito bem os recursos disponíveis no NWS. A aplicação cliente 3 carece ainda de um maior tempo de programação para a sua finalização e a inclusão de melhorias, entre elas:

a) A especificação e a atualização dos valores de cada elemento do sistema elaborado na interface gráfica;
b) A geração do arquivo de dados de entrada a partir do sistema genérico elaborado na interface gráfica, para que possa ser enviado ao servidor que irá realizar a simulação desejada;
c) A exibição dos resultados da simulação, na forma de tabelas e/ou gráfico das variáveis analisadas.

No próximo capítulo serão apresentados e analisados os testes computacionais realizados com o NWS para demonstrar a sua confiabilidade e deixar mais claros a modo de operação dos principais recursos disponíveis.

Capítulo 6

TESTES COMPUTACIONAIS E ANÁLISE DOS RESULTADOS

6.1 Considerações Iniciais

São apresentadas neste capítulo os testes computacionais realizados. Todos eles foram realizados em um notebook com processador Intel Core i3 de 2.2GHz e 4GBytes de memória RAM, e sistema operacional Windows 10.

Apesar de ser uma aplicação *web* os testes realizados localmente num único computador são suficientes para atestar o seu funcionamento, pois o servidor web, os navegadores e os protocolos são os mesmos caso os testes tivessem sido realizados utilizando a estrutura disponível na internet.

Assim foram testadas as aplicações para o cálculo do fluxo de potência e para a análise de estabilidade transitória, e também as aplicações clientes.

6.2 Teste da aplicação para o cálculo do fluxo de potência

A aplicação desenvolvida em PHP para o cálculo do fluxo de potência no NWS foi testada para os sistemas de 9, 57 e 118 barras do IEEE. Os dados de entrada utilizados para a realização dos testes encontram-se listados no Apêndice A.

Os resultados foram comparados com os do MATPOWER considerando os mesmos parâmetros de simulação apresentados na Tabela 6.1, e utilizando os seus próprios arquivos de testes disponíveis, que seguem praticamente os mesmos dados adotados neste teste

Tabela 6.1 – Parâmetros padrões de simulação para o cálculo do fluxo de potência

Potência base (MVA)	Número máximo de iterações	Tolerância	Método de solução
100	10	1e-3	Newton-Raphson

Fonte Elaborado pelo Autor

Todos os testes realizados resultaram na obtenção de valores coincidentes para as variáveis elétricas vinculadas às barras e ramos. Houve apenas diferenças nos tempos de execução e divergências nos valores das perdas de potências reativas totais e parciais nos ramos relativos às linhas de transmissão. Essas divergências são mostradas no sub-tópico 6.4.2 deste capítulo na Figura 6.10 e Figura 6.15 e estão relacionadas a forma de cálculo realizada pelo MATPOWER, pois as perdas de potências ativas totais, parciais e individuais em cada ramo foram coincidentes.

A Tabela 6.2 contém uma comparação entre o MATPOWER e o NWS quanto ao número de iterações necessárias para a convergência e os menores tempos de simulação obtidos.

Tabela 6.2 – Comparação entre MATPOWER e NWS

Nº de barras	MATPOWER		NWS	
	nº de iterações	tempo de simulação (s)	nº de iterações	tempo de simulação (s)
9	3	0,01	3	0,14
57	2	0,01	2	10,182
118	2	0,02	2	45,578

Fonte Elaborado pelo Autor

As diferenças nos tempos de simulação são bem expressivas. A principal razão disso é porque no NWS a maioria das rotinas matemáticas utilizadas não são nativas do PHP. O MATPOWER, apesar de seu código ter sido escrito numa linguagem de *script*, utiliza muitas rotinas matemáticas prontas, que foram devidamente compiladas para o uso no MATLAB. Isso garante um melhor desempenho de execução do que uma aplicação com muita parte de seu código escrito em linguagem de *script*. Porém, como visto anteriormente, é possível gerar

as chamadas extensões que são rotinas geralmente escritas nas linguagens C ou C++ e compiladas para serem utilizadas pelo interpretador do PHP.

6.3 Teste da aplicação para a análise de estabilidade transitória

A aplicação desenvolvida em PHP para a análise de estabilidade transitória no NWS foi testada para o sistema de 9 barras do IEEE considerando um curto-circuito trifásico na barra 2, durante o intervalo de 0,2 a 0,3 s. Os dados de entrada utilizados para a realização do teste encontram-se listados no Apêndice A.

Os resultados encontrados são fornecidos na forma de gráficos gerados no navegador utilizando a biblioteca em jqPlot e comparados com os do MatDyn. Todos eles indicam que o sistema simulado atinge uma nova condição estável de operação após a retirada do curto-circuito. Para a execução do teste foram considerados os mesmos parâmetros de simulação apresentados na Tabela 6.3 e utilizando os seus próprios arquivos de testes disponíveis, que seguem praticamente os mesmos dados adotados neste teste.

Tabela 6.3 – Parâmetros padrões de simulação para a análise de estabilidade transitória

Frequência base (Hz)	Tempo inicial (s)	Tempo final (s)	Passo de integração	Método de solução
60	0	14	1e-2	Euler modificado (Trapezoidal implícito)

Fonte Elaborado pelo Autor

As Figuras 6.1 a 6.5 apresentam os gráficos gerados relativos aos três geradores síncronos presentes no sistema de 9 barras. Nessas figuras os gráficos relativos ao MatDyn estão dispostos à esquerda e ao NWS à direita.

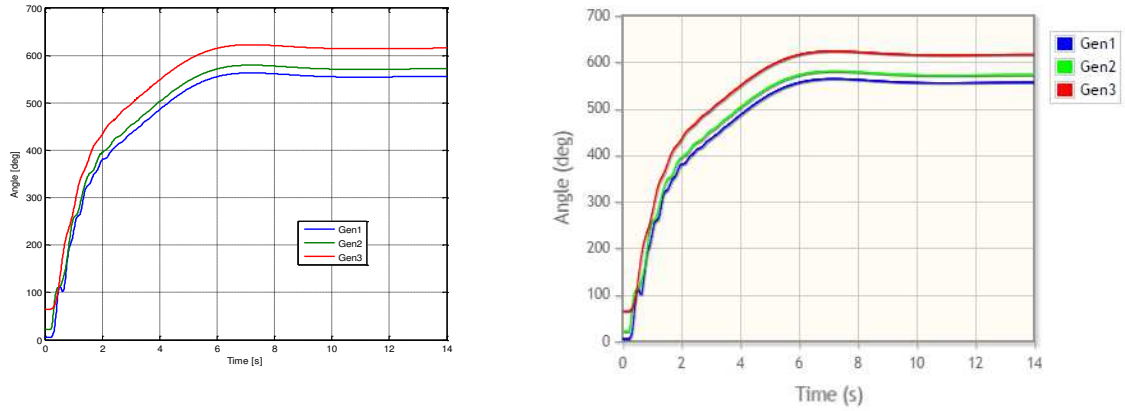


Figura 6.1 – Ângulos Delta dos geradores

Fonte Elaborado pelo Autor

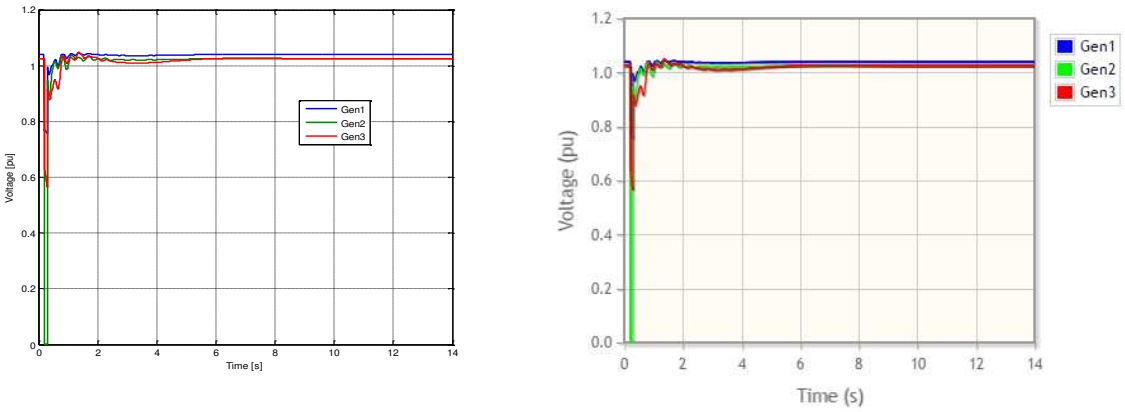


Figura 6.2 – Tensões terminais dos geradores

Fonte Elaborado pelo Autor

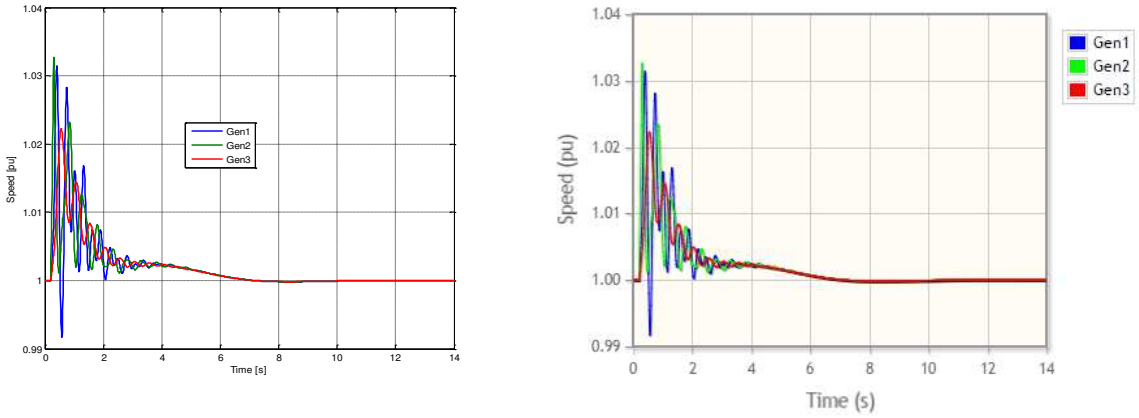


Figura 6.3 – Velocidades angulares (ω) dos rotores dos geradores

Fonte Elaborado pelo Autor

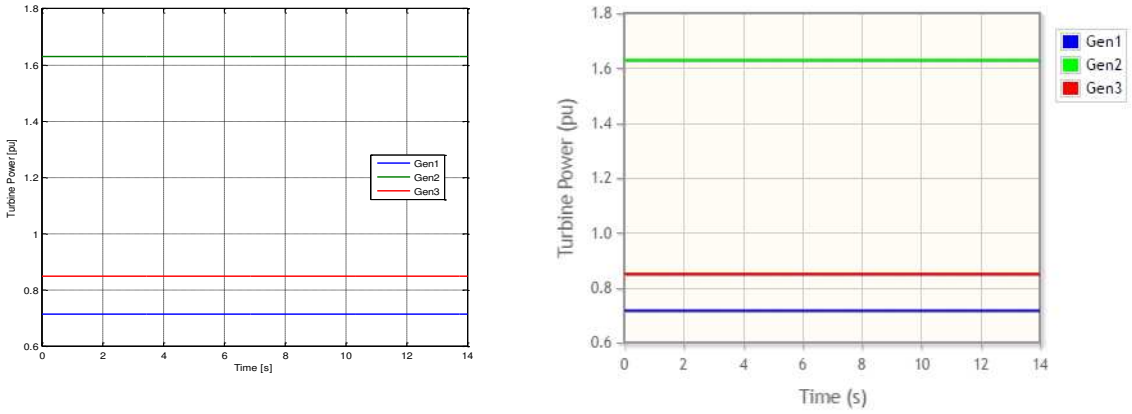


Figura 6.4 – Potência mecânica (P_m) dos geradores

Fonte Elaborado pelo Autor

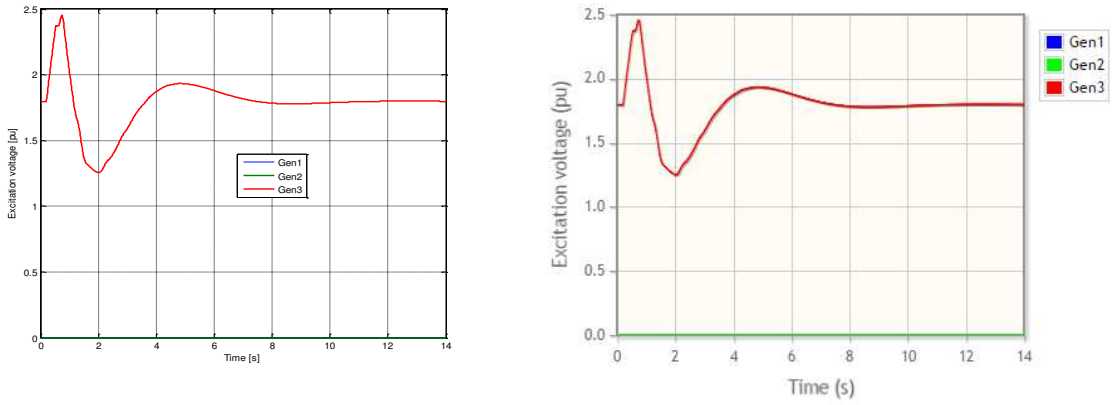


Figura 6.5 – Tensões de campo (E_{fd}) dos geradores

Fonte Elaborado pelo Autor

De modo a comparar quantitativamente todos esses gráficos, e tomando os do MatDyn como referência, foi obtido para a Figura 6.4 o erro percentual absoluto, Equação 6.1 (KHOURY, 2011), e para cada uma das demais figuras foi calculado o erro percentual absoluto médio (MAPE), Equação 6.2 (KHOURY, 2011). No caso da Figura 6.4 só foi utilizado essa métrica pois todos os valores obtidos foram constantes em todo o intervalo da simulação. E para a Figura 6.5 foi calculado o MAPE somente para o gerador 3 (Gen3), pois os demais estão especificados pelo modelo clássico, que no programa sempre possui tensão de campo igual a zero.

$$err_{\%} = \left| \frac{X_i - X_i^{'}}{X_i} \right| \cdot 100 \tag{6.1}$$

$$MAPE = \frac{\sum_{i=1}^{n} err_{\%}}{n} \tag{6.2}$$

Onde:

i – instante de tempo;

n – número total de instantes de tempo;

X_i – Valor real ou de referência no instante i;

$X_i^{'}$ – Valor previsto ou calculado no instante i.

Os valores obtidos encontram-se na Tabela 6.4.

Tabela 6.4 – Valores de MAPE e erro percentual absoluto

	Gerador 1 (GEN1)	**Gerador 2 (GEN2)**	**Gerador 3 (GEN3)**
	MAPE (%)		
Ângulos Delta	$1,7x10^{-3}$	$1,4x10^{-3}$	$1,2x10^{-3}$
Tensões terminais	$0,77x10^{-4}$	$79x10^{-4}$	$1,48\ x10^{-4}$
Velocidades angulares	$1,17x10^{-4}$	$0,42x10^{-4}$	$0,14\ x10^{-4}$
Tensões de campo	-	-	$3,53x10^{-4}$
	Erro Percentual Absoluto (%)		
Potência mecânica	$4x10^{-15}$	$0,13\ x10^{-15}$	$0,91x10^{-15}$

Fonte Elaborado pelo Autor

É possível verificar visualmente e quantitativamente que existe uma ótima correlação entre as curvas gráficas para a mesma grandeza e mesma máquina. Os resultados obtidos validam a aplicação escrita em PHP para a análise de estabilidade transitória.

6.4 Teste das aplicações clientes

Uma vez que os resultados produzidos pelas aplicações para o cálculo do fluxo de potência e para a análise de estabilidade transitória foram validados, os testes das aplicações clientes tiveram como foco a verificação de somente dois aspectos:

- a comunicação com o servidor, para o envio e o recebimento dos dados no formato JSON;
- os recursos disponíveis na interface como a colaboração em tempo real;

6.4.1 Cliente 1

Esta aplicação é a mais simples de todas e foi construída para facilitar o envio e a visualização dos dados recebidos do servidor. Com isso foi possível realizar de uma forma mais prática os primeiros testes das aplicações hospedadas no servidor. Além disso, serviu como base para a análise e a configuração da aplicação de colaboração em tempo real, a biblioteca TogetherJS.

A Figura 6.6 e Figura 6.7 apresentam, respectivamente, as telas do cálculo do fluxo de carga e da análise de estabilidade transitória após o recebimento dos resultados das simulações similares realizadas com o sistema de 9 barras.

Figura 6.6 – Tela da aplicação cliente 1 realizando o cálculo do fluxo de carga

Fonte Elaborado pelo Autor

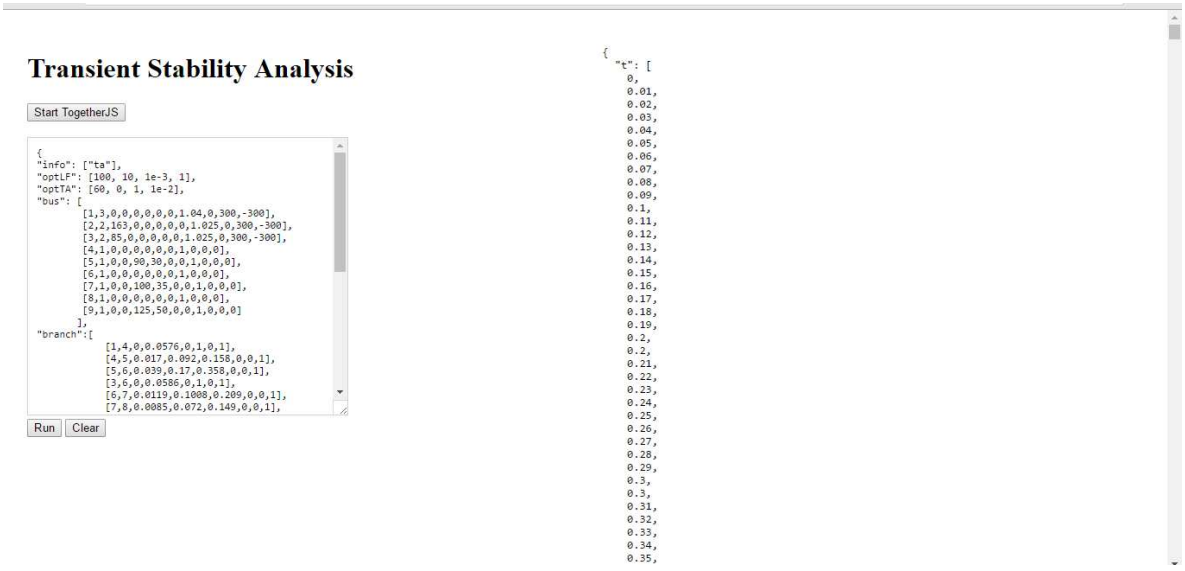


Figura 6.7 – Tela da aplicação cliente 1 realizando a análise de estabilidade transitória

Fonte Elaborado pelo Autor

É uma interface bastante simples, mas que dificulta a visualização dos resultados, principalmente da análise de estabilidade transitória, pois são listados somente os pontos dos gráficos a serem plotados. Porém foi importante para a realização dos testes e validação dos resultados das aplicações no servidor, e também para o primeiro contato e uso da ferramenta de colaboração em tempo real.

A Figura 6.8 e Figura 6.9 mostram essa ferramenta funcionando de forma simultânea em dois navegadores diferentes, Google Chrome à esquerda e Mozilla Firefox à direita. Na primeira figura os dados contidos na área de texto na janela do Chrome são editados por um usuário e as modificações realizadas aparecem automaticamente na janela do Firefox utilizado por outro usuário. Já na segunda figura é mostrada o uso da aplicação de mensagens de texto. Nesse cliente foi necessário somente incluir no código HTML a biblioteca do TogetherJS e o botão para iniciar a sua execução. Essa biblioteca já vem preparada para o envio de mensagens referentes às modificações efetuadas nos elementos HTML, como áreas de texto, não sendo necessário a escrita de nenhum código JAVASCRIPT adicional.

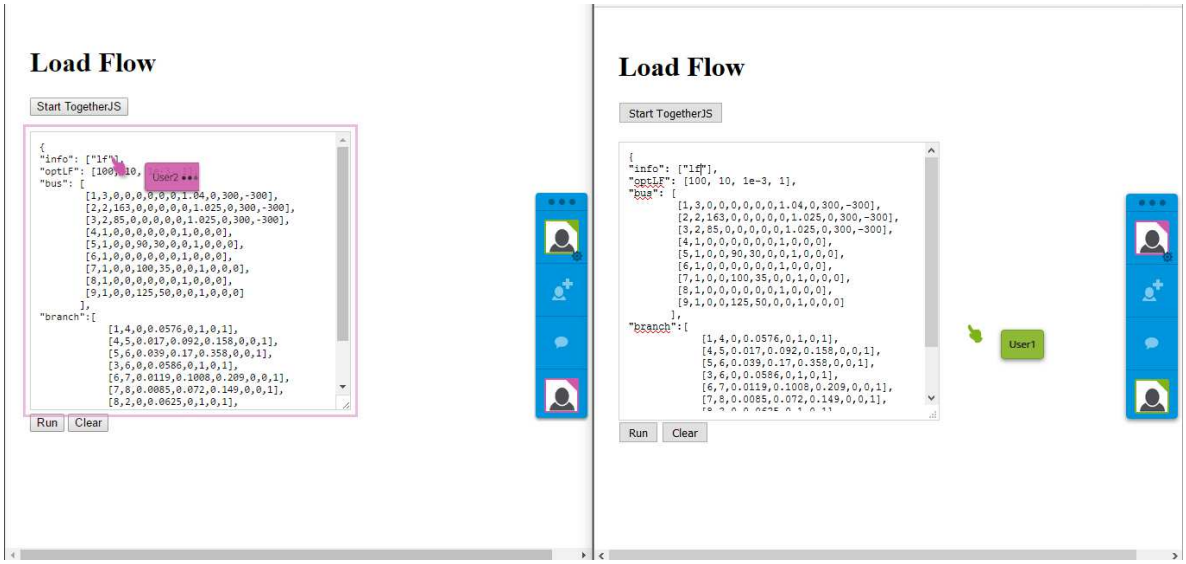


Figura 6.8 – Tela da aplicação cliente 1 com a colaboração em tempo real em execução

Fonte: Elaborado pelo Autor

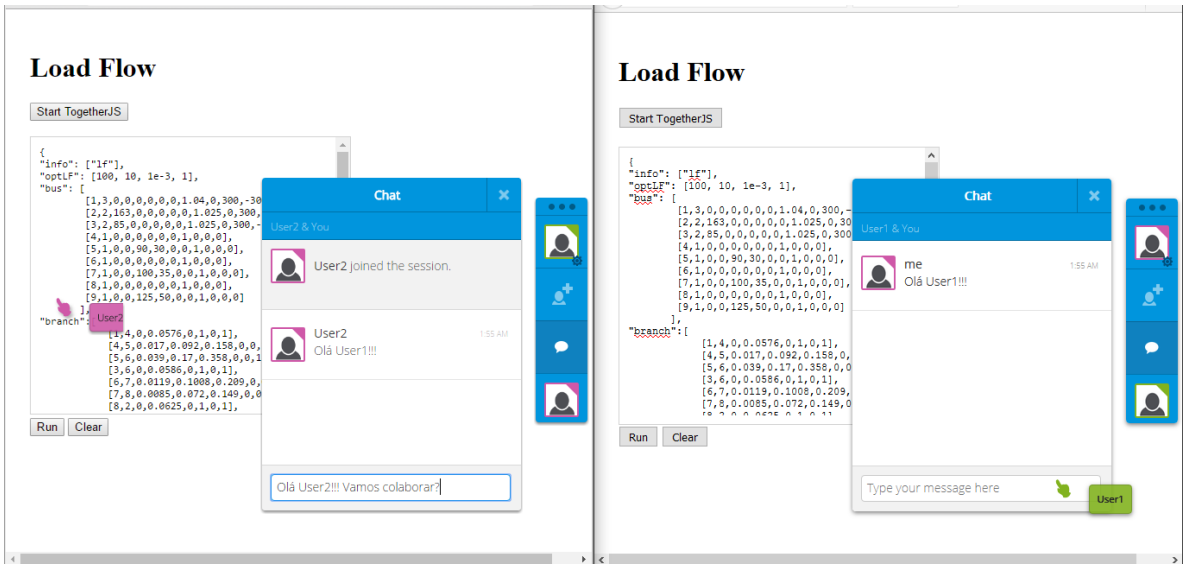


Figura 6.9 – Tela da aplicação cliente 1 com o recurso de mensagens de texto em execução

Fonte: Elaborado pelo Autor

6.4.2 Cliente 2

Esta segunda aplicação, em relação a primeira, trouxe melhorias na forma da iteração do usuário. Neste cliente os dados de entrada e saída foram colocados em tabelas da biblioteca DHTMLX de modo a facilitar a edição e a visualização das informações. Em contrapartida foi necessária a escrita de código JAVASCRIPT adicional para que a ferramenta de colaboração em tempo real tivesse sua utilidade.

A Figura 6.10 e Figura 6.11 apresentam os resultados da execução do cálculo do fluxo de potência para o sistema de 9 barras, respectivamente, no NWS e no MATPOWER. Seguindo essa mesma sequência, a Figura 6.12 e Figura 6.13 mostram os nove primeiros resultados para o sistema de 57 barras, e a Figura 6.14 e Figura 6.15 trazem os nove primeiros resultados para o sistema de 118 barras

Run Load Flow

Number of iterations: 3

BUS

Bus	U (pu)	Theta (degree)	Pgen (MW)	Qgen (MVAR)	Pload (MW)	Qload (MVAR)	Qgmax (MVAR)	Qgmin (MVAR)
1	1.04	0	71.641021474482	27.045923533492	0	0	300	-300
2	1.025	9.2800054816428	163	6.653620705352	0	0	300	-300
3	1.025	4.6647513331368	85	-10.859733267893	0	0	300	-300
4	1.025788392844	-2.2167877999498	0	0	0	0	0	0
5	1.0126543240178	-3.6873961701571	0	0	90	30	0	0
6	1.0323529490024	1.9667160744491	0	0	0	0	0	0
7	1.0158825836275	0.72753607687428	0	0	100	35	0	0
8	1.0257693723865	3.7197011546217	0	0	0	0	0	0
9	0.99563085804829	-3.9888052728515	0	0	125	50	0	0

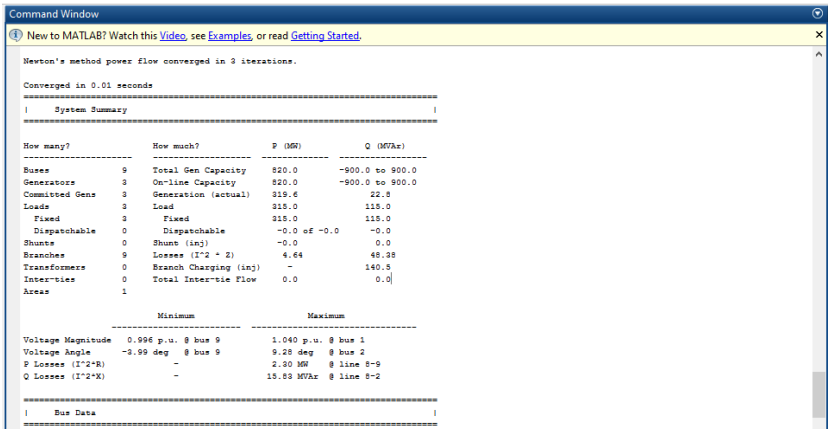
BRANCH

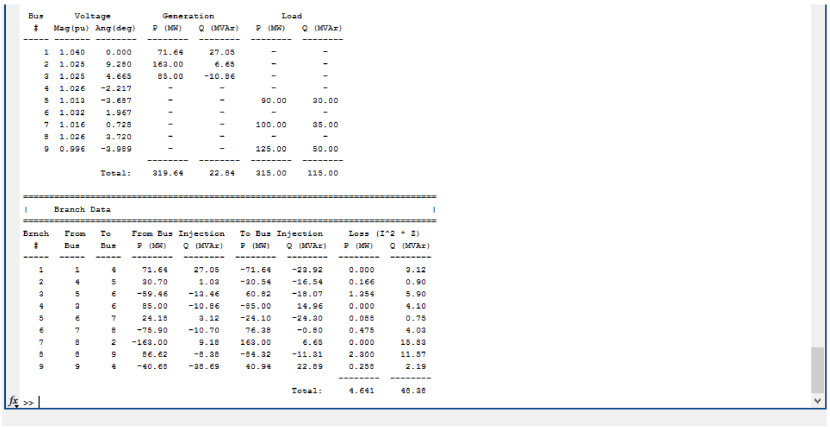
Total loss: P (MW) = 4.6410214744829 Q (MW) = -92.160125219065

From	To	P (MW) (From)	Q (MVAR) (From)	P (MW) (To)	Q (MVAR) (To)	P (MW) (Loss)	Q (MVAR) (Loss)
1	4	71.641021474482	27.045923533492	-71.641021474482	-23.92312699863	0	3.1227965348628
4	5	30.703669762308	1.0300063738835	-30.53726287414	-16.54336524376	0.16640688816783	-15.513358869876
5	6	-59.46273712586	-13.45663475624	60.816585977109	-18.074835718896	1.3538488512493	-31.531470475135
3	6	85	-10.859709070989	-85	14.955327300832	0	4.0956182298426
6	7	24.183414022891	3.1195084180639	-24.095417457393	-24.295822611685	0.08799656549861	-21.176314193621
7	8	-75.904582542608	-10.704177388315	76.379866166836	-0.79733144224901	0.47528362422787	-11.501508830564
8	2	-163	9.1781488401884	163	6.6536603184273	2.8421709430404e-	15.831809158616
8	9	86.620133833166	-8.3808173979382	-84.32016251845	-11.312751170506	2.2999713147159	-19.693568568444
9	4	-40.679837481551	-38.687248829493	40.937351712174	22.893120624746	0.25751423062336	-15.794128204747

Figura 6.10 – Resultado do fluxo de potência para o sistema de 9 barras no cliente 2 do NWS

Fonte: Elaborado pelo Autor





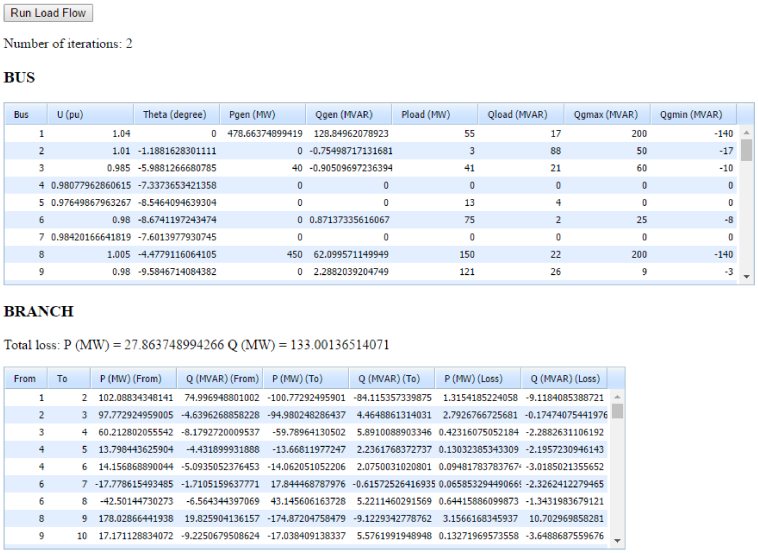
Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (MW)	Generation Q (MVAr)	Load P (MW)	Load Q (MVAr)
1	1.040	0.000	71.64	27.05	-	-
2	1.025	9.280	163.00	6.65	-	-
3	1.025	4.665	85.00	-10.86	-	-
4	1.026	-2.217	-	-	-	-
5	1.013	-3.687	-	-	90.00	30.00
6	1.032	1.967	-	-	-	-
7	1.016	0.728	-	-	100.00	35.00
8	1.026	3.720	-	-	-	-
9	0.996	-2.989	-	-	125.00	50.00
		Total:	319.64	22.84	315.00	115.00

Branch Data

Branch #	From Bus	To Bus	From Bus Injection P (MW)	From Bus Injection Q (MVAr)	To Bus Injection P (MW)	To Bus Injection Q (MVAr)	Loss (I^2 * Z) P (MW)	Loss (I^2 * Z) Q (MVAr)
1	1	4	71.64	27.05	-71.64	-23.92	0.000	3.12
2	4	5	30.70	1.03	-30.54	-16.54	0.166	0.90
3	5	6	-59.46	-13.46	60.82	-18.07	1.354	5.90
4	3	6	85.00	-10.86	-85.00	14.96	0.000	4.10
5	6	7	24.18	3.12	-24.10	-24.30	0.088	0.75
6	7	8	-75.90	-10.70	76.38	-0.80	0.475	4.03
7	8	2	-163.00	9.18	163.00	6.65	0.000	15.83
8	8	9	86.62	-8.38	-84.32	-11.31	2.300	11.57
9	9	4	-40.68	-38.69	40.94	22.89	0.256	2.19
						Total:	4.641	48.38

Figura 6.11 – Resultado do fluxo de potência para o sistema de 9 barras no MATPOWER

Fonte: Elaborado pelo Autor



Run Load Flow

Number of iterations: 2

BUS

Bus	U (pu)	Theta (degree)	Pgen (MW)	Qgen (MVAR)	Pload (MW)	Qload (MVAR)	Qgmax (MVAR)	Qgmin (MVAR)
1	1.04	0	478.66374899419	128.84962078923	55	17	200	-140
2	1.01	-1.1881628301111	0	-0.75498717131681	3	88	50	-17
3	0.985	-5.9881266680785	40	-0.90509697236394	41	21	60	-10
4	0.98077962860615	-7.3373653421358	0	0	0	0	0	0
5	0.97649867963267	-8.5464094639304	0	0	13	4	0	0
6	0.98	-8.6741197243474	0	0.87137335616067	75	2	25	-8
7	0.98420166641819	-7.6013977930745	0	0	0	0	0	0
8	1.005	-4.4779116064105	450	62.099571149949	150	22	200	-140
9	0.98	-9.5846714084382	0	2.2882039204749	121	26	9	-3

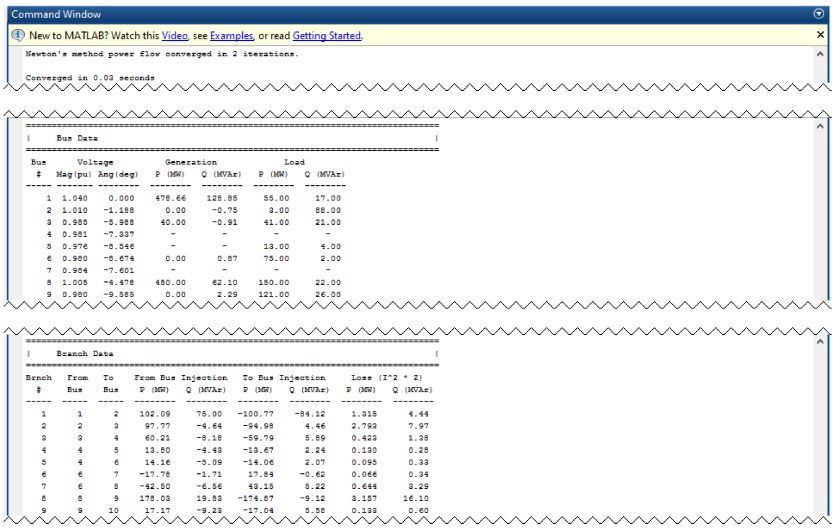
BRANCH

Total loss: P (MW) = 27.863748994266 Q (MW) = 133.00136514071

From	To	P (MW) (From)	Q (MVAR) (From)	P (MW) (To)	Q (MVAR) (To)	P (MW) (Loss)	Q (MVAR) (Loss)
1	2	102.08834348141	74.996948801002	-100.77292495901	-84.115357339875	1.3154185224058	-9.1184085388721
2	3	97.772924959005	-4.6396268858228	-94.980248286437	4.4648861314031	2.7926766725681	-0.17474075441976
3	4	60.212802055542	-8.1792720009537	-59.78964130502	5.8910088903346	0.42316075052184	-2.2882631106192
4	5	13.798443625904	-4.431899931888	-13.668119777247	2.2361768372737	0.13032385343309	-2.1957230946143
4	6	14.156868890044	-5.0935052376453	-14.062051052206	2.0750031020801	0.09481783783767	-3.0185021355652
6	7	-17.778615493485	-1.7105159637771	17.844468787976	-0.61572526416935	0.065853294490665	-2.3262412279465
6	8	-42.50144730273	-6.5643443970 69	43.145606163728	5.2211460291569	0.64415886099873	-1.3431983679121
8	9	178.02866441938	19.825904136157	-174.87204758479	-9.1229342778762	3.1566168345937	10.702969858281
9	10	17.171128834072	-9.2250679508624	-17.038409138337	5.5761991948948	0.13271969573558	-3.6488687559676

Figura 6.12 – Nove primeiros resultados do fluxo de potência para o sistema de 57 barras no cliente 2 do NWS

Fonte: Elaborado pelo Autor



Command Window

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

Newton's method power flow converged in 2 iterations.

Converged in 0.03 seconds

Bus Data

Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (MW)	Generation Q (MVAr)	Load P (MW)	Load Q (MVAr)
1	1.040	0.000	478.66	128.85	55.00	17.00
2	1.010	-1.188	0.00	-0.75	3.00	88.00
3	0.985	-5.988	40.00	-0.91	41.00	21.00
4	0.981	-7.337	-	-	-	-
5	0.976	-8.546	-	-	13.00	4.00
6	0.980	-8.674	0.00	0.87	75.00	2.00
7	0.984	-7.601	-	-	-	-
8	1.005	-4.478	450.00	62.10	150.00	22.00
9	0.980	-9.585	0.00	2.29	121.00	26.00

Branch Data

Branch #	From Bus	To Bus	From Bus Injection P (MW)	From Bus Injection Q (MVAr)	To Bus Injection P (MW)	To Bus Injection Q (MVAr)	Loss (I^2 * Z) P (MW)	Loss (I^2 * Z) Q (MVAr)
1	1	2	102.09	75.00	-100.77	-84.12	1.315	6.44
2	2	3	97.77	-4.64	-94.98	4.46	2.792	7.97
3	3	4	60.21	-8.18	-59.79	5.89	0.423	1.38
4	4	5	13.80	-4.43	-13.67	2.24	0.130	0.28
5	4	6	14.16	-5.09	-14.06	2.07	0.095	0.33
6	6	7	-17.78	-1.71	17.84	-0.62	0.066	0.34
7	6	8	-42.50	-6.96	43.15	5.22	0.644	3.29
8	8	9	178.03	19.83	-174.87	-9.12	3.157	16.10
9	9	10	17.17	-9.23	-17.04	5.58	0.133	0.60

Figura 6.13 – Nove primeiros resultados do fluxo de potência para o sistema de 57 barras no MATPOWER

Fonte: Elaborado pelo Autor

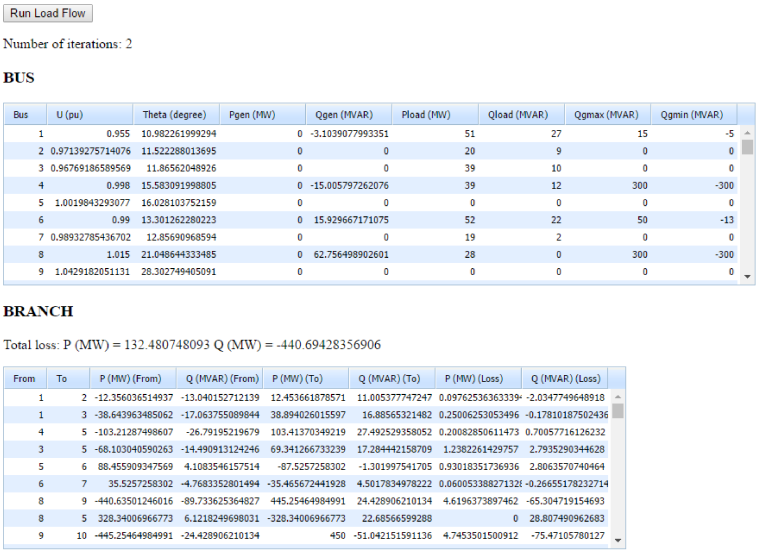


Figura 6.14 – Nove primeiros resultados do fluxo de potência para o sistema de 118 barras no cliente 2 do NWS

Fonte: Elaborado pelo Autor

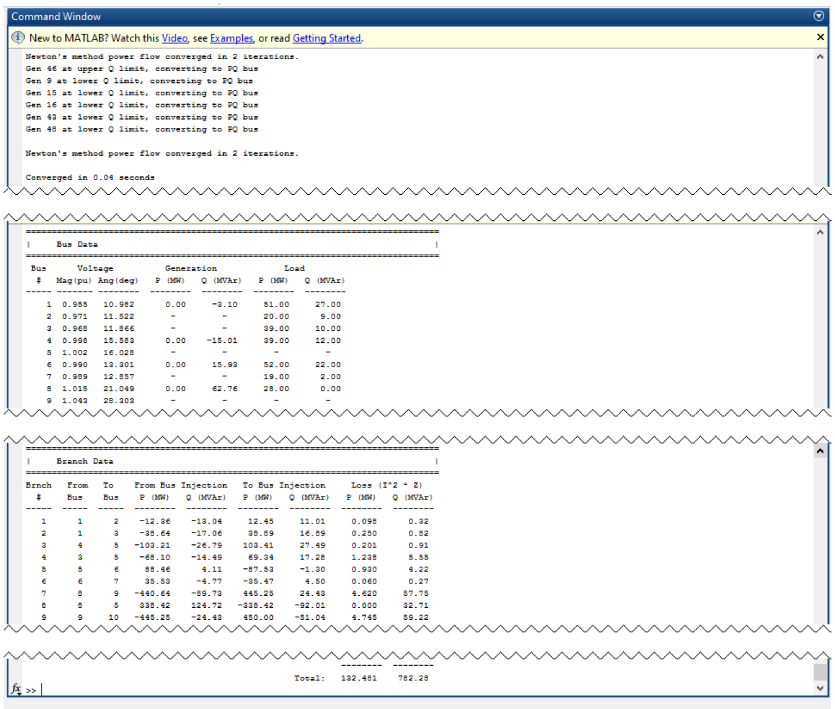


Figura 6.15 – Nove primeiros resultados do fluxo de potência para o sistema de 118 barras no MATPOWER

Fonte: Elaborado pelo Autor

Os testes da ferramenta de colaboração em tempo real também foram feitos em dois navegadores diferentes, Google Chrome e Mozilla Firefox. Como especificado nos códigos foi possível adicionar e remover dados de barras e de ramos e obter a atualização imediata na tela dos dois navegadores. Os novos usuários ao se conectarem também visualizam as modificações realizadas nas tabelas de opções de simulação, dados de barras e de ramos, uma vez que o TogetherJS já tenha sido iniciado.

6.4.3 Cliente 3

No caso desta versão de cliente foram testadas os recusos disponíveis como a criação do sistema elétrico, o armazenamento e a abertura de arquivos salvos no Google Drive (Figura 6.16 e Figura 6.17), e os recursos de zoom, grade e guias (Figura 6.18).

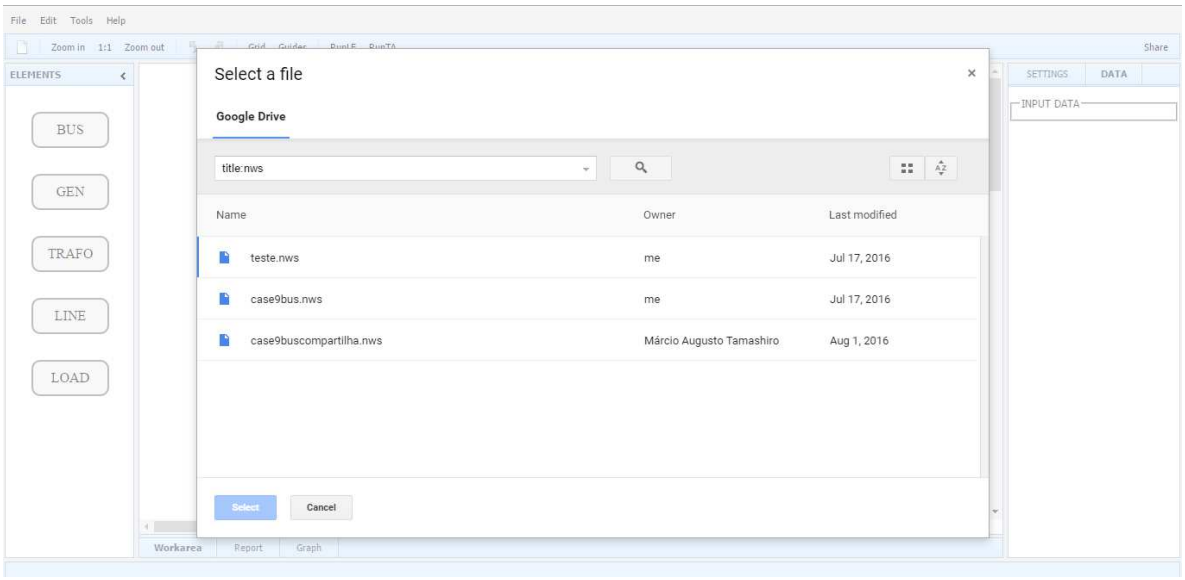


Figura 6.16 – Janela do Google Drive contendo três arquivos armazenados pelo usuário

Fonte: Elaborado pelo Autor

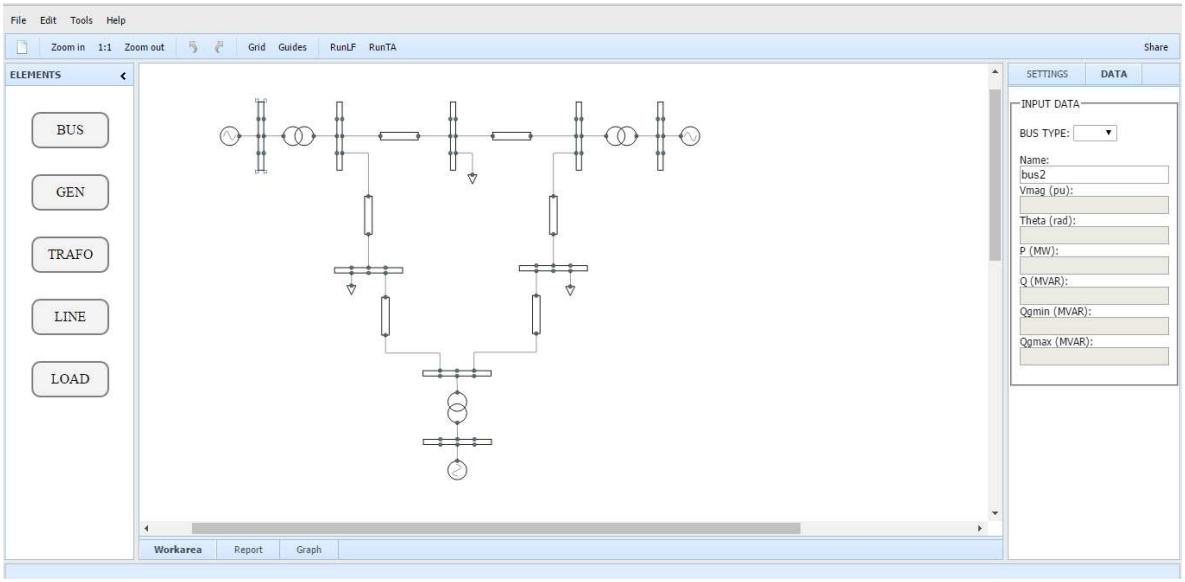


Figura 6.17 – Tela do NWS após abertura do arquivo do sistema de 9 barras

Fonte: Elaborado pelo Autor

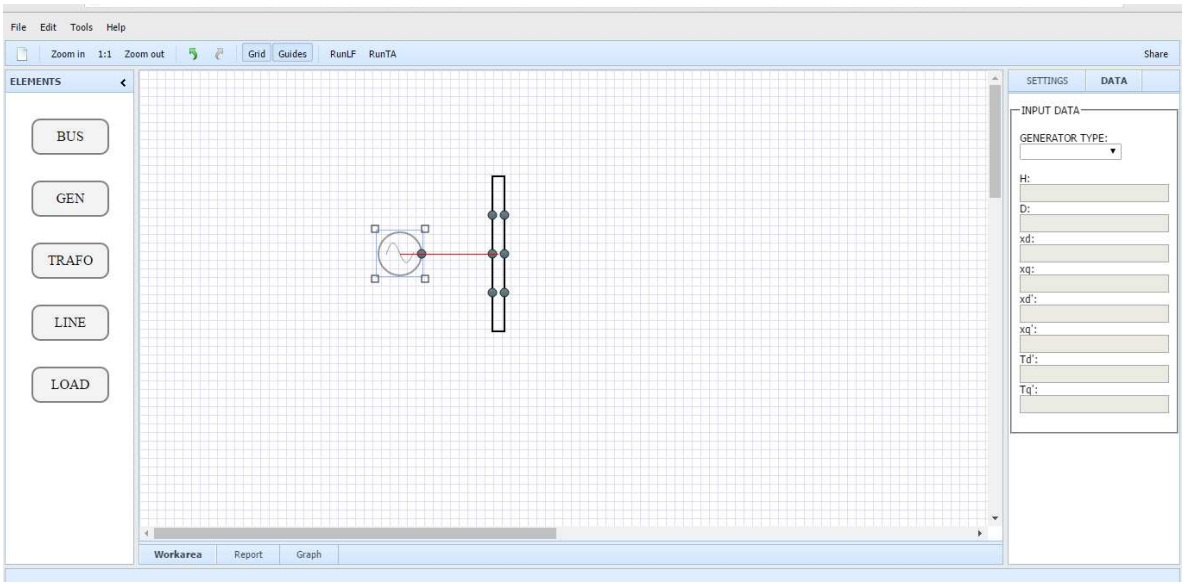


Figura 6.18 – Tela do NWS com o recurso de grade e de guia ativos

Fonte: Elaborado pelo Autor

6.5 Considerações Finais

Os cálculos do fluxo de potência efetuados produziram resultados com uma ótima exatidão, idênticos aos produzido pela aplicação de referência, MATPOWER. Os gráficos

gerados pelo NWS para a análise de estabilidade transitória também foram idênticos aos produzidos pela aplicação de referência, MatDyn.

Com a utilização das aplicações clientes 1 e 2 foi possível testar a comunicação e execução da aplicação no servidor, que ocorreu conforme o planejado e sem erros. A ferramenta de colaboração em tempo real integrada as essas aplicações também foi executada conforme o planejado e sem maiores problemas.

A aplicação cliente 3 possui uma interface mais usual para o usuário, mas precisa ser mais trabalhada para que possa receber os recursos disponíveis nas aplicações 1 e 2.

O próximo capítulo apresenta as conclusões finais deste trabalho e lista algumas propostas de trabalhos futuros

Capítulo 7

CONCLUSÕES

Ao final deste trabalho foi possível constatar o elevado grau de dificuldade para a implementação de uma aplicação *web*, mas cujas vantagens elencadas na Introdução compensam o esforço empregado. O desenvolvimento de *software* para *web* lida com muitas tecnologias diferentes e que estão em constante evolução.

Para a construção do NDSE WEB *Simulator* foi utilizado o que há de mais atual em tecnologias para *web*. Os testes computacionais realizados atestam sua confiabilidade e o tornam uma alternativa viável a ser considerada. Ele elimina as principais desvantagens dos pares de programas utilizados como referência para a sua concepção: MATPOWER/MatDyn e UFUFlow/TransUFU.

A organização do seu código e as soluções desenvolvidas podem muito bem servir de modelo para projetos similares, e permitem ainda a realização de melhorias, como a adição de novos recursos.

Convém destacar também o recurso de colaboração em tempo, não disponível até o momento em nenhuma outra aplicação similar. E ainda a sua implementação como serviço *web* que garante a interoperabilidade com outras aplicações por meio da internet.

As aplicações clientes implementadas para a realização dos testes e a integração desses recursos, apesar de terem sido implementadas como aplicações *web*, podem ser construídas também como aplicações *desktop*, em outras linguagens de programação e para qualquer sistema operacional ou dispositivo. Ou seja, existe essa flexibilidade raramente observada, ou mesmo inexistente, em outras aplicações similares.

Ao fim deste trabalho espera-se que a aplicação implementada possa:

a) Servir de referência para terceiros, ou que sirva de base para a criação de outras;
b) Garantir resultados confiáveis e que possa ainda ser utilizado para a homologação de análises realizadas em outras aplicações;
c) Estimular e permitir o recebimento de contribuições para a correção de erros e a adição de novos recursos;
d) Estimular o desenvolvimento e o uso de programas de código aberto para a análise de sistemas elétricos;
e) Estimular a cooperação com foco na garantia de implementações computacionais cada vez melhores e mais eficientes;
f) Estimular o compartilhamento e a discussão de conhecimentos nesta área;
g) Contribuir de forma efetiva com o ensino e a pesquisa voltados para a análise de sistemas elétricos;
h) Fomentar a formação de novos grupos de pesquisa nessa área.

O aprimoramento de qualquer programa computacional é uma tarefa contínua. E cabe aos desenvolvedores e possíveis colaboradores o trabalho da correção de erros, que sempre existirão, e a adição de novos recursos conforme novas necessidades forem surgindo. Nesse sentido lista-se a seguir algumas sugestões para trabalhos futuros:

a) Adição de outras ferramentas para a análise computacional de sistemas elétricos, como por exemplo, a análise de curto-circuito e a análise de transitórios eletromagnéticos;
b) Adição de outros métodos para o cálculo do fluxo de potência, como o método de Gauss-Seidel, o de Newton Desacoplado e o Desacoplado Rápido, e ainda a possibilidade do cálculo do fluxo de potência linearizado ou CC;
c) Adição de outros métodos para a solução das equações diferenciais, como por exemplo, os métodos de Runge-Kutta;
d) Adição de outros modelos de máquinas síncronas, de reguladores de tensão e de velocidade;
e) Investigação de ferramentas e algoritmos para o uso de processamento paralelo na análise computacional de sistemas elétricos;
f) Desenvolvimento de outras aplicações clientes e mais próximas de uma aplicação *desktop*, e que possa ficar disponível na internet para qualquer usuário;
g) Gerar as denominadas extensões PHP das principais rotinas de cálculo com o objetivo de garantir um melhor desempenho na execução da aplicação.

REFERÊNCIAS

ABDEL-AKER, M.; EL-NEMR, M. K.; ABDEL-GALIL, K. **Distributed Interactive Power System simulator Using Web Technology**. IEEE International Conference on Power and Energy (PECon). Kuala Lumpur, Malásia, 2010.

AGAMAH, S.; EKONOMOU, L. **A PHP application library for web-based power systems analysis**. IEEE European Modelling Symposium on Mathematical Modelling and Computer Simulation. Madri, Espanha, 2015.

AGOSTINI, M. N. **Nova Filosofia para o Projeto de Software para Sistemas de Energia Eletrica usando Modelagem Oreintada a Objetos**. Tese de Doutorado. Universidade Federal de Santa Catarina. Florianópolis. 2002.

Anarede. Disponível em: <http://www.anarede.cepel.br/cprog.html>. Acesso em 30 de agosto de 2016.

APACHE. Disponivel em: <https://httpd.apache.org>. Acesso em 30 de agosto de 2016.

ATP. Disponível em <http://eeug-test.hostingkunde.de>. Acesso em 30 de agosto de 2016.

BALDUINO, P. **Dominando JavaScript com jQuery**. São Paulo: Casa do Código, 2015.

BECKER, D. et al. **Standards-Based Approach Integrates Utility Applications**. IEEE Computer Applications in Power, v. 13, n. 4, pp. 13-20, 2000.

BENTO, E. J. **Desenvolvimento web com PHP e MySQL**. São Paulo: Casa do Código, 2015.

BIBIN, H. et al. **Application of Design Patterns in Power System Transient Simulator**. IEEE International Conference on Computer Science and Automation Engineering. Zhangjiajie, China, 2012.

BOSE, **A. Digital simulation of power systems**. IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control, v. 2, pp. 444–447, 1998.

Caso Teste – 9 Barras. *Power Systems and Evolutionary Algorithms* - 9-*Bus System*. Disponível em <http://www.al-roomi.org/power-flow/9-bus-system>. Acesso em 30 de agosto de 2016.

Caso Teste – 57 Barras. *Power Systems and Evolutionary Algorithms* - 57-*Bus System*. Disponível em <http://www.al-roomi.org/power-flow/57-bus-system>. Acesso em 30 de agosto de 2016.

Caso Teste – 118 Barras. *Power Systems and Evolutionary Algorithms* - 118-*Bus System* . Disponível em <http://www.al-roomi.org/power-flow/118-bus-system>. Acesso em 30 de agosto de 2016.

CHEN, S.; LU, F. Y. **Web-based simulations of power systems**. IEEE Computer Applications in Power, v. 15, n. 1, pp. 35-40, 2002.

COLE, S. **Manual MatDyn *Version* 1.2**, 2010. Disponível em: <https://www.esat.kuleuven.be/electa/teaching/matdyn/MatDyn1.2manual>. Acesso em 30 de agosto de 2016.

_____; BELMANS, R. MatDyn, **A New Matlab-Based Toolbox for Power System Dynamic Simulation**. IEEE Transactions on Power Systems, v. 26, n. 3, pp. 1129-1136, 2011.

Cyme. Disponível em: <http://www.cyme.com/software/cymecymflow>. Acesso em 30 de agosto de 2016.

DALL'OGLIO, P. **PHP: Programando com orientação a objetos**. São Paulo: Novatec, 2007.

DHTMLX. Disponivel em: <http://dhtmlx.com>. Acesso em 30 de agosto de 2016.

Digsilent. Disponível em: <http://www.digsilent.de/index.php/products-powerfactory.html>. Acesso em 30 de agosto de 2016.

DIPAOLO, I. F. et al. **Creational Object-oriented Design Pattern Applied to the Development of Software Tools for Electric Power Systems Dynamic Simulations**. IEEE Latin America Transactions, v. 8, n. 3, pp. 287-295, 2010.

DOMMEL, H. W.; SATO, N.. **Fast Transient Stability Solutions**. IEEE Transactions on Power Apparatus and Systems, vol. PAS-91, n. 4, pp. 1643-1650, 1972.

DRAW2D. Disponivel em: <http://www.draw2d.org>. Acesso em 30 de agosto de 2016.

DZAFIC, I. **An Object-Oriented Graphical Framework for Power System Analysis**. International Conference on Power Engineering - Energy and Electrical Drives. Setúbal, Portugal, 2007.

ELGERD, O. I. **Electric energy theory: an introduction**. New York: McGraw-Hill, 1971.

EMTP-RV. Disponível em: < http://emtp-software.com/>. Acesso em 30 de agosto de 2016.

FLINN, D. G.; DUGAN, R. C. **A Database for Diverse Power System Simulation Applications**. IEEE Transactions on Power Systems, v. 7, n. 2, pp. 784-790, 1992.

FOLEY, M. et al. **An Object Based Graphical User Interface for Power Systems**. IEEE Transactions on Power Systems, v. 8, n. 1, pp. 97-104, 1993.

FUERTE-ESQUIVEL, C. R. et al. **Efficient Object Oriented Power Systems Software for the Analysis of Large-Scale Networks Containing FACTS-Controlled Branches**. IEEE Transactions on Power Systems, v. 13, n. 2, pp. 464-472, 1998.

GOOGLE API. Disponivel em: <https://developers.google.com/api-client-library>. Acesso em 30 de agosto de 2016.

GUIMARÃES, G. C. **Manual do TransUFU – Programa de Análise de Estabilidade Transitória da UFU**. Uberlândia: Universidade Federal de Uberlândia, 2000.

_____, **Manual do UFUFlow - Programa de fluxo de carga**. Uberlândia: Universidade Federal de Uberlândia, 2000.

HAKAVIK, B.; HOLEN, A. T. **Power System Modelling and Sparse Matrix Operations Using Object-Oriented Programming**. IEEE Transactions on Power Systems, v. 9, n. 2, pp. 1045-1051, 1994.

HANDSCHIN, E. et al. **Object-Oriented Software Engineering for Transmission Planning in Open Access Schemes**. IEEE Transactions on Power Systems, v. 13, n. 1, pp. 94-100, 1998.

Interpss. Disponível em: <http://www.interpss.org>. Acesso em 30 de agosto de 2016.

KHOURY, F. K. C. B. **Minimização de custos de produção via programação inteira mista: estudo de caso de planejamento de produção de luminárias**. Dissertação de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro. 2011.

KOBER, F.; MANZONI, A.; LEMOS, F. A. B. **An Objected-Oriented Approach to Development and Integrationof Graphical User Interface and Power System Framework**. IEEE Bologna PowerTech Conference. Bologna, Itália, 2003.

LECHETA, R. R. **Web Services RESTful: Aprenda a crir web services RESTful em Java na nuvem do Google**. São Paulo: Novatec, 2015.

LEE, S.-H.; CHU, C.-C. **A Web-based Power Flow Calculation of Large-Scale Power Systems Embedded with VSC-based HVDC systems**. IEEE/PES Transmission and Distribution Conference and Exhibition: Asia and Pacific. Dalian, China, 2005.

LEOU, R.-C.; GAING, Z.-L. **A Web-Based Load Flow Simulation of Power Systems**. IEEE Power Engineering Society Summer Meeting. Chicago, EUA, 2002.

MALARVIZHI, R.; VEENA, S. T. **Power Flow Analysis using Matlab Builder JA**. International Conference on Recent Advancements in Electrical, Electronics and Control Engineering. Sivakasi, Índia, 2011.

MANZONI, A. **Desenvolvimento de um Sistema Computacional Orientado a Objetos para Sistemas Elétricos de Potência: aplicação a simulação rápida e análise da estabilidade de tensão**. Tese de Doutorado. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2005.

MARINHO, J. M. T. **Simulação em Sistemas de Energia Elétrica com Modelagem Flexível: monofásica e trifásica**. Tese de Doutorado. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2008.

MatDyn. Disponível em : <https://www.esat.kuleuven.be/electa/teaching/matdyn>. Acesso em 30 de agosto de 2016.

MATPOWER. Disponível em : <http://www.pserc.cornell.edu/matpower>. Acesso em 30 de agosto de 2016.

MILANO, F. **An Open Source Power System Analysis Toolbox**. IEEE Transactions on Power Systems, v. 20, 2005.

_____, **A Python-based Software Tool for Power System Analysis**. IEEE Power and Energy Society General Meeting. Vancouver, Canadá, 2013.

_____; VANFRETTI, L.; MORATAYA, J. C. **An Open Source Power System Virtual Laboratory: The PSAT Case and Experience**. IEEE Transactions on Education, v. 51, n. 1, pp. 17-23, 2008.

_____; ZHOU, M.; HOU, G. **Open model for exchanging power system data**. IEEE Power & System General Meeting. Calgary, Canadá, 2009.

MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de Software II: Introdução ao desenvolvimento web com HTML, CSS, JAVASCRIPT e PHP**. Porto Alegre: Bookman, 2014.

NETBEANS. Disponivel em: <https://netbeans.org>. Acesso em 30 de agosto de 2016.

NEYER, A. F.; WU, F. F.; IMHOF, K. **Object-Oriented Programming for Flexible Software : Example of a Load Flow**. IEEE Transactions on Power Systems, v. 5, n. 3, pp. 689-696, 1990.

NUMERICJS. Disponivel em: <http://www.numericjs.com>. Acesso em 30 de agosto de 2016.

ONG, Y. S.; GOOI, H. B. **A Web-based Power Flow Simulator for Power Engineering Education**. IEEE Power Engineering Society Summer Meeting. Edmonton, Canadá, 1999.

OPENELECTRICAL. Disponivel em: <www.openelectrical.org/wiki/index.php?title=Power_Systems_Analysis_Software>. Acesso em 30 de agosto de 2016.

PANDIT, S.; SOMAN, S. A.; KHAPARDE, S. A. **Object-Oriented Design for Power System Applications**. IEEE Computer Applications in Power, pp. 43-47, 2000.

PHILLIPS, N. B. P.; GANN, J. O.; IRVING, M. R. **The SIMIAN Architecture - An Object Orientated Framework for Integrated Power System Modelling Analysis and Control**. Fourth International Conference on Power System Control and Management. London, UK, 1996.

PowerWorld. Disponível em: <http://www.powerworld.com>. Acesso em 30 de agosto de 2016.

PSAT. Disponível em: < http://faraday1.ucd.ie/psat.html>. Acesso em 30 de agosto de 2016.

PSS/E. Disponível em: <http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/software-solutions/planning-data-management-software/planning-simulation/Pages/PSS-E.aspx#>. Acesso em 30 de agosto de 2016.

SCHAFFNER, C. **An Internet-Based Load Flow Visualization Software for Education in Power Engineering**. IEEE Power Engineering Society Winter Meeting. New York, EUA, 2002.

SENA, J. A. D. S. **Desenvolvimento de Framework para Análise e Simulação Dinâmica de Sistemas Elétricos de Potência**. Tese de Doutorado. Universidade Federal do Pará. Belém. 2013.

SHAHRIARI, A.; MOKHLIS, H.; BAKAR, A. **Critical Reviews of Load Flow Methods for Well, Ill and Unsolvable Condition**. Journal of Electrical Engineering, 63, 2012.

SimPowerSystems. Disponível em: <http://www.mathworks.com/products/simpower>. Acesso em 30 de agosto de 2016.

SOMAN, S. A.; KHAPARDE, S. A.; PANDIT, S. **Computational Methods for Large Sparse Power Systems Analysis, An Object Oriented Approach**. New York: Springer, 2002.

STAGG, W. G.; EL-ABIAD, A. H. **Computer methods in power system analysis**. New York: McGraw-Hill, 1968.

THE RedMonk Programming Language Rankings: June 2015. RedMonk - Analysis for the people, by the people, 2015. Disponivel em: <http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15>. Acesso em: 01 ago. 2016.

TOGETHERJS. Mozilla Labs: TogetherJS, 2016. Disponivel em: <https://togetherjs.com>. Acesso em: 01 ago. 2016.

WEIJIANG, Q.; ZOU WEIMEI, Z.; YONGFENG, S. **Design Patterns Applied in Power System Analysis Software Package**. International Conference on Industrial Control and Electronics Engineering. Xi'an, China, 2012.

ZHOU, E. **Object-oriented programming, C++ and power system simulation**. IEEE Transactions on Power Systems, v. 11, pp. 206-215, 1996.

ZHOU, M.; ZHOU, S. **Internet, Open-source and Power System Simulation**. IEEE Power Engineering Society General Meeting. Tampa, China, 2007.

ZHU, J.; JOSSMAN, P. **Application of Design Patterns for Object-Oriented Modeling of Power Systems**. IEEE Transactions on Power Systems, v. 14, n. 2, pp. 532-537, 1999.

ZHU, J.; LUBKEMAN, D. **Object-Oriented Development of Software Systems for Power System Simulations**. IEEE Transactions on Power Systems, v. 12, n. 2, pp. 1002-1007, 1997.

ZIMMERMAN, R. D.; MURILLO-SÁNCHEZ, C. E.; THOMAS, R. J.. MATPOWER: **Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education**. IEEE Transactions on Power Systems, v. 26, n. 1, pp. 12-19, 2011.

_____; THOMAS, R. J. **PowerWeb: a tool for evaluating economic and reliability impacts of electric power market designs**. IEEE Power Systems Conference and Exposition. New York. 2004. pp. 1562-1567.

Apêndice A

DADOS DE ENTRADA PARA O NWS

A.1 – Dados de entrada para o cálculo do fluxo de potência do sistema de 9 barras

Obs.: O diagrama unifilar do sistema de 9 barras e os arquivos com seus dados de entrada para diversos programas de simulação estão disponíveis na referência Caso Teste – 9 Barras (2016).

```
{
"info": ["lf"],
"optLF": [100, 10, 1e-3, 1],
"bus": [
      [1,3,0,0,0,0,0,0,1.04,0,300,-300],
      [2,2,163,0,0,0,0,0,1.025,0,300,-300],
      [3,2,85,0,0,0,0,0,1.025,0,300,-300],
      [4,1,0,0,0,0,0,0,1,0,0,0],
      [5,1,0,0,90,30,0,0,1,0,0,0],
      [6,1,0,0,0,0,0,0,1,0,0,0],
      [7,1,0,0,100,35,0,0,1,0,0,0],
      [8,1,0,0,0,0,0,0,1,0,0,0],
      [9,1,0,0,125,50,0,0,1,0,0,0]
     ],
"branch":[
         [1,4,0,0.0576,0,1,0,1],
         [4,5,0.017,0.092,0.158,0,0,1],
         [5,6,0.039,0.17,0.358,0,0,1],
         [3,6,0,0.0586,0,1,0,1],
         [6,7,0.0119,0.1008,0.209,0,0,1],
         [7,8,0.0085,0.072,0.149,0,0,1],
         [8,2,0,0.0625,0,1,0,1],
         [8,9,0.032,0.161,0.306,0,0,1],
```

```
        [9,4,0.01,0.085,0.176,0,0,1]
      ]
}
```

A.2 – Dados de entrada para o cálculo do fluxo de potência do sistema de 57 barras

Obs.: O diagrama unifilar do sistema de 57 barras e os arquivos com seus dados de entrada para diversos programas de simulação estão disponíveis na referência Caso Teste – 57 Barras (2016).

```
{
"info": ["lf"],
"optLF": [100, 10, 1e-3, 1],
"bus": [
        [1,3,128.9,-16.1,55,17,0,0,1.04,0,200,-140],
        [2,2,0,-0.8,3,88,0,0,1.01,-1.18,50,-17],
        [3,2,40,-1,41,21,0,0,0.985,-5.97,60,-10],
        [4,1,0,0,0,0,0,0,0.981,-7.32,0,0],
        [5,1,0,0,13,4,0,0,0.976,-8.52,0,0],
        [6,2,0,0.8,75,2,0,0,0.98,-8.65,25,-8],
        [7,1,0,0,0,0,0,0,0.984,-7.58,0,0],
        [8,2,450,62.1,150,22,0,0,1.005,-4.45,200,-140],
        [9,2,0,2.2,121,26,0,0,0.98,-9.56,9,-3],
        [10,1,0,0,5,2,0,0,0.986,-11.43,0,0],
        [11,1,0,0,0,0,0,0,0.974,-10.17,0,0],
        [12,2,310,128.5,377,24,0,0,1.015,-10.46,155,-150],
        [13,1,0,0,18,2.3,0,0,0.979,-9.79,0,0],
        [14,1,0,0,10.5,5.3,0,0,0.97,-9.33,0,0],
        [15,1,0,0,22,5,0,0,0.988,-7.18,0,0],
        [16,1,0,0,43,3,0,0,1.013,-8.85,0,0],
        [17,1,0,0,42,8,0,0,1.017,-5.39,0,0],
        [18,1,0,0,27.2,9.8,0,-10,1.001,-11.71,0,0],
        [19,1,0,0,3.3,0.6,0,0,0.97,-13.2,0,0],
        [20,1,0,0,2.3,1,0,0,0.964,-13.41,0,0],
        [21,1,0,0,0,0,0,0,1.008,-12.89,0,0],
        [22,1,0,0,0,0,0,0,1.01,-12.84,0,0],
        [23,1,0,0,6.3,2.1,0,0,1.008,-12.91,0,0],
        [24,1,0,0,0,0,0,0,0.999,-13.25,0,0],
        [25,1,0,0,6.3,3.2,0,-16.949,0.982,-18.13,0,0],
        [26,1,0,0,0,0,0,0,0.959,-12.95,0,0],
```

```
            [27,1,0,0,9.3,0.5,0,0,0.982,-11.48,0,0],
            [28,1,0,0,4.6,2.3,0,0,0.997,-10.45,0,0],
            [29,1,0,0,17,2.6,0,0,1.01,-9.75,0,0],
            [30,1,0,0,3.6,1.8,0,0,0.962,-18.68,0,0],
            [31,1,0,0,5.8,2.9,0,0,0.936,-19.34,0,0],
            [32,1,0,0,1.6,0.8,0,0,0.949,-18.46,0,0],
            [33,1,0,0,3.8,1.9,0,0,0.947,-18.5,0,0],
            [34,1,0,0,0,0,0,0,0.959,-14.1,0,0],
            [35,1,0,0,6,3,0,0,0.966,-13.86,0,0],
            [36,1,0,0,0,0,0,0,0.976,-13.59,0,0],
            [37,1,0,0,0,0,0,0,0.985,-13.41,0,0],
            [38,1,0,0,14,7,0,0,1.013,-12.71,0,0],
            [39,1,0,0,0,0,0,0,0.983,-13.46,0,0],
            [40,1,0,0,0,0,0,0,0.973,-13.62,0,0],
            [41,1,0,0,6.3,3,0,0,0.996,-14.05,0,0],
            [42,1,0,0,7.1,4.4,0,0,0.966,-15.5,0,0],
            [43,1,0,0,2,1,0,0,1.01,-11.33,0,0],
            [44,1,0,0,12,1.8,0,0,1.017,-11.86,0,0],
            [45,1,0,0,0,0,0,0,1.036,-9.25,0,0],
            [46,1,0,0,0,0,0,0,1.05,-11.89,0,0],
            [47,1,0,0,29.7,11.6,0,0,1.033,-12.49,0,0],
            [48,1,0,0,0,0,0,0,1.027,-12.59,0,0],
            [49,1,0,0,18,8.5,0,0,1.036,-12.92,0,0],
            [50,1,0,0,21,10.5,0,0,1.023,-13.39,0,0],
            [51,1,0,0,18,5.3,0,0,1.052,-12.52,0,0],
            [52,1,0,0,4.9,2.2,0,0,0.98,-11.47,0,0],
            [53,1,0,0,20,10,0,-15.873,0.971,-12.23,0,0],
            [54,1,0,0,4.1,1.4,0,0,0.996,-11.69,0,0],
            [55,1,0,0,6.8,3.4,0,0,1.031,-10.78,0,0],
            [56,1,0,0,7.6,2.2,0,0,0.968,-16.04,0,0],
            [57,1,0,0,6.7,2,0,0,0.965,-16.56,0,0]
        ],
    "branch": [
            [1,2,0.0083,0.028,0.129,0,0,1],
            [2,3,0.0298,0.085,0.0818,0,0,1],
            [3,4,0.0112,0.0366,0.038,0,0,1],
            [4,5,0.0625,0.132,0.0258,0,0,1],
            [4,6,0.043,0.148,0.0348,0,0,1],
            [6,7,0.02,0.102,0.0276,0,0,1],
            [6,8,0.0339,0.173,0.047,0,0,1],
```

[8,9,0.0099,0.0505,0.0548,0,0,1],
[9,10,0.0369,0.1679,0.044,0,0,1],
[9,11,0.0258,0.0848,0.0218,0,0,1],
[9,12,0.0648,0.295,0.0772,0,0,1],
[9,13,0.0481,0.158,0.0406,0,0,1],
[13,14,0.0132,0.0434,0.011,0,0,1],
[13,15,0.0269,0.0869,0.023,0,0,1],
[1,15,0.0178,0.091,0.0988,0,0,1],
[1,16,0.0454,0.206,0.0546,0,0,1],
[1,17,0.0238,0.108,0.0286,0,0,1],
[3,15,0.0162,0.053,0.0544,0,0,1],
[4,18,0,0.555,0,0.97,0,1],
[4,18,0,0.43,0,0.978,0,1],
[5,6,0.0302,0.0641,0.0124,0,0,1],
[7,8,0.0139,0.0712,0.0194,0,0,1],
[10,12,0.0277,0.1262,0.0328,0,0,1],
[11,13,0.0223,0.0732,0.0188,0,0,1],
[12,13,0.0178,0.058,0.0604,0,0,1],
[12,16,0.018,0.0813,0.0216,0,0,1],
[12,17,0.0397,0.179,0.0476,0,0,1],
[14,15,0.0171,0.0547,0.0148,0,0,1],
[18,19,0.461,0.685,0,0,0,1],
[19,20,0.283,0.434,0,0,0,1],
[21,20,0,0.7767,0,1.043,0,1],
[21,22,0.0736,0.117,0,0,0,1],
[22,23,0.0099,0.0152,0,0,0,1],
[23,24,0.166,0.256,0.0084,0,0,1],
[24,25,0,1.182,0,0,0,1],
[24,25,0,1.23,0,0,0,1],
[24,26,0,0.0473,0,1.043,0,1],
[26,27,0.165,0.254,0,0,0,1],
[27,28,0.0618,0.0954,0,0,0,1],
[28,29,0.0418,0.0587,0,0,0,1],
[7,29,0,0.0648,0,0.967,0,1],
[25,30,0.135,0.202,0,0,0,1],
[30,31,0.326,0.497,0,0,0,1],
[31,32,0.507,0.755,0,0,0,1],
[32,33,0.0392,0.036,0,0,0,1],
[34,32,0,0.953,0,0.975,0,1],
[34,35,0.052,0.078,0.0032,0,0,1],

```
        [35,36,0.043,0.0537,0.0016,0,0,1],
        [36,37,0.029,0.0366,0,0,0,1],
        [37,38,0.0651,0.1009,0.002,0,0,1],
        [37,39,0.0239,0.0379,0,0,0,1],
        [36,40,0.03,0.0466,0,0,0,1],
        [22,38,0.0192,0.0295,0,0,0,1],
        [11,41,0,0.749,0,0.955,0,1],
        [41,42,0.207,0.352,0,0,0,1],
        [41,43,0,0.412,0,0,0,1],
        [38,44,0.0289,0.0585,0.002,0,0,1],
        [15,45,0,0.1042,0,0.955,0,1],
        [14,46,0,0.0735,0,0.9,0,1],
        [46,47,0.023,0.068,0.0032,0,0,1],
        [47,48,0.0182,0.0233,0,0,0,1],
        [48,49,0.0834,0.129,0.0048,0,0,1],
        [49,50,0.0801,0.128,0,0,0,1],
        [50,51,0.1386,0.22,0,0,0,1],
        [10,51,0,0.0712,0,0.93,0,1],
        [13,49,0,0.191,0,0.895,0,1],
        [29,52,0.1442,0.187,0,0,0,1],
        [52,53,0.0762,0.0984,0,0,0,1],
        [53,54,0.1878,0.232,0,0,0,1],
        [54,55,0.1732,0.2265,0,0,0,1],
        [11,43,0,0.153,0,0.958,0,1],
        [44,45,0.0624,0.1242,0.004,0,0,1],
        [40,56,0,1.195,0,0.9580,0,1],
        [56,41,0.553,0.549,0,0,0,1],
        [56,42,0.2125,0.354,0,0,0,1],
        [39,57,0,1.355,0,0.98,0,1],
        [57,56,0.174,0.26,0,0,0,1],
        [38,49,0.115,0.177,0.003,0,0,1],
        [38,48,0.0312,0.0482,0,0,0,1],
        [9,55,0,0.1205,0,0.94,0,1]
    ]
}
```

A.3 – Dados de entrada para o cálculo do fluxo de potência do sistema de 118 barras

Obs.: O diagrama unifilar do sistema de 118 barras e os arquivos com seus dados de entrada para diversos programas de simulação estão disponíveis na referência Caso Teste – 118 Barras (2016).


```
{
"info": ["lf"],
"optLF": [100, 10, 1e-3, 1],
"bus":    [
        [1,2,0,0,51,27,0,0,0.955,10.67,15,-5],
        [2,1,0,0,20,9,0,0,0.971,11.22,0,0],
        [3,1,0,0,39,10,0,0,0.968,11.56,0,0],
        [4,2,-9,0,30,12,0,0,0.998,15.28,300,-300],
        [5,1,0,0,0,0,0,2.5,1.002,15.73,0,0],
        [6,2,0,0,52,22,0,0,0.99,13,50,-13],
        [7,1,0,0,19,2,0,0,0.989,12.56,0,0],
        [8,2,-28,0,0,0,0,0,1.015,20.77,300,-300],
        [9,1,0,0,0,0,0,0,1.043,28.02,0,0],
        [10,2,450,0,0,0,0,0,1.05,35.61,200,-147],
        [11,1,0,0,70,23,0,0,0.985,12.72,0,0],
        [12,2,85,0,47,10,0,0,0.99,12.2,120,-35],
        [13,1,0,0,34,16,0,0,0.968,11.35,0,0],
        [14,1,0,0,14,1,0,0,0.984,11.5,0,0],
        [15,2,0,0,90,30,0,0,0.97,11.23,30,-10],
        [16,1,0,0,25,10,0,0,0.984,11.91,0,0],
        [17,1,0,0,11,3,0,0,0.995,13.74,0,0],
        [18,2,0,0,60,34,0,0,0.973,11.53,50,-16],
        [19,2,0,0,45,25,0,0,0.962,11.05,24,-8],
        [20,1,0,0,18,3,0,0,0.958,11.93,0,0],
        [21,1,0,0,14,8,0,0,0.959,13.52,0,0],
        [22,1,0,0,10,5,0,0,0.97,16.08,0,0],
        [23,1,0,0,7,3,0,0,1,21,0,0],
        [24,2,-13,0,0,0,0,0,0.992,20.89,300,-300],
        [25,2,220,0,0,0,0,0,1.05,27.93,140,-47],
        [26,2,314,0,0,0,0,0,1.015,29.71,1000,-1000],
        [27,2,-9,0,62,13,0,0,0.968,15.35,300,-300],
        [28,1,0,0,17,7,0,0,0.962,13.62,0,0],
        [29,1,0,0,24,4,0,0,0.963,12.63,0,0],
        [30,1,0,0,0,0,0,0,0.968,18.79,0,0],
        [31,2,7,0,43,27,0,0,0.967,12.75,300,-300],
        [32,2,0,0,59,23,0,0,0.963,14.8,42,-14],
```

[33,1,0,0,23,9,0,0,0.972,10.63,0,0],
[34,2,0,0,59,26,0,-7.143,0.984,11.3,24,-8],
[35,1,0,0,33,9,0,0,0.981,10.87,0,0],
[36,2,0,0,31,17,0,0,0.98,10.87,24,-8],
[37,1,0,0,0,0,0,4,0.992,11.77,0,0],
[38,1,0,0,0,0,0,0,0.962,16.91,0,0],
[39,1,0,0,27,11,0,0,0.97,8.41,0,0],
[40,2,-46,0,20,23,0,0,0.97,7.35,300,-300],
[41,1,0,0,37,10,0,0,0.967,6.92,0,0],
[42,2,-59,0,37,23,0,0,0.985,8.53,300,-300],
[43,1,0,0,18,7,0,0,0.978,11.28,0,0],
[44,1,0,0,16,8,0,-10,0.985,13.82,0,0],
[45,1,0,0,53,22,0,-10,0.987,15.67,0,0],
[46,2,19,0,28,10,0,-10,1.005,18.49,100,-100],
[47,1,0,0,34,0,0,0,1.017,20.73,0,0],
[48,1,0,0,20,11,0,-6.667,1.021,19.93,0,0],
[49,2,204,0,87,30,0,0,1.025,20.94,210,-85],
[50,1,0,0,17,4,0,0,1.001,18.9,0,0],
[51,1,0,0,17,8,0,0,0.967,16.28,0,0],
[52,1,0,0,18,5,0,0,0.957,15.32,0,0],
[53,1,0,0,23,11,0,0,0.946,14.35,0,0],
[54,2,48,0,113,32,0,0,0.955,15.26,300,-300],
[55,2,0,0,63,22,0,0,0.952,14.97,23,-8],
[56,2,0,0,84,18,0,0,0.954,15.16,15,-8],
[57,1,0,0,12,3,0,0,0.971,16.36,0,0],
[58,1,0,0,12,3,0,0,0.959,15.51,0,0],
[59,2,155,0,277,113,0,0,0.985,19.37,180,-60],
[60,1,0,0,78,3,0,0,0.993,23.15,0,0],
[61,2,160,0,0,0,0,0,0.995,24.04,300,-100],
[62,2,0,0,77,14,0,0,0.998,23.43,20,-20],
[63,1,0,0,0,0,0,0,0.969,22.75,0,0],
[64,1,0,0,0,0,0,0,0.984,24.52,0,0],
[65,2,391,0,0,0,0,0,1.005,27.65,200,-67],
[66,2,392,0,39,18,0,0,1.05,27.48,200,-67],
[67,1,0,0,28,7,0,0,1.02,24.84,0,0],
[68,1,0,0,0,0,0,0,1.003,27.55,0,0],
[69,3,516.4,0,0,0,0,0,1.035,30,300,-300],
[70,2,0,0,66,20,0,0,0.984,22.58,32,-10],
[71,1,0,0,0,0,0,0,0.987,22.15,0,0],
[72,2,-12,0,0,0,0,0,0.98,20.98,100,-100],

[73,2,-6,0,0,0,0,0,0.991,21.94,100,-100],
[74,2,0,0,68,27,0,-8.33,0.958,21.64,9,-6],
[75,1,0,0,47,11,0,0,0.967,22.91,0,0],
[76,2,0,0,68,36,0,0,0.943,21.77,23,-8],
[77,2,0,0,61,28,0,0,1.006,26.72,70,-20],
[78,1,0,0,71,26,0,0,1.003,26.42,0,0],
[79,1,0,0,39,32,0,-5,1.009,26.72,0,0],
[80,2,477,0,130,26,0,0,1.04,28.96,280,-165],
[81,1,0,0,0,0,0,0,0.997,28.1,0,0],
[82,1,0,0,54,27,0,-5,0.989,27.24,0,0],
[83,1,0,0,20,10,0,-10,0.985,28.42,0,0],
[84,1,0,0,11,7,0,0,0.98,30.95,0,0],
[85,2,0,0,24,15,0,0,0.985,32.51,23,-8],
[86,1,0,0,21,10,0,0,0.987,31.14,0,0],
[87,2,4,0,0,0,0,0,1.015,31.4,1000,-100],
[88,1,0,0,48,10,0,0,0.987,35.64,0,0],
[89,2,607,0,0,0,0,0,1.005,39.69,300,-210],
[90,2,-85,0,78,42,0,0,0.985,33.29,300,-300],
[91,2,-10,0,0,0,0,0,0.98,33.31,100,-100],
[92,2,0,0,65,10,0,0,0.99,33.8,9,-3],
[93,1,0,0,12,7,0,0,0.987,30.79,0,0],
[94,1,0,0,30,16,0,0,0.991,28.64,0,0],
[95,1,0,0,42,31,0,0,0.981,27.67,0,0],
[96,1,0,0,38,15,0,0,0.993,27.51,0,0],
[97,1,0,0,15,9,0,0,1.011,27.88,0,0],
[98,1,0,0,34,8,0,0,1.024,27.4,0,0],
[99,2,-42,0,0,0,0,0,1.01,27.04,100,-100],
[100,2,252,0,37,18,0,0,1.017,28.03,155,-50],
[101,1,0,0,22,15,0,0,0.993,29.61,0,0],
[102,1,0,0,5,3,0,0,0.991,32.3,0,0],
[103,2,40,0,23,16,0,0,1.01,24.44,40,-15],
[104,2,0,0,38,25,0,0,0.971,21.69,23,-8],
[105,2,0,0,31,26,0,-5,0.965,20.57,23,-8],
[106,1,0,0,43,16,0,0,0.962,20.32,0,0],
[107,2,-22,0,28,12,0,-16.67,0.952,17.53,200,-200],
[108,1,0,0,2,1,0,0,0.967,19.38,0,0],
[109,1,0,0,8,3,0,0,0.967,18.93,0,0],
[110,2,0,0,39,30,0,-16.67,0.973,18.09,23,-8],
[111,2,36,0,0,0,0,0,0.98,19.74,1000,-100],
[112,2,-43,0,25,13,0,0,0.975,14.99,1000,-100],

```
            [113,2,0,0,6,0,0,0,0.993,13.74,200,-100],
            [114,1,0,0,8,3,0,0,0.96,14.46,0,0],
            [115,1,0,0,22,7,0,0,0.96,14.46,0,0],
            [116,2,-184,0,0,0,0,0,1.005,27.12,1000,-1000],
            [117,1,0,0,20,8,0,0,0.974,10.67,0,0],
            [118,1,0,0,33,15,0,0,0.949,21.92,0,0]
        ],
    "branch": [
            [1,2,0.0303,0.0999,0.0254,0,0,1],
            [1,3,0.0129,0.0424,0.01082,0,0,1],
            [4,5,0.00176,0.00798,0.0021,0,0,1],
            [3,5,0.0241,0.108,0.0284,0,0,1],
            [5,6,0.0119,0.054,0.01426,0,0,1],
            [6,7,0.00459,0.0208,0.0055,0,0,1],
            [8,9,0.00244,0.0305,1.162,0,0,1],
            [8,5,0,0.0267,0,0.985,0,1],
            [9,10,0.00258,0.0322,1.23,0,0,1],
            [4,11,0.0209,0.0688,0.01748,0,0,1],
            [5,11,0.0203,0.0682,0.01738,0,0,1],
            [11,12,0.00595,0.0196,0.00502,0,0,1],
            [2,12,0.0187,0.0616,0.01572,0,0,1],
            [3,12,0.0484,0.16,0.0406,0,0,1],
            [7,12,0.00862,0.034,0.00874,0,0,1],
            [11,13,0.02225,0.0731,0.01876,0,0,1],
            [12,14,0.0215,0.0707,0.01816,0,0,1],
            [13,15,0.0744,0.2444,0.06268,0,0,1],
            [14,15,0.0595,0.195,0.0502,0,0,1],
            [12,16,0.0212,0.0834,0.0214,0,0,1],
            [15,17,0.0132,0.0437,0.0444,0,0,1],
            [16,17,0.0454,0.1801,0.0466,0,0,1],
            [17,18,0.0123,0.0505,0.01298,0,0,1],
            [18,19,0.01119,0.0493,0.01142,0,0,1],
            [19,20,0.0252,0.117,0.0298,0,0,1],
            [15,19,0.012,0.0394,0.0101,0,0,1],
            [20,21,0.0183,0.0849,0.0216,0,0,1],
            [21,22,0.0209,0.097,0.0246,0,0,1],
            [22,23,0.0342,0.159,0.0404,0,0,1],
            [23,24,0.0135,0.0492,0.0498,0,0,1],
            [23,25,0.0156,0.08,0.0864,0,0,1],
            [26,25,0,0.0382,0,0.96,0,1],
```

[25,27,0.0318,0.163,0.1764,0,0,1],
[27,28,0.01913,0.0855,0.0216,0,0,1],
[28,29,0.0237,0.0943,0.0238,0,0,1],
[30,17,0,0.0388,0,0.96,0,1],
[8,30,0.00431,0.0504,0.514,0,0,1],
[26,30,0.00799,0.086,0.908,0,0,1],
[17,31,0.0474,0.1563,0.0399,0,0,1],
[29,31,0.0108,0.0331,0.0083,0,0,1],
[23,32,0.0317,0.1153,0.1173,0,0,1],
[31,32,0.0298,0.0985,0.0251,0,0,1],
[27,32,0.0229,0.0755,0.01926,0,0,1],
[15,33,0.038,0.1244,0.03194,0,0,1],
[19,34,0.0752,0.247,0.0632,0,0,1],
[35,36,0.00224,0.0102,0.00268,0,0,1],
[35,37,0.011,0.0497,0.01318,0,0,1],
[33,37,0.0415,0.142,0.0366,0,0,1],
[34,36,0.00871,0.0268,0.00568,0,0,1],
[34,37,0.00256,0.0094,0.00984,0,0,1],
[38,37,0,0.0375,0,0.935,0,1],
[37,39,0.0321,0.106,0.027,0,0,1],
[37,40,0.0593,0.168,0.042,0,0,1],
[30,38,0.00464,0.054,0.422,0,0,1],
[39,40,0.0184,0.0605,0.01552,0,0,1],
[40,41,0.0145,0.0487,0.01222,0,0,1],
[40,42,0.0555,0.183,0.0466,0,0,1],
[41,42,0.041,0.135,0.0344,0,0,1],
[43,44,0.0608,0.2454,0.06068,0,0,1],
[34,43,0.0413,0.1681,0.04226,0,0,1],
[44,45,0.0224,0.0901,0.0224,0,0,1],
[45,46,0.04,0.1356,0.0332,0,0,1],
[46,47,0.038,0.127,0.0316,0,0,1],
[46,48,0.0601,0.189,0.0472,0,0,1],
[47,49,0.0191,0.0625,0.01604,0,0,1],
[42,49,0.0715,0.323,0.086,0,0,1],
[42,49,0.0715,0.323,0.086,0,0,1],
[45,49,0.0684,0.186,0.0444,0,0,1],
[48,49,0.0179,0.0505,0.01258,0,0,1],
[49,50,0.0267,0.0752,0.01874,0,0,1],
[49,51,0.0486,0.137,0.0342,0,0,1],
[51,52,0.0203,0.0588,0.01396,0,0,1],

[52,53,0.0405,0.1635,0.04058,0,0,1],
[53,54,0.0263,0.122,0.031,0,0,1],
[49,54,0.073,0.289,0.0738,0,0,1],
[49,54,0.0869,0.291,0.073,0,0,1],
[54,55,0.0169,0.0707,0.0202,0,0,1],
[54,56,0.00275,0.00955,0.00732,0,0,1],
[55,56,0.00488,0.0151,0.00374,0,0,1],
[56,57,0.0343,0.0966,0.0242,0,0,1],
[50,57,0.0474,0.134,0.0332,0,0,1],
[56,58,0.0343,0.0966,0.0242,0,0,1],
[51,58,0.0255,0.0719,0.01788,0,0,1],
[54,59,0.0503,0.2293,0.0598,0,0,1],
[56,59,0.0825,0.251,0.0569,0,0,1],
[56,59,0.0803,0.239,0.0536,0,0,1],
[55,59,0.04739,0.2158,0.05646,0,0,1],
[59,60,0.0317,0.145,0.0376,0,0,1],
[59,61,0.0328,0.15,0.0388,0,0,1],
[60,61,0.00264,0.0135,0.01456,0,0,1],
[60,62,0.0123,0.0561,0.01468,0,0,1],
[61,62,0.00824,0.0376,0.0098,0,0,1],
[63,59,0,0.0386,0,0.96,0,1],
[63,64,0.00172,0.02,0.216,0,0,1],
[64,61,0,0.0268,0,0.985,0,1],
[38,65,0.00901,0.0986,1.046,0,0,1],
[64,65,0.00269,0.0302,0.38,0,0,1],
[49,66,0.018,0.0919,0.0248,0,0,1],
[49,66,0.018,0.0919,0.0248,0,0,1],
[62,66,0.0482,0.218,0.0578,0,0,1],
[62,67,0.0258,0.117,0.031,0,0,1],
[65,66,0,0.037,0,0.935,0,1],
[66,67,0.0224,0.1015,0.02682,0,0,1],
[65,68,0.00138,0.016,0.638,0,0,1],
[47,69,0.0844,0.2778,0.07092,0,0,1],
[49,69,0.0985,0.324,0.0828,0,0,1],
[68,69,0,0.037,0,0.935,0,1],
[69,70,0.03,0.127,0.122,0,0,1],
[24,70,0.00221,0.4115,0.10198,0,0,1],
[70,71,0.00882,0.0355,0.00878,0,0,1],
[24,72,0.0488,0.196,0.0488,0,0,1],
[71,72,0.0446,0.18,0.04444,0,0,1],

[71,73,0.00866,0.0454,0.01178,0,0,1],
[70,74,0.0401,0.1323,0.03368,0,0,1],
[70,75,0.0428,0.141,0.036,0,0,1],
[69,75,0.0405,0.122,0.124,0,0,1],
[74,75,0.0123,0.0406,0.01034,0,0,1],
[76,77,0.0444,0.148,0.0368,0,0,1],
[69,77,0.0309,0.101,0.1038,0,0,1],
[75,77,0.0601,0.1999,0.04978,0,0,1],
[77,78,0.00376,0.0124,0.01264,0,0,1],
[78,79,0.00546,0.0244,0.00648,0,0,1],
[77,80,0.017,0.0485,0.0472,0,0,1],
[77,80,0.0294,0.105,0.0228,0,0,1],
[79,80,0.0156,0.0704,0.0187,0,0,1],
[68,81,0.00175,0.0202,0.808,0,0,1],
[81,80,0,0.037,0,0.935,0,1],
[77,82,0.0298,0.0853,0.08174,0,0,1],
[82,83,0.0112,0.03665,0.03796,0,0,1],
[83,84,0.0625,0.132,0.0258,0,0,1],
[83,85,0.043,0.148,0.0348,0,0,1],
[84,85,0.0302,0.0641,0.01234,0,0,1],
[85,86,0.035,0.123,0.0276,0,0,1],
[86,87,0.02828,0.2074,0.0445,0,0,1],
[85,88,0.02,0.102,0.0276,0,0,1],
[85,89,0.0239,0.173,0.047,0,0,1],
[88,89,0.0139,0.0712,0.01934,0,0,1],
[89,90,0.0518,0.188,0.0528,0,0,1],
[89,90,0.0238,0.0997,0.106,0,0,1],
[90,91,0.0254,0.0836,0.0214,0,0,1],
[89,92,0.0099,0.0505,0.0548,0,0,1],
[89,92,0.0393,0.1581,0.0414,0,0,1],
[91,92,0.0387,0.1272,0.03268,0,0,1],
[92,93,0.0258,0.0848,0.0218,0,0,1],
[92,94,0.0481,0.158,0.0406,0,0,1],
[93,94,0.0223,0.0732,0.01876,0,0,1],
[94,95,0.0132,0.0434,0.0111,0,0,1],
[80,96,0.0356,0.182,0.0494,0,0,1],
[82,96,0.0162,0.053,0.0544,0,0,1],
[94,96,0.0269,0.0869,0.023,0,0,1],
[80,97,0.0183,0.0934,0.0254,0,0,1],
[80,98,0.0238,0.108,0.0286,0,0,1],

```
        [80,99,0.0454,0.206,0.0546,0,0,1],
        [92,100,0.0648,0.295,0.0472,0,0,1],
        [94,100,0.0178,0.058,0.0604,0,0,1],
        [95,96,0.0171,0.0547,0.01474,0,0,1],
        [96,97,0.0173,0.0885,0.024,0,0,1],
        [98,100,0.0397,0.179,0.0476,0,0,1],
        [99,100,0.018,0.0813,0.0216,0,0,1],
        [100,101,0.0277,0.1262,0.0328,0,0,1],
        [92,102,0.0123,0.0559,0.01464,0,0,1],
        [101,102,0.0246,0.112,0.0294,0,0,1],
        [100,103,0.016,0.0525,0.0536,0,0,1],
        [100,104,0.0451,0.204,0.0541,0,0,1],
        [103,104,0.0466,0.1584,0.0407,0,0,1],
        [103,105,0.0535,0.1625,0.0408,0,0,1],
        [100,106,0.0605,0.229,0.062,0,0,1],
        [104,105,0.00994,0.0378,0.00986,0,0,1],
        [105,106,0.014,0.0547,0.01434,0,0,1],
        [105,107,0.053,0.183,0.0472,0,0,1],
        [105,108,0.0261,0.0703,0.01844,0,0,1],
        [106,107,0.053,0.183,0.0472,0,0,1],
        [108,109,0.0105,0.0288,0.0076,0,0,1],
        [103,110,0.03906,0.1813,0.0461,0,0,1],
        [109,110,0.0278,0.0762,0.0202,0,0,1],
        [110,111,0.022,0.0755,0.02,0,0,1],
        [110,112,0.0247,0.064,0.062,0,0,1],
        [17,113,0.00913,0.0301,0.00768,0,0,1],
        [32,113,0.0615,0.203,0.0518,0,0,1],
        [32,114,0.0135,0.0612,0.01628,0,0,1],
        [27,115,0.0164,0.0741,0.01972,0,0,1],
        [114,115,0.0023,0.0104,0.00276,0,0,1],
        [68,116,0.00034,0.00405,0.164,0,0,1],
        [12,117,0.0329,0.14,0.0358,0,0,1],
        [75,118,0.0145,0.0481,0.01198,0,0,1],
        [76,118,0.0164,0.0544,0.01356,0,0,1]
    ]
}
```

A.4 – Dados de entrada para a análise de estabilidade transitória do sistema de 9 barras

```
{
"info": ["ta"],
```

```
"optLF": [100, 10, 1e-3, 1],
"optTA": [60, 0, 1, 1e-2],
"bus": [
      [1,3,0,0,0,0,0,0,1.04,0,300,-300],
      [2,2,163,0,0,0,0,0,1.025,0,300,-300],
      [3,2,85,0,0,0,0,0,1.025,0,300,-300],
      [4,1,0,0,0,0,0,0,1,0,0,0],
      [5,1,0,0,90,30,0,0,1,0,0,0],
      [6,1,0,0,0,0,0,0,1,0,0,0],
      [7,1,0,0,100,35,0,0,1,0,0,0],
      [8,1,0,0,0,0,0,0,1,0,0,0],
      [9,1,0,0,125,50,0,0,1,0,0,0]
     ],
"branch":[
         [1,4,0,0.0576,0,1,0,1],
         [4,5,0.017,0.092,0.158,0,0,1],
         [5,6,0.039,0.17,0.358,0,0,1],
         [3,6,0,0.0586,0,1,0,1],
         [6,7,0.0119,0.1008,0.209,0,0,1],
         [7,8,0.0085,0.072,0.149,0,0,1],
         [8,2,0,0.0625,0,1,0,1],
         [8,9,0.032,0.161,0.306,0,0,1],
         [9,4,0.01,0.085,0.176,0,0,1]
      ],
"gen": [
      [1,1,1.20,0.02,0.14,0.14,0,0,0,0],
      [2,1,2.40,0.01,0.14,0.14,0,0,0,0],
      [3,2,5.74,0.02,1.93,1.77,0.25,0.25,5.2,0.81]
     ],
"exc": [
      [3,1,50,0.05,-0.17,0.95,0.04,1,0.014,1.55,-1.7,1.7]
     ],
"gov": [],
"event": [
       [1,2,7,0.2,1e-10],
       [1,2,7,0.3,0]
      ]
}
```