UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MIGUEL ANGELO DIMAS CASEMIRO GONÇALVES ADERALDO

DEMOCRATIZANDO A ANÁLISE DE REDES ELÉTRICAS DE DISTRIBUIÇÃO BRASILEIRAS COM BDGD-TOOLS: CONTRIBUIÇÕES DE DESENVOLVIMENTO DE APLICAÇÃO OPEN SOURCE PARA CONVERSÃO DE BDGD EM MODELOS DO OPENDSS

FORTALEZA

2023

MIGUEL ANGELO DIMAS CASEMIRO GONÇALVES ADERALDO

DEMOCRATIZANDO A ANÁLISE DE REDES ELÉTRICAS DE DISTRIBUIÇÃO BRASILEIRAS COM BDGD-TOOLS: CONTRIBUIÇÕES DE DESENVOLVIMENTO DE APLICAÇÃO OPEN SOURCE PARA CONVERSÃO DE BDGD EM MODELOS DO OPENDSS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Lucas Silveira Melo.
Coorientador: Eng. M.Sc Paulo Radatz.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A1d Aderaldo, Miguel Angelo Dimas Casemiro Gonçalves.
Democratizando a análise de redes elétricas de distribuição brasileiras com BDGD-Tools : contribuições de desenvolvimento de aplicação open source para conversão de bdgd em modelos do OpenDSS / Miguel Angelo Dimas Casemiro Gonçalves Aderaldo. – 2023.
90 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2023.
Orientação: Prof. Dr. Lucas Silveira Melo.

1. OpenDSS. 2. Base de Dados Geográfica (BDGD). 3. BDGD-Tools. 4. Redes Elétricas de Distribuição. I. Título.

CDD 621.3

MIGUEL ANGELO DIMAS CASEMIRO GONÇALVES ADERALDO

DEMOCRATIZANDO A ANÁLISE DE REDES ELÉTRICAS DE DISTRIBUIÇÃO BRASILEIRAS COM BDGD-TOOLS: CONTRIBUIÇÕES DE DESENVOLVIMENTO DE APLICAÇÃO OPEN SOURCE PARA CONVERSÃO DE BDGD EM MODELOS DO OPENDSS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em: 12/12/2023.

BANCA EXAMINADORA

Prof. Dr. Lucas Silveira Melo (Orientador)
Universidade Federal do Ceará (UFC)

Eng. M.Sc Paulo Radatz (Coorientador)
Electric Power Research Institute (EPRI)

Eng. M.Sc. Felipe Sampaio
Casa dos Ventos

Prof. Dr. Arnaldo José Pereira Rosentino Junior
Universidade Federal do Triângulo Mineiro (UFTM)

À minha mãe, por seu amor e apoio inabaláveis.

AGRADECIMENTOS

A minha família, em especial Antonia Casemiro e Luciene Casemiro, por sempre apoiarem minhas decisões, contribuírem para minha formação como cidadão e zelarem pelo meu bem-estar e felicidade, acima de tudo.

Expresso minha profunda gratidão ao Prof. Dr. Lucas Silveira Melo pela excelente orientação, fornecendo uma visão crítica fortemente fundamentada que contribuiu para o sucesso deste trabalho.

Ao Eng. M.Sc. Paulo Radatz pela sua orientação precisa e contribuição, fornecendo recursos e conexões fundamentais que elevaram a qualidade deste trabalho, alinhando-o às demandas do mercado.

Aos professores participantes da banca examinadora Eng. M.Sc. Felipe Sampaio e Prof. Dr. Arnaldo José Pereira Rosentino Junior pelo tempo dedicado e valiosas sugestões.

Aos meus amigos da engenharia elétrica, Mário Monteiro, Weslley Gonçalves e Lucas Costa, pelas reflexões, momentos de descontração e troca de conhecimentos ao longo da graduação.

A todos os meus professores da Universidade Federal do Ceará e da Ecole Centrale de Lille pelo papel fundamental que desempenham em nossa sociedade e, especialmente, pelo papel que tiveram em meu desenvolvimento como engenheiro e como pessoa.

Ao ensino público superior brasileiro por ser de altíssima qualidade, totalmente gratuito e acessível a todos, na esperança de que um dia possa alcançar até mesmo os cidadãos mais inacessíveis.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela bolsa de estudos e oportunidade que me foi concedida para realizar meu programa de duplo diploma na Ecole Centrale de Lille, onde pude adquirir competências complementares para este trabalho.

"O software é uma grande combinação entre arte e engenharia." (Bill Gates, 2005)

RESUMO

Estudos nas redes de distribuição de energia no Brasil desempenham um papel fundamental para garantir o funcionamento eficiente desse serviço vital, podendo ser feitos através de representações em softwares. A modelagem fiel dessas redes pode ser viabilizada através das informações fornecidas pela Base de Dados Geográfica da Distribuidora (BDGD) para simulações no OpenDSS. No entanto, os métodos disponíveis atualmente para esse processo impõem barreiras significativas, como complexidade elevada ou limitações de acesso. Nesse contexto, a aplicação livre BDGD-Tools surge como uma solução para a conversão das informações da BDGD em formato OpenDSS, buscando democratizar a análise das redes de distribuição do Brasil. O objetivo deste trabalho é contribuir para o desenvolvimento da BDGD-Tools, desde seu estágio inicial até uma versão totalmente funcional.A metodologia adotada envolveu o estudo aprofundado da biblioteca, identificação e implementação dos recursos essenciais remanescentes para sua funcionalidade e validação. Destacam-se, entre os recursos implementados, a definição das classes Python dos elementos, aprimoramentos nos módulos de conversão e utilidades, bem como a geração dos arquivos .dss para simulação. A validação da BDGD-Tools foi realizada através da comparação das redes elétricas modeladas pela ferramenta com o método oficial estabelecido pela ANEEL, que utiliza o programa GeoPerdas. Utilizou-se como estudo de caso representativo os dados da distribuidora Creluz do ano de 2022, localizada no Rio Grande do Sul. A avaliação concentrou-se na estrutura da rede elétrica convertida, no fluxo de potência e no perfil de tensão gerado. Os resultados obtidos demonstraram a consistência na representação dos elementos da rede elétrica. Ao comparas as duas redes, obteve-se erro aceitável de 4,1% para potência ativa total e 0,206 p.p. em perdas ativas totais no fluxo de potência e erros de 0% e 0,143% para tensões mínima e máxima identificadas nos perfis de tensão. Essa validação reforça a confiabilidade da BDGD-Tools na precisa tradução da rede elétrica da BDGD para o OpenDSS, proporcionando uma análise detalhada do desempenho da biblioteca em relação aos métodos convencionais da ANEEL. Por fim, são identificadas áreas potenciais para melhorias futuras, focando na experiência do usuário, flexibilidade na configuração da modelagem da rede e maior precisão na conversão.

Palavras-chave: OpenDSS; Base de Dados Geográfica (BDGD); BDGD-Tools, Redes Elétricas de Distribuição

ABSTRACT

Studies on energy distribution networks in Brazil play a fundamental role in ensuring the efficient operation of this vital service and can be carried out through representations in software. Accurate modeling of these networks can be made feasible by utilizing information provided by the Distributor's Geographic Database (BDGD) for simulations in OpenDSS. However, the currently available methods for this process impose significant barriers, such as high complexity or access limitations. In this context, the open-source application BDGD-Tools emerges as a solution for converting BDGD information into OpenDSS format, aiming to democratize the analysis of Brazil's distribution networks. The objective of this work is to contribute to the development of BDGD-Tools, from its initial stage to a fully functional version. The adopted methodology involved an in-depth study of the library, identifying and implementing the remaining essential features for its functionality and validation. Among the implemented features, the definition of Python classes for elements, improvements in conversion modules and utilities, as well as the generation of .dss files for simulation, stand out. Validation of BDGD-Tools was conducted by comparing the electric networks modeled by the tool with the official method established by ANEEL, which uses the GeoPerdas program. A representative case study utilized the data from the Creluz distributor in 2022, located in Rio Grande do Sul. The evaluation focused on the converted electric network structure, power flow, and voltage profile generated. The obtained results demonstrated consistency in the representation of the electric network elements. When comparing the two networks, an acceptable error of 4.1% for total active power and 0.206 percentage points in total active losses in power flow were achieved, along with errors of 0% and 0.143% for minimum and maximum voltages identified in the voltage profiles. This validation reinforces the reliability of BDGD-Tools in accurately translating the electric network from BDGD to OpenDSS, providing a detailed analysis of the library's performance compared to conventional ANEEL methods. Finally, potential areas for future improvements are identified, focusing on user experience, flexibility in network modeling configuration, and increased accuracy in conversion

Keywords: OpenDSS; Distributor Geographic Database; BDGD-Tools; Electrical Distribution Networks

LISTA DE FIGURAS

Figura 1 – Dados de linhas de média tensão da Rede de Distribuição do Ceará no software QGIS. 26
Figura 2 – Dados das entidades em tabelas acessados pelo QGIS 26
Figura 3 – Resumo das formas alternativas para conversão da BDGD em modelos elétricos. 35
Figura 4 – Mapeamento entre as entidades da BDGD e elementos do OpenDSS. As entidades são classificadas em diferentes elementos através das cores. 37
Figura 5 – Arquitetura da biblioteca BDGD-Tools. 38
Figura 6 – Funcionamento sob a ótica interna. 40
Figura 7 – Funcionamento sob a ótica do usuário. 41
Figura 8 – Arquitetura da BDGD-Tools com os recursos identificados a serem implementados em destaque. 43
Figura 9 – Etapas do processo de criação da conversão de entidades da BDGD para elemento do OpenDSS. A parte em verde é realizada no arquivo JSON. A parte azul na classe Elemento em Python. 45
Figura 10 – Fluxograma para o método *calculated*. 47
Figura 11 – Arquivo JSON. 49
Figura 12 – UML. 50
Figura 13 – Método __repr__() responsável por criar a linha de comando do elemento Line 52
Figura 14 – Configuração das colunas para importação da BDGD no arquivo JSON. . . 56
Figura 15 – UML Transformer. 57
Figura 16 – Método __repr__() responsável por criar a linha de comando do elemento Transformer 58
Figura 17 – Configuração das colunas para importação da tabela EQRE da BDGD através do arquivo JSON. 61
Figura 18 – UML RegControl. 62
Figura 19 – Método __repr__() responsável por criar a linha de comando do elemento RegControl 63
Figura 20 – Configuração das colunas para importação da tabela CRVCRG da BDGD através do arquivo JSON. 65
Figura 21 – UML LoadShape. 66

Figura 22 – Método __repr__() responsável por criar a linha de comando do elemento LoadShape 67
Figura 23 – Visão geral da representação de uma unidade consumidora para modelagem no OpenDSS. 69
Figura 24 – Configuração das colunas para importação da tabela da entidade UCMT através do arquivo JSON. 73
Figura 25 – UML Load. 74
Figura 26 – Método __repr__() responsável por criar a linha de comando para os elementos *Loads* 75
Figura 27 – Representação geográfica da rede elétrica de distribuição gerida pela Creluz através de sua BDGD de 2022 no software QGIS. 80
Figura 28 – Resultado do fluxo de potência para a rede elétrica resultado da conversão da bdgd-tool para a BDGD da Creluz. 84
Figura 29 – Resultado do fluxo de potência para a rede elétrica resultado do método da ANEEL para a BDGD da Creluz. 84
Figura 30 – Resultado do perfil de tensão para a rede elétrica da conversão da bdgd-tool para a BDGD da Creluz. 85
Figura 31 – Resultado do perfil de tensão para a rede elétrica do método da ANEEL para a BDGD da Creluz. 85

LISTA DE TABELAS

Tabela 1 – Método Estático 48
Tabela 2 – Método Direto 48
Tabela 3 – Método Indireto 48
Tabela 4 – Método Calculado 49
Tabela 5 – Método Direto UNTRMT 55
Tabela 6 – Método Indireto UNTRMT 55
Tabela 7 – Método Calculado UNTRMT 55
Tabela 8 – Método Estático - RegControl 60
Tabela 9 – Método Direto - RegControl 60
Tabela 10 – Método Indireto - RegControl 60
Tabela 11 – Método Calculado - RegControl 60
Tabela 12 – Método Estático Loadshape 65
Tabela 13 – Método Direto Loadshape 65
Tabela 14 – Método Estático Load - UCMT 71
Tabela 15 – Método Direto Load - UCMT 72
Tabela 16 – Método Indireto Load - UCMT 72
Tabela 17 – Informações sobre a rede elétrica no OpenDSS gerada a partir do resultado da conversão. 81
Tabela 18 – Informações sobre a rede elétrica no OpenDSS gerada a partir do método da ANEEL. 82

SUMÁRIO

1 INTRODUÇÃO 15
1.0.1 Estrutura do Trabalho 16
2 BASE DE DADOS GEOGRÁFICA DA DISTRIBUIDORA - BDGD . . 18
2.1 Apresentação do Capítulo 18
2.2 Contextualização 18
2.3 Estrutura 19
2.3.1 Entidades 19
2.3.1.1 Circuito de Média Tensão - CTMT *20*
2.3.1.2 Segmento Condutor - SEGCON *20*
2.3.1.3 Segmento do Sistema de Distribuição de Média Tensão - SSDMT *20*
2.3.1.4 Segmento do Sistema de Distribuição de Baixa Tensão - SSDBT *21*
2.3.1.5 Unidade Seccionadora de Média Tensão - UNSEMT *21*
2.3.1.6 Ramal de Ligação - RAMAL *21*
2.3.1.7 Unidade Reguladora de Média Tensão - UNREMT *22*
2.3.1.8 Equipamento Regulador - EQRE *22*
2.3.1.9 Unidade Transformadora de Média Tensão - UNTRMT *23*
2.3.1.10 Unidade Transformadora de Média Tensão - UNTRMT *23*
2.3.1.11 Equipamento Transformador de Média Tensão - EQTRMT *24*
2.3.1.12 Unidade Consumidora de Média Tensão - UCMT *24*
2.3.1.13 Unidade Consumidora de Baixa Tensão - UCBT *24*
2.3.1.14 Ponto de Iluminação Pública - PIP *25*
2.3.1.15 Curva de Carga - CRVCRG *25*
2.4 Quantum Gis: software para leitura da BDGD 26
2.5 Considerações Finais 27
3 OPENDSS 28
3.1 Introdução ao capitulo 28
3.2 Elementos 28
3.2.1 Circuit 28
3.2.2 Transformer 29
3.2.3 LineCode 30

3.2.4 ***Line*** . . . 31
3.2.5 ***RegControl*** . . . 32
3.2.6 ***Reactor*** . . . 32
3.3 **Considerações Finais** . . . 33
4 **BDGD-TOOLS** . . . 34
4.1 **Apresentação do Capítulo** . . . 34
4.2 **Contextualização** . . . 34
4.3 **Características Principais** . . . 36
4.4 **Mapeamento** . . . 36
4.5 **Arquitetura** . . . 38
4.6 **Funcionamento** . . . 40
4.7 **Considerações Finais** . . . 41
5 **METODOLOGIA** . . . 42
5.1 **Contextualização** . . . 42
5.2 **Ferramentas e Tecnologia** . . . 43
5.2.1 ***Ferramentas de Desenvolvimento*** . . . 43
5.2.2 ***Bibliotecas para Implementação de Recursos*** . . . 44
5.3 **Implementação** . . . 44
5.3.1 ***Line*** . . . 46
5.3.1.1 *Considerações Iniciais e Implementação de Pré-Requisitos* . . . *46*
5.3.1.2 *Arquivo JSON* . . . *48*
5.3.1.3 *Classe Line* . . . *49*
5.3.2 ***Transformer com Reator*** . . . 54
5.3.2.1 *Considerações Iniciais e Implementação de pré-requisitos* . . . *54*
5.3.2.2 *Arquivo JSON* . . . *55*
5.3.2.3 *Classe Transformer* . . . *57*
5.3.3 ***RegControl*** . . . 59
5.3.3.1 *Considerações Iniciais/Implementação de pré-requisitos/Desafios* . . . *59*
5.3.3.2 *Arquivo JSON* . . . *60*
5.3.3.3 *Classe RegControl* . . . *62*
5.3.4 ***LoadShape*** . . . 63
5.3.4.1 *Considerações Iniciais/Implementação de pré-requisitos* . . . *64*

5.3.4.2 Arquivo JSON *64*
5.3.4.3 Classe LoadShape *66*
5.3.5 Load 67
5.3.5.1 Considerações Iniciais *68*
5.3.5.2 Cálculo da Demanda Máxima (kW) *69*
5.3.5.3 Arquivo JSON *71*
5.3.5.4 Classe Load *73*
5.3.6 Utilidades e Conversões 75
5.3.6.1 Utilidades *75*
5.3.6.1.1 Manipulação de Dados 75
5.3.6.1.2 Criação de Arquivos .DSS 76
5.3.6.2 Conversões *77*
5.3.6.2.1 Processamento de Dados de Carga 77
5.4 Considerações Finais do Capítulo 78
6 RESULTADOS 79
6.1 Considerações Iniciais 80
6.2 Resultados e Validação: Estrutura da Rede Elétrica no OpenDSS 80
6.2.1 Resultados da Estrutura da Rede Elétrica no OpenDSS 80
6.2.2 Validação da Estrutura da Rede Elétrica no OpenDSS 82
6.3 Resultados e Validação: Fluxo de Carga 83
6.4 Resultados e Validação: Perfil de Tensão 85
6.5 Ambiente de Registro de Erros em Comparação com a Rede de Referência 86
6.6 Trabalhos Futuros 87
6.7 Considerações Finais 88
7 CONCLUSÕES E TRABALHOS FUTUROS 89
REFERÊNCIAS 90

1 INTRODUÇÃO

A distribuição de energia elétrica desempenha um importante papel na sociedade contemporânea, sendo fundamental para garantir um fornecimento confiável e eficiente de eletricidade. Assim, análises na rede de forma contínua se fazem essenciais para garantir o bom funcionamento, ainda mais no contexto dos desafios para a rede com a transição energética.

Em um contexto de estudos para estas redes de distribuição, faz-se necessário o uso de simulações. De uma perspectiva acadêmica, isso dá-se para investigação de impactos na rede com a inserção de novas tecnologias, principalmente. Do ponto de vista do mercado, ou seja, das distribuidoras, isso dá-se para planejamento e cálculo de perdas técnicas.

Hoje, para análise e simulação das redes elétricas de distribuição do Brasil, pode-se usar as informações adaptadas da Base de Dados Geográfica da Distribuidora (BDGD) para modelá-las em um software de sistemas de potência, como o OpendSS. Contudo, redes de distribuição possuem uma quantidade de equipamentos escalável, com acentuada complexidade em sua estrutura. Assim, torna-se inviável modelá-las em um software manualmente.

Nesse contexto, existem alternativas que automatizam esse processo, porém não asseguram um acesso fácil e aberto para todos.

Entre eles, está o tradicional método da ANEEL para modelar as redes no OpenDSS e calcular as perdas técnicas das redes de distribuição, que impactam nos reajustes das tarifas. Neste processo, a complexidade dos softwares oficiais da ANEEL, GeoPerdas e ProgGeoPerdas, gera uma forte barreira para acesso comum, além de ter uso direcionado para as concessionárias.

Além disso, embora existam outras ferramentas para conversão da BDGD em sistemas elétricos modelados no OpenDSS, como o SigPerdas, a acessibilidade é restrita e sua aplicação limitada ao âmbito comercial.

Neste cenário, é visto que a análise e simulação das redes elétricas de distribuição do Brasil, especialmente no contexto das exigências técnicas estabelecidas pela Agência Nacional de Energia Elétrica (ANEEL), enfrentam desafios a ser superados pelas concessionárias e pela academia.

Diante dessas limitações, a BDGD-Tools foi concebida como uma resposta às lacunas identificadas na conversão de dados da BDGD para o formato OpenDSS. Seu propósito não se restringe apenas a replicar os cálculos de perdas da ANEEL, mas também visa abrir novas possibilidades acadêmicas e práticas. Representa um avanço significativo ao permitir estudos em redes reais de distribuição de energia de forma acessível e livre, promovendo um ambiente

colaborativo e de ampla utilização (BDGD-Tools, 2023).

Assim, este trabalho busca realizar contribuições para o desenvolvimento da ferramenta desde seu estado inicial até sua versão plenamente funcional. Para alcançar este objetivo, este trabalho delineia objetivos específicos. Primeiramente, visa-se converter os dados da BDGD de forma pontual para o formato OpenDSS, preenchendo os parâmetros de modelagem de cada equipamento da rede e considerando todas as suas nuances. Em segundo lugar, busca-se estabelecer um ambiente de validação robusto para aferir a consistência e precisão dos resultados obtidos. Esses objetivos específicos são fundamentais para cumprir o propósito mais amplo de aprimorar e tornar a ferramenta completamente funcional.

1.0.1 Estrutura do Trabalho

Este estudo está estruturado em capítulos que abordam a fundamentação teórica, a metodologia empregada e os resultados obtidos.

No Capítulo 1 é apresentada a introdução, com as motivações e objetivos do trabalho, assim como a estrutura geral.

Para o Capítulo 2 é discutida a Base de Dados da Distribuidora (BDGD) e sua importância no contexto da distribuição de energia. São abordados aspectos como a estrutura da base de dados, assim como a descrição das entidades que representam os equipamentos da rede. O Capítulo é de fundamental importância para a base de conhecimento para o entendimento da ideia de conversão dos dados neste trabalho.

No Capítulo 3 são explorados os fundamentos do OpenDSS, software essencial para a análise de sistemas elétricos de distribuição. É feita uma breve explicação dos elementos e seus parâmetros para modelagem das redes elétricas. Estes serão discorridos ao longo do trabalho, visto que é o formato para o qual os dados da BDGD são convertidos.

Já no Capítulo 4 é abordada a apresentação detalhada da BDGD-Tools, desde sua arquitetura até seu funcionamento interno, assim como seu estado inicial antes do desenvolvimento deste trabalho.

Em seguida, o Capítulo 5 descreve a metodologia adotada. Inicialmente, é feita uma contextualização das preparações para o trabalho, seguida da explicação das ferramentas e tecnologias utilizadas no desenvolvimento. Além disso, é explicada a implementação, estando este tópico em função dos elementos do OpenDSS: L*ineCode, Line, Transformer, Load, LoadShape e RegControl*. Para cada elemento, são descritas considerações iniciais e etapas do processo de

aplicação da conversão. Por último, uma seção descrevendo funções de utilidades e conversão é apresentada, que complementam a implementação dos elementos.

Por fim, no Capítulo 6, este trabalho é concluído ao apresentar os resultados obtidos e suas análises. É realizada uma comparação entre uma rede gerada pela BDGD-Tools e outra resultante dos métodos oficiais da ANEEL no OpenDSS. A análise compara suas estruturas, perfis de tensão e fluxo de carga. Adicionalmente, é desenvolvido um ambiente gerador de log de erros para justificar discrepâncias e identificar possíveis trabalhos futuros.

2 BASE DE DADOS GEOGRÁFICA DA DISTRIBUIDORA - BDGD

2.1 Apresentação do Capítulo

O capítulo tem como objetivo explicar a BDGD e seus conceitos que são utilizados neste trabalho. De início, é feita uma contextualização sobre a existência dessa base de dados no Brasil. No tópico seguinte, é explorada a estrutura da base de dados. Em seguida, as principais entidades são abordadas. Por fim, são apresentadas considerações finais sobre o capítulo.

2.2 Contextualização

Uma base de dados georreferenciada é de extrema importância para o setor elétrico. Segundo Crispino, o Sistema de Informação Geográfica (SIG) desempenha papel vital em atividades de planejamento, manutenção, monitoramento, operação, regulação e fiscalização. Essa importância se destaca ainda mais quando se trata de empresas de distribuição de energia elétrica, visto que gerenciam extensas redes e grande quantidade de equipamentos (CRISPINO, 2001).

No Brasil, como parte do seu papel regulador, a Agência Nacional de Energia Elétrica (ANEEL) utiliza o Sistema de Informações Georreferenciadas da Regulação (SIG-R) (ANEEL, 2009). O SIG-R trata-se de um conjunto de diversos sistemas e uma base de dados com informações sobre o sistema de distribuição. No PRODIST 10, a Base de Dados Geográfica da Distribuidora (BDGD) é definida como um dos principais elementos do SIG-R (ANEEL, 2021b).

A BDGD é uma representação do sistema elétrico real em um determinado período, onde são apresentadas a situação dos ativos e informações técnicas e comerciais de interesse.

Anualmente, cada distribuidora tem como obrigação coletar e formatar os dados sobre a estrutura da rede elétrica nos padrões fornecidos pela ANEEL, indicando o método de aquisição de dados (ANEEL, 2021b).

As geodatabases, BDGDs, são disponibilizadas no Portal Dados Abertos da ANEEL (ANEEL, 2021a), onde é possível acessar as informações de cada distribuidora do Brasil (ANEEL, 2017).

Este tópico estabelece o contexto da BDGD. Nas seções subsequentes, serão abordados a estrutura e detalhes sobre a BDGD.

2.3 Estrutura

Esta seção tem como objetivo apresentar a estrutura da BDGD e suas principais entidades que serão abordadas neste trabalho.

O módulo 10 do PRODIST é responsável por definir todos os padrões da BDGD, seja de formato de dados ou dos arquivos da entrega.

Segundo o documento (PRODIST, 10), a BDGD informa dados técnicos de distribuição, dados comerciais e dados físico-contábeis da base de ativos. Em relação ao modelo geográfico, ela contém o traçado geométrico dos segmentos da rede; a localização geográfica das estruturas de suporte, dos acessantes e equipamentos; e a delimitação das subestações e outras regiões.

A estrutura da BDGD está organizada em dois principais conjuntos:

- **Entidades geográficas:** representam componentes do mundo real necessariamente com um contexto geográfico - serão representadas no SIG-R. Cada entidade deste tipo possui nome, sigla, tipo de feição e descrição. Por exemplo, subestações, unidades consumidoras e redes elétricas.
- **Entidades não geográficas:** representam estruturas de informação que estão atreladas a outras entidades, mas não possuem contexto geográfico. Cada entidade deste tipo possui nome, sigla e descrição. Por exemplo, registro de perdas técnicas e não técnicas.

Uma entidade é um modelo de dados que representa um equipamento ou estrutura de interesse. Elas possuem campos de dados que podem ser abertos ou fechados. Para campo aberto é permitido o livre preenchimento, enquanto que para o campo fechado são seguidos determinados padrões pré-estabelecidos.

Como parte desses padrões, as informações constantes na BDGD são definidas por regras compreendidas no Dicionário de Dados da ANEEL (DDA). O DDA também é definido no módulo 10 do PRODIST (ANEEL, 2021b).

Nas seções seguintes, abordaremos as principais entidades e suas informações básicas que têm relevância para o desenvolvimento deste trabalho.

2.3.1 Entidades

Nesta seção serão apresentadas as entidades mais relevantes para modelar um sistema elétrico de distribuição real.

2.3.1.1 Circuito de Média Tensão - CTMT

Representa um circuito de média tensão existente no sistema de distribuição.

- Código de identificação exclusiva do circuito
- Código do barramento associado
- Subestação associada
- Código do ponto de acoplamento comum
- Tensão de operação do sistema em p.u.
- Energia ativa mensal do ano referente à BDGD

2.3.1.2 Segmento Condutor - SEGCON

Essa entidade representa um agrupamento de tipo de condutores do sistema de distribuição.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada segmento condutor
- Número de condutores por fase
- Resistência e reatância da sequência positiva
- Correntes nominal e máxima do condutor
- Informações de controle patrimonial

2.3.1.3 Segmento do Sistema de Distribuição de Média Tensão - SSDMT

Essa entidade representa o traçado de um segmento de rede de distribuição em nível de média tensão.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada segmento de rede
- Código do circuito de média tensão atrelado
- Código dos pontos de acoplamento.
- Comprimento do segmento em metros, considerando os efeitos da catenária e do relevo.
- Código do tipo de segmento condutor
- Código das fases de conexão
- Subestação
- Informações de controle patrimonial

2.3.1.4 Segmento do Sistema de Distribuição de Baixa Tensão - SSDBT

Essa entidade representa o traçado de um segmento de rede de distribuição em nível de baixa tensão.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada segmento de rede
- Código do circuito de baixa tensão atrelado
- Código dos pontos de acoplamento.
- Comprimento do segmento em metros, considerando of efeitos da catenária e do relevo.
- Código do tipo de segmento condutor
- Código das fases de conexão
- Subestação atrelada
- Informações de controle patrimonial

2.3.1.5 Unidade Seccionadora de Média Tensão - UNSEMT

Essa entidade representa a localização de uma unidade seccionadora instalada na rede de distribuição em nível de média tensão.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada seccionadora
- Código do circuito de baixa tensão atrelado
- Código dos pontos de acoplamento.
- Código do tipo de segmento condutor
- Capacidade do elo fusível
- Corrente nominal
- Código das fases de conexão
- Subestação atrelada
- Informações de controle patrimonial

2.3.1.6 Ramal de Ligação - RAMAL

Cada registro desta entidade representa um ramal de ligação de um acessante

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada ramal.

- Código do circuito atrelado.
- Código dos pontos de acoplamento.
- Código do tipo de segmento condutor
- Código das fases de conexão
- Comprimento do ramal
- Subestação atrelada
- Informações de controle patrimonial

2.3.1.7 Unidade Reguladora de Média Tensão - UNREMT

Essa entidade representa a localização de uma unidade reguladora instalada na rede de distribuição em nível de média tensão.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada unidade reguladora
- Código das fases de conexão
- Código dos pontos de acoplamento.
- Código do circuito de média tensão atrelado
- Capacidade do elo fusível
- Subestação atrelada
- Informações de controle patrimonial

2.3.1.8 Equipamento Regulador - EQRE

Essa entidade representa um equipamento regulaor do distema de distribuição, que compõe a entidade unidade reguladora.

Possui as seguintes informações básicas:

- Código de identificação exclusiva para cada equipamento regulador
- Código das fases de conexão no primário e secundário
- Potência e corrente nominais
- Tensão d regulação (p.u.)
- TP e TC
- Código dos pontos de acoplamento
- Perdas
- Resistência e Reatância

– Informações de controle patrimonial

2.3.1.9 Unidade Transformadora de Média Tensão - UNTRMT

Deve incluir todas as unidades transformadoras de média tensão, que possuam informação referente ao período dos dados, representado geograficamente pela sua localização. Possui as seguintes informações básicas:

– Código de identificação exclusiva para cada unidade transformadora.
– Referência das fases de conxões primárias, secundárias e terciárias.
– Potência nominal aparente.
– Código dos pontos de acoplamento.
– Código do circuito de média tensão atrelado.
– Capacidade do elo fusível.
– Subestação atrelada.
– Informações de controle patrimonial.
– Porcentagem de perda no ferro (%).
– Porcentagem de perda total (%).
– Tipo do transformador.

2.3.1.10 Unidade Transformadora de Média Tensão - UNTRMT

Deve incluir todas as unidades transformadoras de média tensão, que possuam informação referente ao período dos dados, representado geograficamente pela sua localização. Possui as seguintes informações básicas:

– Código de identificação exclusiva para cada unidade transformadora.
– Referência das fases de conexões primárias, secundárias e terciárias.
– Potência nominal aparente.
– Código dos pontos de acoplamento.
– Código do circuito de média tensão atrelado.
– Capacidade do elo fusível.
– Subestação atrelada.
– Informações de controle patrimonial.
– Porcentagem de perda no ferro (%).
– Porcentagem de perda total (%).

– Tipo do transformador.

2.3.1.11 Equipamento Transformador de Média Tensão - EQTRMT

Cada registro desta entidade representa um equipamento transformador de média tensão instalado no sistema de distribuição Possui as seguintes informações básicas:

– Código de identificação exclusiva para cada unidade transformadora.
– Referência das fases de conexão.
– Tensão nominal do primário, secundário e terciário
– Código dos pontos de acoplamento.
– Informações de controle patrimonial.
– Porcentagem de perda no ferro (%).
– Porcentagem de perda total (%).

2.3.1.12 Unidade Consumidora de Média Tensão - UCMT

Essa entidade representa a localização da unidade consumidora com característica de consumo em média tensão do sistema de distribuição.

Possui as seguintes informações básicas:

– Código de identificação da unidade consumidora.
– Código do ponto de acoplamento comum.
– Código das fases de conexão.
– Dados de geração, se for o caso.
– Identificador do circuito de média tensão, subestação e transformador.
– Tipo da tipologia de curva de carga associada.
– Carga instalada.
– Energia ativa e demanda ativa mensal para os meses do período da BDGD.

2.3.1.13 Unidade Consumidora de Baixa Tensão - UCBT

Essa entidade representa a localização da unidade consumidora com característica de consumo em baixa tensão do sistema de distribuição.

Possui as seguintes informações básicas:

– Código de identificação da unidade consumidora.

- Código do ponto de acoplamento comum.
- Código das fases de conexão.
- Dados de geração, se for o caso.
- Identificador do circuito de média tensão, subestação e transformador.
- Tipo da tipologia de curva de carga associada.
- Carga instalada.
- Energia ativa para os meses do período da BDGD.

2.3.1.14 Ponto de Iluminação Pública - PIP

Essa entidade representa um ponto de iluminação pública sem medição individual no sistema de distribuição.

Possui as seguintes informações básicas:

- Código identificador único do ponto de iluminação pública
- Código da subestação e transformador e circuito de média tensão atrelados
- Referência das fases de conexão
- Grupo tarifário
- Ponto de acoplamento comum
- Carga instalada (W)
- Tipologia de curva de carga associada
- Potência Unitária do Ponto de ilumunicação, Reator e Relefotoelétrico
- Perdas unitárias dos equiamentos auxiliares
- Energia ativa mensal

2.3.1.15 Curva de Carga - CRVCRG

A curva de carga não é definida como uma entidade da BDGD. No entanto, é uma tabela da base de dados que possui informações para definir os tipos de unidades consumidoras e suas determinadas curvas de carga.

Possui as seguintes informações básicas:

- Referência do tipo de carga: residencial ou rual, por exemplo.
- Potência da carga para períodos de 15 minutos de um dia, totalizando 96 valores.
- Tipo de dia da carga específica: podendo ser dias úteis, sábados ou domingos e feriados.

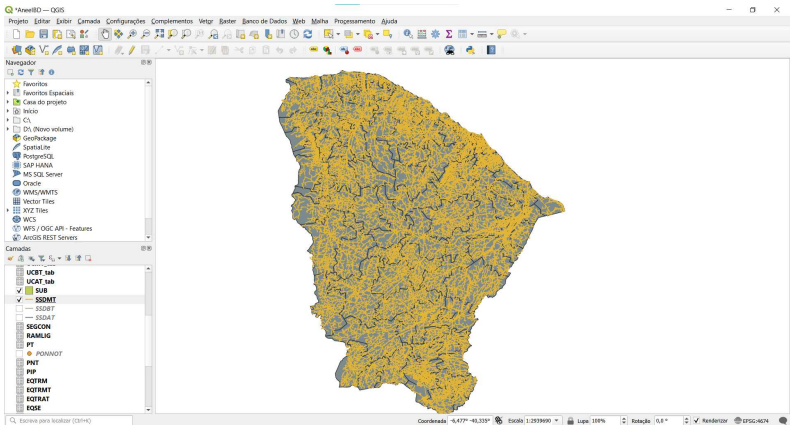
2.4 Quantum Gis: software para leitura da BDGD

QGIS é um software aberto com uma interface amigável para o Sistema de Informação Geográfico. Ele é gerenciado pela Fundação Geoespacial de Còdigo Aberto (OSGEO).

O software é uma alternativa para leitura dos dados da BDGD, visto que possui suporte para dados geográficos. A partir dele, é possível visualizar todas as entidades geográficas em um mapa, podendo filtrar as informações desejadas. Ainda, as tabelas de cada entidade podem ser acessadas facilmente.

A Figura 1 ilustra informações da BDGD da concessionária ENEL CE de 2021 (ANEEL, 2023), com as camadas de conjuntos e linhas de média tensão expostas (ANEEL, 2023).

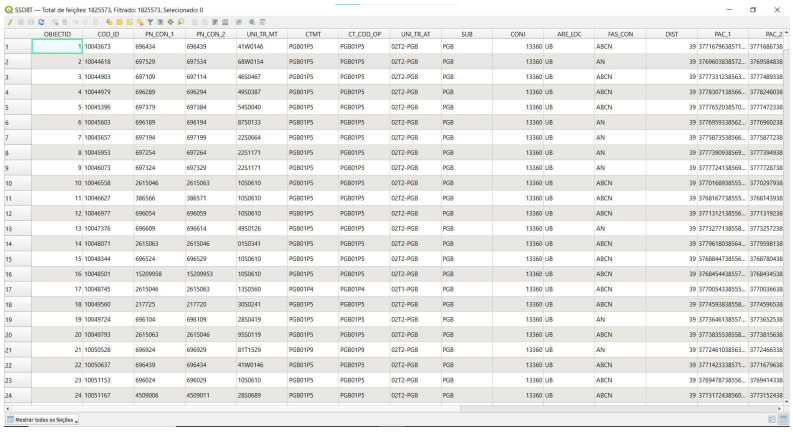
Figura 1 – Dados de linhas de média tensão da Rede de Distribuição do Ceará no software QGIS.



Fonte: Autor

A Figura 2 apresenta os dados em formato de tabela. As informações são sobre todas as entidades do tipo SEGCON, mencionado anteriormente na Seção 2.3.1.2.

Figura 2 – Dados das entidades em tabelas acessados pelo QGIS



Fonte: Autor

2.5 Considerações Finais

Neste capítulo, foram aprofundados os conhecimentos na BDGD no cenário do setor elétrico brasileiro. Antes de tudo, foi destacada a importância de bases geo-referenciadas para o setor de distribuição, progredindo para a estrutura detalhada da BDGD e suas principais entidades. O capítulo 2 é concluído com uma introdução ao software Quantum GIS como uma ferramenta para a leitura e manipulação de dados da BDGD. Todas essas informações dão suporte para a aplicação da metodologia discutida nos próximos capítulos.

3 OPENDSS

3.1 Introdução ao capitulo

Este capítulo é de fundamental importância para uma melhor compreensão da conversão da Base de Dados de Geográfica da Distribuidora (BDGD) para o OpenDSS. Ele explora os elementos essenciais do OpenDSS, fundamentais para modelar sistemas elétricos. Esse conhecimento é vital para implementar com precisão a conversão da BDGD para o OpenDSS, sendo utilizado pelas distribuidoras de energia e reconhecido pela ANEEL para cálculos de perdas em sistemas elétricos.

A interpretação precisa da modelagem dos equipamentos elétricos no OpenDSS é essencial para garantir uma conversão correta. Esses elementos formam a base teórica necessária para representar sistemas elétricos, assegurando resultados precisos na simulação.

3.2 Elementos

Nesta seção, cada elemento-chave do OpenDSS é apresentado de forma concisa, incluindo sua definição sucinta, modelagem matemática quando aplicável, e exemplificação do modelo no software, detalhando seus parâmetros relevantes.

3.2.1 Circuit

Definindo-o, esse componente representa uma fonte trifásica de tensão simétrica, ou seja, três fontes senoidais de tensão com a mesma amplitude e um espaçamento angular de 120 graus entre elas. Além disso, as impedâncias próprias e as impedâncias mútuas são equivalentes entre si.

Para o entendimento deste trabalho, é de fundamental importância entender a declaração dos elementos. Para o circuito, o Comando 4 é um exemplo de definição do elemento com seus respectivos parâmetros. (RADATZ; ROCHA, 2017)

```
1 New Circuit.TheveninEquivalente bus1=PontoThevenin pu=1.1
   basekv=13.8 Z0=[0.025862916, 0.077588748] Z1
   =[0.023094242, 0.092376969]
```

Comando 1 – Linha de declaração de um elemento Line

Onde,

- **New Circuit**: Declaração para criar um novo circuito
- **TheveninEquivalente**: Nome atribuído ao circuito
- **bus1=PontoThevenin pu=1.1 basekv=13.8**: Especifica a barra de conexão do elemento TheveninEquivalente. PontoThevenin é o nome da barra, pu=1.1 indica a tensão em per unit (pu) de operação, e basekv=13.8 especifica a tensão base em kilovolts
- **Z0=[0.025862916, 0.077588748]**: Define a impedância de sequência zero do elemento TheveninEquivalente. Os valores [0.025862916, 0.077588748] correspondem à parte real e imaginária, respectivamente
- **Z1=[0.023094242, 0.092376969]**: Define a impedância de sequência positiva do elemento TheveninEquivalente. Os valores [0.023094242, 0.092376969] correspondem à parte real e imaginária, respectivamente

3.2.2 Transformer

Os transformadores de potência desempenham um papel crucial nos sistemas elétricos de potência. Eles viabilizam a geração e transmissão eficientes de eletricidade em níveis de tensão mais econômicos e tecnicamente apropriados.

Diversos circuitos equivalentes de transformadores são discutidos na literatura, e a escolha do modelo mais apropriado geralmente está condicionada ao contexto de aplicação. Neste trabalho, os modelos apresentados foram alinhados com os padrões adotados pelo OpenDSS, identificados como:

- **Transformador Monofásico**: Transfere eficientemente energia entre diferentes níveis de tensão, possuindo enrolamentos primário e secundário.
- **Transformador Trifásico**: Crucial para transferir energia entre redes com diferentes níveis de tensão.
- **Transformador de Fase Dividida**: Específico para aplicações particulares, o transformador de fase dividida, uma variação monofásica, oferece flexibilidade ao conectar sistemas em diferentes níveis de tensão.

A declaração de um modelo de transformador no OpenDSS pode ser exemplificada pelo seguinte comando:

Em que:

- **New "Transformer.TRF_1_165"**: Cria um novo transformador com o nome especificado.

```
New "Transformer.TRF_1_165" phases=3 windings=2 buses
   =["86851.1.2.3" "ET1_165.1.2.3.4"] conns=[Delta Wye] kvs
   =[13.8 0.38] taps=[1.045 1.045] kvas=[45.0 45.0] %
   loadloss=1.853 %noloadloss=0.478
```

Comando 2 – Linha de declaração de um elemento Transformador

- **phases=3**: Define que o transformador possui três fases.
- **windings=2**: Indica que o transformador possui dois enrolamentos.
- **buses**: Lista os barramentos conectados aos enrolamentos do transformador.
- **conns**: Especifica a configuração de conexão para cada enrolamento (Delta, Wye, etc.).
- **kvs**: Define as tensões nominais de cada enrolamento em kilovolts.
- **taps**: Especifica os ajustes de tap para cada enrolamento.
- **kvas**: Define as capacidades de cada enrolamento em kilovolt-amperes.
- **%loadloss** e **%noloadloss**: Indicam as perdas carregadas e sem carga do transformador, respectivamente.

Esta declaração mostra detalhes específicos do modelo de um transformador no OpenDSS, incluindo suas configurações de conexão, tensões, ajustes de tap e capacidades.

3.2.3 LineCode

Os LineCodes no OpenDSS desempenham um papel crucial na definição das características elétricas das linhas. Eles representam os parâmetros que descrevem o comportamento elétrico das linhas, incluindo resistência, reatância e condutância, entre outros.

Para exemplificar a declaração de um LineCode no OpenDSS, é considerado o seguinte comando:

```
New LineCode.NomeDoLineCode R1=0.5 X1=0.2 R0=1.0 X0=0.4 C1
   =0.001 C0=0.0005 units=km
```

Comando 3 – Linha de declaração de um LineCode

Em que:

- **New LineCode.NomeDoLineCode**: Cria um novo LineCode com o nome especificado.
- **R1, X1, R0, X0, C1, C0**: Representam os parâmetros de resistência, reatância e condutân-

cia do LineCode para fases e sequências zero.

– **units=km**: Define a unidade de comprimento da linha para quilômetros.

Este exemplo ilustra a declaração de um LineCode no OpenDSS, demonstrando como são especificados os parâmetros que definem as características elétricas das linhas no software.

3.2.4 Line

As linhas no OpenDSS desempenham um papel crucial na configuração dos sistemas elétricos. Elas representam as conexões físicas entre os equipamentos e possuem diferentes configurações, como linhas monofásicas ou trifásicas, com ou sem neutro. Elas têm a capacidade de adotar várias configurações, como exemplificado por uma linha trifásica à 4 fios.

Para ilustrar a declaração de uma linha no OpenDSS, é considerado o seguinte comando:

```
New Line.NomeDaLinha phases=3 bus1=BarraOrigem bus2=
   BarraDestino linecode=NomeDoLineCode length=5 units=km
```

Comando 4 – Linha de declaração de um elemento Line

Em que:

– **New Line.NomeDaLinha**: Cria uma nova linha com o nome especificado.

– **phases=3**: Define que a linha possui três fases.

– **bus1, bus2**: Especifica as barras de origem e destino da linha.

– **linecode=NomeDoLineCode**: Associa a linha ao LineCode específico que define suas características elétricas.

– **length=5 units=km**: Define o comprimento da linha em quilômetros.

Este exemplo ilustra a declaração de uma linha no OpenDSS, demonstrando como são especificados os parâmetros que definem as conexões físicas e as características elétricas da linha no software.

3.2.5 RegControl

Os RegControls no OpenDSS desempenham um papel fundamental no controle de reguladores de tensão. Eles permitem o monitoramento e ajuste automático das tensões em pontos específicos do sistema elétrico, garantindo um fornecimento adequado de energia.

Para exemplificar a declaração de um RegControl no OpenDSS, considere o seguinte comando:

```
New RegControl.NomeDoRegControl Element=NomeDoRegulador
   Terminal=1 Transformer=TrafoControlado PTRatio=60 Band
   =2.0 VSet=120 TapNum=32 TapWinding=1 MaxTapChange=16
```

Comando 5 – Linha de declaração de um elemento RegControl

Em que:

- **New RegControl.NomeDoRegControl**: Cria um novo RegControl com o nome especificado.
- **Element**: Especifica o elemento ao qual o RegControl está associado (por exemplo, um regulador de tensão).
- **Terminal**: Define o terminal do elemento ao qual o RegControl está conectado.
- **Transformer**: Define o transformador controlado pelo RegControl.
- **PTRatio**: Especifica a relação de transformação primária do regulador.
- **Band**: Define a largura da banda de regulação.
- **VSet**: Especifica a tensão alvo para regulação.
- **TapNum, TapWinding**: Configuram o número de taps e o enrolamento do transformador a ser controlado.
- **MaxTapChange**: Define o número máximo de alterações de tap permitidas.

Este exemplo ilustra a declaração de um RegControl no OpenDSS, demonstrando como são especificados os parâmetros para controle de reguladores de tensão no software.

3.2.6 Reactor

Os Reactors no OpenDSS são elementos utilizados para modelar reatores em sistemas elétricos. Eles representam componentes que podem ser usados para controlar a corrente,

restringir curto-circuitos ou fornecer reatâncias para compensação de sistemas.

Para exemplificar a declaração de um Reactor no OpenDSS, considere o seguinte comando:

```
New Reactor.NomeDoReactor phases=3 Bus=BarraLocal kvar=500
   %R=0 X=5
```

Comando 6 – Linha de declaração de um elemento Reactor

Em que:

- **New Reactor.NomeDoReactor**: Cria um novo Reactor com o nome especificado.
- **phases=3**: Define que o Reactor possui três fases.
- **Bus**: Especifica a barra à qual o Reactor está conectado.
- **kvar**: Define a potência reativa do Reactor em kilovars.
- **%R, %X**: Representam a parte real e imaginária da impedância do Reactor.

Este exemplo ilustra a declaração de um Reactor no OpenDSS, demonstrando como são especificados os parâmetros para modelagem de reatores no software.

3.3 Considerações Finais

No capítulo, foram apresentados os elementos essenciais do OpenDSS, fundamentais para a modelagem precisa de sistemas elétricos. Ter uma compreensão aprofundada desses elementos é crucial para entender as contribuições realizadas neste trabalho.

4 BDGD-TOOLS

4.1 Apresentação do Capítulo

O capítulo possui como objetivo apresentar a biblioteca BDGD-Tools. A biblioteca é o foco deste trabalho, o qual é uma contribuição direta para o software.

Neste capítulo, são apresentadas as motivações da criação e desenvolvimento da biblioteca, seguido por uma exposição de suas principais características. Uma visão geral da arquitetura é oferecida, incluindo diagramas para facilitar a compreensão. Em seguida, o funcionamento da biblioteca é detalhado, abrangendo tanto a perspectiva do usuário quanto a do desenvolvedor. Por fim, é realizada uma discussão sobre o estado da arte da biblioteca, expondo o que falta para uma versão funcional.

Compreender os conceitos abordados nas seções subsequentes é fundamental para um entendimento pleno do conteúdo do Capítulo 5 sobre a metodologia do trabalho.

4.2 Contextualização

A principal motivação final da criação da biblioteca dá-se pela necessidade de simular redes elétricas de distribuição brasileira a partir das informações da BDGD de forma democrática e eficiente.

Atualmente, existem outras maneiras de fazer a conversão da BDGD para modelos elétricos. Apesar disso, deixam os pontos de caráter livre ou eficiência a desejar. Tais formas são:

- **Manualmente**: criação das linhas de modelagem de cada equipamento da rede por um operador de forma manual. Este processo, além de ser exaustivo, está susceptível à erros humanos e possui um tempo de execução total excessivo, tornando este inviável.
- **Softwares proprietários:** realizam a conversão da BDGD para modelagem dos sistemas elétricos de distribuição. Um exemplo seria o SigPerdas (SINAPSIS, 2023). A natureza das soluções desta categoria é fechada e direcionada exclusivamente para as concessionárias, fato que limitações à acessibilidade e à capacidade de adaptação para outros propósitos além dos estritamente comerciais.
- **Método oficial da ANEEL:** Tradicionalmente, a ANEEL faz uso do software GeoPerdas para extrair as informações da BDGD para realizar, em especial, os cálculos de perdas

utilizando o ProgGeoPerdas ((PAULISTA *et al.*, 2023). O método se mostra complexo e requer forte expertise para ser reproduzido.

Na Figura 3, é exibido o resumo sobre as possibilidades listadas.

Figura 3 – Resumo das formas alternativas para conversão da BDGD em modelos elétricos.

Fonte: Autor

De forma geral, a BDGD-Tools surge como uma resposta direta às limitações enfrentadas pelas concessionárias de energia ao tentar reproduzir o complexo método de perdas técnicas exigidos pela ANEEL (ANEEL, 2021c), conforme relatado por profissionais do setor. Assim, essa dependência do Geoperdas e sua complexidade gerou uma lacuna significativa para as concessionárias e outros interessados no setor. O depoimento do engenheiro Jamenson Guilherme expõe esse cenário, onde explicita a necessidade de mão de obra com competências técnicas avançadas para manipular o método da ANEEL, limitando-se àqueles com expertise especializada.

> "Para a implementação do cálculo de perdas, conforme estipulado no módulo 7 do Prodist, emprega-se o GeoPerdas, uma aplicação dedicada à conversão dos dados provenientes da BDGD para o formato OpenDSS. Entretanto, essa conversão demanda a utilização de ferramentas especializadas e um conhecimento técnico avançado por parte do operador encarregado dessa tarefa. Tal requisito impõe desafios à replicação do cálculo de perdas, restringindo-se aos técnicos que tenham adquirido expertise nesse procedimento. Um método alternativo, que emprega o Python como ferramenta, simplifica a reprodução do cálculo, promove a disseminação do conhecimento entre os profissionais técnicos e amplia o emprego desses modelos para finalidades além da obtenção de perdas."**-*Jamenson Guilherme, Eng. Gerente de Regulamentação da CRELUZ desde 2009.***

Diante dessa lacuna e limitações, foi iniciado um esforço para implementação de uma ferramenta de software chamada de BDGD-Tools (RADATZ; VIANA, 2023a) a ser

disponibilizada de forma livre em repositório web. O objetivo principal não se restringe apenas a possibilitar que as concessionárias repliquem os cálculos da ANEEL, mas também busca abrir novas fronteiras no campo acadêmico. A BDGD-Tools representa um avanço significativo ao permitir estudos em redes reais de distribuição de energia sem demandar grandes investimentos, além de ser uma ferramenta livre e acessível para todos os interessados (BDGD-Tools, 2023).

Nas próximas seções, serão abordados mais detalhes sobre a biblioteca.

4.3 Características Principais

A BDGD-Tools é uma ferramenta dedicada à conversão do formato da BDGD para o formato do OpenDSS. Seu principal propósito é permitir a extração de dados da BDGD e a conversão desses sistemas elétricos de distribuição para um formato compatível com o OpenDSS, possibilitando a realização de cálculos e análises específicas nessa plataforma.

Os objetivos da biblioteca foram delineados para atender à demanda das concessionárias de energia, possibilitando a replicação dos cálculos da ANEEL e, ao mesmo tempo, abrindo novas possibilidades de estudos com redes reais do sistema elétrico brasileiro. Os requisitos da ferramenta foram projetados para garantir eficiência na extração de dados, precisão na conversão, flexibilidade de uso e acessibilidade tanto para profissionais quanto para acadêmicos interessados no campo, com uma documentação clara e abrangente (BDGD-Tools, 2023).

Esse software, desenvolvido em *Python*, oferece uma instalação via pip, facilitando a integração em projetos. Distribuída sob a licença MIT e disponível no GitHub (RADATZ; VIANA, 2023a), essa biblioteca permite aos usuários total liberdade para modificar, distribuir e utilizar a ferramenta conforme necessário. A documentação pode ser encontrada no servidor da *Read the Docs*, uma plataforma de hospedagem de documentação de software gratuita e de código aberto (BDGD-Tools, 2023).

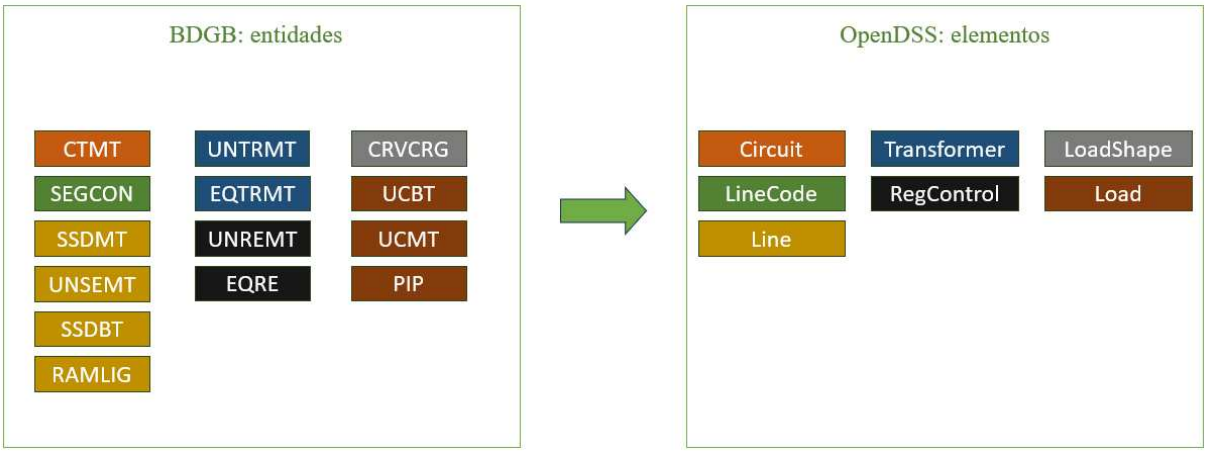
4.4 Mapeamento

A ideia do software está baseada no mapeamento entre as entidades da BDGD e os elementos do OpenDSS que representam o sistema elétrico real, apresentados nos Capítulos 2 e 3, respectivamente.

Antes de tudo, a visão é que para cada elemento do OpenDSS, são categorizadas entidades da BDGD, como é mostrado na Figura 4. A figura mostra a fonte de dados para

modelagem de cada elemento do software de simulação do sistema elétrico de potência.

Figura 4 – Mapeamento entre as entidades da BDGD e elementos do OpenDSS. As entidades são classificadas em diferentes elementos através das cores.



Fonte: Autor

Por exemplo, para modelar o Line, são necessários os dados da tabela da entidade SEGCON. Já para modelar o elemento transformer, é necessário extrair os dados das tabelas das entidades UNTRMT e EQTRMT.

O mapeamento entre as colunas das tabelas e os parâmetros dos elementos pode ser resumido nos seguintes métodos:

- **Estático:** Refere-se a valores fixos, como, por exemplo, a frequência da linha de transmissão que pode ser um valor constante como 60Hz. Neste caso, não é necessário a busca na BDGD e o valor é definido internamente no software.
- **Mapeamento direto:** Utilizado quando os atributos de um elemento podem ser preenchidos diretamente com os valores encontrados na base de dados. Isso significa que a informação de uma coluna pode ser simplesmente direcionada para o parâmetro do elemento, sem realizar procedimentos de conversão.
- **Mapeamento indireto:** Empregado quando existe uma relação mais complexa entre os dados na base e as propriedades dos elementos. Normalmente se dá quando algum padrão definido no módulo 10 do PRODIST é empregado. Neste caso, o mesmo documento possui tabelas de conversão.
- **Calculado:** Refere-se a propriedades que exigem uma expressão ou cálculo para serem obtidas.

Com esses métodos de mapeamento, a biblioteca popula os parâmetros dos elementos do OpenDSS de forma precisa e organizada.

É importante ressaltar que os padrões de modelagem seguem as diretrizes do Módulo 7 do PRODIST (ANEEL, 2021c), o qual estabelece as premissas para o cálculo das perdas.

4.5 Arquitetura

A biblioteca é formada pelos seguintes componentes:

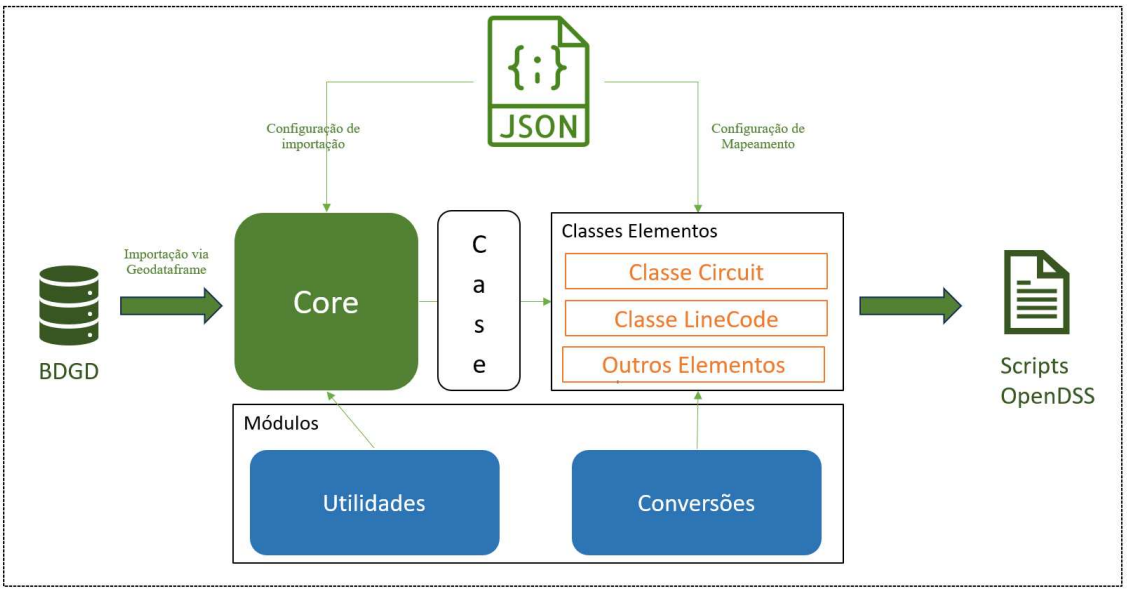
1. BDGD.
2. Classes dos elementos.
3. Classe do cenário (*Case*).
4. Módulo de Utilidades.
5. Módulo de Conversão.
6. Arquivo JSON.

A estratégia da arquitetura é concentrar a configuração principal do software em torno do arquivo JSON, sigla para "JavaScript Object Notation". JSON é um formato leve de troca de dados. Sua escolha deu-se por ser legível por humanos e também fácil de ser interpretado e gerado por máquinas.

Assim, os demais componentes da arquitetura utilizarão este recurso para desempenhar a importação precisa dos dados da BDGD e conversão de acordo com o mapeamento com as informações lá detalhadas.

A Figura 5 ilustra uma visão geral da arquitetura da biblioteca.

Figura 5 – Arquitetura da biblioteca BDGD-Tools.



Fonte: Autor

Os componentes da arquitetura exibida na 5 são descritos a seguir:

- **Base de Dados:** a base de dados utilizada é a BDGD, 2.
- **Core:** orquestrador dos diferentes recursos da biblioteca.
- **Classe Case:** representa um cenário, formando um conjunto de elementos do sistema elétrico de potência.
- **Classes Elementos:** na BDGD-Tools, cada elemento do OpenDSS é implementado através de uma classe. Elas acessam e convertem os dados para o padrão de declaração do elemento. Estas classes podem ser LoadShape, Load, Tansformer, Linecode, Line, Circuito e Regulador de Tensão.
- **Módulo Utilidades:** Funções auxiliares, tais como manipulação do arquivo JSON ou criação dos arquivos dss.
- **Módulo Conversão:** Funções conversoras utilizadas nas classes elementos.
- **Arquivo JSON BDGD2DSS:** é um arquivo de configuração centralizado, contendo metadados cruciais para o funcionamento do software. Ele orienta como importar e converter os dados da BDGD, incluindo a indicação do método de mapeamento para cada variável. O seu uso flexibiliza manutenções no software, permitindo ajustes na maneira como os dados são interpretados e processados sem alterações no código-fonte.

O *core* se comunica diretamente com a BDGD por meio de funções do módulo *utilidades* que faz uso da *GeoPandas* (JORDAHL *et al.*, 2020), biblioteca opensource do python desenvolvida para facilitar o trabalho com dados geoespaciais. O core utiliza as diretrizes do arquivo *BDGD2DSS.JSON*, o que permite uma importação precisa, selecionando apenas as tabelas e colunas necessárias, garantindo eficiência no uso dos recursos. Além disso, o script Core possui o gerenciamento da classe case.

As classes *elementos* acessam os dados já importados da BDGD e os traduzem de acordo com a configuração do arquivo *BDGD2DSS.JSON* para o formato DSS. Elas se utilizam dos módulos disponíveis, *conversão e utilidades*, seja para converter informações ou criar arquivos *.dss*.

De modo geral, a arquitetura possui uma estrutura de inteligência concentrada com flexibilidade nos acesso e interpretação dos dados.

4.6 Funcionamento

Esta seção busca explicar o funcionamento a partir de duas perspectivas: interna (desenvolvimento) e externa (usuário).

A Figura 6 mostra o funcionamento interno. É apresentado de forma geral o fluxograma do software.

Figura 6 – Funcionamento sob a ótica interna.

Fonte: Autor

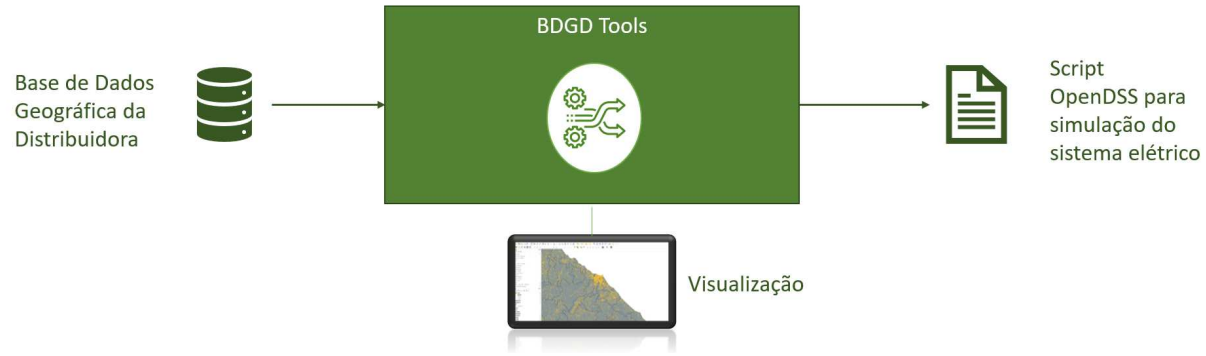
A importação é feita através da criação de tabelas pré-configuradas, no arquivo *JSON*, que são populadas pelos dados da BDGD.

Em seguida, cada classe elemento recebe um dataframe referente a tabela da entidade alvo. Nisso, é feita a implementação das conversões de acordo com o mapeamento definido anteriormente. O processo é realizado através de uma varredura do arquivo json, onde as variáveis do objeto são preenchidos pelo dataframe recebido como parâmetro.

Em seguida, a classe realiza a junção desses dados para formar a linha de código da declaração do equipamento no OpenDSS. Após este processo para o conjunto total de entidades, é criado o arquivo .dss com as linhas geradas.

A Figura 7 exibe o funcionamento externo, da perspectiva de usuário. Basicamente, é fornecida uma BDGD como entrada e a biblioteca prevê gerar scripts para modelar o sistema fornecido para o OpenDSS. O usuário pode fornecer alguns parâmetros, tais como um alimentador em específico da rede. A biblioteca tem como recurso a ser implementado a visualização de dados geográficos das entidades do sistema.

Figura 7 – Funcionamento sob a ótica do usuário.



Fonte: Autor

4.7 Considerações Finais

Com este capítulo, foi vista a essência da BDGD-Tools, compreendendo sua origem diante das lacunas do setor elétrico e seu importante papel na democratização da análise de redes elétricas brasileiras no OpenDSS. Ainda foi possível abordar seu funcionamento e arquitetura.

Importante ressaltar que, até meados de agosto de 2023, a biblioteca ainda se encontrava em um estado não funcional, apesar de sua base estrutural estar já bem definida até então.

De forma geral, este capítulo mostra os diferentes componentes da biblioteca e como eles interagem entre si - conhecimento que ajudará no entendimento da metodologia executada no Capítulo 5.

5 METODOLOGIA

Este capítulo tem como objetivo apresentar a metodologia adotada para o desenvolvimento das atividades. Inicia-se com uma concisa contextualização, seguida pela análise das tecnologias empregadas e uma visão abrangente do estado da arte em questão. Posteriormente, é explicada a implementação dos recursos adicionados, percorrendo cada elemento detalhadamente.

Recomenda-se a leitura prévia do Capítulo 4 para um melhor entendimento deste capítulo.

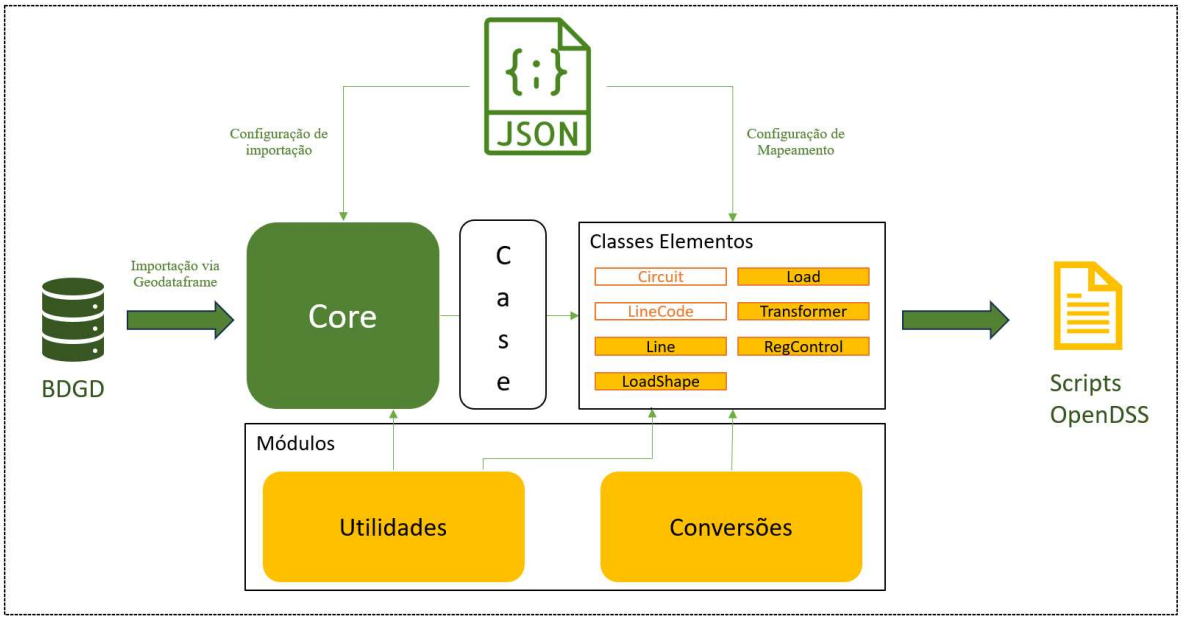
5.1 Contextualização

Seguido do reconhecimento inicial do desenvolvimento da biblioteca Python BDGD-Tools do primeiro semestre de 2023, foi feito um levantamento de recursos a serem implementados para seu estado funcional. Este processo se deu a partir da leitura da documentação, entendimento do código fonte e reuniões com os engenheiros criadores da biblioteca Paulo Radatz e Enio Viana.

A Figura 8 exibe esses pontos identificados com destaque em laranja. São estes pontos a definição das classes dos elementos e incrementos aos módulos de conversão e utilidades, assim como a geração dos arquivos .dss (ver 4). Ainda, outras implementações secundárias foram efetuadas nos outros componentes, porém com menor relevância, sendo muitas vezes ajustes em decorrência de atualizações.

A implementação desses recursos identificados é o tema principal deste trabalho e será abordada nas seções seguintes.

Figura 8 – Arquitetura da BDGD-Tools com os recursos identificados a serem implementados em destaque.



Fonte: Autor

5.2 Ferramentas e Tecnologia

Ao longo do projeto, diferentes ferramentas e tecnologias foram utilizadas para o desenvolvimento do software. A lista pode ser dividida em dois pontos, Sub-Seção 5.2.1 e Sub-Seção 5.2.2.

5.2.1 Ferramentas de Desenvolvimento

- O **Visual Studio Code** foi escolhido como ambiente de desenvolvimento integrado (IDE) devido à sua popularidade, flexibilidade e suporte a uma variedade de linguagens de programação (Python, json e dss, no contexto do projeto). Sua extensibilidade, integração com ferramentas de controle de versão (git) e a comunidade ativa de desenvolvedores foram fatores determinantes na escolha.
- O **Virtualenv** foi utilizado para criar ambientes virtuais isolados para o projeto. Isso permite a gestão independente das dependências do projeto, garantindo consistência nas versões de pacotes e evitando conflitos entre diferentes projetos que possam exigir versões específicas de bibliotecas ou pacotes Python.
- A plataforma **GitHub** foi escolhida para o controle de versionamento do código-fonte. Essa escolha é respaldada pela ampla adoção na comunidade de desenvolvimento, além

de fornecer recursos robustos para colaboração, controle de alterações e rastreamento de problemas. O uso de branches e pull requests facilita a colaboração entre os membros da equipe.

– Utilizando **Setuptools** em modo desenvolvedor, simplificamos configuração, empacotamento e distribuição. Essa escolha não apenas agiliza a instalação do projeto, mas também facilita ambientes propícios ao debugging

5.2.2 Bibliotecas para Implementação de Recursos

– A biblioteca **Geopandas** foi escolhido para lidar com dados geoespaciais da BDGD devido à sua integração eficiente com o ecossistema Pandas e suas capacidades de manipulação de dados espaciais.
– A escolha da biblioteca **JSON** se dá pela necessidade de manipulação do arquivo em .JSON que comporta a inteligência do software.
– A biblioteca Python py-dss-interface (GitHub Python DSS Tools, 2023) foi selecionada para interação e manipulação de elementos do OpenDSS, permitindo a criação de um ambiente de depuração de erros das saídas da ferramenta.

5.3 Implementação

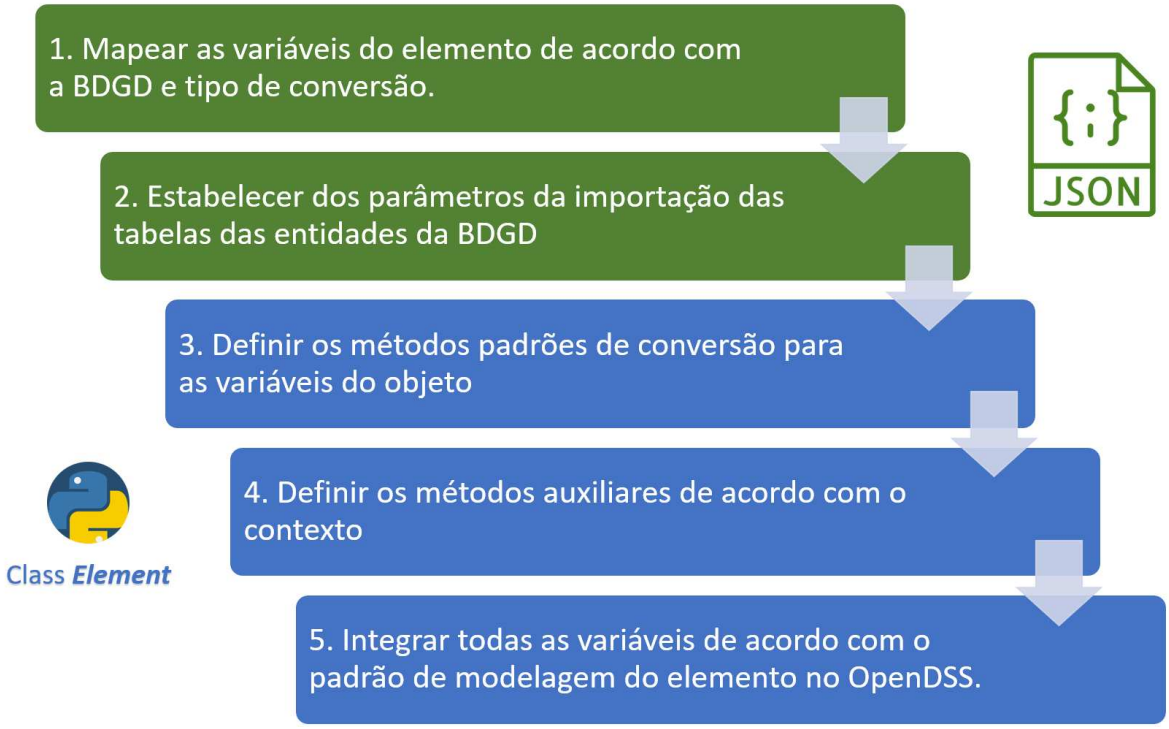
Esta seção apresentará a metodologia usada para implementação de recursos para a biblioteca BDGD-Tools. Esses recursos foram definidos na Seção 5.1.

Seguindo os padrões apresentados no Capítulo 4, a conversão de informações da base de dados para um sistema elétrico modelado no OpenDSS é realizada em função de cada elemento da rede. Nesse contexto, focaremos as explicações em torno dos elementos.

De forma genérica, a Figura 9 apresenta as etapas de implementaçao da conversão para um elemento. O processo é dividido em duas partes: modificações do arquivo JSON e definição da classe do elemento. Cada etapa é definida abaixo:

1. Trata-se de definir, no arquivo JSON, quais são as variáveis necessárias para modelar o elemento no OpenDSS. Além disso, especificar como encontrar e tratar essas informações de acordo com a BDGD. Detalhes sobre as estratégias de mapeamento podem ser encontradas no Capítulo 4.
2. Para uma otimização da biblioteca, é importado da BDGD apenas as informações ne-

Figura 9 – Etapas do processo de criação da conversão de entidades da BDGD para elemento do OpenDSS. A parte em verde é realizada no arquivo JSON. A parte azul na classe Elemento em Python.



Fonte: Autor

cessárias da(s) entidade(s) referente(s) ao elemento. Ou seja, colunas específicas da(s) tabela(s) da(s) entidade(s). Portanto, é preciso inserir no arquivo JSON as entidades e seus parâmetros como configuração para importação.

3. Em seguida, na classe Elemento em Python, define-se os métodos de conversão. Os métodos de conversão são responsáveis por preencher as variáveis com os dados da BDGD e, se necessário, tratá-los.
4. Métodos auxiliares, tais como a criação dos arquivos .dss para aquele elemento, são criados.
5. O último passo está referente a criar a linha de código no formato do OpenDSS para modelar o elemento alvo no simulador. Isso se dá com a integração de todas as variáveis relevantes no padrão requerido.

Além disso, cada elemento dispõe de suas particularidades e desafios. A seguir, será abordada a aplicação da metodologia da Figura 9.

Importante ressaltar que toda a implementaçao dos elementos deu-se levando em conta suas modelagens seguindo as diretrizes do múdulo 7 do PRODIST da ANEEL (ANEEL, 2021c), que determina os pré-requisitos para o cálculo das perdas técnicas.

5.3.1 Line

O elemento representa diferentes componentes do sistema elétrico, identificados como entidades da BDGD. São estes Segmentos de Baixa e Média Tensão, e Ramais de Ligação.

Para o elemento *Line*, o objetivo da biblioteca é gerar a linha de comando responsável pela sua modelagem no software OpenDSS.

O *Line* tem sua modelagem no software OpenDSS descrita no Capítulo 3, onde é explicado cada parâmetro. Sendo assim, o foco deste capítulo é o contexto da implementação da conversão.

5.3.1.1 Considerações Iniciais e Implementação de Pré-Requisitos

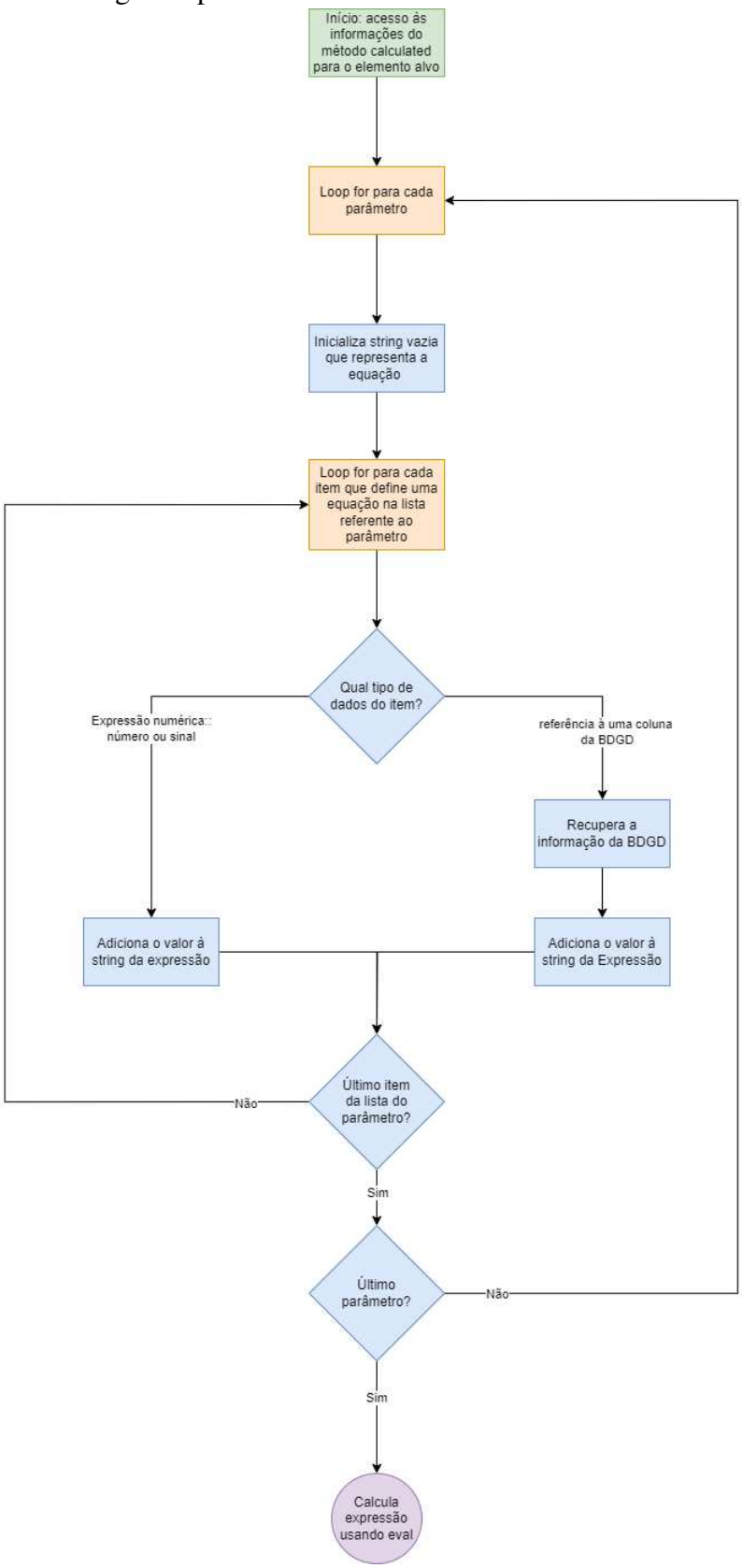
Em relação ao mapeamento, foram utilizados os métodos estático, direto, indireto e calculado, sendo este último o único até então não implementado.

Neste contexto, como o primeiro elemento a requisitar a forma de conversão "calculada", o método *"calculated"* foi criado para atender a esse requisito. Essa metodologia de conversão se dá quando um ou mais valores da BDGD precisam passar por uma equação matemática antes de serem atribuídos ao elemento em questão.

A expressão matemática é definida como uma string no arquivo json relacionada a uma variável do elemento. Dentro da expressão matemática estão termos da BDGD, cujo valores serão extraidos e adicionados à expresão para calcular-se o resultado. A Figura 10 exibe o fluxograma do método implementado.

No geral, o método tem esse funcionamnto com o intuito de estar alinhado com a estratégia global de centralizar a inteligência do softwre no arquivo JSON. Sendo assim, as modificações quanto ao método são realziadas somente no arquivo *"bdgd2dss.json"*.

Figura 10 – Fluxograma para o método *calculated*.



Fonte: Autor

O método é flexível, sendo apenas necessário isolar a referência da BDGD como um único elemento da lista. Por exemplo, na Sub-Seção 5.3.5.3, tem-se a variável length, definida pela expressão em uma lista ["COMP", "/","1000"]. COMP deve estar separado para permitir a captura daquele valor da BDGD.Por sua vez "/" e '1000", elementos matemáticos, podem estar juntos ou separados em uma mesma string.

Assim, os pré-requisitos da classe Line foram atendidos e o desenvolvimento passa para o arquivo JSON.

5.3.1.2 Arquivo JSON

Como descrito anteriormente, o arquivo JSON funciona como a inteligência do software. Neste arquivo, são definidas as configurações de importação e mapeamento.

Em relação ao mapeamento das variáveis[1] , ficou definido conforme mostrado nas Tabelas 1, 2, 3 e 4.

Tabela 1 – Método Estático

Variável	Valor
units	km

Tabela 2 – Método Direto

Variável	Coluna Correspondente da BDGD
bus1	PAC_1
bus2	PAC_2
line	COD_ID
linecode	TIP_CND
feeder	CTMT

Tabela 3 – Método Indireto

Variável	Correspondente da BDGD	Funções Conversoras
phases	FAS_CON	convert_tfascon_phases
bus_nodes	FAS_CON	convert_tfascon_bus
suffix_linecode	FAS_CON	convert_tfascon_quant_fios

[1] Para entendimento do mapeamento das variáveis, rever o Capítulo 4

Tabela 4 – Método Calculado

Variável	Fórmula
length	COMP / 1000 (km)

Para importação dos dados da BDGD de forma eficiente, são definidas as colunas especificas das tabelas das entidades representadas pelo elemento Line. A Figura 11 mostra um exemplo para a tabela da entidade SSDMT.

Figura 11 – Arquivo JSON.

Fonte: Autor

Assim, dá-se a configuração do arquivo JSON para implementação do elemento Line.

5.3.1.3 Classe Line

Após ter definido no arquivo JSON como encontrar os valores das variáveis e convertê-las, a classe implementa de fato a conversão dos ados em elementos do OpenDSS (ver Seção 4.6).

No que se refere a classe Line, a Figura 12 ilustra sua implementação através de um diagrama UML, onde são definidos variáveis, métodos de conversão e métodos auxiliares.

Com o dataframe contendo as informações da BDGD para a entidade alvo como parâmetro do método *create_new_line_from_json()*, é possível acessar e converter os dados.

Figura 12 – UML.

Fonte: Autor

Em um segundo momento, é, então, construida a linha de declaração do elemento no OpenDSS. O Comando 7 mostra um exemplo para uma entidade de segmento de baixa tensão da BDGD. O texto declara a entidade no software de simulação com as informações da distribuidora.

Iss foi realizado através do método *___rep__()*, que foi definido para que cada objeto gerado seja representado diretamente com sua linha de comando. A Figura 13 ilustra o método *___rep__()*, expondo como foi criada a linha. A ideia, como explicada anteriormente, é a junção das variáveis definidas para o modelo do OpenDSS[2].

As peculiaridades de sua construção giram em torno das variáveis *bus1, bus2 e linecode*. Para as variáveis que definem a barra, é preciso o nome da barra e definição da ligação dos nós, como indica a figura. Já para *linecode*, que relaciona a linha a um tipo de condutor, é

[2] Para entendimento do significado dos parâmetros, rever o Capítulo 3

formado pelo seu nome e um sufixo que indica a quantidade de fios. A definição da ligação dos nós e a quantidade de fios foram extraidas e convertidas do tipo das fases de conexão da BDGD, segundo a Tabela TFASCON do Módulo 10 do PRODIST.

```
1 New "Line.SBT_5466" phases=3 bus1="BT17342.1.2.3.4" bus2="
    BT73301.1.2.3.4" linecode="85_4" length=0.02501 units=km
```

Comando 7 – Linha de declaração de um elemento Line

Figura 13 – Método __repr__() responsável por criar a linha de comando do elemento Line

```
def __repr__(self):
    return  f'New \"Line.{self.entity}_{self.line}" phases={self.phases} ' \
            f'bus1="{self.bus1}.{self.bus_nodes}" bus2="{self.bus2}.{self.bus_nodes}" ' \
            f'linecode="{self.linecode}_{self.suffix_linecode}" length={self.length:.5f} ' \
            f'units={self.units}'
```

Fonte: Autor

Importante ressalvar que a mesma classe é responsável por instancializar outras entidades como Segmento de Média Tensão e Ramal de Ligação. Uma das variáveis que possui alteração é o pré-fixo do nome do elemento, armazenado em *entity*.

Por fim, é chamado o método *create_output_file()* para criar o arquivo .dss definido no módulo *Utilidades*, ver Subseção 5.3.6.1. O nome do arquivo é passado para a classe case, onde será utilizado para criação dos arquivos *master* posteriormente.

Assim, dá-se a implementação do elemento Line.

5.3.2 Transformer com Reator

O elemento representa um transformador do sistema elétrico, possuindo suas informações em duas entidades da BDGD. São estas Equipamento Transformador de Média Tensão e Unidade Transforamdora de Media Tensão.

Para o elemento *Transformer*, o objetivo da biblioteca é gerar a linha de comando responsável pela sua modelagem no software OpenDSS.

O *Transformer* tem sua modelagem no software OpenDSS descrita no Capítulo 3, onde é explicado cada parâmetro. Sendo assim, o foco deste capítulo é o contexto da implementação da conversão.

5.3.2.1 Considerações Iniciais e Implementação de pré-requisitos

A implementação do transformador apresenta considerações importantes. Em primeiro lugar, há a exigência do Módulo 7 do PRODIST, que estipula a existência de um aterramento de 15 ohms. Em segundo lugar, há a necessidade de integrar dados de diferentes tabelas. Outro ponto é o aumento da complexidade ao lidar com a criação da linha de comando através dos métodos *__rep__() e full_string()*.

Para cumprir as diretrizes do módulo 7 do PRODIST, foi incorporado um reator com resistência aterrada de 15 ohms. Este elemento foi modelado junto ao transformador, sendo relacionado a cada nova instância deste último.

Com o objetivo de contemplar os parâmetros de modelagem do transformador e do reator, duas tabelas da BDGD foram utilizadas, *UNTRMT e EQTRMT*. Assim, um método foi desenvolvido no módulo *utilidades* para unir de forma apropriada os dois dataframes, mantendo o mesmo padrão do processo de conversão.

Adicionalmente, o método *adapting_string_variables()* foi adicionado para adaptar as variáveis para o formato OpenDSS. Com isso, tem-se um código mais legível e flexível à possíveis manutenções.

É relevante ressaltar que a implementação considera sempre tensão no primário e secundário, sendo o terciário opcional - fato implementado no método destacado acima.

Em relação ao mapeamento, foram empregados os métodos direto, indireto e calculado.

Dessa forma, os requisitos preliminares para o elemento Transformer foram atendidos

e o desenvolvimento passa para o arquivo JSON.

5.3.2.2 Arquivo JSON

Como descrito anteriormente, o arquivo JSON funciona como o cérebro do software. Neste arquivo, são definidas as configurações de importação e mapeamento.

Em relação ao mapeamento das variáveis[3] no JSON , ficou definido conforme mostrado nas Tabelas 5, 6 e 7.

Tabela 5 – Método Direto UNTRMT

Variável	**Coluna Correspondente da BDGD**
bus1	PAC_1
bus2	PAC_2
bus3	PAC_3
transformer	COD_ID
kvas	POT_NOM
tap	TAP
feeder	CTMT

Tabela 6 – Método Indireto UNTRMT

Variável	**Correspondente da BDGD**	**Funções Conversoras**
phases	FAS_CON_P	convert_tfascon_phases
bus1_nodes	FAS_CON_P	convert_tfascon_bus
bus2_nodes	FAS_CON_S	convert_tfascon_bus
bus3_nodes	FAS_CON_T	convert_tfascon_bus
kv1	TEN_PRI	convert_tten
kv2	TEN_SEC	convert_tten
kv3	TEN_TER	convert_tten
windings	TIP_TRAFO	convert_ttranf_windings
conn_p	FAS_CON_P	convert_tfascon_conn
conn_s	FAS_CON_S	convert_tfascon_conn
conn_t	FAS_CON_T	convert_tfascon_conn

Tabela 7 – Método Calculado UNTRMT

Variável	**Fórmula**
loadloss	$\left(\frac{PER\_TOT - PER\_FER}{POT\_NOM/1000}\right) \times 100$
noloadloss	$\left(\frac{PER\_FER}{POT\_NOM/1000}\right) \times 100$

As *Funções Conversoras* exibidas na Tabela 6 são definidas no módulo de conversão de acordo com o módulo 7 do PRODIST (ANEEL, 2021c).

[3] Para entendimento do mapeamento das variáveis, rever o Capítulo 4

Em seguida, para importação dos dados da BDGD de forma eficiente, são definidas as colunas especificas das tabelas das entidades representadas pelo elemento Transformer. A Figura 14 mostra um exemplo para a tabela da entidade *UNTRMT*.

Figura 14 – Configuração das colunas para importação da BDGD no arquivo JSON.

Fonte: Autor

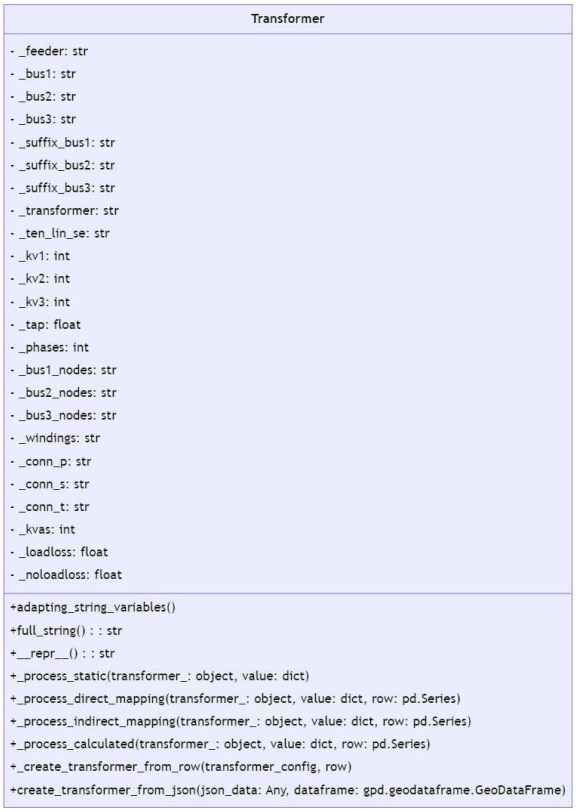
Assim, dá-se a configuração do arquivo JSON para implementação do elemento Transformador com reator.

5.3.2.3 Classe Transformer

Após ter definido no arquivo JSON como encontrar os valores das variáveis e convertê-las, a classe busca colocar em prática a conversão (ver Seção 4.6).

No que se refere a classe *Transformer*, a Figura 15 ilustra sua implementação através de um diagrama UML, onde são definidos variáveis, métodos de conversão e métodos auxiliares.

Figura 15 – UML Transformer.



Fonte: Autor

Com o dataframe contendo as informações da BDGD para a entidade alvo como parâmetro do método *create_new_transformer_from_json()*, é possível acessar e converter os dados.

Em um segundo momento, é, então, construida a linha de declaração do elemento no OpenDSS. O Comando 8 mostra um exemplo para uma entidade de segmento de baixa tensão da BDGD. O texto declara a entidade no software de simulação com as informações da distribuidora.

Isso foi realizado através do método *___rep__()*, que foi definido para que cada objeto gerado seja representado diretamente com sua linha de comando, onde possui o reator e o transformador conectados. A Figura 16 ilustra o método *___rep__()*, expondo como foi criada a

linha. A ideia, como explicada anteriormente, é a junção das variáveis definidas para o modelo do OpenDSS[4].

As peculiaridades de sua construção giram em torno das variáveis *buses e kvs*. Para essas variáveis, foi adaptado para tensão no terciário, caso exista. O *Reator* possui o mesmo nome que o transformer, está ligado à barra 2, nó 4 e diretamente para o aterramento.

```
1 New "Transformer.TRF_1_165" phases=3 windings=2 buses=["
     86851.1.2.3" "ET1_165.1.2.3.4"] conns=[Delta Wye ] kvs
     =[13.8 0.38] taps=[1.045 1.045] kvas=[45.0 45.0] %
     loadloss=1.853 %noloadloss=0.478
2 New "Reactor.TRF_1_165 phases=1 bus1="ET1_165.4" R=15 X=0
     basefeq=60
```

Comando 8 – Linha de declaração de um elemento Transformer com reator

Figura 16 – Método __repr__() responsável por criar a linha de comando do elemento Transformer

```
def __repr__(self):

    self.kvs, self.buses, self.kvas, self.taps = Transformer.adapting_string_variables(self)

    return  (
f'New \"Transformer.TRF_{self.transformer}" phases={self.phases} '
f'windings={self.windings} '
f'buses=[{self.buses}] '
f'conns=[{self.conn_p} {self.conn_s} {self.conn_t}] '
f'kvs=[{self.kvs}] '
f'taps=[{self.taps}] '
f'kvas=[{self.kvas}] '
f'%loadloss={self.loadloss:.3f} %noloadloss={self.noloadloss:.3f}\n'
f'New "Reactor.TRF_{self.transformer} phases=1 bus1="{self.bus2}.4" R=15 X=0 basefeq=60\n'
)
```

Fonte: Autor

Por fim, é chamado o método *create_output_file()* para criar o arquivo .dss definido no módulo *Utilidades*, ver Subseção 5.3.6.1. O nome do arquivo é passado para a classe case, onde será utilizado para criação dos arquivos *master* posteriormente.

Assim, dá-se a implementação do elemento Transformer.

[4] Para entendimento do significado dos parâmetros, rever o Capítulo 3

5.3.3 RegControl

O elemento representa um regulador de tensão, ou auto-transformador do sistema elétrico, possuindo suas informações em duas entidades da BDGD. São estas Equipamento Regulador e Unidade Reguladora de Média Tensão.

Para o elemento *RegControl*, o objetivo da biblioteca é gerar a linha de comando responsável pela sua modelagem no software OpenDSS.

O *RegControl* tem sua modelagem no software OpenDSS descrita no Capítulo 3, onde é explicado cada parâmetro. Sendo assim, o foco deste capítulo é o contexto da implementação da conversão.

5.3.3.1 Considerações Iniciais/Implementação de pré-requisitos/Desafios

A implementação do RegControl, que tem como foco simulador o regulador de tensão, é o conjunto de um transformador relacionado a um objeto RegControl do OpenDSS que é responsável pelo controle do TAP do transformador.

Nesse contexto, a modelagem realizada na Seção 5.3.2 foi usada como inpiração. Assim, foi repetida a ideia do transformador, retirando o reator e adicionando a linha de declaração do *RegControl*.

Hoje, no sistema brasileiro, tem-se um regulador de tensão por fase. Assim, segundo a BDGD, cada unidade reguladora de tensão possui três equipamentos reguladores monofásicos. Na prática, para fazer a modelagem corretamente, é preciso juntar e relacionar os dataframes dessas duas entidades para fornecer as informações completas para a modelagem. Para isto, o método *inner_entities_tables()* foi criado no módulo de utilidades.

O método *inner_entities_tables()* une os dois dataframes através das colunas de identificação de cada um, sendo uma entidade EQRE relacionada a uma entidade UNREMT.

Em relação ao mapeamento, foram empregados os métodos estático, direto, indireto e calculado.

Dessa forma, os requisitos preliminares para o elemento RegCotnrol foram atendidos e o desenvolvimento passa para o arquivo JSON.

5.3.3.2 Arquivo JSON

Como descrito anteriormente, o arquivo JSON funciona como o cérebro do software. Neste arquivo, são definidas as configurações de importação e mapeamento.

Em relação ao mapeamento das variáveis[5] no JSON , ficou definido conforme mostrado nas Tabelas 8, 9, 10 e 11.

Tabela 8 – Método Estático - RegControl

Variável	**Valor**
windings	2
band	2

Tabela 9 – Método Direto - RegControl

Variável	**Coluna Correspondente da BDGD**
bus1	PAC_1
bus2	PAC_2
transformer	COD_ID
vreg	TEN_REG
prefix_transformer	LIG_FAS_P
xhl	XHL
feeder	CTMT

Tabela 10 – Método Indireto - RegControl

Variável	**Correspondente da BDGD**	**Funções Conversoras**
kvas	POT_NOM	convert_tpotaprt
phases	LIG_FAS_P	convert_tfascon_phases
bus1_nodes	LIG_FAS_P	convert_tfascon_bus
bus2_nodes	LIG_FAS_S	convert_tfascon_bus
conn_p	LIG_FAS_P	convert_tfascon_conn
conn_s	LIG_FAS_S	convert_tfascon_conn

Tabela 11 – Método Calculado - RegControl

Variável	**Equação**
ptratio	$TEN\_REG \times 100$
loadloss	$\left(\frac{PER\_TOT - PER\_FER}{POT\_NOM/1000}\right) \times 100$
noloadloss	$\left(\frac{PER\_FER}{POT\_NOM/1000}\right) \times 100$

[5] Para entendimento do mapeamento das variáveis, rever o Capítulo 4

As *Funções Conversoras* exibidas na Tabela 10 são definidas no módulo de conversão de acordo com o módulo 7 do PRODIST (ANEEL, 2021c).

As colunas correspondentes se referem às tabelas das entidades do elemento, UNREMT e EQRE.

Em seguida, para importação dos dados da BDGD de forma eficiente, são definidas as colunas especificas das tabelas das entidades representadas pelo elemento RegControl. A Figura 17 mostra um exemplo para a tabela da entidade *EQRE*.

Figura 17 – Configuração das colunas para importação da tabela EQRE da BDGD através do arquivo JSON.

Fonte: Autor

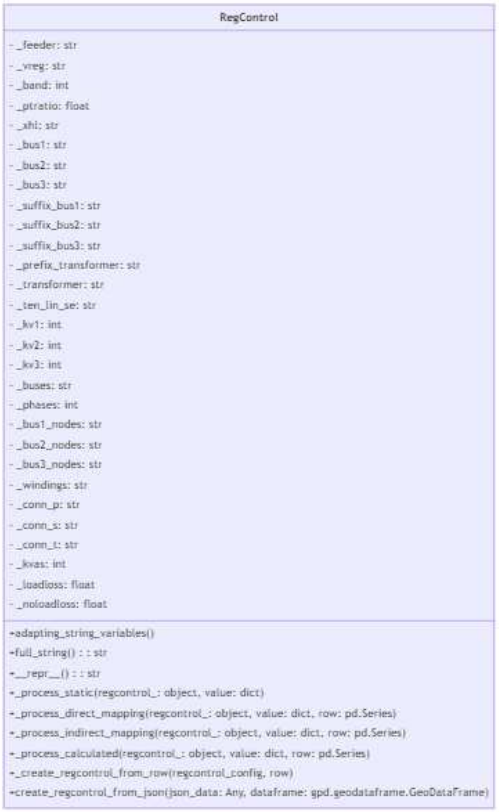
Assim, dá-se a configuração do arquivo JSON para implementação do elemento RegControl.

5.3.3.3 Classe RegControl

Após ter definido no arquivo JSON como encontrar os valores das variáveis e convertê-las, a classe busca colocar em prática a conversão (ver Seção 4.6).

No que se refere a classe *RegControl*, a Figura 18 ilustra sua implementação através de um diagrama UML, onde são definidos variáveis, métodos de conversão e métodos auxiliares.

Figura 18 – UML RegControl.



Fonte: Autor

Com o dataframe contendo as informações da BDGD para a entidade alvo como parâmetro do método *create_new_regcontrol_from_json()*, é possível acessar e converter os dados.

Em um segundo momento, é, então, construida a linha de declaração do elemento no OpenDSS. O Comando 9 mostra um exemplo para um regulador de tensão da BDGD.

Isso foi realizado através do método *___rep__()*, que foi definido para que cada objeto gerado seja representado diretamente com sua linha de comando, onde possui o transformador e o controlador conectados. A Figura 19 ilustra o método *___rep__()*, expondo como foi criada a linha. A ideia, como explicada anteriormente, é a junção das variáveis definidas para o modelo

do OpenDSS[6].

```
New "Transformer.AB1_3000" phases=2 windings=2 buses=["
   RT4711.1.2" "82261.1.2"] conns=[Delta Delta ] kvs=["
   tensao"] kvas=[667 667] xhl=4.0 %loadloss=8.571 %
   noloadloss=3.810
New "Regcontrol.AB1_3000" transformer="AB1_3000" windings=2
    vreg=1.05 band=2 ptratio=105.0
```

Comando 9 – Linha de declaração de um elemento RegControl

Figura 19 – Método __repr__() responsável por criar a linha de comando do elemento RegControl

```
def __repr__(self):

    if self.buses == "":
        self.buses, self.kvas = RegControl.adapting_string_variables(self)

    return  (
f'New \"Transformer.{self.prefix_transformer}{self.transformer}" phases={self.phases} '
f'windings={self.windings} '
f'buses=[{self.buses}] '
f'conns=[{self.conn_p} {self.conn_s} {self.conn_t}] '
f'kvs=["tensao"] '
f'kvas=[{self.kvas}] '
f'xhl={self.xhl} '
f'%loadloss={self.loadloss:.3f} %noloadloss={self.noloadloss:.3f}'
f'\nNew \"Regcontrol.{self.prefix_transformer}{self.transformer}" transformer="{self.prefix_transformer}{self.transformer}" '
f'windings={self.windings} '
f'vreg={self.vreg} '
f'band={self.band} '
f'ptratio={self.ptratio}\n'
)
```

Fonte: Autor

Por fim, é chamado o método *create_output_file()* para criar o arquivo .dss definido no módulo *Utilidades*, ver Subseção 5.3.6.1. O nome do arquivo é passado para a classe case, onde será utilizado para criação dos arquivos *master* posteriormente.

Assim, dá-se a implementação do elemento RegControl.

5.3.4 LoadShape

O elemento representa curvas de carga típicas de consumidores do sistema elétrico. Na BDGD, informações de curva de carga estão presentes na tabela CRVCRG da BDGD.

Para o elemento *LoadShape*, o objetivo da biblioteca é gerar as linhas de comando para modelar as curvas de carga no software OpenDSS.

[6] Para entendimento do significado dos parâmetros, rever o Capítulo 3

O *LoadShape* tem sua modelagem no software OpenDSS descrita no Capítulo 3, onde é explicado cada parâmetro. Sendo assim, o foco deste capítulo é o contexto da implementação da conversão.

5.3.4.1 Considerações Iniciais/Implementação de pré-requisitos

A implementação do LoadShape, que tem como foco definir as curvas de carga da BDGD para o software OpenDSS, aporta um pré-requisito principal: adaptar as informações de consumo em potência trazidas na BDGD para o formato do OpenDSS.

A tabela *CRVCRG* apresenta perfis de consumo divididos em três categorias distintas: dias úteis (DU), sábados (SA) e domingos e feriados (DO). Dentro de cada categoria, são fornecidos dados de potência ativa diária em intervalos de 15 minutos, totalizando 96 valores ao longo do dia.

Para o OpenDSS, os valores esperados são a cada hora e em *p.u.*, ou seja, 24 pontos com valores entre 0 e 1. Assim, é necessário converter os dados.

Para solucionar esse problema, foi desenvolvido o método *process_loadshape()*, disponível no módulo de conversão (Sub-Seção 5.3.6.2). Inicialmente, o método converte os 96 valores de potência para 24 por meio da média a cada 4 valores. Em seguida, normaliza esses valores para obter, assim, as unidades em *p.u.*.

Por fim, o método *compute_loadshape_curve()* trata os valores e os adiciona ao dataframe central com as informações de todas as curvas de carga. Em seguida, a classe utilizará esse dataframe para popular as variáveis do objeto com os diferentes métodos de mapeamento.

Em relação ao mapeamento, foram empregados os métodos estático, direto e calculado.

Dessa forma, os requisitos preliminares para o elemento LoadShape foram atendidos e o desenvolvimento passa para o arquivo JSON.

5.3.4.2 Arquivo JSON

Como descrito anteriormente, o arquivo JSON funciona como o cérebro do software. Neste arquivo, são definidas as configurações de importação e mapeamento.

Em relação ao mapeamento das variáveis[7] no JSON , ficou definido conforme mostrado nas Tabelas 12 e 13.

[7] Para entendimento do mapeamento das variáveis, rever o Capítulo 4

Tabela 12 – Método Estático Loadshape

Variável	Valor
interval	1
npts	24

Tabela 13 – Método Direto Loadshape

Variável	Coluna Correspondente da BDGD
tipocc	COD_ID
tipodia	TIP_DIA
grupotensao	GRU_TEN

As colunas correspondentes se referem à tabela CRVCRG da BDGD.

O método calculado foi utilizado como flag para utilização dos métodos *compute_loadshape_curve()* e emphprocess_loadshape() para calcular os valores do parâmetro *mult* para modelagem do objeto LoadShape.

Em seguida, para importação dos dados da BDGD de forma eficiente, são definidas as colunas especificas da tabela CRVCRG. A Figura 20 ilustra tal fato.

Figura 20 – Configuração das colunas para importação da tabela CRVCRG da BDGD através do arquivo JSON.

```
"CRVCRG": {
    "columns": [
        "COD_ID", "TIP_DIA", "GRU_TEN", "POT_01", "POT_02", "POT_03", "POT_04", "POT_05", "POT_06", "POT_07", "POT_08",
        "POT_09", "POT_10", "POT_11", "POT_12", "POT_13", "POT_14", "POT_15", "POT_16", "POT_17", "POT_18", "POT_19",
        "POT_20", "POT_21", "POT_22", "POT_23", "POT_24", "POT_25", "POT_26", "POT_27", "POT_28", "POT_29", "POT_30",
        "POT_31", "POT_32", "POT_33", "POT_34", "POT_35", "POT_36", "POT_37", "POT_38", "POT_39", "POT_40", "POT_41",
        "POT_42", "POT_43", "POT_44", "POT_45", "POT_46", "POT_47", "POT_48", "POT_49", "POT_50", "POT_51", "POT_52",
        "POT_53", "POT_54", "POT_55", "POT_56", "POT_57", "POT_58", "POT_59", "POT_60", "POT_61", "POT_62", "POT_63",
        "POT_64", "POT_65", "POT_66", "POT_67", "POT_68", "POT_69", "POT_70", "POT_71", "POT_72", "POT_73", "POT_74",
        "POT_75", "POT_76", "POT_77", "POT_78", "POT_79", "POT_80", "POT_81", "POT_82", "POT_83", "POT_84", "POT_85",
        "POT_86", "POT_87", "POT_88", "POT_89", "POT_90", "POT_91", "POT_92", "POT_93", "POT_94", "POT_95", "POT_96"
    ],
    "type": {
        "COD_ID": "category",
        "TIP_DIA": "category",
        "GRU_TEN": "category"
    },
    "ignore_geometry": "False"
},
```

Fonte: Autor

Assim, dá-se a configuração do arquivo JSON para implementação do elemento.

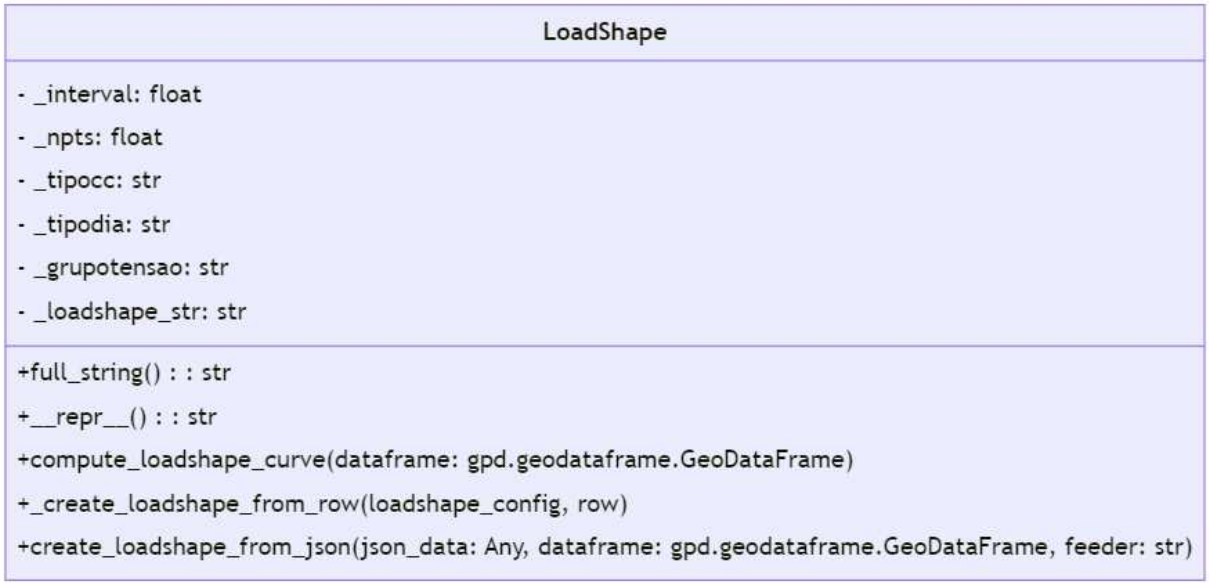
LoadShape.

5.3.4.3 Classe LoadShape

Após ter definido no arquivo JSON como encontrar os valores das variáveis e convertê-las, a classe busca colocar em prática a conversão (ver Seção 4.6).

No que se refere a classe *LoadShape*, a Figura 21 ilustra sua implementação através de um diagrama UML, onde são definidos variáveis, métodos de conversão e métodos auxiliares.

Figura 21 – UML LoadShape.



Fonte: Autor

Com o dataframe contendo as informações da BDGD para a entidade alvo como parâmetro do método *create_new_LoadShape_from_json()*, é possível acessar e converter os dados.

Em um segundo momento, é, então, construida a linha de declaração das curvas de carga no OpenDSS. O Comando 10 mostra um exemplo para o tipo de carga *"RES_11"*, o qual compreende uma curva para cada categoria de dia. Assim, são modeladas três *Loadshapes*. Cada linha de declaração se refere a um objeto *Loadshape* diferente, no OpenDSS e no BDGD Tools.

Isso foi realizado através do método *___rep__()*, que foi definido para que cada objeto gerado seja representado diretamente com sua linha de comando. A Figura 22 ilustra o método *___rep__()*, expondo como foi criada a linha. A ideia, como explicada anteriormente, é

a junção das variáveis definidas para o modelo do OpenDSS[8].

```
New "Loadshape.RES_11_DU" 24 1 mult=(0.055648, 0.007342,
   0.0, 0.008778, 0.001932, 0.000166, 0.012918, 0.100475,
   0.10467, 0.139616, 0.26267, 0.606768, 0.741636,
   0.547146, 0.275643, 0.201888, 0.17837, 0.325991,
   0.625152, 0.854422, 1.0, 0.709341, 0.350392, 0.122999)
New "Loadshape.RES_11_SA" 24 1 mult=(0.110923, 0.050782,
   0.015135, 0.008961, 0.001494, 0.0, 0.108135, 0.357065,
   0.194364, 0.290551, 0.489794, 0.585084, 0.767301,
   0.830828, 0.633775, 0.473066, 0.7442, 1.0, 0.970627,
   0.754954, 0.605297, 0.905407, 0.755153, 0.295927)
New "Loadshape.RES_11_DO" 24 1 mult=(0.081868, 0.040666,
   0.033083, 0.026641, 0.007583, 0.0, 0.01114, 0.064891,
   0.117098, 0.383439, 0.180513, 0.26842, 0.326198,
   0.546302, 0.413904, 0.368407, 0.202993, 0.442625, 1.0,
   0.438465, 0.58972, 0.514092, 0.489867, 0.251241)
```

Comando 10 – Linhas de declaração das curvas de carga de um perfil de consumidor.

Figura 22 – Método __repr__() responsável por criar a linha de comando do elemento LoadShape

```
def __repr__(self):
    return f"New \"Loadshape.{self.tipocc}_{self.tipodia}\" {self.npts} " \
           f"{self.interval} mult=({self.loadshape_str})"
```

Fonte: Autor

Por fim, é chamado o método *create_output_file()* para criar o arquivo .dss. Este foi definido no módulo *Utilidades*, ver Subseção 5.3.6.1. O nome do arquivo é passado para a classe *case*, onde será utilizado para criação dos arquivos *master* posteriormente.

Assim, dá-se a implementação do elemento LoadShape.

5.3.5 Load

O elemento representa cargas do sistema elétrico. Na BDGD, informações de carga estão presentes nas entidades de Unidade de Consumo, seja de baixa ou média tensão. Estas são UCBT e UCMT, respectivamente.

Para o elemento *Load*, o objetivo da biblioteca é gerar as linhas de comando para modelar as cargas no software OpenDSS.

[8] Para entendimento do significado dos parâmetros, rever o Capítulo 3

O *Load* tem sua modelagem no software OpenDSS descrita no Capítulo 3, onde é explicado cada parâmetro. Sendo assim, o foco deste capítulo é o contexto da implementação da conversão.

5.3.5.1 Considerações Iniciais

A implementação da conversão das informações das unidades consumidoras presentes na BDGD para o OpenDSS aporta um grau de complexidade maior em relação às implementações dos outros elementos. A intenção desta seção é apresentar os principais desafios, soluções encontradas e premissas adotadas.

Antes de tudo, é preciso definir a modelagem matemática a ser utilizada, ou seja, qual o vaor do parâmetro *modelo* do elemento *Load* no OpenDSS. Para o cálculo de perdas, é especificado que a parcela ativa da carga seja modelada em 50% como potência constante e 50% como impedância constante (ANEEL, 2021c). Sendo assim, cada carga é modelada a partir de dois elementos *Load*, sendo um do modelo 2 e outro modelo 3 do software OpenDSS. Na biblioteca, abordamos o conjunto de cargas M2 e M3 como um objeto Load da BDGD-Tools.

Na BDGD, para cada unidade consumidora, é associada uma classe. Uma unidade consumidora pode ser relacionada a uma classe de diferentes categorias, tais como residencial, industrial ou rural. Por exemplo, a classe de carga *RES_1*, da categoria residencial, assim como toda classe, é composta por três curvas de carga de tipo de dia, como foi visto na Seção 5.3.4: dias úteis (DU), sábados (SA) e domingos e feriados(DO). Consequentemente, cada unidade consumidora é modelada três vezes no OpenDSS, cada uma associada a um desses formatos descritos, visando capturar de forma mais precisa a realidade.

Adicionalmente, foi considerada a variação no consumo mensal. Dessa forma, levando em conta que três objetos (*Load*) são modelados para cada mês, resultando em um total de 36 desses objetos para representar uma unidade consumidora na Base de Dados de Grande Dimensão (BDGD) para análise anual.

A Figura 23 ilustra o processo de modelagem para uma unidade consumidora. No primeiro retângulo, da linha DU, a classe *Load* da biblioteca BDGD-Tools gera a linha de comando para instancializar duas cargas no OpenDSS: uma do modelo de potência constante e outra impedância constante - conjunto que chamamos de objeto load. Esses conjuntos são relacionados a curva de dias úteis (DU) de um tipo de carga. Em seguida, é iniciado um processo iterativo para cada mês, modificando apenas o valor da demanda máxima *(kw)*, gerando 12

conjuntos de carga. Consecutivamente, isso é realizado para SA e DO, totalizando em 36 conjuntos de carga, ou objetos *Load*, modelados no OpenDSS.

Figura 23 – Visão geral da representação de uma unidade consumidora para modelagem no OpenDSS.

Fonte: Autor

Outro ponto é o cálculo da demanda máxima, valor que não está diretamente disponível na BDGD. Este será descrito a seguir.

5.3.5.2 Cálculo da Demanda Máxima (kW)

O cálculo da demandá máxima da carga relacionada a uma curva de carga específica foi inspirado no processo feito pelo software Geoperdas da ANEEL.

Para a fórmula final, são utilizados as informações da BDGD de curva de carga e também energia consumida mensal de cada unidade consumidora. Além disso, é levado em consideração a quantidade de dias de cada tipo (DU, SA e DO) para cada mês específico.

Suponhamos, então, que queremos calcular a demanda máxima de uma carga relacionada a curva de carga DU de uma classe X.

Primeiramente, através do método *process_loadshape()*, descrito na Seção 5.3.4, tem-se os valores de potência em 24 pontos normalizada em p.u. das curvas de carga.

É definida, então, a energia total de uma classe através da soma da energia de cada tipo de dia. A Equação 5.1 exibe o cálculo.

$$\text{ener\_classe} = \sum_{i=0}^{24}(\text{POT\_DU}_i) + \sum_{i=0}^{24}(\text{POT\_SA}_i) + \sum_{i=0}^{24}(\text{POT\_DO}_i)(kW). \tag{5.1}$$

Em seguida, é definida uma proporção para cada tipo de dia da classe.

$$prop\_DU = \frac{\sum_{i=0}^{24}(\text{POT\_DU}_i)}{\text{ener\_classe}}. \tag{5.2}$$

$$prop\_SA = \frac{\sum_{i=0}^{24}(\text{POT\_SA}_i)}{\text{ener\_classe}}. \tag{5.3}$$

$$prop\_DO = \frac{\sum_{i=0}^{24}(\text{POT\_DO}_i)}{\text{ener\_classe}}. \tag{5.4}$$

Estes passos descritos até agora são efetuados através do método *compute_pre_kw()*, que armazena as informações de cada classe em um dataframe para uso posterior.

Com o objetivo de calcular a demanda máxima a cada mês, o **proporcional de cada curva de tipo de dia do mês** foi calculado. Assim:

$$ener\_Prop\_DU\_MES = prop\_DU \times qtd\_tipdia\_mes. \tag{5.5}$$

$$ener\_Prop\_SA\_MES = prop\_SA \times qtd\_tipdia\_mes. \tag{5.6}$$

$$ener\_Prop\_DO\_MES = prop\_DO \times qtd\_tipdia\_mes. \tag{5.7}$$

Para determinar a quantidade do tipo de dia de um mês específico, foi criado o método *qt_tipdia_mes*, que recebe como parâmetro o tipo de dia e o mês.

Em seguida, calculamos o valor de potência proporcional da curva DU mensal, como mostra a Equação 5.8:

$$ener\_Prop\_DU\_MENSAL = \frac{ener\_Prop\_DU\_MES}{ener\_Prop\_DU\_MES + ener\_Prop\_SA\_MES + ener\_Prop\_DO\_MES} \tag{5.8}$$

É preciso calcular o parâmetro fator de carga. Para isto, temos as Equções 5.9 e 5.11.

$$Pot\_ATV\_Media\_DU = \frac{\sum_{i=0}^{24}(\text{POT\_DU}_i)}{24}(kw). \tag{5.9}$$

$$Pot\_ATV\_Max\_DU = \max(POT\_DU_i)(kw). \tag{5.10}$$

Logo, podemos calcular o fator de carga:

$$fc = \frac{Pot\_ATV\_Media\_DU}{Pot\_ATV\_Max\_DU}. \tag{5.11}$$

Assim, podemos enfim calcular a demanda máxima da carga para a curva de carva de dias úteis da classe X para um mês específico, como mostra a Equação 5.12.

$$kw = \frac{Energia\_Mes \times (ener\_Prop\_DU\_MENSAL_MES \times 1000)}{\text{qt\_tipdia\_mes} \times 24 \times \text{fc}}. \tag{5.12}$$

Em relação ao mapeamento, foram empregados os métodos estático, direto, indireto e iterativo.

Dessa forma, os requisitos preliminares para o elemento Load foram atendidos e o desenvolvimento passa para o arquivo JSON.

5.3.5.3 Arquivo JSON

Como descrito anteriormente, o arquivo JSON funciona como o cérebro do software. Neste arquivo, são definidas as configurações de importação e mapeamento.

Em relação ao mapeamento das variáveis[9] no JSON , ficou definido de acordo com as tabelas a seguir:

Tabela 14 – Método Estático Load - UCMT

Variável	**Valor**
pf	0.92
vminpu	0.93
vmaxpu	1.50
status	variable

As tabelas mostram essa configuração para a entidade UCMT da BDGD. Processo similar foi feito para as outras entidades que são representadas pelo elemento Load do OpenDSS: UCBT e PIP.

[9] Para entendimento do mapeamento das variáveis, rever o Capítulo 4

Tabela 15 – Método Direto Load - UCMT

UCMT_tab	**Coluna Correspondente da BDGD**
bus1	PAC
load	PN_CON
daily	TIP_CC
feeder	CTMT
energia_01	ENE_01
... (continua da energia_02 a energia_12)	

Tabela 16 – Método Indireto Load - UCMT

UCMT_tab	**Correspondente da BDGD**	**Funções Conversoras**
phases	FAS_CON	convert_tfascon_phases
conn	FAS_CON	convert_tfascon_conn
bus_nodes	FAS_CON	convert_tfascon_bus
kv	TEN_FORN	convert_tten

Em seguida, para importação dos dados da BDGD de forma eficiente, são definidas as colunas especificas das tabelas das entidades da BDGD. A Figura 24 ilustra tal processo para a entidade UCMT.

Em relação às conexões, em um primeiro momento, foram todas consideradas em *delta.*

Assim, dá-se a configuração do arquivo JSON para implementação do elemento.

Figura 24 – Configuração das colunas para importação da tabela da entidade UCMT através do arquivo JSON.

Fonte: Autor

5.3.5.4 Classe Load

Após ter definido no arquivo JSON como encontrar os valores das variáveis e convertê-las, a classe busca colocar em prática a conversão (ver Seção 4.6).

No que se refere a classe *Load*, a Figura 25 ilustra sua implementação através de um

diagrama UML, onde são definidos variáveis, métodos de conversão e métodos auxiliares.

Figura 25 – UML Load.

Fonte: Autor

Com o dataframe contendo as informações da BDGD para a entidade alvo como parâmetro do método *create_new_Load_from_json()*, é possível acessar e converter os dados. Esse dataframe pode ser referente às unidades mencionadas anteriormente: UCBT, UCMT ou PIP.

Em um segundo momento, é, então, construída a linha de declaração das cargas no OpenDSS. O Comando 10 mostra um exemplo para uma carga da baixa tensão, entidade UCBT. O código compreende dois elementos *Load*, um para potência constante e outro impedância constante, como descrito na Seção anterior.

Isso foi realizado através do método *___rep__()*, que foi definido para que cada objeto gerado seja representado diretamente com sua linha de comando. A Figura 22 ilustra o método *___rep__()*, expondo como foi estruturada a linha de comando. A ideia, como explicada anteriormente, é a junção das variáveis definidas para o modelo do OpenDSS[10].

Por fim, é chamado o método *create_output_file()* para criar o arquivo .dss. Este foi definido no módulo *Utilidades*, ver Subseção 5.3.6.1. O nome do arquivo é passado para a classe

[10] Para entendimento do significado dos parâmetros, rever o Capítulo 3

```
New "Load.BT_2836080_22697_M1" bus1="UC2836080.1.2.3.4"
   phases=3 conn=Delta model=2 kv=0.38 kw = 0.3714684 pf
   =0.92 status=variable vmaxpu=1.5 vminpu=0.92 daily="
   CO5_DO"
New "Load.BT_2836080_22697_M2" bus1="UC2836080.1.2.3.4"
   phases=3 conn=Delta model=3 kv=0.38 kw = 0.3714684 pf
   =0.92 status=variable vmaxpu=1.5 vminpu=0.92 daily="
   CO5_DO"
```

Comando 11 – Linhas de declaração das curvas de carga de um perfil de consumidor.

Figura 26 – Método __repr__() responsável por criar a linha de comando para os elementos *Loads*

```
def __repr__(self):


    return f'New \"Load.{self.entity}_{self.load}_{self.id}_M1" bus1="{self.bus1}.{self.bus_nodes}" ' \
        f'phases={self.phases} conn=Delta model=2 kv={self.kv} kw = {self.kw/2:.7f} '\
        f'pf={self.pf} status=variable vmaxpu={self.vmaxpu} vminpu={self.vminpu} ' \
        f'daily="{self.daily}_{self.tip_dia}" \n'\
        f'New \"Load.{self.entity}_{self.load}_{self.id}_M2" bus1="{self.bus1}.{self.bus_nodes}" ' \
        f'phases={self.phases} conn=Delta model=3 kv={self.kv} kw = {self.kw/2:.7f} '\
        f'pf={self.pf} status=variable vmaxpu={self.vmaxpu} vminpu={self.vminpu} ' \
        f'daily="{self.daily}_{self.tip_dia}"\n '
```

Fonte: Autor

case, onde será utilizado para criação dos arquivos *master* posteriormente.

Assim, dá-se a implementação do elemento Load.

5.3.6 Utilidades e Conversões

5.3.6.1 Utilidades

O módulo de utilidades é um conjunto de funções encapsuladas em um único arquivo, destinado a fornecer funcionalidades genéricas e úteis para um conjunto variado de aplicações. Neste caso, as contribuições envolvem funções para manipulação de dados geoespaciais e geração de arquivos .dss, oferecendo um conjunto de ferramentas para facilitar tarefas específicas encontradas durante o desenvolvimento.

5.3.6.1.1 Manipulação de Dados

– **Função *merge_entities_tables(dataframe1, dataframe2)***

Esta função tem como objetivo mesclar dois GeoDataFrames de entidades com base

nos seus índices. Aqui está uma visão detalhada do que ela realiza:

Propósito: Fusão de GeoDataFrames.

Parâmetros: dataframe1 e dataframe2: São os dois GeoDataFrames que serão mesclados. Normalmente refere-se a duas entidades que precisam ser unidas para fornecer as informações à modelagem de um elemento do OpenDSS.

Processo: Utiliza o método join para juntar dataframe1 e dataframe2, especificando um sufixo para colunas que possam ter nomes duplicados. Identifica colunas duplicadas que foram criadas durante a junção, com sufixo _left. Remove as colunas duplicadas para manter apenas uma cópia de cada uma. Retorno:

Retorna o GeoDataFrame resultante da fusão, contendo os dados mesclados de dataframe1 e dataframe2 sem as colunas duplicadas.

5.3.6.1.2 Criação de Arquivos .DSS

– **Função *create_output_file(object_list, file_name, object_lists, file_names, feeder)***

Esta função é responsável por criar arquivos de saída e escrever dados a partir de uma lista de objetos. Foi arquitetada para ser flexível podendo lidar com diferentes tipos de objetos, no caso elementos. Além disso, é adaptada para o contexto onde é preciso gerar mais de um arquivo por elemento, como no caso das cargas.

Propósito: Criação de arquivos de saída a partir de listas de objetos.

Parâmetros:

1. object_list: Lista de objetos que serão escritos no arquivo.
2. file_name: Nome do arquivo de saída.
3. object_lists e file_names: Listas de listas de objetos e nomes de arquivos (possivelmente para lidar com vários conjuntos de objetos e nomes de arquivos simultaneamente).
4. feeder: Parece ser uma variável indicando a qual "alimentador"os arquivos pertencem.

Processo:

1. Verifica se existe um diretório chamado "output"e o cria se não existir.
2. Verifica e cria um subdiretório específico para o "alimentador"dentro do diretório "output".
3. Cria um arquivo no subdiretório com o nome especificado (file_name) e escreve os dados dos objetos no arquivo.

Retorn:

Retorna o nome do arquivo de saída gerado.

5.3.6.2 Conversões

O módulo de conversões é um conjunto de funções encapsuladas em um único arquivo, destinado a fornecer conversões para um conjunto de variáveis. Neste caso, as contribuições envolvem as conversões já presentes no módulo 10, assim como funções adicionais que foram necessárias ao longo do trabalho. As funções já presentes do módulo foram simplesmente implementadas em forma de dicionário interno que mapeia códigos para seus reais valores. Nesta seção, serão mencionadas apenas as funções não presentes no módulo 10 do PRODIST.

5.3.6.2.1 Processamento de Dados de Carga

– **Função *process_loadshape(loadshape_list)***

Esta função é responsável por processar dados de carga, calculando a média de conjuntos de quatro números em uma lista de 96 pontos e normalizando os valores resultantes entre 0 e 1.

Propósito: Processamento de dados de carga para condensação e normalização.

Parâmetros:

1. loadshape_list: Lista de 96 números representando dados de carga ao longo do dia, ou seja, a cada 15 minutos.

Processo:

1. Divide a lista de 96 números em conjuntos de quatro.
2. Calcula a média de cada conjunto de quatro números, tendo um valor médio para cada hora.
3. Normaliza os valores resultantes entre 0 e 1, considerando o maior valor na lista das médias calculadas.

Retorno:

Retorna duas listas:

Lista normalizada entre 0 e 1, representando a potência média de cada conjunto de quatro números. Lista original contendo as potências médias dos conjuntos de quatro números da lista de entrada.

Esse método é útil para compactar e normalizar dados de carga, adequando ao formato do OpenDSS.

– Função *convert_tfascon_bus(case)*

Esta função converte a representação de um tipo específico de barra (nó). Mapeia diferentes códigos para uma convenção de numeração ou identificação das barras.

Propósito: Conversão da representação de barras em sistemas elétricos.

Parâmetros:

case: Código representando a ligação, qual definirá as ligações dos nós.

Processo:

A função recebe um código representativo da ligação da barra. Utiliza um dicionário interno para mapear esse código para uma convenção específica de numeração ou identificação das barras.

Retorno: Retorna a identificação específica. Se o código não for encontrado no dicionário interno, retorna *'Invalid case'* para facilitar o varrimento de erros.

– Função *convert_tfascon_bus_prim(case)*, *convert_tfascon_bus_sec(case)* e *convert_tfascon_bus_terc(ca*

Semelhantes à função anterior, essas convertem tipos de ligação em representação das ligações dos nós das barras primárias, secundárias e terciárias, quando for o caso, de transformadores para o OpenDSS. São mapedados diferentes códigos de ligação para uma identificação específica de ligação dos nós para o OpenDSS.

Propósito: Conversão do tipo de ligação para identificação da ligação dos nós nas barras primárias, secundárias e terciárias do transformador.

Parâmetros:

case: Código representando o tipo de ligação.

Retorno: Retorna a identificação específica à ligação fornecida.

5.4 Considerações Finais do Capítulo

Foi possível adentrar nos detalhes da metodologia abordada para efetuar o trabalho. Assim, as principais ferramentas e recursos de utilidade e conversão foram apresentados; assim como a implementação de cada elemento modelado para o OpenDSS a partir da BDGD.

t

6 RESULTADOS

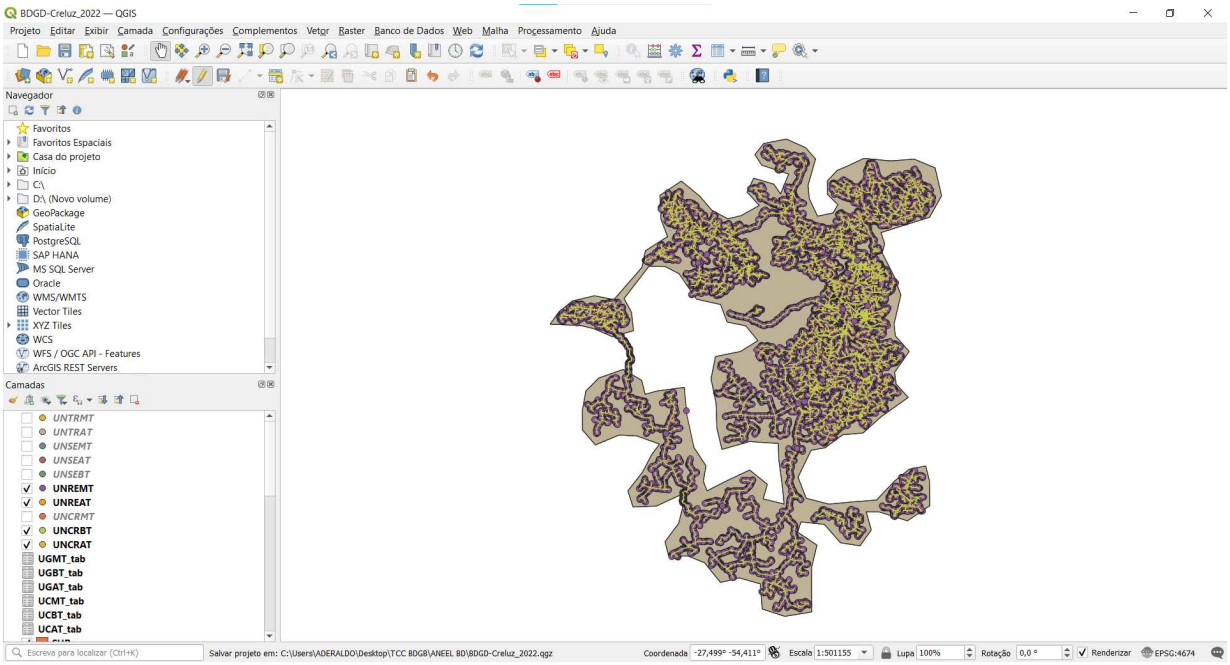
A validação de ferramentas e métodos é um passo crucial no desenvolvimento de soluções tecnológicas. No contexto deste trabalho, o presente capítulo apresenta os resultados obtidos a partir da BDGD-Tools, focando na comparação com o método estabelecido pela ANEEL (Agência Nacional de Energia Elétrica), que faz uso dos softwares Geoperdas e ProgGeoperdas (PAULISTA *et al.*, 2023). Este método é utilizado como parâmetro de referência nesta primeira versão funcional da biblioteca, pelo seu reconhecimento no setor, visto que é estabelecido por um órgão oficial do governo e influencia diretamente em reajustes tarifários. É válido destacar ainda que a comparação com o método da ANEEL é uma abordagem para esta primeira versão funcional, sendo outras formas de validação necessárias em trabalhos futuros, como mencionado no Capítulo **??**.

Para conduzir esta validação, utilizamos a base de dados geográfica da distribuidora *Creluz* do ano de 2022 (CRELUZ, 2022), localizada no Rio Grande do Sul, como um estudo de caso representativo. A Figura 27 expõe sua visualização geográfica através do software QGIS.

Este capítulo explora os três principais pontos de validação: a **estrutura da rede elétrica** resultante da conversão de dados no *OpenDSS*, **fluxo de potência** e o **perfil de tensão** gerado. Esses pontos são fundamentais para avaliar a precisão e a confiabilidade da BDGD-Tools, permitindo uma análise detalhada da performance da biblioteca em relação aos métodos convencionais da ANEEL.

Ao apresentar e analisar os resultados dessa validação, este capítulo visa contribuir para o entendimento da eficácia da BDGD-Tools como uma ferramenta viável na conversão de dados da BDGD para o OpenDSS, além de identificar possíveis áreas para melhorias e refinamentos futuros. A compreensão desses resultados é essencial para aprimorar a confiabilidade e a utilidade prática dessa biblioteca no contexto da análise e simulação de sistemas elétricos de distribuição.

Figura 27 – Representação geográfica da rede elétrica de distribuição gerida pela Creluz através de sua BDGD de 2022 no software QGIS.



Fonte: Autor

6.1 Considerações Iniciais

Para realizar a avaliação da estrutura da rede elétrica, o procedimento adotado baseou-se na análise dos arquivos .dss gerados pela BDGD-Tools e na utilização do OpenDSS como plataforma de simulação e visualização. Os arquivos .dss representam a estrutura convertida da rede elétrica, contendo informações detalhadas sobre os elementos e parâmetros do sistema.

O OpenDSS foi empregado para carregar e interpretar os arquivos .dss, permitindo uma análise mais detalhada e visualização da rede elétrica convertida.

Para a BDGD alvo, foi escolhido um alimentador em específico para ser feita a análise, o *"1_3PAS_1"*. Sendo assim, todos resultados exibidos serão referentes a este alimentador.

As seções a seguir entrarão em detalhes sobre as diferentes formas de resultados e validação.

6.2 Resultados e Validação: Estrutura da Rede Elétrica no OpenDSS

6.2.1 Resultados da Estrutura da Rede Elétrica no OpenDSS

Esta seção busca oferecer uma análise detalhada da estrutura da rede elétrica construída no OpenDSS a partir dos resultados da conversão realizada pela BDGD-Tools. O objetivo

é fornecer uma visão abrangente dos diferentes elementos presentes na rede, destacando a quantidade e distribuição de cada componente.

Essa análise quantitativa é fundamental para avaliar a fidelidade e a representatividade da estrutura convertida, proporcionando insights valiosos sobre a composição da rede elétrica modelada no OpenDSS. A Tabela 17 traz essas informações.

Tabela 17 – Informações sobre a rede elétrica no OpenDSS gerada a partir do resultado da conversão.

Categorias	**Contagem**
buses	1856
nodes	4403
ckt elements	3094
vsource	1
line	1720
transformer	135
reactor	135
energymeter	1
load	1102

Esses dados fornecem uma visão abrangente da estrutura da rede elétrica construída no OpenDSS a partir da saída da BDGD-Tools. Cada categoria representa diferentes elementos e sua quantidade na rede convertida. Vamos analisar algumas informações específicas:

- **Buses (Barramentos)**: 1867 barramentos foram identificados na rede. Os barramentos são pontos de conexão onde os dispositivos elétricos estão ligados.
- **Nodes (Nós):** 4432 nós foram definidos na rede. Os nós são pontos específicos nas barra. Em um nó, pode estar conectada uma fase, por exemplo.
- **Ckt Elements (Elementos de Circuito):** Totalizando 3107 elementos de circuito na rede, essa categoria inclui uma variedade de componentes, como linhas, transformadores, reatores, e outros dispositivos que compõem o circuito.
- **Vsource (Fonte de Tensão):** Uma única fonte de tensão foi identificada na rede, referente ao circuito de média tensão, identificado como *"1_3PAS_1"*, alimentador objet de estudo.
- **Line (Linhas):** 1729 linhas estão presentes na estrutura da rede. As linhas representam os segmentos físicos de transmissão de energia. São resultados das entidades UNSEMT, UNSEBT e RAMLIG.
- **Transformer (Transformadores):** Foram encontrados 135 transformadores na rede, que desempenham um papel crucial na alteração dos níveis de tensão entre diferentes partes do sistema.

- **Reactor (Reatores):** Também foram identificados 135 reatores na rede, coerente com o valor de transformadores. Os reatores são utilizados para modelar a resistência de aterramento dos transformadores.
- **Energymeter (Medidor de Energia):** Um único medidor de energia foi incluído na estrutura, essencial para medição e monitoramento, colocado propositalmente para obter resultados. Esta informação não provém da BDGD.
- **Load (Cargas):** Um total de 1126 cargas foram definidas na rede, representando os consumidores de energia elétrica. Este valor se refere aos elementos instancializados, ou seja, no total, estão sendo representadas

À primeira vista, é evidente que os números estão alinhados com a escala do alimentador. No entanto, ao examinar mais minuciosamente, torna-se aparente que esses números podem não corresponder completamente ao esperado, considerando a quantidade de elementos presentes na BDGD. A discrepância na quantidade é relativamente pequena.

Por outro lado, outros valores como o da quantidade de transformadores, reatores e cargas está de acordo com a BDGD.

6.2.2 Validação da Estrutura da Rede Elétrica no OpenDSS

Após um primeiro olhar sob a estrutura da rede elétrica no OpenDSS a partir da saída da BDGD-Tools, é possível comparar com a rede elétrica gerada pelo método da ANEEL para a mesma base de dados.

A Tabela 18 expões os dados da modelagem da rede elétrica no OpenDSS a partir dos arquivos dss obtidos com o método da ANEEL.

Tabela 18 – Informações sobre a rede elétrica no OpenDSS gerada a partir do método da ANEEL.

Categoria	Contagem
buses	1856
nodes	4403
ckt elements	3094
vsource	1
line	1720
transformer	135
reactor	135
energymeter	1
load	1102

Ao comparar com a Tabela 17, pode-se perceber uma mesma quantidade de barras, transformadores, reatores e circuitos.

No entanto, divergem em valores de nós, quantidade de elementos do circuito, cargas e linhas. Essa divergência dá-se a processos ainda não implementados na biblioteca BDGD Tools, tais como o recurso de validação efetuado pelo GeoPerdas diretamente em SQL. Exemplo disso, é a discrepância entre o número esperado para linhas, que é de 1729, e o valor modelado, 1720. Outro exemplo são as cargas, com uma quantidade de 1102 cargas ao invés das 1126 presentes na BDGD da Creluz de 2022.

A análise da estrutura da rede elétrica no OpenDSS revelou uma representação fiel das entidades da BDGD. A comparação com a modelagem da ANEEL destacou semelhanças e divergências, apontando áreas ainda não completamente integradas na BDGD-Tools. Essas discrepâncias fornecem direcionamento para melhorias futuras, evidenciando a necessidade contínua de refinamento da ferramenta.

A próxima etapa de validação do fluxo de potência buscará uma comparação detalhada entre os resultados obtidos, aprofundando a compreensão da fidelidade da conversão.

6.3 Resultados e Validação: Fluxo de Carga

Nesta seção, são apresentados os resultados gerais do alimentador, focando especialmente no início do alimentador.

Os resultados obtidos para a potência ativa total no início do alimentador revelam uma estreita semelhança entre os valores gerados pela BDGD-Tools e os obtidos pelo método da ANEEL.

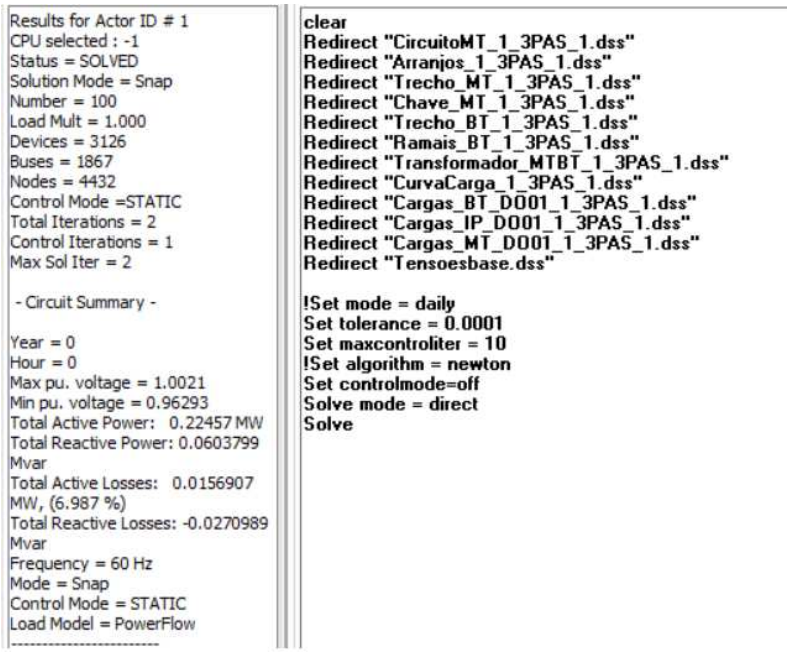
- **BDGD-Tools:**
  - Potência Ativa Total: 0.22457 MW
  - Perdas Ativas Totais: 0.0156907 MW (6.987%)
- **Método da ANEEL:**
  - Potência Ativa Total: 0.234208 MW
  - Perdas Ativas Totais: 0.015882 MW (6.781%)

Em relação a potência ativa total, o erro é de aproximadamente 4,1%, enquanto que as perdas ativas totais variam em 0,206 p.p. entre as duas redes elétricas. O erro da potência ativa total gera um ponto de investigação para trabalhos futuros: as cargas modeladas.

A Figura 28 apresenta os resultados da BDGD-Tools para o fluxo de potência, enquanto a Figura 29 representa os resultados do método da ANEEL.

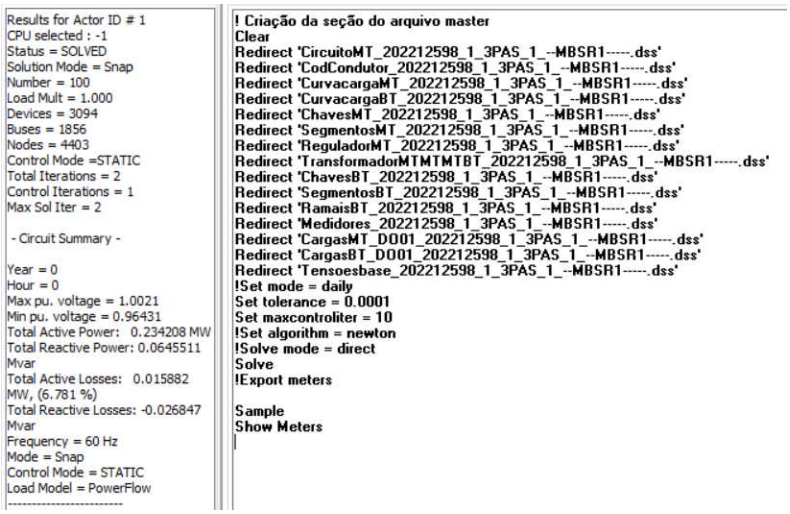
A comparação direta entre os resultados mostra um erro aceitável para a potência

Figura 28 – Resultado do fluxo de potência para a rede elétrica resultado da conversão da bdgd-tool para a BDGD da Creluz.



Fonte: Autor

Figura 29 – Resultado do fluxo de potência para a rede elétrica resultado do método da ANEEL para a BDGD da Creluz.



Fonte: Autor

ativa total e perdas ativas fornecidas pela BDGD-Tools e pelo método da ANEEL. A diferença observada nos valores pode ser atribuída a pequenas variações nos parâmetros ou métodos de cálculo utilizados por cada abordagem.

Essa proximidade sugere que a BDGD-Tools foi capaz de gerar resultados comparáveis e confiáveis em relação aos métodos estabelecidos pela ANEEL, demonstrando uma consistência notável na determinação da potência ativa total e das perdas ativas no sistema elétrico modelado.

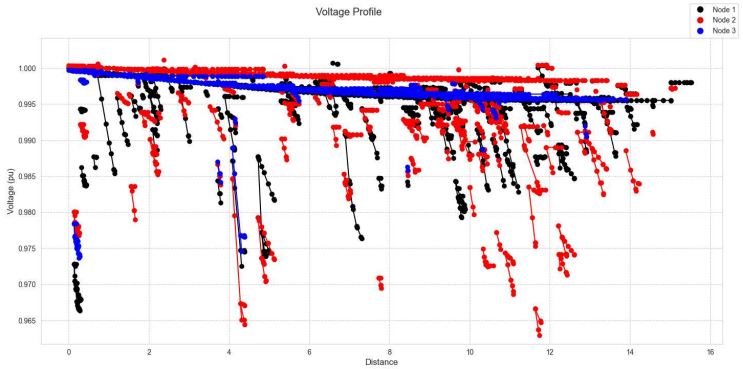
A análise dos resultados do fluxo de potência reforça a confiabilidade da BDGD-Tools na estimativa da potência ativa total e das perdas ativas, evidenciando a eficácia da ferramenta na conversão dos dados da BDGD para simulação do sistema elétrico no OpenDSS.

6.4 Resultados e Validação: Perfil de Tensão

A análise do perfil de tensão é essencial para compreender a distribuição e estabilidade da tensão ao longo da rede elétrica. Esta seção apresenta os resultados obtidos pela BDGD-Tools e pelo método da ANEEL em relação ao perfil de tensão, destacando as variações nos valores de tensão ao longo do sistema.

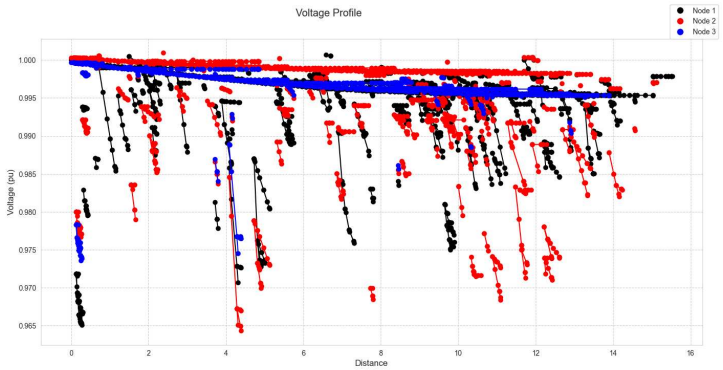
A Figura 30 correspondente ao perfil de tensão no OpenDSS para a BDGD-Tools e a Figura (31) para o método da ANEEL. Elas dão suporte na visualização e compreensão das diferenças nas tensões ao longo da rede elétrica.

Figura 30 – Resultado do perfil de tensão para a rede elétrica da conversão da bdgd-tool para a BDGD da Creluz.



Fonte: Autor

Figura 31 – Resultado do perfil de tensão para a rede elétrica do método da ANEEL para a BDGD da Creluz.



Fonte: Autor

Pode-se compreender, a partir da Figura 30 que as tensões para as fases apresentam valores em pu que diminuem com a distância do alimentador como esperado de um sistema de distribuição, confirmando que os equipamentos elétricos tem forte evidência de estarem conectados corretamente.

Ao comparar com o perfil de tensão da rede elétrica resultado dos métodos da ANEEL, na Figura 31, é possível ver uma distorção no gap entre as cargas de média e baixa tensão. No perfil de tensão da rede gerada pela BDGD Tools, esse gap é maior, com a tensão mínima chegando até mesmo a 0,96293 pu. Isto dá indícios de que alguns transformadores podem não estar modelados como deveriam.

Assim, com uma investigação utilizando *py-dss-interface* (RADATZ; VIANA, 2023b), divergências nos transformadores foram encontradas devido à algumas nuances ainda não contempladas na biblioteca

BDGD-Tools:

- Tensão Mínima: Valores próximos a 0.96293 pu
- Tensão Máxima: Valores próximos a 1.0021 pu

Método da ANEEL:

- Tensão Mínima: Valores próximos a 0.96431 pu
- Tensão Máxima: Valores próximos a 1.0021 pu

No geral, o erro entre as tensões máxima e mínima foram de 0,143% e 0%, respectivamente. Isso demonstra a eficácia da ferramenta.

6.5 Ambiente de Registro de Erros em Comparação com a Rede de Referência

Com o propósito de detectar erros de maneira precisa e automatizada, desenvolveu-se uma ferramenta em Python, um ambiente de registro de erros.

Nesse ambiente, é realizada uma minuciosa comparação entre a rede resultante do método da ANEEL e aquela produzida pela ferramenta BDGD-Tools. A ferramenta de análise é construída com a biblioteca Python *py-dss-interface* (RADATZ; VIANA, 2023b). Por meio dela, é possível acessar os dados modelados no OpenDSS em *dataframes*, simplificando a análise das redes por elemento.

Consequentemente, cada elemento das redes é cuidadosamente confrontado para identificar discrepâncias. Essas disparidades são meticulosamente documentadas em arquivos de saída no formato CSV, sendo atribuído um arquivo específico para cada entidade analisada. No

arquivo, são encontradas as seguintes informações:

- Identificação do elemento.
- Parâmetro.
- Valor do parâmetro para a rede criada pela BDGD-Tools.
- Valor do parâmetro para a rede criada pelo método da ANEEL.

Esse processo oferece uma visão detalhada das diferenças entre as redes, facilitando a identificação e correção de possíveis erros ou inconsistências, aspecto crucial para manter a integridade e precisão dos dados.

6.6 Trabalhos Futuros

No geral, reconhece-se que o método da ANEEL abstrai características da rede para simplificar a modelagem, o que pode limitar a fidelidade em relação à rede elétrica real. Nesse contexto, a BDGD-Tools tem como objetivo, a médio prazo, não apenas replicar os propósitos da ANEEL, mas ser utilizada para estudos complexos nas redes de distribuição. Esta seção aborda estratégias e os pontos necessários para alcançar esse patamar almejado.

Inicialmente, ao replicar o método da ANEEL, foi desenvolvida uma metodologia de conversão da BDGD com uma arquitetura flexível, visando possíveis alterações futuras. Isso se torna uma base sólida para a construção de uma versão mais detalhada e fiel à rede do que o método atual da ANEEL.

O primeiro passo é equiparar a BDGD-Tools ao método da ANEEL para, posteriormente, aprimorar a biblioteca.

Para replicar o método da ANEEL com maior fidelidade, alguns pontos precisam ser abordados:

- **Criação de arquivos dll restantes**: implementar a criação do arquivo dll *tensão de bases* ainda não criado de forma automática pela biblioteca.
- **Aprimoramento da precisão:** correção dos pontos identificados nos registros de erros detalhado na Seção **??**
- **Validação em diferentes bases de dados:** para testar completamente a biblioteca, é necessário validar em diferentes BDGDs de outras distribuidoras.
- **Personalização de Parâmetros:** adição de recursos como no GeoPerdas, permitindo que o usuário ajuste a tensão mínima nas cargas ou limite a distância do ramal em 30 metros, por exemplo.

Uma vez que uma réplica do método da ANEEL seja alcançada, o próximo estágio no desenvolvimento da ferramenta é buscar uma modelagem ainda mais fidedigna à rede elétrica da distribuidora. Isso inclui a implementação das características abstraídas pelo método da ANEEL, como a consideração de capacitores, cabos com geometria, entre outros recursos que impactam na modelagem final. Nesse processo, métodos alternativos de validação devem ser considerados.

Para fins de visualização, a incorporação de mapas interativos ou gráficos para as coordenadas geográficas é uma possibilidade a ser explorada. Além disso, a implementação de uma interface visual de usuário completaria o ciclo rumo à democratização da ferramenta.

6.7 Considerações Finais

Cada análise realizada reforçou a confiabilidade da BDGD-Tools na tradução precisa da rede elétrica da BDGD para o OpenDSS.

- **Estrutura da Rede Elétrica:** A consistência e coerência na representação dos elementos fundamentais conferem solidez à estrutura convertida.
- **Fluxo de Potência:** O erro aceitável de 4,1% de potência ativa e 0,206 p.p para perdas ativas entre a BDGD-Tools e o método da ANEEL ressalta a precisão da simulação.
- **Perfil de Tensão:** O comportamento geral no perfil de tensão reflete uma representação consistente da rede, justificando que os elementos estão conectados corretamente, apresentando erro de 0% e 0,143% para tensões mínima e máxima, se comparadas com o método da ANEEL. No mais, apresenta divergências justificadas pelas nuances ainda não implementadas da modelagem dos transformadores.

Assim, cada validação contribuiu com uma perspectiva diferente que, em conjunto, reforça a confiabilidade e utilidade prática da BDGD-Tools na modelagem precisa e coerente de sistemas elétricos a partir da BDGD.

Além disso, é apresentada neste capítulo uma ferramenta de registro de erros para auxiliar na identificação de problemas de forma precisa e automatizada.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este estudo representa um passo significativo em direção à superação dos desafios enfrentados pelas concessionárias de energia na análise e simulação de redes elétricas de distribuição, especialmente diante das exigências técnicas da ANEEL. A BDGD-Tools emergiu como uma resposta inovadora para preencher lacunas identificadas na conversão de dados da Base de Dados da Distribuidora para o formato do OpenDSS.

A BDGD-Tools demonstrou sua importância prática ao viabilizar a simulação precisa de redes elétricas de distribuição brasileiras no ambiente do OpenDSS a partir de uma BDGD. Sua contribuição para a reprodução fiel desses sistemas elétricos se destaca como um marco significativo na análise e avaliação da infraestrutura do setor de distribuição, além trazer seu caráter livre com simplicidade de utilização.

Portanto, este estudo cumpriu com o objetivo de tornar funcional a biblioteca Python BDGD-Tools e demonstrar a sua eficácia como uma ferramenta de conversão da BDGD em sistema elétrico modelado. Além disso, delineia claramente as direções futuras para o seu aprimoramento e desenvolvimento, visando atender às demandas crescentes e complexas do setor de distribuição de energia elétrica.

REFERÊNCIAS

ANEEL. **Dados Abertos - ANEEL**. São Paulo: SMA, 2021. Disponível em: https://dadosabertos.aneel.gov.br/. Acesso em: 8 mar. 2023.

ANEEL. **Módulo 10 - Sistema de Informação Geográfica Regulatório**. São Paulo: SMA, 2021. Disponível em: https://www.gov.br/aneel/pt-br/centrais-de-conteudos/procedimentos-regulatorios/prodist. Acesso em: 8 mar. 2023.

ANEEL. **Módulo 7 - Cálculo de Perdas na Distribuição**. São Paulo: SMA, 2021. Disponível em: https://www.gov.br/aneel/pt-br/centrais-de-conteudos/procedimentos-regulatorios/prodist. Acesso em: 8 mar. 2023.

ANEEL. **Base de Dados Geografica da Distribuidora ENEL CE 39 - 2021**. 2023. Disponível em: https://dadosabertos-aneel.opendata.arcgis.com/datasets/aneel::enel-ce-39-2021-12-31-v10-20230124-0859-gdb-zip/about. Acesso em: 11 nov. 2023.

BDGD-Tools. **Documentação da BDGD Tools**. 2023. https://bdgd-tools.readthedocs.io/. Acessado em 14 de novembro de 2023.

CRELUZ. **CRELUZ-D 598 2022-12-31 V11 20230831-0921**. 2022. Disponível em: https://dadosabertos-aneel.opendata.arcgis.com/datasets/b3ba770ad80d49da8460d7244d515701/about. Acesso em: 14 nov. 2023.

GitHub Python DSS Tools. **GitHUB BDGD Python DSS Tools**. 2023. https://github.com/PauloRadatz/py-dss-tools. Acessado em 14 de novembro de 2023.

JORDAHL, K.; BOSSCHE, J. V. den; FLEISCHMANN, M.; WASSERMAN, J.; MCBRIDE, J.; GERARD, J.; TRATNER, J.; PERRY, M.; BADARACCO, A. G.; FARMER, C.; HJELLE, G. A.; SNOW, A. D.; COCHRAN, M.; GILLIES, S.; CULBERTSON, L.; BARTOS, M.; EUBANK, N.; MAXALBERT; BILOGUR, A.; REY, S.; REN, C.; ARRIBAS-BEL, D.; WASSER, L.; WOLF, L. J.; JOURNOIS, M.; WILSON, J.; GREENHALL, A.; HOLDGRAF, C.; FILIPE; LEBLANC, F. **geopandas/geopandas: v0.8.1**. 2020. https://bdgd-tools.readthedocs.io/. Acessado em 14 de novembro de 2023.

PAULISTA, V.; OLIVEIRA, R. F.; NASCIMENTO. Impacto da implementação de bancos de capacitores na apuração das perdas regulatórias. **São Judas**, 2023. Orientador: Prof. Dr. André Luiz de Oliveira.

RADATZ; ROCHA. **notas-tecnicas-opendss**. São Paulo: USP EPRI, 2017.

RADATZ; VIANA. **GitHUB BDGD Tools**. 2023. https://github.com/eniocc/bdgd-tools/. Acessado em 14 de novembro de 2023.

RADATZ; VIANA. **GitHUB BDGD Tools**. 2023. https://py-dss-interface.readthedocs.io/en/latest/. Acessado em 1 de dezembro de 2023.

SINAPSIS. **SIG Perdas - software para cálculo de perdas técnicas**. 2023. Disponível em: https://sinapsisenergia.com.br/produto/sigperdas-software-para-calculo-de-perdas-tecnicas/. Acesso em: 14 nov. 2023.