FERRAMENTA COMPUTACIONAL PARA CRIAÇÃO DE MODELOS DE ALIMENTADORES REAIS DE DISTRIBUIÇÃO NO OPENDSS A PARTIR DE DADOS DISPONIBILIZADOS PELA ANEEL

Henrique da Silva, John Jefferson Antunes Saldanha
Instituto Federal de Santa Catarina
Câmpus Jaraguá do Sul – Rau – Curso de Bacharelado em Engenharia Elétrica
e-mail: mfp101999@hotmail.com, john.saldanha@ifsc.edu.br
Trabalho de Conclusão de Curso – 06/07/2022

***Resumo* – Este trabalho propõe uma ferramenta computacional para conversão automática dos dados de alimentadores de distribuição reais disponibilizados pela ANEEL na sintaxe do simulador OpenDSS. A ferramenta é baseada nos softwares QGIS, onde são carregados os dados provenientes da BDGD, e o MATLAB, que converte os dados em arquivos de texto na sintaxe compatível com o OpenDSS. A ferramenta desenvolvida é validada através da modelagem de alimentadores reais da concessionária Celesc. Verificou-se que a ferramenta apresentou resultados consistentes, demonstrando ser útil para pesquisadores, profissionais e empresas do setor. A ferramenta foi disponibilizada de forma eletrônica para ser utilizada, analisada e aprimorada.**

***Palavras-chave* – Alimentadores de distribuição, BDGD, MATLAB, OpenDSS.**

COMPUTATIONAL TOOL FOR CREATING MODELS OF REAL DISTRIBUTION FEEDERS IN OPENDSS FROM DATA PROVIDED BY ANEEL

***Abstract* – This work proposes a computational tool for automatic conversion of data from real distribution feeders provided by ANEEL into the syntax of the OpenDSS simulator. The tool is based on QGIS software, which loads data from BDGD - Geographic Database of the Distributor - and MATLAB, which converts data into text files compatible with OpenDSS syntax. The validation of the tool is presented through the modeling of real feeders of the power distribution company Celesc. It was found that the tool presented consistent results, proving to be a useful tool for researchers, professionals and companies in the sector. The tool was made available electronically to be used, analyzed and improved.**

***Keywords* – BDGD, Distribution feeders, MATLAB, OpenDSS.**

I. INTRODUÇÃO

As redes de distribuição de energia elétrica são fundamentais para o funcionamento de um sistema elétrico de potência. Estas redes atendem os consumidores finais, entregando energia elétrica, e possuem característica predominantemente radiais, onde o fluxo de energia flui das centrais geradoras para as cargas. Entretanto, as redes de distribuição têm passado por modificações e atualizações devido a inserção de novas tecnologias [1].

De acordo com [2] a "*smart grid*" (rede inteligente) é a modernização da rede de distribuição de energia elétrica automatizada, ou seja, a próxima geração do sistema elétrico de potência, com gestão integrada de sua infraestrutura e serviços, planejada para melhorar o sistema tradicional em termos de qualidade de energia, confiabilidade, eficiência, resiliência e aspectos ambientais. A rede inteligente originou-se devido aos novos desafios gerados pela modernização, como [3]:

- Inserção de novas fontes de geração e armazenamento de energia
- Advento de veículos elétricos
- Mudança do perfil de consumo do consumidor
- Bidirecionalidade energética e de informação
- Gestão otimizada do crescimento significativo da carga
- Maior preocupação dos consumidores em relação a qualidade do fornecimento
- Medidas mais restritivas das agências reguladoras de distribuição de energia elétrica

Desta forma, devido à alta complexidade do novo sistema é de suma importância que o mesmo seja planejado e operado de forma adequada. Um software que tem ganhado bastante espaço para modelar e simular o comportamento de redes de distribuição é o The Open Distribution System Simulator (OpenDSS) [4]–[6].

O OpenDSS teve origem no ano de 1997 devido ao fato dos usuários precisarem de respostas mais precisas envolvendo problemas de geração distribuída (GD) [7]. O mesmo é um software de código aberto e foi desenvolvido para simular sistemas elétricos de potência, sendo recomendado em Notas

Técnicas da Agência Nacional de Energia Elétrica (ANEEL) no cálculo de perdas e níveis de tensão [8].

Aliado a isso, a ANEEL, normatiza através do "Módulo 10 - Sistema de Informação Geográfica Regulatório" [9] do PRODIST (Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional), a Base de Dados Geográfica da Distribuidora (BDGD). A BDGD disponibiliza dados das redes de distribuição de todas as distribuidoras de energia elétrica do Brasil, e esta base é atualizada anualmente.

Com isso, a partir dos dados da BDGD, é possível obter modelos elétricos de redes reais para estudos. Entretanto, a modelagem de forma manual de um alimentador real no OpenDSS a partir da BDGD demanda muito tempo, devido a necessidade de conversão dos dados obtidos da BDGD para a linguagem usada no OpenDSS, somado com o número elevado de elementos nas redes, além da possibilidade de erros de digitação. Neste contexto, uma ferramenta automatizada que auxiliaria na criação de modelos para simulação a partir de dados da BDGD é de grande valia, portanto, este trabalho desenvolverá uma ferramenta para interpretar os dados da BDGD e convertê-los para a sintaxe utilizada no OpenDSS. Deste modo, espera-se que a ferramenta contribua para a modernização das redes de distribuição, facilitando a modelagem e integração dos dados no simulador.

Este trabalho esta dividido nas seguintes seções: A Seção II, apresenta a fundamentação teórica do trabalho. A metodologia é apresentada na Seção III, enquanto na Seção IV demonstram-se algumas informações sobre o desenvolvimento da ferramenta. Já na Seção V, apresenta-se a validação da ferramenta e por fim na Seção VI são expostas as conclusões obtidas.

II. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais conceitos utilizados no trabalho. Os seguintes conceitos são abordados: modelos utilizados dos principais elementos do sistema de distribuição, a Base de Dados Geográfica da Distribuidora (BDGD) contendo as principais entidades para modelagem de um alimentador, curvas de carga, software QGIS e OpenDSS.

A. Modelos

Para a modelagem de um sistema de distribuição é necessário ter conhecimento prévio de alguns elementos básicos que o constituem, entre estes estão: linhas de distribuição, transformadores de distribuição e as cargas.

1) Modelo de linha curta: Para modelagem das linhas de distribuição é utilizado o modelo de linha curta, pois o mesmo aplica-se a segmentos com extensão de até 80 km e níveis de tensão de até 69 kV. Neste modelo é desconsiderado o efeito da capacitância [10] e é representado por uma indutância série, conforme a Fig. 1.

A Eq. (1) define o cálculo da impedância para a linha curta:

$$Z = (r + j\omega L)\, l = R + jX \tag{1}$$

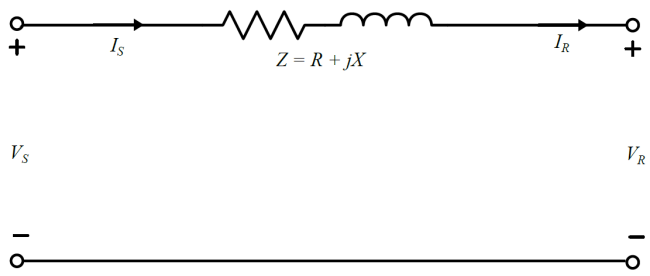

Fig. 1. Modelo de linha curta [10].

Onde: r é resistência por unidade de comprimento ($\Omega/u.c$), L é a indutância por unidade de comprimento ($H/u.c$) e l o comprimento. Os parâmetros V_s e I_s são respectivamente, tensão e corrente no lado emissor da linha, já V_R e I_R, tensão e corrente no lado receptor da linha.

2) Transformador: Os transformadores são os equipamentos responsáveis pela alteração do nível de tensão das linhas de distribuição para níveis de consumo. O modelo de um transformador monofásico está representado na Fig. 2.

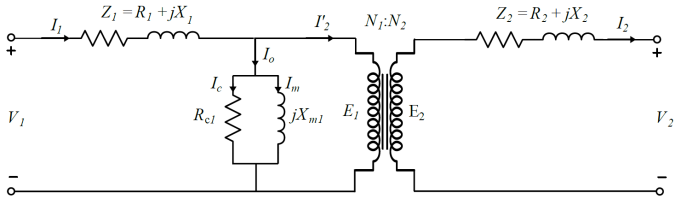

Fig. 2. Modelo de um transformador real monofásico [11].

Onde:

- $N_1 : N_2$: relação de transformação;
- E_1: tensão induzida no enrolamento primário;
- E_2: tensão induzida no enrolamento secundário;
- Z_1: perdas do enrolamento primário;
- Z_2: perdas do enrolamento secundário;
- R_{c1} e X_{m1}: componentes perdas no núcleo;
- V_1 e V_2: tensões terminais;
- I_c: corrente de perdas no ferro;
- I_m: corrente de magnetização;
- I_o: corrente de excitação;
- I_1 e I_2: correntes do primário e secundário;
- I'_2: corrente do secundário refletida no primário.

3) Modelo de carga polinomial ZIP: O ZIP, conforme [12], é o modelo mais utilizado para descrever o comportamento estático de cargas, sendo largamente empregado em estudos de fluxo de potência e de estabilidade de tensão. O mesmo está apresentado nas Eqs. (2) e (3).

$$P = P_o \left[\alpha_{Z_p} \left(\frac{V}{V_o} \right)^2 + \beta_{I_p} \left(\frac{V}{V_o} \right) + \gamma_{P_p} \right] \quad (2)$$

$$Q = Q_o \left[\alpha_{Z_q} \left(\frac{V}{V_o} \right)^2 + \beta_{I_q} \left(\frac{V}{V_o} \right) + \gamma_{P_q} \right] \quad (3)$$

Onde: P é a potência ativa em função das variações de tensão e Q é a potência reativa em função das variações de tensão. Este modelo associa o comportamento da potência que flui para a carga através de 3 parcelas: uma parcela da carga representada pelo modelo de impedância constante (α_*), outra de corrente constante (β_*) e de potência constante (γ_*) [12]. Estes parâmetros (α_*, β_* e γ_*) devem seguir a restrição apresentada na Eq. (4), onde o somatório das parcelas não deve exceder 100%.

$$\alpha_* + \beta_* + \gamma_* = 1 \quad (4)$$

B. Base de Dados Geográfica da Distribuidora (BDGD)

A BDGD compreende o conjunto de informações que são encaminhadas pelas distribuidoras nacionais para a ANEEL. Nesta base de dados constam: o modelo geográfico que simplifica o sistema elétrico real, possuindo o traçado geométrico dos segmentos das redes de baixa, média e alta tensão, a localização geográfica das estruturas de suporte, dos equipamentos e das subestações, informações sobre os dados técnicos do sistema de distribuição e informações comerciais e dados físico-contábeis da base de ativos [9]. O Módulo 10 do PRODIST estabelece o padrão e a estrutura das informações, o formato dos arquivos digitais, os prazos e a forma de envio à ANEEL. A BDGD pode ser solicitada pela Plataforma Integrada de Ouvidoria e Acesso à Informação (Fala.BR).

As entidades básicas necessárias da BDGD para a modelagem de um sistema de distribuição real estão apresentadas nas próximas subseções, para maiores detalhes, estes dados podem ser encontrados no Módulo 10 do PRODIST [9].

1) SUB - subestação: Inclui todas as subestações de interesse, sendo representado geograficamente pela área ocupada da subestação [9]. Os dados básicos principais desta entidade são: o código identificador da subestação e o nome.

2) CTMT - circuito de média tensão: Inclui todos os circuitos de média tensão [9]. E os principais dados básicos são: o código do circuito, nome, barramento conectado, ponto de acoplamento, tensão nominal e energia ativa medida entre 12 meses.

3) SEGCON - segmento condutor: Inclui todos os tipos de segmentos condutores [9]. Os principais dados básicos desta entidade são: o código do segmento, geometria, formação, bitola, isolação e material do cabo, resistência e reatância de sequência positiva, corrente nominal e máxima do condutor.

4) EQTRD - equipamento transformador de distribuição: Inclui todos os equipamentos transformadores de distribuição instalados no sistema [9]. As principais informações básicas desta entidade são: o código identificador, ponto de conexão, potência nominal, classe de tensão, tensões nominais, perdas e impedâncias dos enrolamentos.

5) UNTRD - unidade transformadora de distribuição: Inclui todas as unidades transformadoras de distribuição, representados geograficamente pela sua localização [9]. E os dados básicos principais seguem a mesma linha da entidade EQTRD.

6) SSDMT - segmento do sistema de distribuição de média tensão: Inclui todos segmentos de rede em média tensão, representados geograficamente pelo seu traçado entre dois pontos [9]. Os seus principais dados básicos são: o código, pontos de conexão e comprimento do segmento.

7) SSDBT - segmento do sistema de distribuição de baixa tensão: Inclui todos segmentos de rede em baixa tensão, representados geograficamente pelo seu traçado [9]. E seus dados básicos principais seguem a mesma linha da entidade SSDMT.

8) UCMT - unidade consumidora de média tensão: Inclui todas as unidades consumidoras em média tensão, representado geograficamente pelo local do ponto de conexão [9]. Seus principais dados básicos são: o código da unidade, ponto de conexão, código da unidade transformador, dados de localização, grupo de tensão, fases de conexão, tipologia da curva de carga e medidas apuradas de 12 meses de demanda e energia ativa.

9) UCBT - unidade consumidora de baixa tensão: Inclui todas as unidades consumidoras em baixa tensão [9]. E os principais dados seguem a mesma linha da entidade UCMT.

10) PIP - ponto de iluminação pública: Inclui todos os pontos de iluminação pública sem medição individual [9]. Contendo dados como: código identificador, fases de conexão, grupo tarifário, carga instalada, potência do ponto e energia ativa medida no período de 12 meses.

C. Curvas de carga

A curva de carga representa o perfil de consumo de uma unidade, e é expressa através da demanda de potência em um determinado intervalo de tempo. Nos sistemas de distribuição nacionais, as curvas de carga típicas são divididas por cada classe de consumo, sendo estas: residenciais, comerciais, industriais, serviço público, iluminação pública e cargas em média e alta tensão.

D. QGIS

O QGIS é um Sistema de Informação Geográfica (SIG) de código aberto licenciado segundo a Licença Pública Geral GNU,

é um software multiplataforma que permite a visualização, edição e análise de dados georreferenciados [13]. Na Fig. 3, pode ser vista a interface do programa juntamente com informações carregadas da BDGD referente a Celesc.

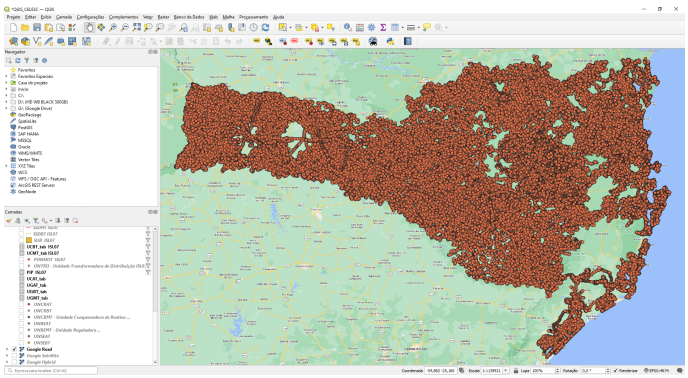

Fig. 3. Informações da Celesc no software QGIS.

E. Software de simulação de redes OpenDSS

O OpenDSS é um software de código aberto desenvolvido no ano de 1997 pela empresa Electrotek Concepts® e comprado no ano de 2007 pelo Electric Power Research Institute (EPRI®). É um programa de simulação no domínio da frequência de uso geral que possui recursos especiais para criar modelos de sistemas de distribuição de energia elétrica e realizar análises relacionadas ao planejamento de distribuição e qualidade de energia. (AJUSTAR CITAÇÃO) O software funciona através de linhas de comandos baseadas em texto e possui uma interface *Component Object Model* (COM) que permite a interoperabilidade com vários outros programas externos, como: MATLAB, Excel, entre outros [7].

A seguir apresentam-se como os elementos de um alimentador são modelados no OpenDSS.

1) Barramento de referência: O ponto de partida da modelagem de um alimentador no OpenDSS é através de uma barra de referência, representado por um equivalente de Thevénin. Este elemento é modelado conforme exemplo a seguir:

New Circuit.JSL07 bus1 = 3017203 basekv = 13.8 pu = 1.00 phases = 3 frequency = 60 mvasc3 = 21000 mvasc1 = 20000

Onde:
- New Circuit: criação da barra de referência;
- bus1: barra que está conectado;
- basekv: tensão de base;
- pu: tensão em pu;
- phases: quantidade de fases do alimentador;
- frequency: frequência do alimentador;
- mvasc3: potência de curto-circuito trifásica em MVA;
- mvasc1: potência de curto-circuito monofásica em MVA.

Uma alternativa para o uso das potências de curto-circuito é a declaração das impedâncias de sequência positiva e zero (R_1, X_1, R_0, X_0). Estes dados são encontrados na entidade "CTMT" da BDGD.

2) Linhas de distribuição: As linhas de distribuição são modeladas através do modelo de linha curta. Estes dados são encontrados nas entidades "SSDMT" e "SSDBT" da BDGD, e sua modelagem no OpenDSS é feita conforme exemplo a seguir:

New line.4056854 phases = 3 bus1 = 2988689 bus2 = 942346 length = 26.8548 units = m linecode = CMT88-3-5

Onde:
- New line: criação de uma linha;
- phases: número de fases deste segmento de linha;
- bus1: barra que o terminal 1 está conectado;
- bus2: barra que o terminal 2 está conectado;
- length: comprimento do segmento de linha;
- units: unidade de medida do comprimento;
- linecode: referência do elemento que contém as características do cabo.

Cada linha de distribuição é referenciada a um elemento que contém as características do cabo. Este elemento no OpenDSS é chamado de "*linecode*" e os dados são encontrados na entidade "SEGCON" da BDGD, e sua modelagem é feita conforme exemplo a seguir:

New linecode.CMT20-3-3 nphases = 3 basefreq = 60 units = km Normamps = 140 R1 = 1.598 !ohm/km X1 = 0.522 !ohm/km C1 = 0.00

Onde:
- New linecode: criação de um novo elemento que contém as características do cabo;
- nphases: número de fases;
- basefreq: frequência de operação;
- units: unidade de medida de comprimento;
- Normamps: corrente nominal do cabo;
- R_1: resistência de sequência positiva por unidade de comprimento;
- X_1: reatância de sequência positiva por unidade de comprimento;
- C_1: capacitância total de sequência positiva por unidade de comprimento.

3) Transformadores: Os dados dos transformadores são encontrados nas entidades "UNTRD" e "EQTRD" da BDGD e no OpenDSS um transformador é modelado conforme exemplo a seguir:

New transformer.7295705 xhl = 4.38 windings = 2 %loadloss = 1.0202 %noloadloss = 0.2598 wdg = 1 bus = 942358 kv = 13.8 kva = 500 conn = delta %r = 1.02 tap = 0.9565 wdg = 2 bus = BT_942358.1.2.3.0 kv = 0.38 kva = 500 conn = wye %r = 1.02 tap = 1

Onde:
- New transformer: criação do transformador;
- xhl: reatância percentual do primário para o secundário;
- windings: quantidade de enrolamentos;
- %loadloss: perdas em carga (% em relação a potência nominal);
- %noloadloss: perdas a vazio (% em relação a potência nominal);
- wdg: referência do enrolamento, sendo 1: primário, 2: secundário;
- bus: barra a qual o enrolamento está conectado;
- kv: tensão nominal do enrolamento em kV;
- kva: potência nominal do enrolamento em kVA;
- conn: tipo de conexão do enrolamento (delta ou estrela (wye));
- %r: resistência percentual do enrolamento;
- tap: tensão em pu que o tap está ajustado.

4) Cargas: Na BDGD as informações das cargas são encontradas nas entidades "UCBT", "UCMT" e "PIP", sendo cargas em baixa e média tensão e pontos de iluminação pública. No OpenDSS estas são modeladas conforme exemplo a seguir:

New load.475 phases = 3 model = 8 ZIPV = [0.5 0 0.5 0 0 1 0.9] daily = A4 bus = 942381.1.2.3 kv = 13.2 pf = 0.92 kw = 5.85841 conn = delta

Onde:
- New load: criação da carga;
- phases: número de fases;
- model: modelo utilizado para caracterização da carga;
- ZIPV: expoentes de potência ativa e reativa do modelo ZIP;
- daily: curva de carga associada a carga;
- bus: barra a qual a carga está conectada;
- kv: tensão nominal da carga;
- pf: fator de potência;
- kw: potência ativa nominal;
- conn: tipo de conexão da carga (delta ou estrela (wye)).

Já as curvas de carga no OpenDSS são modeladas conforme:

New loadshape.Residencial npts = 24 interval = 1 mult = (0.875718121, 0.7379304, ..., 1.081178899)

Onde:
- New loadshape: criação da curva;
- npts: número de pontos;
- interval: intervalo de tempo entre cada ponto;
- mult: valores que multiplicam a potência da carga.

III. METODOLOGIA

A proposta deste trabalho é desenvolvida a partir dos passos descritos na Fig. 4. A seguir apresentam-se detalhes de cada etapa.

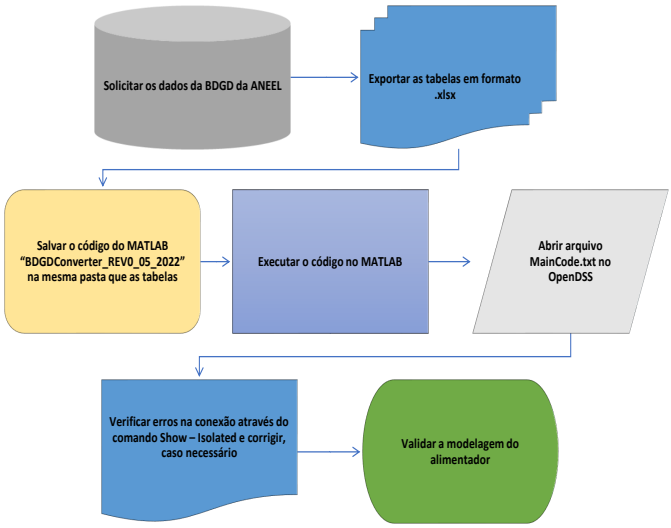

Fig. 4. Etapas da ferramenta computacional desenvolvida.

A. Curvas de carga

As curvas de carga típicas utilizadas neste trabalho foram obtidas por meio da campanha de medições da ANEEL, referentes à 4ª Revisão Tarifária Periódica da Celesc, realizada em 2016 [14]. As curvas representam a média entre todos os tipos de consumidores dentro de uma das classes de consumo e são segmentadas em 3 períodos, dias úteis, sábados e domingos. Para cada curva utilizou-se uma potência de base, obtida pela média dos valores registrados para cada classe. As mesmas podem ser vistas nas Fig. 5, 6 e 7.

B. Solicitação da BDGD através do Fala.BR

A solicitação da base de dados das distribuidoras é feita através da Plataforma Integrada de Ouvidoria e Acesso à Informação (Fala.BR) [15]. Após a conexão no sistema, deve-se cadastrar um pedido para a ANEEL através do menu "Acesso à Informação" e requisitar a Base de Dados Geográfica da Distribuidora (BDGD), informando o motivo por trás da solicitação. Após o acesso ser concedido, um e-mail será enviado informando como os dados devem ser obtidos. Em adição, um tutorial também apresenta como os dados podem ser importados no QGIS.

C. Exportar as tabelas em formato .xlsx

Com os dados da concessionária desejada carregados nas camadas do QGIS, primeiro é recomendado clicar com o botão

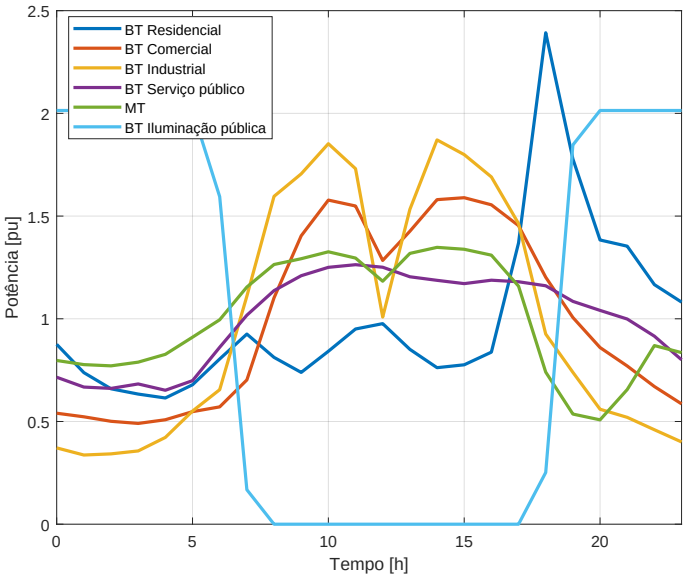

Fig. 5. Curva de carga em dias úteis.

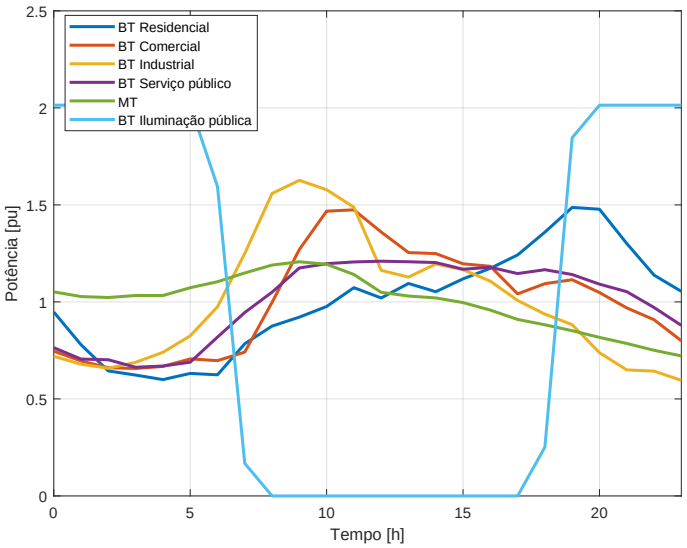

Fig. 6. Curva de carga nos sábados.

direito em cada entidade e filtrá-las conforme a Fig. 8 . Esta etapa não é obrigatória, pois o código irá filtrar todas tabelas com o alimentador desejado, porém, se for escolhido exportar as tabelas sem filtrá-las, o tempo de processamento da ferramenta poderá ser grande.

O “COD_ID” ou o “NOM” da subestação podem ser obtidos clicando com o direito na entidade “SUB” e selecionando “Abrir tabela de atributos”, nesta tabela é encontrada a subestação desejada. De posse do código da subestação verifica-se na tabela de atributos da entidade “CTMT” o “COD_ID” do alimentador desejado, e o mesmo é utilizado para filtrar as demais entidades necessárias para modelar a rede. Com as entidades citadas filtradas, o próximo passo é exportar cada uma delas clicando com o direito e selecionando “Exportar - Guardar elementos

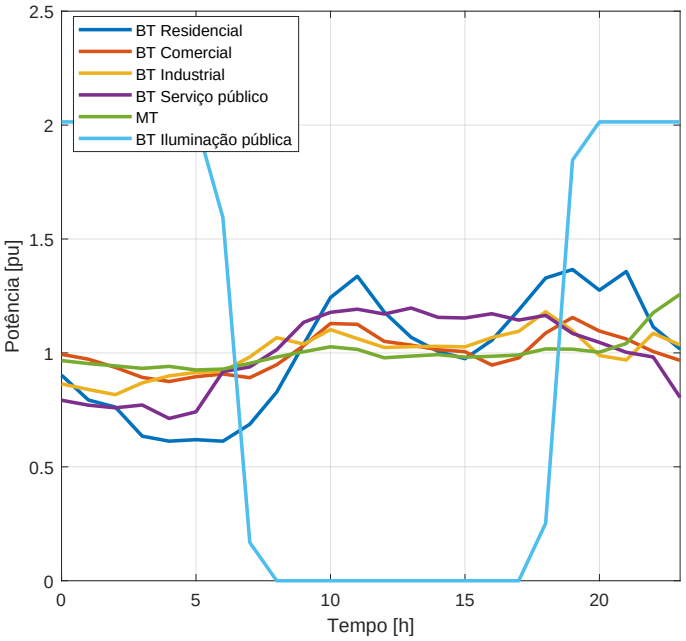

Fig. 7. Curva de carga nos domingos.

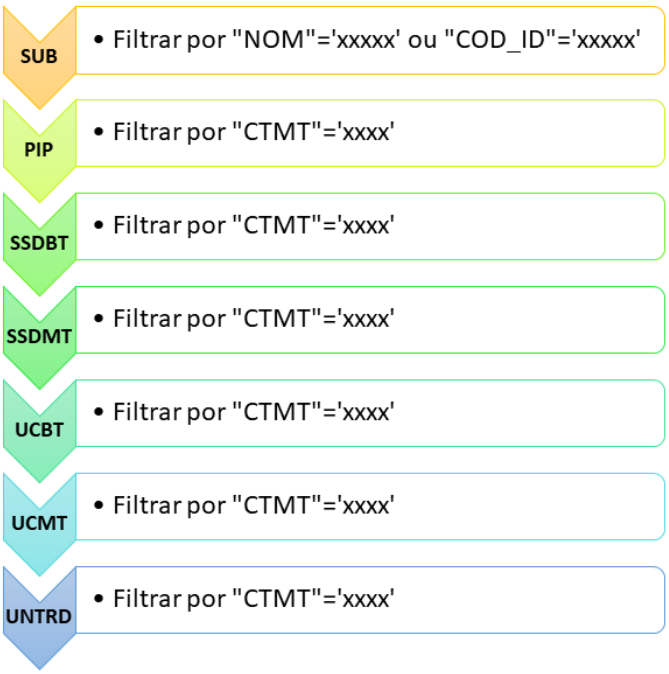

Fig. 8. Filtragem das entidades.

como”. Na janela aberta deve-se escolher o formato “Shapefile” e salvar todas no mesmo diretório. O resultado deverá ser o mesmo conforme a lista a seguir, sendo que os arquivos adicionais criados com extensão diferente de .dbf podem ser excluídos da pasta.

- CTMT.dbf
- EQTRD.dbf

- PIP.dbf
- SEGCON.dbf
- SSDBT.dbf
- SSDMT.dbf
- UCBT.dbf
- UCMT.dbf
- UNTRD.dbf

O último passo necessário desta etapa é de converter os arquivos do formato .dbf para .xlsx. Para isso, abra cada arquivo em um editor de planilhas e salve no formato .xlsx. O resultado final desta etapa, com a adição da ferramenta (arquivo "BDGDConverter_REV0_05_2022.m"), deverá apresentar os seguintes arquivos na mesma pasta:

- BDGDConverter_REV0_05_2022.m
- CTMT.xlsx
- EQTRD.xlsx
- PIP.xlsx
- SEGCON.xlsx
- SSDBT.xlsx
- SSDMT.xlsx
- UCBT.xlsx
- UCMT.xlsx
- UNTRD.xlsx

D. Execução do código e validação do alimentador

Após a conversão e adição do código realizado, deve-se executar esse no software MATLAB. Essa execução deve ser realizada onde encontram-se as nove planilhas necessárias para a modelagem do alimentador. Ao executar o código o usuário deverá inserir alguns dados de entrada, sendo esses: o "COD_ID" do alimentador, que pode ser encontrado na entidade "CTMT", a potência de curto-circuito trifásica e monofásica em [MVA] e se a análise será feita para dias úteis, sábados ou domingos. O resultado da execução será a criação de dez arquivos de texto .txt, conforme a lista a seguir. O código do alimentador é inserido no final de cada nome do arquivo, para organização.

- LineCode_SSDBT_JSL07.txt
- LineCode_SSDMT_JSL07.txt
- Lines_SSDBT_JSL07.txt
- Lines_SSDMT_JSL07.txt
- Loads_UCBT_JSL07.txt
- Loads_UCMT_JSL07.txt
- LoadShapes.txt
- MainCode.txt
- PIP_JSL07.txt
- Transformers_JSL07.txt

No OpenDSS deve-se abrir o arquivo denominado "MainCode" e executa-lo, conforme a Fig. 9. Nele constam todos parâmetros iniciais necessários para rodar as simulações desejadas. Uma etapa de verificação de erros de conexão é necessária, pois podem existir inconsistências no banco de dados, visto que, devido a grande quantidade de informações e ramificações de uma rede de distribuição o preenchimento manual por parte da concessionária normalmente resulta em alguns erros de digitação. Para a verificação deve-se selecionar o comando "Show - Isolated", caso haja alguma inconsistência no relatório basta corrigir excluindo o trecho incorreto dos arquivos de texto.

```
clear

New Circuit.JSL07 bus1 = 3017203 basekv = 13.8 pu = 1.00 phases = 3 frequency = 60 mvasc3 = 21000 mvasc1 = 20000

Redirect LineCode_SSDBT_JSL07.txt
Redirect LineCode_SSDMT_JSL07.txt
Redirect Lines_SSDMT_JSL07.txt
Redirect Lines_SSDBT_JSL07.txt
Redirect LoadShapes.txt
Redirect Loads_UCBT_JSL07.txt
Redirect Loads_UCMT_JSL07.txt
Redirect Transformers_JSL07.txt
Redirect PIP_JSL07.txt

Set VoltageBases = [13.8, 0.38]
CalcVoltageBases

New energymeter.medidor element = line.4947729 terminal = 1

set tolerance = 0.0000001
set Maxiter=500
set mode = daily
set stepsize = 1h
set number = 24
solve
```

Fig. 9. MainCode no OpenDSS.

Após a correção de eventuais erros, o último passo necessário é a validação do consumo anual do alimentador, utilizando a Eq. (5).

$$\left| 100 \cdot \left(\frac{C_{sim,ano}}{C_{BDGD,ano}} - 1 \right) \right| \leq 10\% \tag{5}$$

Onde:

$C_{sim,ano}$ = Consumo anual simulado [kWh]
$C_{BDGD,ano}$ = Consumo anual real [kWh]

Através do comando "Show - Energy Meters", é obtido o valor de $C_{sim,dia}$, encontrado no relatório gerado como "Reg 1" em [kWh], basta multiplica-lo por 365 dias e a variável $C_{sim,ano}$ é obtida. Já a variável $C_{BDGD,ano}$ é encontrada na entidade "CTMT" da BDGD, basta encontrar a linha referente o alimentador simulado e somar os valores das células contidas entre as colunas "ENE_01" até "ENE_12". Para este trabalho, adotou-se que diferenças de até 10% são consideradas aceitáveis para a validação da modelagem [16].

IV. DESENVOLVIMENTO DA FERRAMENTA

A ferramenta foi disponibilizada de forma eletrônica em: "https://abre.ai/ferramentabdgd2opendss" para ser utilizada, analisada e aprimorada por demais pesquisadores e profissionais do setor. O código também pode ser visto no Apêndice deste

documento.

O desenvolvimento da ferramenta ocorreu a partir dos passos informados na sequência. Primeiramente, o código parte da leitura das planilhas exportadas e criação da tabela "TTEN - Tipo de Tensão", encontrada no DDA (Dicionários de Dados) ANEEL no Módulo 10 do PRODIST, conforme trecho entre as linhas 1 a 20 do código exposto no Apêndice.

Já entre as linhas 23 a 38, são solicitados ao usuário alguns dados de entrada necessários para simulação, como: código do alimentador, potência de curto-circuito monofásica e trifásica, e se a análise será feita para os dias úteis, sábados ou domingos.

Entre as linhas 40 a 55, são filtradas algumas tabelas com o alimentador inserido e é feito a coleta de informações necessárias para a modelagem do barramento principal.

Já entre as linhas 57 a 82, é feito a coleta de informações necessárias para a modelagem das caraterísticas elétricas dos condutores de baixa e média tensão, para a modelagem das linhas de distribuição e também os transformadores de distribuição.

Entre as linhas 85 a 144, são coletados dados para modelagem das cargas tanto em baixa quanto em média tensão do sistema e também escrita do arquivo de texto referente as curvas de carga.

Nas linhas 149 até 174 é gerado o arquivo principal nomeado MainCode.txt. E por fim, entre as linhas 176 até 615, são gerados os arquivos de texto referente a modelagem das linhas de distribuição e suas características elétricas, dos transformadores de distribuição e também das cargas.

V. RESULTADOS

Inicialmente, a ferramenta foi testada atráves da simulação de dois alimentadores pertencentes a concessionária Celesc do estado de Santa Catarina. O primeiro sendo o JSL07, sendo escolhido devido a dois motivos, primeiro por ser um alimentador de pequeno porte, ou seja, com relativamente poucas unidades consumidoras e segundo pois foi utilizado em um artigo acadêmico feito no próprio Câmpus Jaraguá do Sul – Rau [16]. Já a escolha do segundo alimentador, o ISL07, foi feita devido a ele ser de um porte maior que o anterior, possuindo aproximadamente 7 vezes mais unidades consumidoras.

A. Subestação JSL - Alimentador JSL07

Na Fig. 10 encontra-se o traçado do alimentador JSL07 proveniente da subestação JSL, da cidade de Jaraguá do Sul - Santa Catarina, as características deste alimentador podem ser vistas na Tabela I.

O consumo anual de energia real registrado deste alimentador foi de aproximadamente $C_{BDGD,JSL07,ano} = 11,282\ GWh$.

Já o consumo obtido através da simulação deste alimentador no OpenDSS utilizando a ferramenta desenvolvida está disposto na Eq. (6).

$$\begin{aligned} C_{sim,JSL07,ano} &= C_{sim,dia} \cdot 365 \\ C_{sim,JSL07,ano} &= 31.120 \cdot 365 = 11,359\ GWh \end{aligned} \tag{6}$$

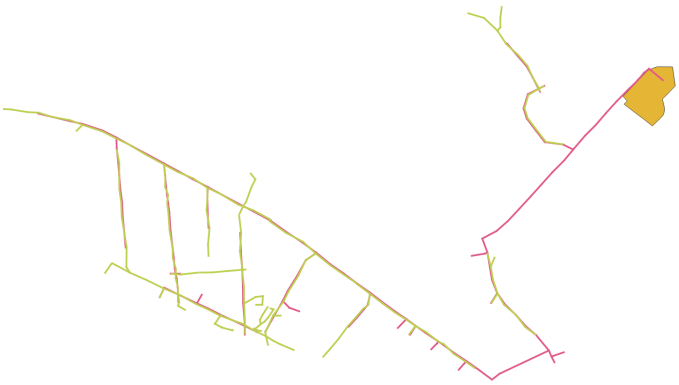

Fig. 10. Representação do alimentador JSL07.

TABELA I
Características do alimentador JSL07.

Descrição	Qnt.
Segmentos de baixa tensão	171
Segmentos de média tensão	148
Unidades transformadoras de distribuição	24
Unidades consumidoras de média tensão	13
Unidades consumidoras de baixa tensão	604
Pontos de iluminação pública (PIP)	192

De posse desses valores, os mesmos são inseridos na Eq. (5) e então é calculado a diferença percentual entre eles, conforme Eq. (7).

$$\begin{aligned} \left| 100 \cdot \left(\frac{C_{sim,JSL07,ano}}{C_{BDGD,JSL07,ano}} - 1 \right) \right| &\leq 10\% \\ \left| 100 \cdot \left(\frac{11,359}{11,282} - 1 \right) \right| &\leq 10\% \\ 0,68\% &\leq 10\% \end{aligned} \tag{7}$$

Como a diferença percentual entre o consumo simulado e o consumo real registrado foi de 0,68%, ou seja, menor que 10%, o sistema gerado através da ferramenta desenvolvida apresenta uma elevada proximidade com o sistema real.

Um resultado importante a ser mencionado é referente ao tempo de execução da ferramenta, os resultados foram coletados em um PC com 32,0 Gb de RAM instalada e um processador com 3,70 GHz, utilizando a versão 2021a do MATLAB. O tempo de execução da ferramenta para o alimentador JSL07 foi de aproximadamente 30,0 segundos.

B. Subestação ISL - Alimentador ISL07

Na Fig. 11 encontra-se o traçado do alimentador ISL07 proveniente da subestação Ilha Sul - ISL, da cidade de Florianópolis - Santa Catarina, as características deste alimentador podem ser vistas na Tabela II. Em comparação com

o alimentador anterior, este apresenta uma rede de dimensão muito maior, servindo como um bom exemplo de teste para o código.

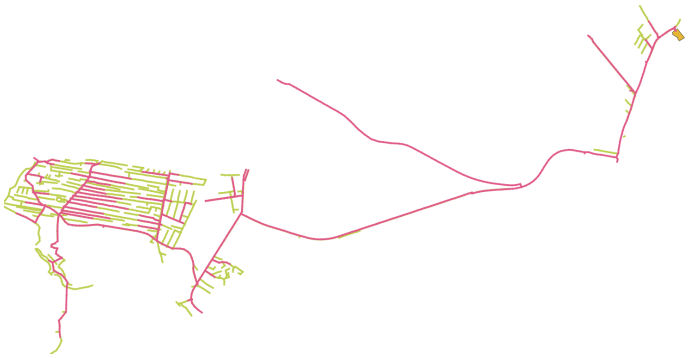

Fig. 11. Representação do alimentador ISL07.

TABELA II
Características do alimentador ISL07.

Descrição	Qnt.
Segmentos de baixa tensão	1431
Segmentos de média tensão	761
Unidades transformadoras de distribuição	86
Unidades consumidoras de média tensão	6
Unidades consumidoras de baixa tensão	4240
Pontos de iluminação pública (PIP)	1300

O consumo anual de energia real registrado deste alimentador foi de aproximadamente $C_{BDGD,ISL07,ano} = 23,112\ GWh$.

Já o consumo obtido através da simulação deste alimentador no OpenDSS utilizando a ferramenta desenvolvida está disposto na Eq. (8).

$$\begin{aligned} C_{sim,ISL07,ano} &= C_{sim,dia} \cdot 365 \\ C_{sim,ISL07,ano} &= 57.750 \cdot 365 = 21,079\ GWh \end{aligned} \tag{8}$$

De posse desses valores, os mesmos são inseridos na Eq. (5) e então é calculado a diferença percentual entre eles, conforme Eq. (9).

$$\begin{aligned} \left| 100 \cdot \left(\frac{C_{sim,ISL07,ano}}{C_{BDGD,ISL07,ano}} - 1 \right) \right| &\leq 10\% \\ \left| 100 \cdot \left(\frac{21,079}{23,112} - 1 \right) \right| &\leq 10\% \\ 8,80\% &\leq 10\% \end{aligned} \tag{9}$$

Como a diferença percentual entre o consumo simulado e o consumo real registrado foi de 8,80%, ou seja, menor que 10%, o sistema gerado através da ferramenta desenvolvida apresenta uma boa proximidade com o sistema real. Em comparação com o resultado obtido do alimentador anterior (JSL07), houve um aumento significativo na diferença percentual do consumo, isto se deve ao fato de que o alimentador em questão (ISL07), apresentou inconsistências em alguns segmentos de rede, em um transformador de distribuição e nas cargas (tais inconsistências encontradas através do comando "Show - Isolated"), então foi necessário excluir estes itens dos seus respectivos arquivos de texto, ou seja, eles são desconsiderados na simulação, perdendo um pouco de precisão no modelo gerado. Estes erros vem dos próprios dados disponibilizados pelas distribuidoras para a ANEEL. Caso o usuário se depare com algumas inconsistências no alimentador modelado e deseje não omitir informações para uma maior precisão, recomenda-se corrigir os segmentos com falha, porém este processo pode ser bem trabalhoso em alguns casos. O tempo de execução da ferramenta para o alimentador ISL07 foi de aproximadamente 36,0 segundos.

Após a validação com os alimentadores da Celesc, foram realizados três testes: alimentador IPR0104 da concessionária Enel-SP, alimentador 328881756 da Light-RJ e alimentador 823620084 da COPEL-PR. Ao realizar os testes, verificou-se que os mesmos apresentaram instabilidades. Isto se deve ao fato de que por mais que a BDGD seja uma base de dados padronizada, ainda existe algumas particularidades de cada concessionária. Ou seja, a ferramenta precisa ser analisada para alguns alimentadores de cada distribuidora de energia elétrica nacional.

VI. CONCLUSÃO

Este trabalho apresentou uma ferramenta computacional para converter os dados de alimentadores de distribuição reais disponibilizados pela ANEEL na sintaxe do OpenDSS. A ferramenta possibilita a conversão rápida e automática dos dados contidos na BDGD para a sintaxe do simulador OpenDSS, evitando erros por parte do usuário e propiciando um elevado ganho de tempo em modelagens de alimentadores reais.

A ferramenta apresentou resultados consistentes na concessionária Celesc, através do estudo de caso de 2 alimentadores, apresentando uma paridade com o sistema real de 0,68% para o primeiro alimentador e 8,80% para o segundo, demonstrando ser eficaz para converter o sistema na sintaxe do OpenDSS. A ferramenta apresentou inconsistências em testes em outras concessionárias.

Para trabalhos futuros recomenda-se: a validação da ferramenta nas outras distribuidoras nacionais, a modelagem de outros equipamentos presentes nas redes de distribuição de energia elétrica, a criação de uma interface GUI "Graphical User Interface" (Interface Gráfica do Usuário, em português) para uma melhor experiência com o usuário, e redigir o código em uma linguagem de programação de código aberto como Python ou GNUOctave, para proporcionar o uso da ferramenta para qualquer usuário sem nenhum custo.

REFERÊNCIAS

[1] R. Bayindir, I. Colak, G. Fulli, K. Demirtas, "Smart grid technologies and applications", *Renewable and Sustainable Energy Reviews*, vol. 66, pp. 499–516, 2016, doi:https://doi.org/10.1016/j.rser.2016.08.002, URL: https://www.sciencedirect.com/science/article/pii/S1364032116304191.

[2] A. A. Shobole, M. Wadi, "Multiagent systems application for the smart grid protection", *Renewable and Sustainable Energy Reviews*, vol. 149, p. 111352, 2021.

[3] F. Toledo, *Desvendando as Redes Elétricas Inteligentes-Smart Grid Handbook*, Brasport, 2012.

[4] S. Monger, R. Vega, H. Krishnaswami, *et al.*, "Simulation of smart functionalities of photovoltaic inverters by interfacing OpenDSS and Matlab", *in 2015 IEEE 16th Workshop on Control and Modeling for Power Electronics (COMPEL)*, pp. 1–6, IEEE, 2015.

[5] A. Hariri, A. Newaz, M. O. Faruque, "Open-source python-OpenDSS interface for hybrid simulation of PV impact studies", *IET Generation, Transmission & Distribution*, vol. 11, no. 12, pp. 3125–3133, 2017.

[6] P. Quesada, A. Arguello, J. Quirós-Tortós, G. Valverde, "Distribution network model builder for OpenDSS in open source GIS software", *in 2016 IEEE PES Transmission & Distribution Conference and Exposition-Latin America (PES T&D-LA)*, pp. 1–6, IEEE, 2016.

[7] EPRI, "OpenDSS Manual", URL: https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Distrib/Doc/OpenDSSManual.pdf.

[8] ANEEL, "Nota Técnica nº 0057/2014 - SRD/ANEEL", URL: http://www2.aneel.gov.br/aplicacoes/audiencia/arquivo/2014/026/documento/nota_tecnica_0057_srd.pdf.

[9] PRODIST, "Módulo 10 – Sistema de Informação Geográfica Regulatório", *Agência Nacional de Energia Elétrica – ANEEL*, 2021.

[10] H. Saadat, *Power system analysis*, McGraw-hill, 1999.

[11] S. J. Chapman, *Fundamentos de máquinas elétricas*, AMGH editora, 2013.

[12] I. F. Visconti, "Modelos de cargas baseados em medições para simulações dinâmicas em sistemas elétricos de potência", *Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro-RJ*, 2010.

[13] OSGeo, "QGIS", URL: https://qgis.org/pt_BR/site/about/index.html.

[14] ANEEL, "Calendário e Resultado dos Processos Tarifários de Distribuição", URL: https://antigo.aneel.gov.br/resultado-dos-processos-tarifarios-de-distribuicao.

[15] Fala.BR, "Acesso a Informação", URL: https://falabr.cgu.gov.br/publico/Manifestacao/SelecionarTipoManifestacao.aspx?ReturnUrl=%2f.

[16] M. S. Jansen, *Análise do impacto da geração distribuída fotovoltaica em uma rede elétrica da cidade de Jaraguá do Sul*, Instituto Federal de Santa Catarina, 2022.

APÊNDICE - CÓDIGO DESENVOLVIDO NO MATLAB

```
clc
clear all

%Leitura das tabelas
%-----------------------------------------//
COD_ID = [0;1;2;3;4;5;...]; %Para dados completos da tabela TTEN verificar no próprio código
TEM = [0;110;115;120;121;125;...];
DESCR={'Não informado';'110 V';'115 V';'120 V';'121 V';'125 V';'...'};
TTEN = table(COD_ID,TEM,DESCR);
clear COD_ID TEM DESCR

CTMT = readtable('CTMT.xlsx');
SSDMT = readtable('SSDMT.xlsx');
SSDBT = readtable('SSDBT.xlsx');
SEGCON = readtable('SEGCON.xlsx');
EQTRD = readtable('EQTRD.xlsx');
UNTRD = readtable('UNTRD.xlsx');
UCBT = readtable('UCBT.xlsx');
UCMT = readtable('UCMT.xlsx');
PIP = readtable('PIP.xlsx');
%-----------------------------------------//

% //Dados de entrada
%-----------------------------------------//
prompt= 'Alimentador (BDGD) COD_ID da camada (CTMT): ';
Ali=input(prompt,'s');

freq=60;

prompt= 'Potência de curto-circuito trifásica (MVA): ';
mvasc3=input(prompt,'s');

prompt= 'Potência de curto-circuito monofásica (MVA): ';
mvasc1=input(prompt,'s');

prompt= 'Análise-> Dias úteis(UT), Sábados(SAB) ou Domingos(DOM): ';
Dia_analise=input(prompt,'s');
%-----------------------------------------//

% //Equivalente Thevenin/Circuit
% //-----------------------------------------//
SUB = CTMT(strcmp(CTMT.COD_ID, Ali), :);
DESCR = TTEN(TTEN.COD_ID == str2double(SUB{1,7}), 3);
DESCR{1,1} = regexprep(DESCR{1, 1}, 'V$', '');
DESCR{1,1} = regexprep(DESCR{1, 1}, 'k$', '');
DESCR{1,1} = regexprep(DESCR{1, 1}, ' ', '');
DESCR=str2double(DESCR{1,1});

SSDBT = SSDBT(strcmp(SSDBT.CTMT, Ali), :);
SSDMT = SSDMT(strcmp(SSDMT.CTMT, Ali), :);

SSDMT_CONTAINS=contains(SSDMT.PN_CON_1,SSDMT.PN_CON_2);
SSDMT_BUS = SSDMT(SSDMT_CONTAINS(:,1) == 0,:);
bus1 = string(SSDMT_BUS{1,3});
% //-----------------------------------------//

% //LineCodes
% //-----------------------------------------//
SSDMT_13 = unique(SSDMT(:,13),'stable');
SEGCON_MT=SEGCON(ismember(SEGCON.COD_ID, SSDMT_13.TIP_CND), :);
rows_MT=size(SEGCON_MT,1);
i=1;

SSDBT_14 = unique(SSDBT(:,14),'stable');
SEGCON_BT=SEGCON(ismember(SEGCON.COD_ID, SSDBT_14.TIP_CND), :);
```

```
rows_BT=size(SEGCON_BT,1);
% //----------------------------------------//

% //Lines
% //----------------------------------------//
rows_SSDMT=size(SSDMT,1);
rows_SSDBT=size(SSDBT,1);
Code_first_line = string(SSDMT_BUS{1,2});
% //----------------------------------------//
% //Trafos
% //----------------------------------------//
UNTRD = UNTRD(strcmp(UNTRD.CTMT, Ali), :);
rows_UNTRD=size(UNTRD,1);
SSDBT_FIL=unique(SSDBT(:,5),'stable');
PN_2_UNTRD=table;
base_kv_ten_lin=UNTRD{1,14};
% //----------------------------------------//

% //----------------------------------------//
% //Loads/LoadsShapes
% //----------------------------------------//
UCBT = UCBT(strcmp(UCBT.CTMT, Ali), :);
UCMT = UCMT(strcmp(UCMT.CTMT, Ali), :);
PIP = PIP(strcmp(PIP.CTMT, Ali), :);
rows_UCBT=size(UCBT,1);
rows_UCMT=size(UCMT,1);
rows_PIP=size(PIP,1);
kwh=table;

R_UTEIS = "0.875718121, 0.7379304, 0.659207807, 0.633476025, 0.61442096, 0.678764605, 0.804322151,
    0.925609108, 0.812108228, 0.738965351, 0.841846193, 0.950237106, 0.976420214, 0.851260093, 0.761985332,
     0.77603361, 0.837538914, 1.369080181, 2.392437728, 1.778169959, 1.383523896, 1.353454872, 1.166310245,
     1.081178899";
C_UTEIS = "0.540290381, 0.522840237, 0.500970213, 0.490532996, 0.508517788, 0.547920027, 0.571156332,
    0.702277961, 1.102078597, 1.403910599, 1.577798928, 1.549089466, 1.283974133, 1.426076057, 1.579967453,
     1.589447048, 1.55488613, 1.453487725, 1.202275416, 1.006271074, 0.859995139, 0.770251199, 0.670245313,
     0.585739789";
IN_UTEIS = "0.371480548, 0.336803009, 0.342382469, 0.356678593, 0.422471297, 0.550753421, 0.654840865,
    1.110008527, 1.596005663, 1.705449919, 1.852359559, 1.730589546, 1.008336908, 1.532508343, 1.870899751,
     1.799217941, 1.690571957, 1.463406208, 0.925729658 0.739420845, 0.559664929, 0.520173023, 0.45983599,
    0.40041103";
SP_UTEIS = "0.715469742, 0.667620058, 0.66107404, 0.682765098, 0.652029173, 0.698830552, 0.861324277,
    1.01744213, 1.137088058, 1.209694169, 1.250450996, 1.263345256, 1.250983732, 1.204525696, 1.187173553,
    1.171167595, 1.187824991, 1.180237526, 1.160738967, 1.084776851, 1.040818453, 0.998823054, 0.915379224,
     0.80041681";
A4_UTEIS = "0.797062444, 0.776931084, 0.770858997, 0.788911111, 0.826906519, 0.91039443, 0.995234812,
    1.154512925, 1.264221535, 1.292436805, 1.326093804, 1.295605964, 1.182864062, 1.318314054, 1.347710884,
     1.338456486, 1.309773825, 1.158663378, 0.740455996, 0.53623314, 0.508044577, 0.65548448, 0.869557771,
    0.835270918";
IP_UTEIS = "2.013976041, 2.013976041, 2.013976041, 2.013976041, 2.013976041, 2.013976041, 1.594394572,
    0.167832588, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.251748882, 1.846143453, 2.013976041, 2.013976041,
    2.013976041, 2.013976041";

R_SAB = "0.9479247 0.778990851 0.644163368 0.622945212 0.599629318 0.631423811 0.624300097 0.784340665
    0.875381538 0.921509082 0.976392549 1.073391253 1.020256255 1.095033215 1.052651324 1.11871709
    1.173223331 1.241993938 1.359572292 1.487065095 1.47764092 1.301080174 1.138071683 1.054302238";
C_SAB = "0.745844961 0.695560012 0.660555887 0.656953401 0.6675735 0.706258203 0.697273586 0.741337801
    0.997543038 1.270040175 1.46779961 1.474424886 1.360264397 1.25427073 1.249090243 1.196283151
    1.18359598 1.041567259 1.094021382 1.114351578 1.048219812 0.969091752 0.908246631 0.799832026";
IN_SAB = "0.719701223 0.678856369 0.65797677 0.688729903 0.740285843 0.824971236 0.974840711 1.249931292
    1.558711053 1.626715041 1.578187507 1.487839002 1.16260595 1.127922538 1.195249562 1.166267331
    1.106884403 1.007627862 0.937038061 0.882564052 0.738361397 0.649525851 0.643271554 0.59593549";
SP_SAB = "0.764124928 0.705296903 0.701858516 0.663284763 0.66956166 0.689300838 0.818684876 0.945400647
    1.050293212 1.174812297 1.197044164 1.205997552 1.209776334 1.206965474 1.202857298 1.168455913
    1.178516186 1.146120392 1.165970522 1.14098204 1.091501285 1.052795445 0.971168566 0.879230189";
A4_SAB = "1.051824318 1.027558766 1.022662475 1.033062567 1.032821183 1.073151932 1.104095753 1.149256352
    1.189917618 1.206903066 1.194106858 1.141280143 1.049368708 1.030409138 1.020566211 0.996503977
    0.957821268 0.909606485 0.88206375 0.851650023 0.817218272 0.785744757 0.750715532 0.721690847";
```

```
107
108 R_DOM = "0.902446549 0.793209114 0.762242413 0.634578965 0.612704677 0.619183942 0.61238264 0.686929606
        0.828068989 1.037885774 1.243109979 1.336588186 1.179317195 1.068303868 1.00666421 0.974474563
        1.055412957 1.188204305 1.328681513 1.366134223 1.275224649 1.357337019 1.114345767 1.016568896";
109 C_DOM = "0.993897432 0.972326705 0.936434735 0.892515887 0.874446892 0.895711227 0.907327765 0.891246254
        0.94811259 1.031648282 1.129049038 1.125414381 1.050651267 1.034211027 1.014083065 1.005260628
        0.946590527 0.978046403 1.087096631 1.155707182 1.096008472 1.061153529 1.006011016 0.967049066";
110 IN_DOM = "0.864017127 0.839475836 0.816636209 0.868310784 0.899163682 0.914429973 0.919112742 0.982321722
        1.066884737 1.03857701 1.102481975 1.062662815 1.02392571 1.027317706 1.028976738 1.026373181
        1.067446416 1.0949689 1.180691398 1.097615015 0.988605309 0.969014647 1.086182031 1.034808338";
111 SP_DOM = "0.792237795 0.770612686 0.758954136 0.771332012 0.712239535 0.741435859 0.917681788 0.938596341
        1.014057683 1.134204229 1.178114981 1.191946067 1.170631599 1.197045856 1.156067675 1.153229582
        1.172034243 1.144135328 1.164647755 1.085417461 1.046097267 1.002677709 0.981975564 0.804626847";
112 A4_DOM = "0.966397294 0.953181849 0.942987579 0.932101485 0.940800001 0.925168058 0.92901207 0.954305205
        0.981923834 1.00508849 1.026816252 1.015473749 0.978544231 0.985969777 0.99257738 0.981376001
        0.98515142 0.99085031 1.017522936 1.016404235 1.003221898 1.04150091 1.176262898 1.257362137";
113
114 if Dia_analise == "UT"
115     fid = fopen( 'LoadShapes.txt', 'wt' );
116     fprintf(fid, 'New loadshape.Residencial npts = 24 interval = 1 mult = (%s) \n', R_UTEIS);
117     fprintf(fid, 'New loadshape.Comercial npts = 24 interval = 1 mult = (%s) \n', C_UTEIS);
118     fprintf(fid, 'New loadshape.Industrial npts = 24 interval = 1 mult = (%s) \n', IN_UTEIS);
119     fprintf(fid, 'New loadshape.Serv_Pub npts = 24 interval = 1 mult = (%s) \n', SP_UTEIS);
120     fprintf(fid, 'New loadshape.A4 npts = 24 interval = 1 mult = (%s) \n', A4_UTEIS);
121     fprintf(fid, 'New loadshape.IP npts = 24 interval = 1 mult = (%s) \n', IP_UTEIS);
122     fclose(fid);
123 elseif Dia_analise == "SAB"
124     fid = fopen( 'LoadShapes.txt', 'wt' );
125     fprintf(fid, 'New loadshape.Residencial npts = 24 interval = 1 mult = (%s) \n', R_SAB);
126     fprintf(fid, 'New loadshape.Comercial npts = 24 interval = 1 mult = (%s) \n', C_SAB);
127     fprintf(fid, 'New loadshape.Industrial npts = 24 interval = 1 mult = (%s) \n', IN_SAB);
128     fprintf(fid, 'New loadshape.Serv_Pub npts = 24 interval = 1 mult = (%s) \n', SP_SAB);
129     fprintf(fid, 'New loadshape.A4 npts = 24 interval = 1 mult = (%s) \n', A4_SAB);
130     fprintf(fid, 'New loadshape.IP npts = 24 interval = 1 mult = (%s) \n', IP_UTEIS);
131     fclose(fid);
132 elseif Dia_analise == "DOM"
133     fid = fopen( 'LoadShapes.txt', 'wt' );
134     fprintf(fid, 'New loadshape.Residencial npts = 24 interval = 1 mult = (%s) \n', R_DOM);
135     fprintf(fid, 'New loadshape.Comercial npts = 24 interval = 1 mult = (%s) \n', C_DOM);
136     fprintf(fid, 'New loadshape.Industrial npts = 24 interval = 1 mult = (%s) \n', IN_DOM);
137     fprintf(fid, 'New loadshape.Serv_Pub npts = 24 interval = 1 mult = (%s) \n', SP_DOM);
138     fprintf(fid, 'New loadshape.A4 npts = 24 interval = 1 mult = (%s) \n', A4_DOM);
139     fprintf(fid, 'New loadshape.IP npts = 24 interval = 1 mult = (%s) \n', IP_DOM);
140     fclose(fid);
141 else
142
143 end
144 % //---------------------------------------//
145
146 % //Escrever TXTs
147 % //---------------------------------------//
148
149 N1 = sprintf("LineCode_SSDBT_%s",Ali);
150 N2 = sprintf("LineCode_SSDMT_%s",Ali);
151 N3 = sprintf("Lines_SSDMT_%s",Ali);
152 N4 = sprintf("Lines_SSDBT_%s",Ali);
153 N5 = sprintf("LoadShapes");
154 N6 = sprintf("Loads_UCBT_%s",Ali);
155 N7 = sprintf("Loads_UCMT_%s",Ali);
156 N8 = sprintf("Transformers_%s",Ali);
157 N9 = sprintf("PIP_%s",Ali);
158
159 fid = fopen( 'MainCode.txt', 'wt' );
160 fprintf(fid, 'clear\n\n');
161 fprintf(fid, 'New Circuit.%s bus1 = %s basekv = %g pu = 1.00 phases = 3 frequency = %g mvasc3 = %s mvasc1 =
        %s\n\n', Ali, bus1, DESCR, freq, mvasc3, mvasc1);
162 fprintf(fid, 'Redirect %s.txt\n',N1);
163 fprintf(fid, 'Redirect %s.txt\n',N2);
```

```
fprintf(fid, 'Redirect %s.txt\n',N3);
fprintf(fid, 'Redirect %s.txt\n',N4);
fprintf(fid, 'Redirect %s.txt\n',N5);
fprintf(fid, 'Redirect %s.txt\n',N6);
fprintf(fid, 'Redirect %s.txt\n',N7);
fprintf(fid, 'Redirect %s.txt\n',N8);
fprintf(fid, 'Redirect %s.txt\n',N9);
fprintf(fid, '\nSet VoltageBases = [%g, %g] \nCalcVoltageBases\n', DESCR, base_kv_ten_lin);
fprintf(fid, '\nNew energymeter.medidor element = line.%s terminal = 1\n', Code_first_line);
fprintf(fid, '\nset tolerance = 0.0000001\nset Maxiter=500\nset mode = daily\nset stepsize = 1h\nset number
    = 24\nsolve');
fclose(fid);

fid = fopen(sprintf('%s.txt', N2) ,'wt');
for i = 1:rows_MT %Escrita linecode MT
    x= string(SEGCON_MT{i,2});
    c= SEGCON_MT{i,18};
    y= SEGCON_MT{i,15};
    z= SEGCON_MT{i,16};
    fprintf(fid,'New linecode.%s nphases = 3 basefreq = %g units = km Normamps = %g \n~ R1 = %g !ohm/km \n~
        X1 = %g !ohm/km \n~ C1 = 0.00\n\n', x, freq, c, y, z);
    if i == rows_MT
        clear x c y z i
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N1) ,'wt');
for i = 1:rows_BT %Escrita linecode BT
    x= string(SEGCON_BT{i,2});

    A = str2double(SEGCON_BT{i,37});
    B = str2double(SEGCON_BT{i,38});
    C = str2double(SEGCON_BT{i,39});
    N = str2double(SEGCON_BT{i,40});
    l=0;
    if A ~= 0
        l=1;
    end
    if B ~= 0
        l=l+1;
    end
    if C ~= 0
        l=l+1;
    end
    if N ~= 0
        l=l+1;
    end
    c= SEGCON_BT{i,18};
    y= SEGCON_BT{i,15};
    z= SEGCON_BT{i,16};
    fprintf(fid,'New linecode.%s nphases = %g basefreq = %g units = km Normamps = %g \n~ R1 = %g !ohm/km \n
        ~ X1 = %g !ohm/km \n~ C1 = 0.00\n\n', x, l, freq, c, y, z);
    if i == rows_BT
        clear x c y z i l A B C N
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N3) ,'wt');
for i = 1:rows_SSDMT %Escrita Lines MT
    x= string(SSDMT{i,2});
    c= string(SSDMT{i,3});
    y= string(SSDMT{i,4});
    z= string(SSDMT{i,19});
    l= string(SSDMT{i,13});
    fprintf(fid,'New line.%s phases = 3 bus1 = %s bus2 = %s \n~ length = %s units = m \n~ linecode = %s\n\n
        ', x, c, y, z, l);
```

```
    if i == rows_SSDMT
        clear x c y z i l
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N8) ,'wt');
for i = 1:rows_UNTRD %Escrita TRAFOS
    x= string(UNTRD{i,2}); % COLETA CODIGO/nome TRAFO
    PAC_2 = string(UNTRD{i,5});
    EQTRD_FIL=EQTRD(strcmp(EQTRD.PAC_2,PAC_2),:);
    xhl=EQTRD_FIL{1,32}; % xhl do trafo da iteração
    PAC_1=string(EQTRD_FIL{1,4}); %coleta PAC_1 do trafo
    SSDMT_PN_2=SSDMT(strcmp(SSDMT.PAC_2, PAC_1), :); %filtra SSDMT referente a conexão com o trafo
    isempty(SSDMT_PN_2);

    if isempty(SSDMT_PN_2) == 1
        SSDMT_PN_1=SSDMT(strcmp(SSDMT.PAC_1, PAC_1), :);
        PN_2=string(SSDMT_PN_1{1,3}); %coleta o codigo da barra (saida SSDMT e entrada trafo)
    else
        PN_2=string(SSDMT_PN_2{1,4}); %coleta o codigo da barra (saida SSDMT e entrada trafo)
    end

    kva=UNTRD{i,20};
    fases=string(UNTRD{i,7});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN" ||
        fases == "ABN" || fases == "BCN" || fases == "CAN" || fases == "ABCN"
        conn1 ="wye";
    elseif fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABC"
        conn1 ="delta";
    else
        continue
    end
    res=EQTRD_FIL{1,31};
    kv=UNTRD{i,14};

    fases=string(UNTRD{i,8});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN" ||
        fases == "ABN" || fases == "BCN" || fases == "CAN" || fases == "ABCN"
        conn2 ="wye";
    elseif fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABC"
        conn2 ="delta";
    else
        continue
    end

    tap = UNTRD{i,17};
    noloadloss = UNTRD{i,21}/(kva*10);
    loadloss = UNTRD{i,22}/(kva*10)-noloadloss;

    if fases == "A"
        conex = ".1";
    elseif fases == "B"
        conex = ".2";
    elseif fases == "C"
        conex = ".3";
    elseif fases == "AN"
        conex = ".1.0";
    elseif fases == "BN"
        conex = ".2.0";
    elseif fases == "CN"
        conex = ".3.0";
    elseif fases == "AB"
        conex = ".1.2";
    elseif fases == "BC"
        conex = ".2.3";
    elseif fases == "CA"
        conex = ".3.1";
```

```
    elseif fases == "ABN"
        conex = ".1.2.0";
    elseif fases == "BCN"
        conex = ".2.3.0";
    elseif fases == "CAN"
        conex = ".3.1.0";
    elseif fases == "ABC"
        conex = ".1.2.3";
    elseif fases == "ABCN"
        conex = ".1.2.3.0";
    end
    PN_2_UNTRD(i,:) = {PN_2};
    fprintf(fid,'New transformer.%s xhl = %g windings = 2 %%loadloss = %g %%noloadloss = %g  \n~ wdg = 1
        bus = %s kv = %g kva = %g conn = %s %%r = %g tap = 0.9565 \n~ wdg = 2 bus = BT_%s%s kv = %g kva = %
        g conn = %s %%r = %g tap = %g\n\n', x, xhl, loadloss, noloadloss, PN_2, DESCR, kva, conn1, res,
        PN_2, conex, kv, kva, conn2, res, tap);
    if i == rows_UNTRD
        clear x c y z i l fases a PAC_1 xhl PAC_2 kv conn2 kva PN_2
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N4) ,'wt');
for i = 1:rows_SSDBT %Escrita Lines BT
    x= string(SSDBT{i,2});

    fases=string(SSDBT{i,10});
    if fases == "A" || fases == "B" || fases == "C" || fases == "N"
            phase = "1";
    elseif  fases == "AN" || fases == "BN" || fases == "CN"
            phase = "2";
    elseif  fases == "AB" || fases == "BC" || fases == "CA"
            phase = "2";
    elseif  fases == "ABN" || fases == "BCN" || fases == "CAN" || fases == "ABC"
            phase = "3";
    elseif  fases == "ABCN"
            phase = "4";
    end

    c= string(SSDBT{i,3});
    y= string(SSDBT{i,4});

    if fases == "A"
        conex2 = ".1";
    elseif fases == "B"
        conex2 = ".2";
    elseif fases == "C"
        conex2 = ".3";
    elseif fases == "N"
        conex2 = ".4";
    elseif fases == "AN"
        conex2 = ".1.4";
    elseif fases == "BN"
        conex2 = ".2.4";
    elseif fases == "CN"
        conex2 = ".3.4";
    elseif fases == "AB"
        conex2 = ".1.2";
    elseif fases == "BC"
        conex2 = ".2.3";
    elseif fases == "CA"
        conex2 = ".3.1";
    elseif fases == "ABN"
        conex2 = ".1.2.4";
    elseif fases == "BCN"
        conex2 = ".2.3.4";
    elseif fases == "CAN"
        conex2 = ".3.1.4";
```

```
    elseif fases == "ABC"
        conex2 = ".1.2.3";
    elseif fases == "ABCN"
        conex2 = ".1.2.3.4";
    end
    PN_2_CONTAIN= find(strcmp(PN_2_UNTRD.Var1,c));

    if PN_2_CONTAIN ~= '0'
        if fases == "A"
            conex1 = ".1";
        elseif fases == "B"
            conex1 = ".2";
        elseif fases == "C"
            conex1 = ".3";
        elseif fases == "N"
            conex1 = ".4";
        elseif fases == "AN"
            conex1 = ".1.0";
        elseif fases == "BN"
            conex1 = ".2.0";
        elseif fases == "CN"
            conex1 = ".3.0";
        elseif fases == "AB"
            conex1 = ".1.2";
        elseif fases == "BC"
            conex1 = ".2.3";
        elseif fases == "CA"
            conex1 = ".3.1";
        elseif fases == "ABN"
            conex1 = ".1.2.0";
        elseif fases == "BCN"
            conex1 = ".2.3.0";
        elseif fases == "CAN"
            conex1 = ".3.1.0";
        elseif fases == "ABC"
            conex1 = ".1.2.3";
        elseif fases == "ABCN"
            conex1 = ".1.2.3.0";
        end
    else
        conex1 = conex2;
    end
    z= string(SSDBT{i,20});
    l= string(SSDBT{i,14});
    fprintf(fid,'New line.%s phases = %s bus1 = BT_%s%s bus2 = BT_%s%s \n~ length = %s units = m \n~
        linecode = %s\n\n', x, phase, c, conex1, y, conex2, z, l);
    if i == rows_SSDBT
        clear x c y z i l fases a conex2 conex1 phase PN_2_CONTAIN
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N7) ,'wt');
for i = 1:rows_UCMT %Escrita loads MT
    x = string(UCMT{i,1});
    c = string(UCMT{i,2});
    kv = TTEN(TTEN.COD_ID == str2double(UCMT{i,18}), 3);
    kv{1,1} = regexprep(kv{1, 1}, 'V$', '');
    kv{1,1} = regexprep(kv{1, 1}, 'k$', '');
    kv{1,1} = regexprep(kv{1, 1}, ' ', '');
    kv=str2double(kv{1,1});
    soma = sum(UCMT{i,36:47},2) /(30*12*24);
    fases = string(UCMT{i,16});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN"
        conn ="wye";
    elseif fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABN" || fases == "BCN" || fases ==
        "CAN" || fases == "ABC" || fases == "ABCN"
        conn ="delta";
```

```
    end
    fprintf(fid,'New load.%s phases = 3 model = 8 ZIPV = [0.5 0 0.5 0 0 1 0.9] daily = A4 bus = %s.1.2.3 kv
        = %g pf = 0.92 kw = %g  conn = %s\n\n', x, c, kv, soma, conn);
    if i == rows_UCMT
        clear x c y z i kv column soma fases conn
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N6) ,'wt');
for i = 1:rows_UCBT %Escrita loads BT
    x = string(UCBT{i,1});

    fases = string(UCBT{i,17});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN" ||
        fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABN" || fases == "BCN" || fases == "
        CAN"
        phase ="1";
    elseif fases == "ABC" || fases == "ABCN"
        phase ="3";
    end

    c = string(UCBT{i,5});
    tip_cc = string(UCBT{i,16});
    if contains(tip_cc,"COM-") == 1
        daily = "Comercial";
    elseif contains(tip_cc,"RES-") == 1
        daily = "Residencial";
    elseif contains(tip_cc,"IND-") == 1
        daily = "Industrial";
    elseif contains(tip_cc,"SP-") == 1
        daily = "Serv_Pub";
    elseif contains(tip_cc,"IP-") == 1
        daily = "IP";
    end

    PN_2_CONTAIN= find(strcmp(PN_2_UNTRD.Var1,c));
    if PN_2_CONTAIN ~= '0'
        if fases == "A"
            conex2 = ".1";
        elseif fases == "B"
            conex2 = ".2";
        elseif fases == "C"
            conex2 = ".3";
        elseif fases == "N"
            conex2 = ".0";
        elseif fases == "AN"
            conex2 = ".1.0";
        elseif fases == "BN"
            conex2 = ".2.0";
        elseif fases == "CN"
            conex2 = ".3.0";
        elseif fases == "AB"
            conex2 = ".1.2";
        elseif fases == "BC"
            conex2 = ".2.3";
        elseif fases == "CA"
            conex2 = ".3.1";
        elseif fases == "ABN"
            conex2 = ".1.2.0";
        elseif fases == "BCN"
            conex2 = ".2.3.0";
        elseif fases == "CAN"
            conex2 = ".3.1.0";
        elseif fases == "ABC"
            conex2 = ".1.2.3";
        elseif fases == "ABCN"
            conex2 = ".1.2.3.0";
```

```
        end
    else
        if fases == "A"
            conex2 = ".1";
        elseif fases == "B"
            conex2 = ".2";
        elseif fases == "C"
            conex2 = ".3";
        elseif fases == "N"
            conex2 = ".4";
        elseif fases == "AN"
            conex2 = ".1.4";
        elseif fases == "BN"
            conex2 = ".2.4";
        elseif fases == "CN"
            conex2 = ".3.4";
        elseif fases == "AB"
            conex2 = ".1.2";
        elseif fases == "BC"
            conex2 = ".2.3";
        elseif fases == "CA"
            conex2 = ".3.1";
        elseif fases == "ABN"
            conex2 = ".1.2.4";
        elseif fases == "BCN"
            conex2 = ".2.3.4";
        elseif fases == "CAN"
            conex2 = ".3.1.4";
        elseif fases == "ABC"
            conex2 = ".1.2.3";
        elseif fases == "ABCN"
            conex2 = ".1.2.3.4";
        end
    end

    kv = TTEN(TTEN.COD_ID == str2double(UCBT{i,19}), 3);
    kv{1,1} = regexprep(kv{1, 1}, 'V$', '');
    kv{1,1} = regexprep(kv{1, 1}, 'k$', '');
    kv{1,1} = regexprep(kv{1, 1}, ' ', '');
    kv=str2double(kv{1,1});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN"
        kv=kv/1000;
    else
        kv = base_kv_ten_lin;
    end

    soma = sum(UCBT{i,25:36},2) /(30*12*24);

    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN"
        conn = "wye";
    elseif fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABN" || fases == "BCN" || fases ==
        "CAN" || fases == "ABC" || fases == "ABCN"
        conn = "delta";
    end
    fprintf(fid,'New load.%s phases = %s model = 8 ZIPV = [0.5 0 0.5 0 0 1 0.85] daily = %s bus = BT_%s%s
        kv = %g pf = 0.92 kw = %g  conn = %s\n\n', x, phase, daily, c, conex2, kv, soma, conn);
    if i == rows_UCBT
        clear x fases phase conex2 soma kv conn tip_cc daily
    end
end
fclose(fid);

fid = fopen(sprintf('%s.txt', N9) ,'wt');
for i = 1:rows_PIP %Escrita PIP
    x = string(PIP{i,2});

    fases = string(PIP{i,12});
    if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN" ||
```

```
        fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABN" || fases == "BCN" || fases == "
        CAN"
556         phase ="1";
557     elseif fases == "ABC" || fases == "ABCN"
558         phase ="3";
559     end
560
561     c = string(PIP{i,10});
562     if fases == "A"
563         conex2 = ".1";
564     elseif fases == "B"
565         conex2 = ".2";
566     elseif fases == "C"
567         conex2 = ".3";
568     elseif fases == "AN"
569         conex2 = ".1.4";
570     elseif fases == "BN"
571         conex2 = ".2.4";
572     elseif fases == "CN"
573         conex2 = ".3.4";
574     elseif fases == "AB"
575         conex2 = ".1.2";
576     elseif fases == "BC"
577         conex2 = ".2.3";
578     elseif fases == "CA"
579         conex2 = ".3.1";
580     elseif fases == "ABN"
581         conex2 = ".1.2.4";
582     elseif fases == "BCN"
583         conex2 = ".2.3.4";
584     elseif fases == "CAN"
585         conex2 = ".3.1.4";
586     elseif fases == "ABC"
587         conex2 = ".1.2.3";
588     elseif fases == "ABCN"
589         conex2 = ".1.2.3.4";
590     end
591     kv = TTEN(TTEN.COD_ID == str2double(PIP{i,14}), 3);
592     kv{1,1} = regexprep(kv{1, 1}, 'V$', '');
593     kv{1,1} = regexprep(kv{1, 1}, 'k$', '');
594     kv{1,1} = regexprep(kv{1, 1}, ' ', '');
595     kv=str2double(kv{1,1});
596
597     if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN"
598         kv=kv/1000;
599     else
600         kv = base_kv_ten_lin;
601     end
602
603     soma = sum(PIP{i,20:31},2) /(30*12*24);
604
605     if fases == "A" || fases == "B" || fases == "C" || fases == "AN" || fases == "BN" || fases == "CN"
606         conn = "wye";
607     elseif fases == "AB" || fases == "BC" || fases == "CA" || fases == "ABN" || fases == "BCN" || fases ==
        "CAN" || fases == "ABC" || fases == "ABCN"
608         conn = "delta";
609     end
610     fprintf(fid,'New load.%s phases = %s model = 8 ZIPV = [0.5 0 0.5 0 0 1 0.85] daily = IP bus = BT_%s%s
        kv = %g pf = 0.92 kw = %g  conn = %s\n\n', x, phase, c, conex2, kv, soma, conn);
611     if i == rows_UCBT
612         clear x fases phase conex2 soma kv conn
613     end
614 end
615 fclose(fid);
616
617 % //--------------------------------------//
```

HENRIQUE DA SILVA

FERRAMENTA COMPUTACIONAL PARA CRIAÇÃO DE MODELOS DE ALIMENTADORES REAIS DE DISTRIBUIÇÃO NO OPENDSS A PARTIR DE DADOS DISPONIBILIZADOS PELA ANEEL.

Este trabalho foi julgado adequado para obtenção do título em Bacharel em Engenharia Elétrica, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Jaraguá do Sul, 06 de julho de 2022.

_____
Prof. Msc. John Jefferson Antunes Saldanha
Orientador
IFSC – Campus Jaraguá do Sul – Rau

_____
Profa. Dra. Ana Paula Carboni de Mello
Universidade Federal do Pampa - UNIPAMPA

_____
Prof. MSc. Vitor Teles Correia
IFSC – Campus Jaraguá do Sul – Rau

Comprovante de Assinatura Eletrônica

contraktor

Datas e horários baseados no fuso horário (GMT -3:00) em Brasília, Brasil
Sincronizado com o NTP.br e Observatório Nacional (ON)
Certificado de assinatura gerado em 09/07/2022 às 12:40:58 (GMT -3:00)

TermoDeAprovacao_TCC_EngEletrica_IFSC_JGS_RAU_06072022_HENRIQUE DA SILVA

ID única do documento: #12fe5022-3eaa-4013-b834-78456c638f73

Hash do documento original (SHA256): 1e60c1bd94c61ac960810208a5e26ab0cfbe6ea278a4bfaf8e46be1ac5840f96

Este Log é exclusivo ao documento número #12fe5022-3eaa-4013-b834-78456c638f73 e deve ser considerado parte do mesmo, com os efeitos prescritos nos Termos de Uso.

Assinaturas (3)

- John Jefferson Antunes Saldanha (Participante)
  Assinou em 09/07/2022 às 09:42:57 (GMT -3:00)
- Ana Paula Carboni de Mello (Participante)
  Assinou em 11/07/2022 às 15:21:29 (GMT -3:00)
- Vitor Teles Correia (Participante)
  Assinou em 13/07/2022 às 09:03:48 (GMT -3:00)

Histórico completo

Data e hora	Evento
11/07/2022 às 18:21:29 (GMT -3:00)	Ana Paula Carboni de Mello (Autenticação: e-mail anamello@unipampa.edu.br; IP: 200.132.136.248) assinou. Autenticidade deste documento poderá ser verificada em https://verificador.contraktor.com.br. Assinatura com validade jurídica conforme MP 2.200-2/01, Art. 10o, §2.

Data e hora	Evento
09/07/2022 às 12:42:57 (GMT -3:00)	John Jefferson Antunes Saldanha (Autenticação: e-mail john.saldanha@ifsc.edu.br; IP: 177.66.146.223) assinou. Autenticidade deste documento poderá ser verificada em https://verificador.contraktor.com.br. Assinatura com validade jurídica conforme MP 2.200-2/01, Art. 10o, §2.
09/07/2022 às 12:40:58 (GMT -3:00)	John Jefferson solicitou as assinaturas.
13/07/2022 às 12:03:48 (GMT -3:00)	Vitor Teles Correia (Autenticação: e-mail vitor.correia@ifsc.edu.br; IP: 177.104.9.62) assinou. Autenticidade deste documento poderá ser verificada em https://verificador.contraktor.com.br. Assinatura com validade jurídica conforme MP 2.200-2/01, Art. 10o, §2.