UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS DOIS VIZINHOS
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

MARCELO CHELLA

ESTUDO EXPLORATÓRIO DA BIBLIOTECA PYTHON PANDAPOWER E SUA UTILIZAÇÃO EM CONJUNTO COM REINFORCEMENT LEARNING PARA ANÁLISE E OTIMIZAÇÃO DE SISTEMAS DE ENERGIA

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2022

MARCELO CHELLA

ESTUDO EXPLORATÓRIO DA BIBLIOTECA PYTHON PANDAPOWER E SUA UTILIZAÇÃO EM CONJUNTO COM REINFORCEMENT LEARNING PARA ANÁLISE E OTIMIZAÇÃO DE SISTEMAS DE ENERGIA

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Prof. Dr. Marcelo Teixeira

DOIS VIZINHOS
2022

MARCELO CHELLA

ESTUDO EXPLORATÓRIO DA BIBLIOTECA PYTHON PANDAPOWER E SUA UTILIZAÇÃO EM CONJUNTO COM REINFORCEMENT LEARNING PARA ANÁLISE E OTIMIZAÇÃO DE SISTEMAS DE ENERGIA

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Data de aprovação: 10/novembro/2022

Marcelo Teixeira
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

Jefferson Tales Oliva
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

Dalcimar Casanova
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

DOIS VIZINHOS
2022

Dedico este trabalho à minha esposa Dani, ao meu pai Eloy e às minhas irmãs Márcia e Rosane, pelo apoio e encorajamento nos momentos difíceis.

Dedico, além disso, à memória da minha mãe, Bernadete, e também à memória dos meus sogros Ivette e Dario. Ambos os últimos vieram a falecer ainda neste ano e deixaram um grande vazio no dia a dia da família, permanecendo apenas a sua lembrança e exemplo de vida.

AGRADECIMENTOS

Agradeço à Deus pela saúde e pelas pessoas que ele colocou na minha vida, e às quais espero corresponder com compreensão, gratidão e amor.

Agradeço à minha esposa Danielle, pela força e ajuda nos tempos difíceis que ambos estivemos enfrentando recentemente, por problemas de saúde próprios e em família.

Agradeço aos meus pais, Eloy e Bernadete, e às minhas irmãs, Rosane e Márcia.

Agradeço à Universidade Tecnológica Federal do Paraná, onde trabalho e estudo, pela oportunidade de usufruir de uma plataforma de ensino à distância eficiente e de qualidade, e de contar com excelentes professores durante esta vivência. Aproveito para agradecer também a todos os professores que transformaram as suas experiências e conhecimentos em conteúdo de valor para este curso de especialização.

Agradeço aos meus colegas de Especialização pelo apoio, companheirismo e generosidade em ajudar nesta caminhada.

Por fim, agradeço especialmente ao meu orientador Prof. Dr. Marcelo Teixeira, que me auxiliou grandemente no desenvolvimento deste TCC, apesar de haver momentos em que, devido aos graves problemas de saúde durante este ano, pareceu que eu seria incapaz de finalizar esta atividade.

"In God we trust. All others must bring data.". (Edwin R. Fisher, 1978).

RESUMO

Atualmente, a maioria das pesquisas e projetos na área de energia se baseiam em software como ferramenta primária. O uso deste tipo de abordagem possibilita a pesquisa, simulação, estudo e fácil alteração de parâmetros de um determinado sistema, viabilizando as pesquisas e o aprendizado relacionados com facilidade e a custos menores. Nesse escopo, estão os aspectos econômicos, ambientais e de engenharia das redes elétricas. Dentro desta linha de ação, a ferramenta pandapower foi desenvolvida para ser um instrumento de simulação para pesquisadores e educadores, de fácil uso e modificação, mantendo o código criado simples de entender e modificar. Essa toolbox é uma ferramenta em Python de código aberto para modelagem, análise e otimização de sistemas de energia com alto grau de automação, e também baseada em uma estrutura de dados tabular da biblioteca Python pandas. As aplicações para esta ferramenta são o cálculo de fluxo de carga trifásico AC, a configuração/execução de estimativa de estado do sistema e balanceamento de carga AC (melhorando a distribuição de cargas ou redimensionando pontos de transformação). Adicionalmente, algoritmos de aprendizado por Aprendizado por Reforço (*Reinforcement Learning*, RL) podem ser utilizados nas tarefas relacionadas a análises de resposta à demanda. Este trabalho visa explorar as aplicações do pandapower fazendo a modelagem e cálculo de fluxo de potência em redes de energia, bem como a demonstrar a realização de previsões de comportamentos destas redes em relação à eventos. Alguns modelos de Inteligência Artificial baseados em Redes Neurais Artificiais e Aprendizado por Reforço serão usados para demonstrações de previsão de estado da rede e exploração de cenários de geração fotovoltaica em resposta a demanda variável, e seus resultados para a otimização de sistemas de energia serão apresentados.

Palavras-chave: Machine Learning, Inteligência Artificial; Python; Sistemas Elétricos; Engenharia Elétrica.

ABSTRACT

Nowadays, most of the research and project in the energy field are based on software as a primary tool. The use of this type of approach makes it possible to research, simulate, study and easily change the parameters of a given system, enabling easy and less expensive research on the matter. Within this scope are the economic, environmental and engineering aspects of electrical networks. Following this model, the pandapower tool was developed to be a simulation tool for researchers and educators, easy to use and modify, keeping the created code simple to understand and modify. This toolbox is an open-source Python tool for modeling, analyzing and optimizing power systems with a high degree of automation, and also based on a tabular data structure from the Python pandas library. Applications for this tool are calculation of three-phase AC load flow, configuration/execution of system state estimation and AC load balancing (improving load distribution or resizing transformer points). Additionally, Reinforcement Learning (RL) algorithms can be used in tasks related to demand response analysis. This work aims to explore the applications of pandapower by modeling and calculating power flow on energy networks, as well as demonstrating the prediction of behavior of these networks in response to triggered events. Some Artificial Intelligence models based on Artificial Neural Networks (ANN) and Reinforcement Learning will be used for demonstrations of grid state prediction and exploration of photovoltaic generation scenarios in response to variable demand, and their results for the optimization of energy systems will be presented.

Keywords: Machine Learning, Artificial Intelligence; Python; Electric Systems; Electrical Engineering.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Recentemente, os sistemas de energia elétrica começaram a passar por mudanças altamente complexas, incluindo a inserção de recursos de energia renovável com disponibilidade sazonal, redes com interligações complexas, necessidade de comunicações de dados entre geradores, consumidores e as empresas distribuidoras de energia, e várias conexões de rede com o recurso de fluxo de energia bidirecional. O desenvolvimento de sistemas inteligentes de energia vem se tornando cada vez mais urgente para promover a construção de um sistema de energia limpo, de baixo carbono, seguro e eficiente. Atualmente, a inteligência artificial (*Artificial Inteligence*, AI) é uma tecnologia emergente neste campo, e está fornecendo um grande suporte para promover e dar suporte a esta evolução nos sistemas de energia. Em particular, a combinação de inteligência artificial com computação em nuvem, *big data*, internet das coisas (*Internet of Things*, IoT) e interconexão móvel pode dotar o sistema de energia com recursos de interação inteligente, segurança e controlabilidade (ZINAMAN; SADAMOR, 2019).

Dentro deste quadro, a ferramenta pandapower para linguagem Python foi criada para ser um instrumento de simulação para pesquisadores e engenheiros, de fácil uso e modificação, mantendo o código criado simples de entender e modificar. Esta ferramenta começou a ser desenvolvida na Universidade de Kassel e no *Fraunhofer Institute for Energy, Economics and Energy System Technology* (IEE) de Kassel, pelos pesquisadores Leon Thurner, Alexander Scheidler e demais colaboradores do projeto (THURNER et al., 2018), com a finalidade de criar um programa de cálculo de redes fácil de usar e voltado para automação da análise e da otimização em sistemas de energia. O pandapower foi baseado na biblioteca de análise de dados pandas e na *toolbox* Python de análise de sistemas de energia pypower, sendo esta última um port da *toolbox* MATPOWER em linguagem MATLAB para a linguagem Python. Uma das vantagens de utilização do pandapower portado em Python é ser publicado em uma licença de codigo aberto "*3-clause BSD*" ao invés da licença proprietária do MATLAB. Adicionalmente, combinados com outras bibliotecas Python, algoritmos de aprendizado por reforço podem ser usados em tarefas de análise e controle complexas, como veremos mais adiante no desenvolvimento desta pesquisa.

Como fonte de dados, uma das bases de dados utilizada foi o conteúdo do projeto SimBench, uma base de dados livre também desenvolvida em estudos na Universidade de Kassel e no *Fraunhofer Institute for Energy, Economics and Energy System Technology* com dados do sistema de energia da Alemanha, contendo parâmetros elétricos para a modelagem de 13 redes elétricas diferentes e interligadas, se necessário, com níveis desde baixa (LV) até extra-alta tensão (EHV). O objetivo do projeto SimBench é desenvolver um conjunto de dados de referência para soluções em análise de rede, planejamento de rede e gerenciamento de operação de rede.

Este conjunto de dados destina-se a desenvolver novos métodos e soluções independentes de conjuntos de dados de rede individuais não publicamente disponíveis para garantir a comparabilidade, transparência e transparência de vários desenvolvimentos neste campo. Além disso, a própria ferramenta pandapower tem um conjunto de dados modelados a partir de estudos, artigos e livros-texto de sistemas elétricos, para uso em ajustes de configurações e comparações de resultados teóricos a partir do uso da ferramenta.

1.1 Problema de Pesquisa

Uma área de aplicação do modelo descrito na introdução está na crescente aceitação de uma política global de combate ao aumento das emissões de gases de efeito estufa. Esta questão tem sido debatida desde o início dos anos 1990 até mais recentemente na Conferência do Clima de Paris (COP21) em dezembro de 2015. Esta política favorece o uso de energia renovável em detrimento de combustíveis fósseis, e necessita da integração destes diversos tipos de fontes de energia no mesmo sistema elétrico, incorporados nas redes elétricas tradicionais. No entanto, esta integração energética pode afetar a estabilidade do sistema devido à localização não ideal das fontes de energia, ligadas à disponibilidade do recurso natural a ser convertido, e pode vir a causar o aumento da perda de transmissão, fluxos de energia indesejados e um nível de tensão nos consumidores em valor inferior ao limite legal. Meios convencionais de controle de rede (carga ajustável do transformador de derivação, transformadores defasados, compensadores seriais ou paralelos comutados por circuitos disjuntores, alteração de ordens de produção, alteração de topologia de rede e ação no geradores de excitação) podem no futuro ser muito lentos e insuficientes para responder às perturbações do sistema. Por consequência, o método de controle de não conformidades do sistema de energia precisa ser preditivo e ter um tempo de resposta curto para configurar o sistema de acordo com as mudanças de fluxo de energia (TOURQUI; BENAKCHA; ALLAOUI, 2018).

1.2 Hipótese de Pesquisa

A resposta à demanda é definida como "uma mudança do padrão de uso de eletricidade dos consumidores de energia em resposta às mudanças no preço da eletricidade ao longo do tempo". Isto pode ser feito como descontos em taxas ou pagamentos de incentivos, destinados a induzir menor uso de eletricidade no consumidor menor em momentos de necessidade de energia no sistema para uso industrial ou quando a confiabilidade do sistema é comprometida. Neste contexto é possível assumir que, por exemplo, existam 1000 domicílios conectados a um determinado barramento e que cada domicílio tenha vários componentes consumidores de energia que podem ser controlados, como carregamento de veículos elétricos, aquecedores e bombas hidráulicas.

Ao montar um modelo de rede de energia experimental, pode ser feito um estudo com um modelo de Aprendizado por Reforço, onde um agente pode manipular a demanda de energia em cargas flexíveis e desta forma pode ser feita uma análise dos melhores cenários de geração renovável fotovoltaica durante períodos de tempo *versus* demanda variável dos usuários *versus* restrições das linhas em corrente e tensão, de forma a embasar o estabelecimento de uma resposta à demanda ideal para este sistema.

1.3 Objetivos

Os objetivos do trabalho são apresentados a seguir.

1.3.1 Objetivo Geral

Desenvolver uma abordagem para especificação de requisitos em trabalhos com Projetos Elétricos em Sistemas de Energia usando pandapower e demonstrar o uso de algoritmo de RL para analisar e configurar um modelo de sistema elétrico de potência através de um programa simplificado e com resposta ideal.

1.3.2 Objetivos Específicos

- Estabelecer a técnica para uso do powerpandas;
- Selecionar critérios de teste de modelos de sistema;
- Definir a especificação de requisitos;
- Avaliar por meio de estudo de caso a especificação de requisitos.

1.4 Organização do Trabalho

Este trabalho é baseado no material de Florian Schaefer que explica os fundamentos do powerpandas e seu uso na modelagem e resolução de cálculos em sistemas elétricos de potência (SCHAEFER, 2020).

Este trabalho também é baseado na análise e estudo da tese de mestrado de Vegard Ulriksen Solberg, intitulada "*Reinforcement learning for grid control in an electric distribution system*"(SOLBERG, 2019), que demonstra a montagem de uma rede elétrica experimental e seu uso em conjunto com bibliotecas Python para montagem de uma estrutura de Aprendizagem por Reforço para prolemas de Sistemas Elétricos de Potência. Além disso, este material aborda as bibliotecas de RL gym e stable_baselines e seus métodos e funções aplicados a este problema do uso de RL em sistemas de energia.

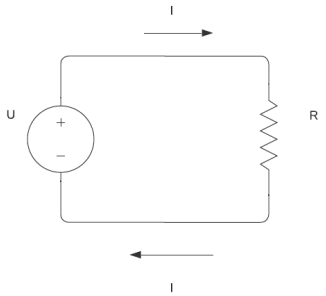
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo vai apresentar os conceitos teóricos necessários para o desenvolvimento da solução proposta neste trabalho. A parte dos conceitos de Circuitos Elétricos e Sistemas Elétricos de Potência envolvida no assunto será adaptada de Solberg (2019) e Bichels (2018), cujas explicações sobre o tema estão detalhadas de maneira clara e concisa. As partes envolvendo Inteligência Artificial (Artificial Inteligence, AI), Aprendizado por Reforço (Reinforcement Learning, RL) e Exploração de Dados têm as suas referências detalhadas nos textos.

2.1 Sistemas Elétricos de Potência (SEP)

De Solberg (2019), temos que um circuito elétrico é um modelo que descreve como a energia elétrica é transferida de uma fonte de energia até uma carga. Um exemplo de fonte é uma tomada na parede e uma carga pode ser uma lâmpada ou um aspirador de pó. Um modelo de um circuito elétrico simples (Figura 1), onde U é a tensão referente à fonte elétrica e R é a carga deste circuito.

Figura 1 – Circuito Elétrico



Fonte: Elaboração Própria

As relações entre tensão, resistência e corrente são definidas pela Lei de Ohm:

$$U = R.I \tag{1}$$

O sistema de transmissão elétrica é análogo a esta configuração, exceto que com várias fontes elétricas (usinas) e cargas (cidades) conectadas (SOLBERG, 2019). Outra versão desta equação é possível introduzindo a condutância ou admitância Y que é definida como o inverso da resistência R, ou seja:

$$Y = \frac{1}{R} \tag{2}$$

A admitância Y é uma medida de como a resistência R permite que um fluxo de cargas elétricas passe em um circuito, e sua unidade é o siemens (S) (SOLBERG, 2019). Usando a admitância Y , a lei de Ohm pode ser expressa como:

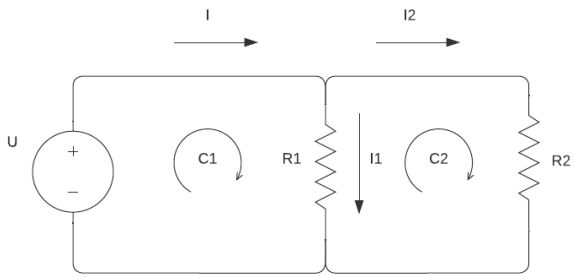
$$I = U.Y \tag{3}$$

A potência elétrica P que uma carga elétrica consome pode ser calculada multiplicando-se a corrente I e queda de tensão U sobre a carga. A unidade de potência elétrica é o watt (W) ou joule por segundo.:

$$P = U.I \tag{4}$$

A Figura 2 mostra um circuito com dois ramos e duas resistências.

Figura 2 – Circuito Elétrico Com Dois Ramos



Fonte: Elaboração Própria

A corrente se divide quando chega a uma interseção, ou nó, de modo que a corrente total entrando para o nó é igual à corrente total saindo dele.

$$I = I_1 + I_2 \tag{5}$$

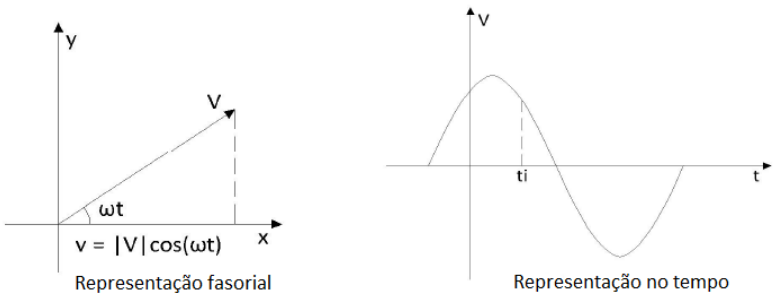
A Equação (5) é a 1º Lei de Kirchoff, ou lei da conservação da corrente ou lei das correntes de Kirchhoff. A 2º Lei de Kirchoff, também conhecida como lei das tensões de Kirchoff, afirma que a soma das tensões U sobre os componentes em um circuito fechado C é igual a zero. Para as duas malhas C_1 e C_2 na Figura 2 temos:

$$U + I_1.R_1 = 0 \tag{6}$$

$$I_1.R_1 - I_2.R_2 = 0 \tag{7}$$

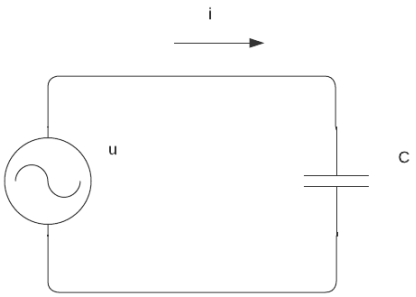
Considerando os circuitos alimentados com fonte de tensão de corrente alternada senoidal, que é o caso do sistema elétrico de potência, existem cargas elétricas que causam um deslocamento de fase entre as formas de onda de corrente e tensão (SOLBERG, 2019).

Figura 3 – Representação da Tensão Senoidal



Fonte: Bichels (2018)

Figura 4 – Circuito Elétrico AC Com Capacitor



Fonte: Elaboração Própria

Por exemplo, o circuito mostrado na Figura 4 tem um capacitor como carga. Um capacitor é um componente que pode armazenar uma carga elétrica Q quando uma fonte de tensão U é aplicada sobre ele. Ele é um elemento passivo projetado para armazenar energia na forma de um campo elétrico.

$$C = Q/U \tag{8}$$

Q é a carga armazenada pelo capacitor e U é a tensão DC aplicada. A unidade de capacitância é o farad (F). Os capacitores são relevantes porque aparecem no Sistema Elétrico de Potência como parâmetros dos modelos de linhas de transmissão. Aplicando a lei das tensões de Kirchoff a este circuito quando a fonte é um sinal senoidal de frequência f, onde t é o tempo (SOLBERG, 2019):

$$u(t) = U.sin(\omega.t) \tag{9}$$

$$\omega = 2.\pi.f \tag{10}$$

$$U.sin(\omega.t) - \frac{Q(t)}{C} = 0 \tag{11}$$

$$Q(0) = 0 \tag{12}$$

A corrente i(t) é a derivada de u(t) em relação ao tempo (SOLBERG, 2019):

$$i(t) = C.\frac{\partial u}{\partial t} \tag{13}$$

Esta corrente é calculada então como:

$$i(t) = U.\omega.C.sin(\omega.t + \frac{\pi}{2}) \tag{14}$$

Esta equação mostra que a corrente i está deslocada de fase 90 graus à frente da tensão u, correspondente ao termo $+\frac{\pi}{2}$ na parte fasorial da equação. Outro conceito convencionado a partir destes cálculos é a reatância capacitiva Xc dada como:

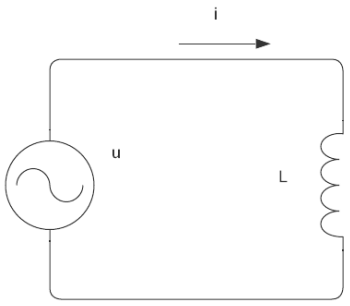
$$X_c = \frac{1}{\omega.C} \tag{15}$$

Onde ω é a frequência angular do sinal senoidal conforme acima, e C é a capacitância do capacitor. A corrente do circuito em corrente alternada agora pode ser expressa da mesma forma que a lei de Ohm para circuitos resistivos em corrente contínua, ou seja:

$$i(t) = U.X_c.sin(\omega.t + \frac{\pi}{2}) \tag{16}$$

O circuito mostrado na Figura 5 é outro exemplo de um componente reativo que desloca a corrente.

Figura 5 – Circuito Elétrico AC Com Indutor



Fonte: Elaboração Própria

A fonte está conectada a uma bobina eletromagnética, também chamada de indutor, um elemento passivo projetado para armazenar energia em um campo magnético. Ele aparece em uma ampla gama de componentes elétricos, como motores e reatores, e também aparece nos modelos de linhas de transmissão elétrica. A tensão sobre um indutor é proporcional à derivada em relação ao tempo da corrente que flui através deste componente, similar ao que foi demonstrado para o capacitor (SOLBERG, 2019):

$$u(t) = L.\frac{\partial i}{\partial t} \tag{17}$$

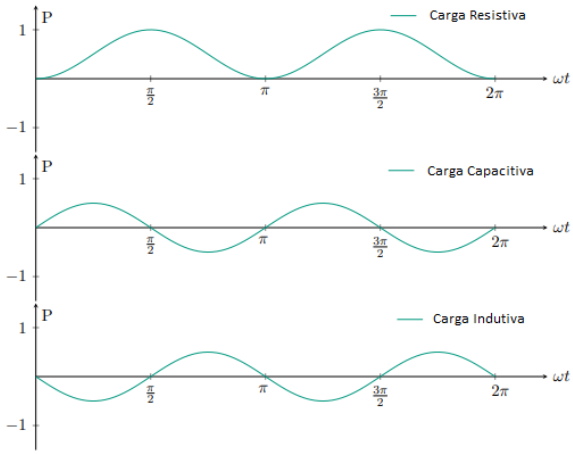
$$i(t) = \frac{U}{\omega.L}.sin(\omega.t - \frac{\pi}{2}) \tag{18}$$

$$X_l = \omega.L \tag{19}$$

Onde ω é a frequência angular da tensão e L é a indutância da bobina. Neste caso, a corrente é deslocada 90 graus atrasada em relação à tensão. Ambos os componentes, indutores e capacitores, deslocam a corrente em 90 graus, mas em direções diferentes.

Por Solberg (2019), para uma carga puramente resistiva, sem deslocamento de fase entre corrente e tensão, a potência instantânea transferida dada pela Equação (4) é sempre positiva. Essa potência é chamada de potência ativa e é medida em watt (W). Para cargas reativas, a mudança de fase entre a corrente e a tensão resulta em uma potência alternante entre a fonte e a carga, conforme mostrado na Figura 6.

Figura 6 – Potência Instantânea Transferida Para a Carga Em Um Circuito Puramente Resistivo, Indutivo e Capacitivo Iguais em Valor

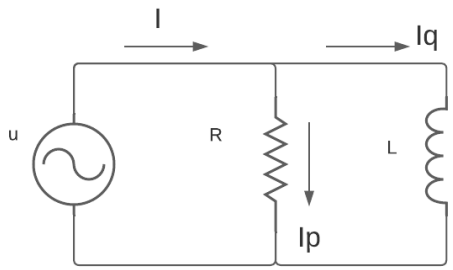


Fonte: Elaboração Própria, baseado em Solberg (2019)

A variação da potência resultante em cargas reativas é chamada de potência reativa Q e possui unidade VAr, para distingui-la da potência ativa. A Figura 6 mostra que a potência instantânea resultante de uma carga indutiva e capacitiva são opostas uma da outra. Linhas de transmissão aéreas são consideradas consumidores de energia reativa, porque têm componentes principalmente indutivos.

Corrente, tensão, impedância e potência são todos expressos como números complexos em um sistema de energia elétrica. Considerando o circuito mostrado na Figura 7, o resistor vai drenar uma corrente Ip que está em fase com a tensão da fonte. O indutor irá drenar uma corrente Iq atrasada em relação à tensão em 90 graus. A corrente resultante drenada da fonte será uma combinação linear de dois sinais senoidais defasados. As equações para somar sinais senoidais com deslocamento de fase são difíceis de trabalhar, e por este motivo é feito o uso de números complexos nestes cálculos.

Figura 7 – Potência Instantânea Transferida Para a Carga Em Um Circuito Puramente Resistivo, Indutivo e Capacitivo Iguais em Valor



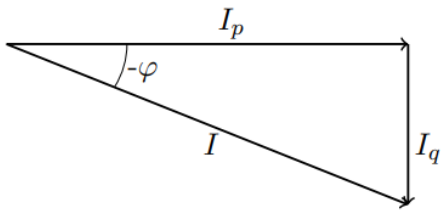
Fonte: Elaboração Própria

A fórmula de Euler afirma que um número complexo A pode ser expresso por (SOLBERG, 2019):

$$A = Re^{j\omega t} = R.cos(\omega.t) + j.R.sin(\omega.t) \tag{20}$$

O número de Euler e é a base do logaritmo natural, j é a unidade imaginária e R é a magnitude de A. As correntes Ip e Iq podem, portanto, ser expressas como a parte imaginária de dois números complexos A1, A2. Tratar as correntes como números complexos torna muito mais fácil somar estes valores porque eles formam um triângulo retângulo no plano complexo, como mostrado na Figura 8. Depois dos cálculos com estes valores, os resultados podem ser convertidos de volta para uma corrente senoidal pelo processo inverso.

Figura 8 – Corrente Expressa Como Números Complexos



Fonte: Solberg (2019)

Na Figura 8, Iq é a corrente circulante por um indutor, usada para gerar o seu campo magnético, e está atrasada em relação à tensão em 90 graus. Ip é a corrente consumida por uma carga resistiva e está em fase com a tensão. A corrente I resultante é uma corrente em atraso, de modo que (SOLBERG, 2019):

$$I = |I|.e^{-j\varphi} \tag{21}$$

Por esta definição, φ é um número real positivo quando a corrente está em atraso em relação à tensão. A corrente resultante da fonte de tensão pode agora ser expressa como (SOLBERG, 2019):

$$I = |I|.e^{-j\varphi} = I_p - jI_q \tag{22}$$

|I| e φ respectivamente são a magnitude e o deslocamento de fase da corrente. A corrente I_q está 90 graus atrás de I_p e, portanto, é multiplicada por -j, que corresponde a 90 graus ($\frac{\pi}{2}$ radianos), em rotação no sentido horário no plano complexo.

Comparando a Equação (20) com a Equação (22), a velocidade angular ω pode ser removida e a corrente pode ser simplesmente ser considerada uma constante complexa. Isso tem a ver com o fato de que na soma de dois sinais sinusoidais síncronos (com a mesma velocidade angular ω), a resultante tem a mesma frequência. Pode-se, portanto, considerar a corrente em algum tempo arbitrário, digamos t = 0, e considerar esta uma constante porque o deslocamento de fase φ é independente do tempo. A corrente complexa resultante também pode ser expressa como |I| $\angle -\varphi$. A amplitude |I| é dada pelo teorema de Pitágoras (SOLBERG, 2019):

$$|I| = \sqrt{I_p^2 + I_q^2} \tag{23}$$

O deslocamento de fase ϕ é calculado por uma relação trigonométrica:

$$\tan\varphi = -\frac{I_q}{I_p} \tag{24}$$

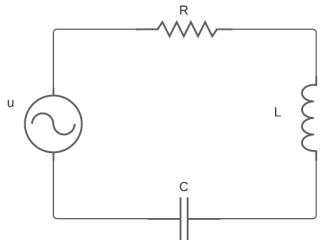
Reatâncias indutivas e capacitivas também são expressas como números complexos.

$$U = I.jX_l \tag{25}$$

$$U = I.(-jX_c) \tag{26}$$

Essa notação complexa também funciona para circuitos mistos resistivos, indutivos e capacitivos, conforme mostrado na Figura 9.

Figura 9 – Circuito RLC Série



Fonte: Elaboração Própria

A impedância Z total do circuito (soma da resistência e reatâncias) é dada como (SOLBERG, 2019):

$$Z = R + jX_l - jX_c \tag{27}$$

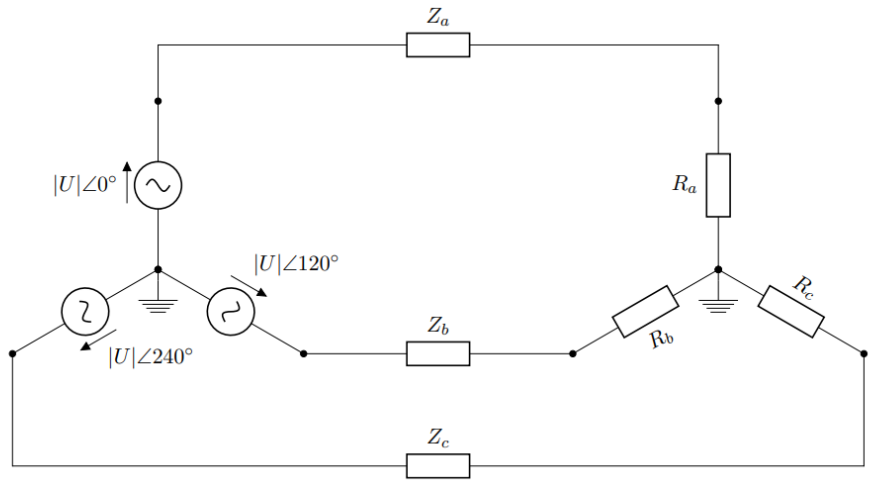
Usando a impedância complexa Z na lei de Ohm, se obtém tanto a amplitude (módulo) resultante |I| e o ângulo de fase da corrente ϕ em um sistema AC. A potência reativa Q que flui pela ligação do circuito também é expressa como um número complexo. A potência aparente S (soma vetorial das potências P e Q) fluindo pelo circuito é definida como:

$$S = U.I* = P + jQ = |S|.e^{j\varphi} \tag{28}$$

U é a tensão, I* é o conjugado complexo da corrente, P e Q são respectivamente a potência ativa e reativa fornecida pela fonte de tensão. De acordo com essa definição, um indutor consome potência reativa enquanto um capacitor é uma fonte reativa. O módulo da potência aparente |S|:

$$|S| = \sqrt{P^2 + Q^2} \tag{29}$$

Figura 10 – Sistema Trifásico de Potência

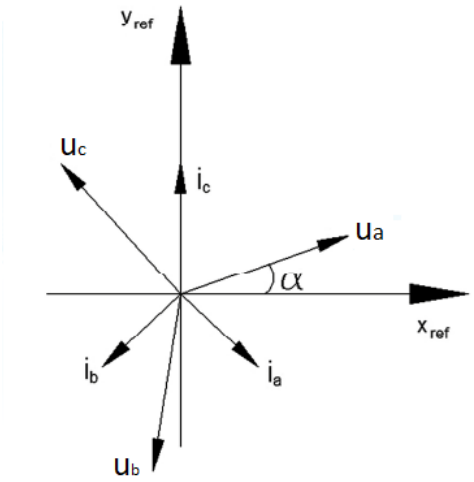


Fonte: Solberg (2019)

Como explicado em Solberg (2019), um sistema de energia elétrica de potência convencional transfere energia em três condutores que possuem a mesma magnitude de tensão senoidal e são defasados 120 graus um em relação ao outro.

Um esquema elétrico trifilar de um sistema de energia trifásico é mostrado na Figura 10. Os três condutores não são normalmente desenhados em ilustrações de uma infraestrutura de sistema de energia, sendo substituído por um diagrama unifilar. Um diagrama unifilar é mais adequado e mais simples para ilustrar o fluxo de potência.

Figura 11 – Representação Fasorial das Tensões e Correntes do Sistema Trifásico



Fonte: Elaboração Própria, baseado em Bichels (2018)

A magnitude da tensão dada em um diagrama unifilar pode ser expressa tanto pela tensão de fase |Uf| que é a tensão relativa entre uma das fases ao terra, ou a tensão entre as linhas |Ul|, que é a tensão relativa entre uma das fases à uma das outras duas. A relação entre estes valores em um sistema trifásico equilibrado é (SOLBERG, 2019):

$$|Ul| = \sqrt{3}.|Uf| \tag{30}$$

A potência aparente |S| transferida em um sistema trifásico é:

$$|S| = \sqrt{3}.|Uf|.|I| \tag{31}$$

|Ull| é o módulo da tensão de linha e |I| é o módulo da corrente em cada condutor. A potência ativa P e a potência reativa Q são determinadas por:

$$P = |S|.cos\varphi \tag{32}$$

$$Q = |S|.sin\varphi \tag{33}$$

φ é o ângulo entre a tensão de fase Uf e a corrente na mesma fase If. Se o sistema é simétrico, então é independente qual fase é usada. Um sistema de energia elétrica geralmente consiste em linhas com diferentes magnitudes de tensão, que pode variar de alguns kV a muitas centenas de kV. Como resultado, é difícil visualizar se o fluxo de potência em uma linha é alto ou baixo, pois deve sempre ser comparado com a tensão desta linha. Para simplificar isso, as quantidades são geralmente medidas em relação aos valores de base de um determinado sistema elétrico. Isto é chamado de sistema por unidade.

Especificamente (SOLBERG, 2019):

$$U = |U_b|.U_{pu} \tag{34}$$

$$I = |I_b|.I_{pu} \tag{35}$$

$$S = |S_b|.S_{pu} \tag{36}$$

$$Z = |Z_b|.Z_{pu} \tag{37}$$

Os índices 'b' e 'pu' denotam valor base e valor 'por unidade', respectivamente. As quantidades por unidade ainda são números complexos, mas são adimensionais.

Uma linha é definida por um valor nominal para potência aparente |Sb| e módulo de tensão |Ub|. A corrente base |Ib| é definido como:

$$|I_b| = \frac{|S_b|}{\sqrt{3}.|U_b|} \tag{38}$$

Por esta fórmula, a potência aparente S_{pu} é:

$$S_{pu} = \frac{S}{|S_b|} = \frac{\sqrt{3}.U_f.I}{\sqrt{3}.|U_b|.|I_b|} = U_{pu}.I_{pu} \tag{39}$$

Essa demonstração faz com que o cálculo da potência aparente tome a forma de um sistema monofásico, embora esteja sendo aplicado a um sistema trifásico. Da mesma forma, a definição do valor por unidade base de impedância |Zb|:

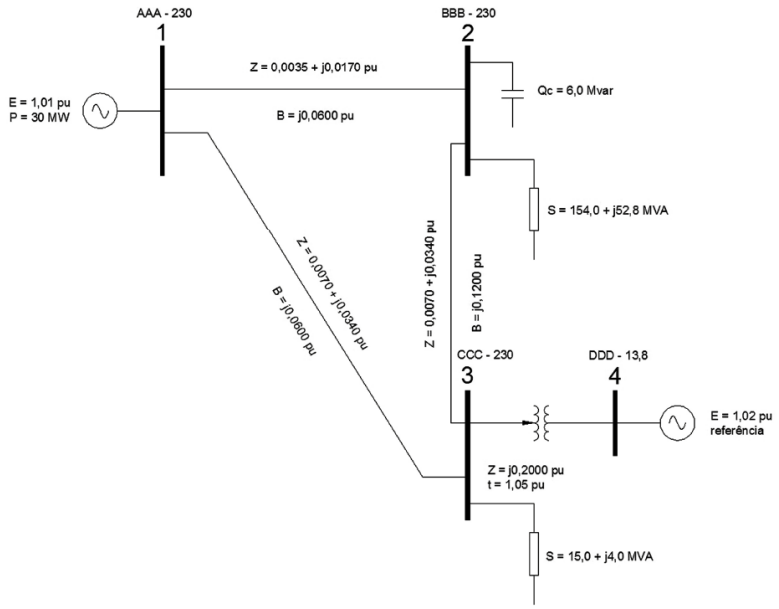
$$|Z_b| = \frac{|U_b|}{\sqrt{3}.|I_b|} \tag{40}$$

Por esta fórmula, a tensão U_{pu} é:

$$Upu = \frac{U}{|U_b|} = \frac{\sqrt{3}.Z.I}{\sqrt{3}.|Z_b|.|I_b|} = Z_{pu}.I_{pu} \tag{41}$$

A notação PU resulta na mesma relação entre corrente, tensão e impedância dada pela lei de Ohm em um sistema monofásico. Conforme Bichels (2018), um sistema elétrico de potência (SEP) consiste em um conjunto de nós N, comumente chamados de barramentos, e um conjunto de ramais L que conectam alguns ou todos os barramentos. Os ramais entre barramentos podem ser linhas de energia, cabos, transformadores ou outros equipamentos. Os barramentos e ramais definem a topologia do sistema elétrico de potência. A Figura 12 é uma ilustração de um sistema de energia elétrica composto por 4 barramentos e 4 ramais. Os ramais são representações de linhas de um sistema trifásico. A convenção brasileira de apresentação destes diagramas também usa E ou V para a tensão.

Figura 12 – Diagrama com Dados Elétricos de um SEP



Fonte: Bichels (2018)

Formalmente, essa rede é descrita como (BICHELS, 2018):

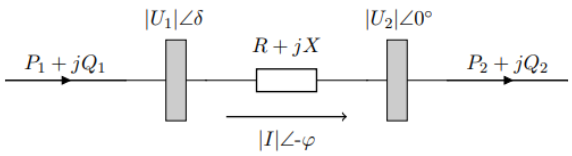
$$N = 1, 2, 3, 4 \tag{42}$$

$$L = (1, 2),(1, 3),(2, 3),(3, 4) \tag{43}$$

Por (SOLBERG, 2019), um barramento é modelado eletricamente como um ponto onde a energia elétrica é injetada. Uma potência injetada positiva corresponde à geração de potência naquele barramento, por exemplo um barramento que está conectado a uma usina. Uma potência injetada negativa corresponde a consumo de energia, como seria o caso de um barramento conectado a uma fábrica de grande capacidade. A soma da produção e do consumo de energia determina a potência líquida injeção nesse barramento. Uma barra k em um sistema elétrico de potência é descrita fisicamente por quatro grandezas: A magnitude da tensão $|U_k|$ ($|E_k|$ ou $|V_k|$ também são utilizados), o ângulo de tensão ϕ_k, a injeção de potência ativa P_k e a injeção de potência reativa Q_k.

A Figura 13 mostra o fluxo de energia entre os barramentos conectados por uma linha de transmissão. Trata-se de um sistema simples de dois barramentos conectado por uma linha. P e Q são o ativo e o reativo fluxo de potência, R e X são a resistência e reatância da linha, U é a tensão e I é a corrente fluindo. A partir de à esquerda temos a potência ativa P_1 e a potência reativa Q_1 fluindo.

Figura 13 – Diagrama com Dados Elétricos de um SEP



Fonte: Solberg (2019)

A potência flui através da linha representada e continua a partir do barramento 2 como P_2 e Q_2. U_1 é a representação complexa da tensão na barra 1 (SOLBERG, 2019):

$$U_1 = |U_1|.e^{j\varphi} \tag{44}$$

Da mesma forma:

$$U_2 = |U_2|.e^{j0} = |U_2| \tag{45}$$

A relação entre tensão e corrente para o sistema é dada pela lei de Ohm:

$$U_1 - U_2 = (R + jX).I \tag{46}$$

R é a resistência e X é a reatância da linha, U_1 e U_2 são as tensões no barramento 1 e 2, e I é a corrente que flui na linha. A impedância Z da linha pode ser expresso como:

$$Z = |Z|.e^{j\epsilon} \tag{47}$$

$$tan(\epsilon) = \frac{X}{R} \tag{48}$$

A corrente I é geralmente definida em atraso (majoritariamente impedâncias indutivas nas cargas e modelos), de modo que:

$$I = |I|.e^{-j\varphi} \tag{49}$$

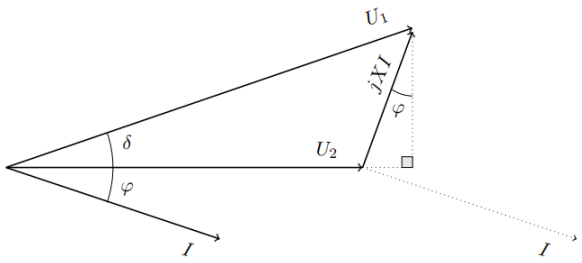
Assim, φ é um número real positivo quando a corrente está atrasada em relação à tensão. Na Figura 14, a corrente, as tensões e a impedância são desenhado como fasores no plano complexo para uma linha de transmissão com resistência zero (R = 0).

$$U_1 - U_2 = (R + jX)I \tag{50}$$

Finalmente, usando a lei de Ohm, a corrente I na linha pode ser expressa como:

$$I = \frac{U_1 - U_2}{Z} = \frac{|U_1|.e^{j\delta} - U_2}{|Z|.e^{j\epsilon}} = \frac{|U_1|.e^{j(\delta-\epsilon)} - U_2.e^{-j\epsilon}}{|Z|} \tag{51}$$

Figura 14 – Fasores de Corrente e Tensão em Uma Rede de Dois Barramentos Conectados por Linha com R=0



Fonte: Solberg (2019)

$|U_1|$ e $|U_2|$ são as tensões no barramento 1 e 2 respectivamente, I é a corrente que flui na linha entre 1 e 2, e |Z| é a impedância da linha. Usando a definição de potência da Equação (4) no barramento 1 temos a potência ativa P1:

$$P_1 = \frac{|U_1|^2}{Z}.cos(\epsilon) - \frac{|U_1|.|U_2|}{|Z|}.cos(\epsilon - \delta) \tag{52}$$

Para facilitar, com perdas resistivas diferentes de zero na linha (R $\neq$ 0), é conveniente usar o ângulo de perda α (SOLBERG, 2019):

$$\alpha = tan(\frac{R}{X}) \tag{53}$$

Pela soma dos ângulos de um triângulo, temos a relação:

$$\alpha + \epsilon = \frac{\pi}{2} \tag{54}$$

Usando o ângulo de perda α, o fluxo de potência ativa na barra 1 (P_1) pode ser expresso como:

$$P1 = \frac{|U_1|^2}{Z}.sen(\alpha) + \frac{|U_1|.|U_2|}{|Z|}.sen(\delta - \alpha) \tag{55}$$

Uma linha sem perdas resistivas é definida com $\alpha = 0$. O resultado fluxo de potência ativa P_1 no barramento 1 se reduz para:

$$P_1 = \frac{|U_1|.|U_2|}{|Z|}.sen(\delta) \tag{56}$$

$|U_1|$ e $|U_2|$ são as tensões nos barramentos 1 e 2, |Z| é a impedância da linha e δ é o ângulo de fase entre a tensão do barramento 1 em relação ao barramento 2. A direção do fluxo de potência ativa é determinada pelo ângulo de fase δ das tensões entre as barras. Em um sistema de dois barramentos, o barramento com a tensão principal está fornecendo energia ativa, enquanto a tensão atrasada está recebendo energia. Para um sistema de energia AC, o valor do módulo da tensão entre o os barramentos podem diferir numericamente e a potência ativa ainda pode fluir em ambas as direções.

De maneira análoga o fluxo de potência reativa Q1 na barra 1 para uma linha sem perdas resistivas é dada por (SOLBERG, 2019):

$$Q_1 = \frac{|U_1|}{|Z|}.(|U_1| - |U_2|) \tag{57}$$

|U1| e |U2| são as tensões nos barramentos 1 e 2, |Z| é a impedância da linha. O ângulo de tensão δ é assumido como pequeno de modo que cos(δ) $\approx$ 1. Pela equação Equação (57) a direção da potência reativa é determinada pela diferença das magnitudes de tensão. Em outras palavras, o barramento com maior tensão em módulo está fornecendo potência reativa em um sistema de dois barramentos.

Conforme Bichels (2018), em um SEP, existe um conjunto de cargas (potências) características dos consumidores, que são representadas no modelo por impedâncias em derivação para a terra em determinados pontos do sistema (nas barras ou nos nós da malhas). Estas cargas devem ser supridas por usinas, geralmente situadas em pontos do sistema distantes das cargas (exceção: geração própria), sendo que entre as barras de geração e de carga existem as linhas de transmissão e os transformadores de subestações do sistema, que formam os ramos da malha do SEP e constituem as impedâncias série do modelo do sistema.

Além destes componentes, têm-se ainda outras impedâncias em derivação para a terra, constituídas por reatores e bancos de capacitores para compensação de queda de tensão, reativos, etc.

$$\sum P_{gerada} = \sum P_{carga} + \sum P_{perdas}$$

Embora as cargas sejam relativamente constantes e a configuração do sistema seja única, a potência gerada em cada usina (a potência despachada), bem como a tensão terminal dos geradores, pode variar dentro de uma faixa ampla, dando origem a uma infinidade de respostas ao problema de fluxo de potência de um SEP que atendem à condição acima. Para cada conjunto de condições de potências geradas e tensões dos nós, há um conjunto de tensões nos demais nós e também um conjunto de fluxos no sistema, e, consequentemente, também de diferentes perdas.

Considerando-se todas as variáveis envolvidas, têm-se os elementos conhecidos:

- a potência ativa e reativa das cargas (valor constante),
- a configuração (topologia) do sistema,
- a potência, ativa e reativa, que cada usina (ou máquina) pode gerar,
- a tensão nos terminais em cada usina.

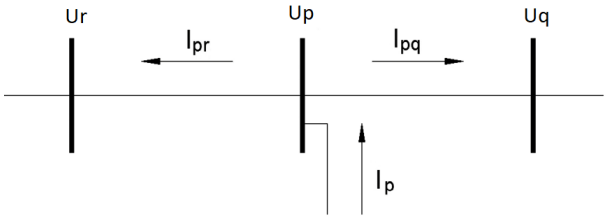
Como elementos desconhecidos, temos:

- a tensão em cada barra de carga,
- o fluxo de potência em cada linha ou transformador,
- as perdas no sistema.

As equações utilizadas para a solução de problemas de fluxo de potência são obtidas com base na aplicação das leis de Kirchhoff aos nós e às malhas do sistema (Equação (5) e Equação (7)). Se o sistema estiver em regime permanente, com tensões senoidais e na frequência nominal do sistema, a soma algébrica das correntes em cada nó deve ser nula (BICHELS, 2018). A solução do sistema das equações escritas para cada nó fornecerá a tensão em cada nó. Com as tensões calculadas e com as admitâncias dos ramos do sistema, calcula-se, então, a corrente dos ramos. Na prática, como se trabalha com a potência dos geradores e das cargas, é melhor trabalhar com a potência que flui nos ramos (linhas e transformadores) e, por isso, calculam-se as potências com estas correntes e as tensões dos barramentos – daí o nome fluxo de potência, ou também fluxo de carga (em inglês *load flow*) para este processo.

Por exemplo, considerando um trecho de sistema com uma barra (nó) na qual um gerador injeta potência sob determinada tensão e à qual estão conectadas duas linhas de transmissão, ligando outras duas barras, conforme a Figura 15.

Figura 15 – Trecho de Sistema Elétrico Para Exemplo



Fonte: Elaboração Própria, baseado em Bichels (2018)

Aplicando a primeira Lei de Kirchhoff (Equação (5)) à barra P temos (BICHELS, 2018):

$$i_p + i_{pq} + i_{pr} = 0$$

Onde ip é a corrente da barra (nó) e i_{pq} e i_{pr} são as correntes dos ramos (neste caso são os fasores, ou seja os números complexos na forma a+jb ou $|M|\angle\theta$).

A aplicação do método em um sistema genérico implica em montar as equações das correntes nos nós do sistema na forma de matrizes. Considerando-se um sistema de n nós, temos a seguinte equação, que relaciona as matrizes de correntes e tensões nodais por meio da matriz de admitâncias nodais (BICHELS, 2018):

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ . \\ i_p \\ . \\ i_n \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15}. & y_{1p} & . & y_{1n} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25}. & y_{2p} & . & y_{2n} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35}. & y_{3p} & . & y_{3n} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45}. & y_{4p} & . & y_{4n} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55}. & y_{5p} & . & y_{5n} \\ . & & & & & & & \\ y_{p1} & y_{p2} & y_{p3} & y_{p4} & y_{p5}. & y_{pp} & . & y_{pn} \\ . & & & & & & & \\ y_{n1} & y_{n2} & y_{n3} & y_{n4} & y_{n5}. & y_{np} & . & y_{nn} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ . \\ u_p \\ . \\ u_n \end{bmatrix} \tag{58}$$

Resumidamente, o vetor de correntes dos nós é igual ao produto da matriz de admitâncias nodais (matriz y_{barra}) pelo vetor das tensões nos nós.

$$\left[i\right] = \left[y\right] \times \left[u\right] \tag{59}$$

$$sp = u_p.i_p^* \therefore i_p = \frac{s_p^*}{u_p^*} \tag{60}$$

onde i_p^* , sp^* , up^* são os conjugados dos números complexos das grandezas. Aplicando a Equação (59) e generalizando para qualquer nó *p* (BICHELS, 2018):

$$u_p = \frac{\frac{s_p^*}{u_p^*} - \sum_{p=1,p\neq q}^{n} y_{pq}.u_q}{y_{pp}} \tag{61}$$

Considerando que um nó de um circuito elétrico é um ponto de conexão de linhas, geradores, transformadores e cargas, este nó nada mais é do que o barramento de uma subestação, daí a nomenclatura de sistemas elétricos utilizar o termo barra. A Equação (62) indica que, para um sistema de n barras, são necessárias n equações semelhantes à Equação (61).

Na prática, uma das barras é usada como referência, considerando-se esta tensão como conhecida. Então, o somatório das potências geradas deve ser igual ao somatório das potências das cargas mais o total das perdas, mas, como as perdas são função do quadrado da corrente e como a corrente nos ramos é uma das incógnitas do problema, o que se pode fazer é fixar a potência das cargas e das gerações, exceto uma, a qual fornecerá a diferença de potência necessária para que a somatória das cargas e perdas seja igual à geração. A tensão desta barra específica será a referência para os cálculos. Ela é chamada de barra de referência ou barra *swing* (do inglês *swing bus* ou barra oscilante) (BICHELS, 2018).

A Equação (59) relaciona correntes e tensões por meio da matriz de admitâncias nodais do sistema. Nesta equação tem-se como conhecida a matriz [y] e como incógnitas as tensões e as correntes nas barras. A Equação (61) é obtida para cada barra, resultante do produto indicado pela Equação (59). Decomposta em suas partes real e imaginária, isto resulta em um sistema de duas equações para cada barra, com quatro incógnitas.

Neste caso, para que seja possível obter uma solução para o sistema de equações, é obrigatório conhecer duas das incógnitas de cada barra, resultando em um número de incógnitas igual ao número de equações. As variáveis associadas a cada barra do sistema são: potência ativa e reativa (corrente na Equação (60)), módulo e ângulo da tensão. Assim, considerando as variáveis de cada barra, podem ser encontrados três tipos de barra nos SEP:

- barras PQ: barra de carga – onde são conhecidas as potências ativa e reativa das cargas e são desconhecidos o módulo e o ângulo da tensão;
- barras PV: barra de geração – onde são conhecidas a potência ativa e o módulo da tensão e são desconhecidas a potência reativa e o ângulo da tensão;
- barras Vθ: barra de referência – onde são conhecidos o módulo e o ângulo da tensão e são desconhecidas a potência ativa e a potência reativa.

As equações para obtenção da tensão das barras, apesar de serem algébricas, não são lineares. Não é possível utilizar os métodos diretos de solução de circuitos elétricos, sendo necessário recorrer a métodos iterativos. A não linearidade das equações se deve ao fato de que tanto os geradores como as cargas são modelados como potência constante e não como tensão constante ou impedância constante, como é considerado na análise de circuitos elétricos (BICHELS, 2018). Se for considerada a potência constante em vez de impedância constante, isso implica em não linearidade para o problema. Também implica em que as cargas e as gerações são especificadas, mas as perdas são desconhecidas, sendo que a Equação (58) deve ser atendida. Uma solução para este problema é especificar a potência ativa de todos os geradores, exceto um e, neste gerador, especificar o módulo da tensão. Neste caso, é feita a mudança das variáveis desconhecidas nesta barra de Qθ para PQ. Após a solução do problema, com a determinação das tensões em todas as barras, calcula-se a corrente nos ramos (linhas e transformadores) e a potência ativa e reativa na barra de referência, de forma a atender que o somatório das potencias geradas seja igual às cargas somadas com as perdas, determinando estas últimas. Duas caraterísticas deste tipo de resolução de problemas são:

- Com as soluções através de métodos iterativos, não há uma solução exata para estes cálculos, mas sim uma solução com um erro menor do que um valor muito pequeno (tolerância), especificado a priori.
- Métodos iterativos para a solução de equações podem ser convergentes ou divergentes, mas no caso da aplicação a sistemas elétricos, geralmente obtém-se convergência.

Dentre os métodos iterativos que podem ser usados, estão: Gauss, Gauss-Seidel, Newton-Raphson, matriz Z_{barra} , desacoplado rápido e DC (corrente contínua). Alguns destes métodos já não são usados na prática, como os métodos de Gauss e de Gauss-Seidel. Atualmente, o método Newton-Raphson se comporta melhor para cálculo de grandes sistemas.

Conforme Bichels (2018), o método de Newton-Raphson para a solução de fluxo de potência se baseia na expansão em série de Taylor de uma função de duas ou mais variáveis.

Vantagens:

- convergência rápida, desde que a estimativa inicial esteja próxima de solução,
- grande região de convergência.

Desvantagens:

- cada iteração leva muito mais tempo de cálculo do que uma iteração Gauss-Seidel,
- mais complicado de codificar.

Por Nirupama (2018), no cálculo de fluxo de potência através do método Newton-Raphson é necessário determinar a magnitude da tensão e o ângulo de fase em cada barramento do sistema que satisfaça o equilíbrio de potência. É necessário resolver uma equação de equilíbrio de potência.

Sejam as variáveis desconhecidas (x_1, x_2, x_3 ..., x_n) e as quantidades especificadas (y_1, y_2, y_34, ...,y_n). Estes estão relacionados pelo conjunto de equações não lineares:

$f_1(x_1, x_2, x_3 \ldots, x_n) = y_1$
$f_2(x_1, x_2, x_3 \ldots, x_n) = y_2$
$f_3(x_1, x_2, x_3 \ldots, x_n) = y_3$
...
$f_n(x_1, x_2, x_3 \ldots, x_n) = y_n$

A solução das equações acima é iniciada com uma solução aproximada:
(x_1^0, x_2^0, x_3^0 ..., x_n^0).

O índice zero sobrescrito é a definição de que esta é a iteração zero no processo de resolver equação. Este "palpite" para a solução das equações não lineares não deve estar muito longe da solução real, caso contrário há chance de a solução divergir ao invés de convergir e pode não ser possível ter uma solução qualquer que seja o tempo do computador. À primeira vista pode parecer uma desvantagem para o método, mas o problema da estimativa inicial para um sistema de energia não é difícil. Um perfil de tensão plano, ou seja, u_i = (1,0 + j0) para i = 1, 2, 3 ... n, exceto o barramento *swing* é considerado satisfatório para quase todos os sistemas práticos.

Sejam Δx_1^0, Δx_2^0, Δx_3^0, ... Δx_n^0 as correções que, somadas aos valores iniciais assumidos, dão a solução real. Então:

$f_1(x_1^0 + \Delta x_1^0, x_2^0 + \Delta x_2^0 , x_3^0 + \Delta x_3^0 \ldots ,x_n^0 + \Delta x_n^0) = y_1$

Expandindo essas equações na série de Taylor em torno da estimativa inicial, temos (NIRUPAMA, 2018):

$f_1(x_1^0,\ x_2^0,\ x_3^0\ \ldots\ ,x_n^0) + (\Delta x_1^0.(\frac{\partial f_1}{\partial x_1})^0 + \Delta x_2^0.(\frac{\partial f_1}{\partial x_2})^0 + \Delta x_3^0.(\frac{\partial f_1}{\partial x_3})^0 + \ldots + \Delta x_n^0.(\frac{\partial f_1}{\partial x_n})^0) +$ termos de ordem superior ...

Onde $(\frac{\partial f_1}{\partial x_1})^0$, $(\frac{\partial f_1}{\partial x_2})^0$, $(\frac{\partial f_1}{\partial x_3})^0$, ... $(\frac{\partial f_1}{\partial x_n})^0$ são as derivadas parciais de f_1 em relação à x_1, x_2, x_3, ..., x_n calculadas em $(x_1^0,\ x_2^0,\ x_3^0\ \ldots,\ x_n^0)$.

Derivadas parciais de segunda ordem e superiores são desprezadas de acordo com o método, por isso há a exigência de que a solução inicial esteja próxima da solução final. Linearizando as equações e organizando na forma matricial, temos:

$$\begin{bmatrix} y_1 - f_1(x_1^0, x_2^0, x_3^0 ..., x_n^0) \\ y_2 - f_2(x_1^0, x_2^0, x_3^0 ..., x_n^0) \\ y_3 - f_3(x_1^0, x_2^0, x_3^0 ..., x_n^0) \\ \cdot \\ y_n - f_n(x_1^0, x_2^0, x_3^0 ..., x_n^0) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \ldots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \ldots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \ldots & \frac{\partial f_3}{\partial x_n} \\ \ldots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} \ldots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \times \begin{bmatrix} \Delta x_1^0 \\ \Delta x_2^0 \\ \Delta x_3^0 \\ \cdot \\ \Delta x_n^0 \end{bmatrix} \quad (62)$$

Esta equação vetorial pode ser resumida como $B = J \times C$. J é a matriz quadrada das derivadas parciais e é chamada de matriz Jacobiana. A solução das equações precisa do cálculo do vetor B, que é a diferença entre as quantidades especificadas e as quantidades calculadas em $(x_1^0,\ x_2^0,\ x_3^0\ \ldots,\ x_n^0)$. Da mesma forma, J é calculado com este "palpite". A solução da equação da matriz fornece $(\Delta x_1^0,\ \Delta x_2^0,\ \Delta x_3^0,\ \ldots\ \Delta x_n^0)$ e as melhores estimativas da solução são dadas por:

$x_1^1 = x_1^0 + \Delta x_1^0$
$x_2^1 = x_2^0 + \Delta x_2^0$
$x_3^1 = x_3^0 + \Delta x_3^0$
...
$x_n^1 = x_n^0 + \Delta x_n^0$

Repetindo o processo de iterações com esses valores obtemos valores estimados ainda melhores. A diferença $(\Delta x_1,\ \Delta x_2,\ \Delta x_3,\ \ldots\ \Delta x_n)$ torna-se cada vez menor a cada iteração e, finalmente, o processo de iteração é interrompido quando $(\Delta x_1,\ \Delta x_2,\ \Delta x_3,\ \ldots\ \Delta x_n)$ são menores que o valor de tolerância pré estabelecido (NIRUPAMA, 2018). Este método pode ser aplicado a problemas de fluxo de potência de várias maneiras, sendo as mais comuns aquelas que usam coordenadas retangulares $(P_i - jQ_i)$ ou coordenadas polares $(|V_i|\angle\delta)$. Para simplificar o método, é reconhecido que o fluxo de potência reativa Q não é muito afetado pela mudança no ângulo de fase δ e, de maneira similar, o fluxo de potência ativa permanece quase inalterado pela variação na magnitude da tensão nodal (ΔV).

A partir destas premissas e usando as coordenadas polares, o conjunto de equações de fluxo de potência linear pode ser escrito em forma de matriz da seguinte forma (NIRUPAMA, 2018):

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \times \begin{bmatrix} \Delta \delta \\ \Delta V \end{bmatrix} \tag{63}$$

J1 corresponde a elementos $\partial \mathrm{P}/\partial\delta$ que existem; J2 corresponde aos elementos $\partial \mathrm{P}/\partial\delta$ que não existem e é igual a zero. J3 corresponde aos elementos $\partial \mathrm{Q}/\partial\delta$ que não existem e, portanto, é zero. J4 corresponde aos elementos $\partial \mathrm{Q}/\partial\delta$ que existem. Isso certamente os cálculos e resulta em um tempo menor de processamento:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & 0 \\ 0 & J_4 \end{bmatrix} \times \begin{bmatrix} \Delta \delta \\ \Delta V \end{bmatrix} \tag{64}$$

Para o barramento controlado por tensão ou barramento de geração, P e a magnitude da tensão V são dadas. A potência P para qualquer barra i é dada como:

$$P_i = Real[V_i^*.\sum_{k=1}^{n} V_k.Y_{ik}] \tag{65}$$

Também para i^{th} barramento, temos:

$$V_i^2 = e_i^2 + f_i^2 \tag{66}$$

Onde V_i é a magnitude da tensão e e_i e f_i são suas componentes real e imaginária. As equações da matriz relacionando as variações nas potências do barramento e o quadrado da magnitude da tensão do barramento, com as mudanças nas componentes real e imaginária da tensão são dadas por:

$$\begin{bmatrix} \Delta P \\ \Delta Q \\ \Delta V_i^2 \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \\ J_5 & J_6 \end{bmatrix} \times \begin{bmatrix} \Delta e \\ ... \\ \Delta f \end{bmatrix} \tag{67}$$

Onde $\Delta(V_i^r)^2 = (V_{iespecificado})^2 - (V_i^r)^2$, e V_i^r é a tensão de barramento calculada após a r-ésima iteração.

Os elementos da matriz jacobiana são calculados usando (NIRUPAMA, 2018):

J_5:
$\frac{\partial(V_i)^2}{e_k} = 0$ para $k \neq i$
$\frac{\partial(V_i)^2}{e_i} = 2.e_i$

J_6:
$\frac{\partial(V_i)^2}{f_k} = 0$ para $k \neq i$
$\frac{\partial(V_i)^2}{f_i} = 2.f_i$

Neste caso V_i^r é a tensão do barramento calculada na r-ésima iteração e $V_{iespecificada}$ é a tensão especificada em qualquer barra i. Após a obtenção das tensões das barras, o fluxo de potência e as perdas nas linhas são calculados usando as considerações a seguir:

- O fluxo de potência na linha i-k na barra i é dado como:
$S_{ik} = P_{ik} + jQ_{ik} = V_i.I_{ik}^* = V_i.(V_i^* - V_k^*).Y_{ik}^* + V_i.V_i^*.Y_{ik0}$

- Igualmente, o fluxo de potência na linha i-k na barra k é dado como (NIRUPAMA, 2018):
$S_{ki} = V_k.(V_k^* - V_i^*).Y_{ik}^* + V_k.V_k^*.Y_{ki0}$

Depois dos cálculos de fluxo de potência estabelecidos, é possível utilizar técnicas mais avançadas para predição de demanda em cenários futuros, tendo como exemplo o trabalho de Solberg (2019). O objetivo deste estudo visava avaliar estratégias de resposta à demanda, que pode ser resumidamente definida como pagamentos de incentivos destinados a induzir um menor uso de eletricidade em momentos de preços elevados do mercado de energia ou quando a confiabilidade do sistema atinge uma criticalidade máxima aceitável nos períodos de maior trânsito de energia, para o valor da potência de pico na rede seja reduzido. Um incentivo financeiro poderia ser dado na conta de energia elétrica para estimular certas classes de consumidores a mudarem seu padrão de consumo, ao invés de investir em infra estrutura de alta capacidade apenas para atendimento de um curto espaço de tempo do horário de pico do sistema. Na execução deste processo de estudo em Solberg (2019), foi criado um vetor de ação onde cada componente determina uma variação percentual no consumo de energia. Num primeiro passo, o consumo e a produção de energia nos nós da rede são alterados de acordo com a previsão de demanda e de geração renovável sazonal, como solar ou eólica. O vetor de ação é então processado pelo algoritmos de Aprendizado por Reforço e atualiza o consumo de energia na rede de cálculo do pandapower. As equações de fluxo de potência para a rede são novamente resolvidas, e uma vez que as novas magnitudes de tensão e corrente em a rede são determinadas, a recompensa pode ser calculada conforme a função estabelecida.

Isso resume as etapas envolvidas pela ação no algoritmo de Aprendizado por Reforço.

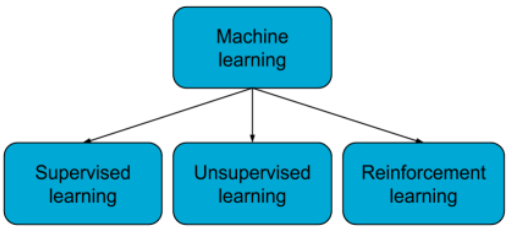
2.2 Inteligência Artificial (AI)

Inteligência Artificial é um programa para uma máquina executar determinadas tarefas que um ser humano poderia aprender a executar (TEOH; RONG, 2022). Esse "aprendizado de determinadas tarefas" geralmente significa o uso de matemática para criar um algoritmo. Porém, algumas destas tarefas, que a maioria das pessoas realizariam com facilidade (por exemplo, reconhecer um rosto na multidão ou reconhecer uma música tocando no rádio) não podem ser explicadas matematicamente de uma maneira fácil por quem executa esta tarefa. Essas pessoas foram capazes de chegar a esses resultados porque elas aprenderam a fazer essas coisas ao longo do tempo. Este aprendizado nas pessoas é executado por meio de repetições da atividade, processando informações e dados fornecidos por quem a está treinando ou pela observação dos resultados das escolhas efetuadas.

A Inteligência Artificial foi desenvolvida para simular o aprendizado humano em um computador. Durante o início das pesquisas em AI, primeiramente foi tentado o desenvolvimento de algoritmos para tentar aproximar a intuição humana. Estes códigos podiam ser vistos como um enorme conjunto de instruções if/else produzindo a resposta desejada. Porém, esta acabou sendo uma abordagem incrivelmente ineficiente devido à complexidade da mente humana. Estas regras if/else são muito rígidas e vão se tornando obsoletas à medida que circunstâncias mudam. Em vez de tentar programar uma máquina para agir como um cérebro, por que não apenas a ensinamos com um grande conjunto de dados para que esta máquina possa descobrir o melhor algoritmo por conta própria? Elaborado dessa maneira, um algoritmo de aprendizado de máquina vem a ser o motor de uma parte dos sistemas atuais de AI.

Por outro lado, a qualidade dos dados utilizados para ensinar a máquina deve ter padrões mínimos de qualidade para poder criar bons programas de inteligência artificial. Os profissionais responsáveis por estes dados (Engenheiros ou Cientistas de Dados) devem ter conhecimento do problema e das variáveis envolvidas no processo para a seleção e processamento do material a ser utilizado como dados de treinamento do sistema. Essa tarefa pode se tornar bastante complexa dependo do processo e do conjunto de dados a ser trabalhado.

Figura 16 – As Três Principais Categorias de Aprendizado de Máquina: Aprendizado Supervisionado, Não Supervisionado e Reinforcement Learning



Fonte: Solberg (2019)

Existem três categorias principais de Aprendizado de Máquina, conforme mostrado na Figura 16. O aprendizado supervisionado (*Supervised Learning*) é aquele no qual há interferência humana. Basicamente, uma pessoa fornece um banco de dados e ensina a máquina a encontrar a resposta desejada de acordo com a necessidade. A aprendizagem não supervisionada (*Unsupervised Learning*) é aquela na qual não acontece interferência humana, ou seja, a máquina cria suas próprias regras de funcionamento com base na identificação de padrões. Por fim, no aprendizado por reforço (*Reinforcement Learning*), a máquina opera baseada em experiência. Nesse processo, há uma penalização para o erro ou uma premiação para o acerto. Depois deste passo, a máquina procura a abordagem correta para corrigir e não repetir um erro novamente. Este processo de aprendizado por reforço não acontece apenas uma vez, na verdade a máquina pode tentar diferentes soluções até encontrar a mais adequada para o problema que está lidando.

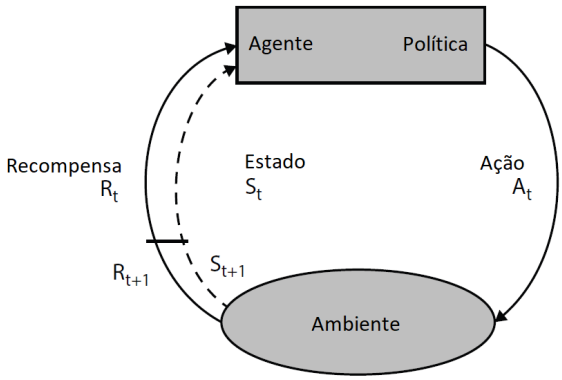
2.3 Aprendizado por Reforço (Reinforcement Learning - RL)

O método de Aprendizado por Reforço consiste no desenvolvimento de um algoritmo capaz de alcançar um objetivo particular por meio de recompensa e punição. O algoritmo terá que balancear entre busca de novas ações e uso de ações previamente conhecidas no processo de aprendizagem (TEOH; RONG, 2022). O processo de RL usa observações coletadas da interação do algoritmo com o ambiente para realizar ações que maximizam a recompensa ou minimizam o risco (MOHAMMED; KHAN; BASHIER, 2016). Para produzir programas inteligentes (também chamados de agentes), o método RL passa pelas seguintes etapas conforme mostrado na Figura 17:

- O estado de entrada é observado pelo agente.
- A função de tomada de decisão é usada para fazer o agente realizar uma ação.
- Após a execução da ação, o agente recebe recompensa ou punição do meio ambiente.
- As informações do par estado-ação sobre a recompensa são armazenadas.
- Usando as informações armazenadas, a política para um estado específico em termos de ação pode ser refinada, ajudando na tomada de decisão ideal para o agente.

A principal diferença entre o Aprendizado por Reforço e outros métodos de Aprendizado de Máquina é que neste caso o agente nunca é informado qual é o caminho correto a seguir em cada situação. Apenas o critério de premiar ou punir é entendido pelo Agente e isso é o que expressa quão boa ou ruim é a sua ação. É tarefa do Agente aprender o que é melhor em cada situação com base nesta informação. Isso faz parte dos pontos fortes do método. Problemas complexos de tomada de decisão podem muitas vezes ser resolvidos fornecendo a menor quantidade de informação necessária para resolver o problema (DULHARE UMA N.; AHMAD, 2020).

Figura 17 – Componentes do Método de Aprendizado por Reforço



Fonte: Elaboração Própria, baseado em Bhatia et al. (2020)

Em muitos animais, o aprendizado por reforço é a única maneira de aprender e também é uma parte essencial do comportamento humano. Quando nossas mãos queimam no calor, aprendemos rapidamente a não repita isso novamente. Recompensa e punição são bons exemplos das retribuições por nossas ações e que compõem nossos padrões de comportamento e o de muitos animais. Em reforço aprendizagem, o objetivo principal da desta é fazer algo ou alcançar um objetivo sem ser alimentado por informações externas diretas (DULHARE UMA N.; AHMAD, 2020). Detalhando melhor cada parte do processo:

- Política: Determina como você lida com cada ação e como você decide em cada uma das diferentes situações. A Política de fato determina como o Agente se comporta em um determinado tempo e orienta o Agente inteligente para melhorar os modos de resolução de problema.
- Função de Recompensa: especifica o alvo na função do Agente. Esta função tem o objetivo de recompensar um Agente para cada Ação, de modo que ao se aproximar do Alvo aumenta a Recompensa. A Recompensa é uma resposta de curto prazo, que deixa o Agente mais perto da sua meta.
- Função de Valor: Para cada estado, uma quantidade definida que quanto mais o Alvo mais perto, como se estivesse em um jogo.
- Modelo: O problema do aprendizado por reforço é probabilístico e estocástico e os Estados são não determinísticos; isto é, para uma Ação, ele pode ir para todos os Estados, mas com uma probabilidade. Cada Ação é uma possibilidade, e é possível passar de um Estado para outro.

O objetivo do Agente é maximizar as recompensas de longo prazo. Em um problema de Aprendizado por Reforço, estamos diante de um fator que interage com o meio por tentativa e erro e aprende a selecionar a ação ideal para atingir o objetivo (DULHARE UMA N.; AHMAD, 2020).

A aprendizagem por RL é, portanto, uma forma de treinar Agentes realizando uma ação por meio de recompensa e punição sem ter que especificar como o Agente executa a ação. Alguns pontos a serem observados (DULHARE UMA N.; AHMAD, 2020):

- Um Algoritmo Guloso na busca de recompensas nem sempre é bom. Há coisas que são fáceis de fazer para gratificação instantânea, e há coisas que oferecem recompensas de longo prazo. O objetivo é não ser ganancioso procurando o recompensas imediatas e rápidas, mas sim otimizar para obter o máximo de recompensas em todo o treino.
- A sequência é importante no aprendizado por reforço
- O Agente de Recompensa não depende apenas do estado atual, mas de todo o história dos estados. Ao contrário do aprendizado supervisionado, a passagem de tempo e sequência de estado-ação-recompensa é importante neste modelo.

Como nota, um Algoritmo Guloso é uma solução comum para problemas de otimização, onde é realizada uma escolha que parece ser a melhor no momento, na esperança de que a mesma seja uma solução em nível global. Porém, ele é míope: ele toma decisões com base nas informações disponíveis na iteração corrente, jamais se arrepende de uma decisão, e não leva em consideração as consequências de suas decisões. Por isso, nem sempre produz a melhor solução (CORMEN et al., 2001).

2.4 Exploração de Dados: Projeto SimBench

O conjunto de dados do SimBench, está disponível de forma pública na área "Download" do site do projeto (<https://simbench.de/en/download/>). Os métodos usados para gerar os dados para o nível de Extra Alta Tensão (EHV) e Alta Tensão (HV) foram baseados em dados públicos e georreferenciados. Além disso, dados públicos de usinas e sobre densidades populacionais, bem como cargas industriais, foram todos usados para modelar a parte de fornecimento de energia. Com base na topologia da rede e na simulação de fornecimento definida, as redes foram dimensionado para o seu estado (N-1) aplicando casos de uso relevantes.

Os dados abertos disponíveis para o nível de Média Tensão (MV), como distribuição de edifícios, mapas de linhas e estradas ou informações sobre a população, foram menos significativos, escassos ou mesmo incertos. Nesse caso, a análise de dados para as grades MV não foi criada para compilar um projeto, mas apenas para validar as redes do SimBench. A fim de considerar e equilibrar os requisitos de cerca de 30 casos de uso relevantes, bem como como as informações encontradas na revisão da literatura, as malhas MV do SimBench foram geradas manualmente. O conjunto de dados do SimBench compreende as 13 redes apresentadas na Figura 18.

Figura 18 – Visão Geral do Conjunto de Dados SimBench

(Exemplary) Subnet	SimBench code	Urbanzation character	Rated voltage [kV]	No. supply points	Transformer types	Generation types	Geo. Information with relation to reality
EHV1	1-EHV-mixed--0-sw	mixed	380, 220	390	209x600MVA	Nuclear, coal, gas	yes
HV1	1-HV-mixed--0-sw	mixed	110	58	2x300MVA, 4x350MVA	Wind	yes
HV2	1-HV-urban--0-sw	urban	110	79	3x300MVA	Wind	yes
MV1.101	1-MV-rural--0-sw	rural	20	92	2x25MVA	Wind, PV, Biomass, Hydro	no
MV2.101	1-MV-semiurb--0-sw	semi-urban	20	112	2x40MVA	Wind, PV, Biomass, Hydro	no
MV3.101	1-MV-urban--0-sw	urban	10	134	2x63MVA	Wind, PV, Hydro	no
MV4.101	1-MV-comm--0-sw	commercial	20	98	2x40MVA	Wind, PV, Biomass, Hydro	no
LV1.101	1-LV-rural1--0-sw	rural	0.4	13	1x160kVA	PV	no
LV2.101	1-LV-rural2--0-sw	rural	0.4	93	1x250kVA	PV	no
LV3.101	1-LV-rural3--0-sw	rural	0.4	118	1x400kVA	PV	no
LV4.101	1-LV-semiurb4--0-sw	semi-urban	0.4	39	1x400kVA	PV	no
LV5.201	1-LV-semiurb5--0-sw	semi-urban	0.4	104	1x630kVA	PV	no
LV6.201	1-LV-urban6--0-sw	urban	0.4	53	1x630kVA	PV	no

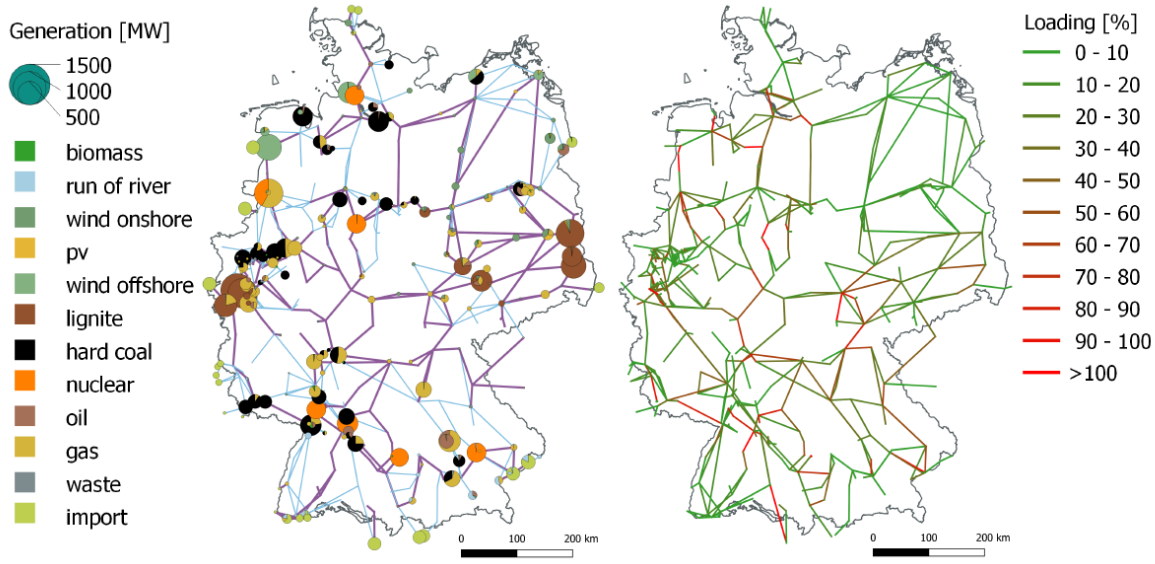
Fonte: Meinecke et al. (2020)

Os códigos do SimBench consistem em um número da versão, os níveis de tensão considerados e tipos de redes, o número do cenário e uma abreviação se os seccionadores estão incluídos em detalhes. O modelo de rede EHV, por exemplo, inclui 32.425 km de linhas, 464 localidades, 530 estações e 209 transformadores. Além da topologia da rede de transmissão (Figura 19) o conjunto de dados também inclui informações sobre a situação da fonte de alimentação correspondente, e a distribuição geográfica das usinas está representada. Existem grandes centrais elétricas nas áreas orientais e ocidentais da Alemanha e grandes usinas a gás nas regiões oeste e sul.

Além da distribuição característica de usinas eólicas e fotovoltaicas terrestres, os parques eólicos *offshore* (geradores em instalações flutuantes no oceano) também são representados por seus respectivos pontos de conexão à rede *onshore* (no continente). Para validar a topologia e, em particular, a situação de alimentação do conjunto de dados, o fluxo de energia da rede SimBench EHV foi comparados com as cargas históricas. O resultado do fluxo de potência de um passo de tempo (2 de janeiro 08:00 da manhã) para o Cenário 1-EHV-mixed–0-sw também é representado na Figura 19. Ele reflete congestionamentos característicos na direção Norte-Sul e Noroeste da Alemanha. Assim, com base nesse estado do grid, casos de uso como a determinação de medidas corretivas de fluxo de potência e estado da rede podem ser calculados.

O SimBench também tem disponível séries temporais, casos de estudo e combinações de redes de diferentes níveis de tensão. Os cenários futuros levam a três variantes em todos os grids, fornecendo informações de comutação e configurações de subestações.

Figura 19 – Distribuição geográfica de geração (esquerda) e carregamentos de linha em uma situação de alta demanda (direita) da rede EHV



Fonte: Meinecke et al. (2020)

3 METODOLOGIA E DESENVOLVIMENTO

O projeto, conforme Subseção 3.2.2, visa aplicar diferentes métodos de AI em sistemas elétricos modelados através da ferramenta pandapower. Um exemplo trata de uma ANN (*Artificial Neural Network*) para gerar um processo de previsão do comportamento da rede elétrica no tempo, e o segundo exemplo trata da aplicação de Aprendizagem por Reforço em um sistema elétrico mais complexo, de forma a gerar informações para adequar a geração e consumo da rede a um perfil que evite violar os limites de tensão mínima, corrente máxima e geração elétrica disponível no sistema em análise. A versão deste trabalho será desenvolvido na linguagem Python 3.10.6, através do notebooks do *Google Colaboratory*. Serão utilizados conjuntos de dados pré carregados na biblioteca do pandapower com redes desenvolvidas em trabalhos na literatura técnica. Cada um dos Casos de Teste foram convertidos do pypower ou MATPOWER (FRAUNHOFER, 2019c). Também serão usados os dados do sistema elétrico de potência da ferramenta simbench para Python, que facilita o carregamento das informações armazenadas no projeto SimBench para estudos e análises. O banco de dados SimBench foi discutido na seção Seção 2.4.

3.1 Materiais

As bibliotecas necessárias para o desenvolvimento apresentado nas seções seguintes são:

- pandapower: ferramentas de cálculo de rede elétrica destinado a automatizar e otimizar a análise de sistemas de energia. Faz uso da biblioteca de análise de dados pandas, permitindo o uso de diferentes solucionadores, entre eles uma implementação aprimorada de fluxo de energia pelo método Newton-Raphson, fluxo de potência AC e DC e fluxo de potência ótimo AC e DC (*Optimal Power Flow*, OPF).

- simbench: este repositório fornece dados e código para usar os dados do projeto SimBench dentro do pandapower.

- pandas: fornece estruturas para o trabalho com dados "relacionais"ou "rotulados"e análise/manipulação de dados.

- numpy: cálculo de matriz N-dimensional, ferramentas de integração de código C/C++ e Fortran, álgebra linear, transformada de Fourier e números aleatórios.

- os: este módulo fornece as funcionalidades do sistema operacional: ler ou escrever um arquivo (open()), manipular caminhos (os.path), ler as linhas de comando (fileinput), criar arquivos e diretórios temporários (tempfile), e manipulação de arquivos e diretórios de alto nível (shutil).

- matplotlib: biblioteca para criar visualizações estáticas, animadas e interativas em Python.

- seaborn: biblioteca de visualização baseada em matplotlib, interface para gráficos estatísticos.

- missingno: Visualização exploratória de dados faltantes.

- requests: biblioteca HTML.

- zipfile: biblioteca para manipular arquivos ZIP.

- beautifulSoup: extração de informações de páginas da web.

- lxml: manuseio de arquivos XML e HTML, e também pode ser usada para web scraping.

- math: funções teoria dos números e teoria da representação.

- datetime: classes para manipulação de datas e horas.

- keras: API projetada para *framework* de *Deep Learning* usado para reduzir a carga cognitiva, minimizar o número de ações do usuário necessárias para casos de uso comuns e fornecer mensagens de erro claras e acionáveis quando necessário.

- sklearn: Scikit-learn (Sklearn) é uma biblioteca para aprendizado de máquina (ML) em Python. Fornece uma seleção de ferramentas ML e modelagem estatística, incluindo classificação, regressão, agrupamento e redução de dimensionalidade.

- pickle: protocolos binários para serializar e de-serializar uma estrutura de objeto Python. "*Pickling*" é o processo pelo qual uma hierarquia de objetos Python é convertida em um fluxo de bytes, e "*unpickling*" é a operação inversa, pela qual um fluxo de bytes (de um arquivo binário ou objeto semelhante a bytes) é convertido em uma hierarquia de objetos.

- dotenv: O Python-dotenv lê pares chave-valor de um arquivo .env e pode defini-los como variáveis de ambiente.

- stable_baselines: conjunto de implementações aprimoradas de algoritmos de aprendizado por reforço baseados em *OpenAI Baselines*.

- gym: API universal para ambientes de aprendizado por reforço

Maiores informações sobre as bibliotecas podem ser encontradas no repositório do *Python Package Index* (PyPI) em <https://pypi.org>.

As bases de dados utilizadas são do próprio pandapower, além do repositório de dados do SimBench. O conjunto de dados SimBench está disponível de forma pública no site do projeto ou na biblioteca Python symbench, com parâmetros para a modelagem estática de redes elétricas com diversos níveis de tensão, de baixa (LV) até extra-alta tensão (EHV). Todas as redes podem ser interconectadas e estão disponíveis em variantes para cenários futuros, séries temporais em resolução de 15 minutos para um ano inteiro e casos de estudo predefinidos.

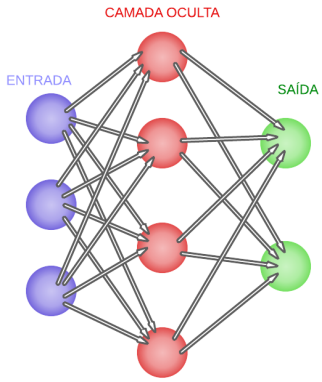
3.2 Métodos

O primeiro processo a ser explorado é a extração de dados de uma rede elétrica padronizada, para obtenção de resultados homogêneos em pesquisas, que faz parte do banco de dados simbench. Foi executado o cálculo de fluxo de potência nesta rede através do pandapower (que implementa um método Newton-Raphson para esta função), a criação de uma série temporal das potências no intervalo de um ano de amostragem que o simbench implementa e a aplicação de uma ANN (*Artificial Neural Network*, rede neural artificial) sobre estes dados para estimativa de comportamento futuro desta rede.

Uma ANN do tipo MLP (*Multi Layer Perceptron*) é constituída por 3 tipos de camadas (Figura 20):

- Camada de Entrada, onde os padrões são apresentados à rede;
- Camadas Intermediárias ou Ocultas: onde é feita a maior parte do processamento, através das conexões ponderadas. Essas redes tem uma ou mais camadas intermediárias ou escondidas;
- Camada de Saída: onde o resultado final é concluído e apresentado.

Figura 20 – ANN do tipo MLP



Fonte: Elaboração Própria

A quantidade de neurônios nas camadas de entrada e saída é dada pelo problema a ser solucionado. Já a quantidade de neurônios nas camadas de processamento (camadas ocultas) são características do projeto. Aumentando o número de neurônios na camada oculta aumenta a capacidade de mapeamento não-linear da rede. Porém, quando esse número for muito grande, o modelo pode se sobrepor aos dados por causa de ruído nas amostras de treinamento. Neste caso a rede está sujeita ao *overfitting*). Por outro lado, uma rede com poucos neurônios na camada oculta pode não ser capaz de realizar o mapeamento desejado, o que é denominado de *underfitting*. O *underfitting* também pode ser causado quando o treinamento é interrompido de forma prematura (HAYKIN, 2001).

Perceptron multicamadas é um algoritmo padrão para muitos processos de aprendizado supervisionado, como reconhecimento de padrões por exemplo. Ele é útil por sua capacidade de resolver problemas estocásticos, porque permite obter soluções aproximadas para problemas complexos. Como limitações deste processo, estas redes neurais artificiais podem ser vistas como "caixas pretas", na qual quase não se sabe porque a rede chega a um determinado resultado, uma vez que os modelos não apresentam justificativas para suas respostas. Outra limitação refere-se ao tempo de treinamento de redes neurais, que tende a ser muito lento. Algumas vezes são necessários milhares de ciclos para se chegar à níveis de erros aceitáveis, principalmente se estiver sendo simulado em computadores seriais, pois a CPU deve calcular as funções para cada unidade e suas conexões separadamente, o que pode ser problemático em redes muito grandes, ou com grande quantidade de dados. Dessa forma, há uma certa dificuldade em definir a rede de forma que ela seja tão grande quanto o necessário para conseguir obter as representações do problema, e ao mesmo tempo pequena o suficiente para se ter um treinamento rápido. Não existem regras claras para se definir quantas unidades devem existir nas camadas ocultas, ou como devem ser as conexões entre essas unidades (HAYKIN, 2001).

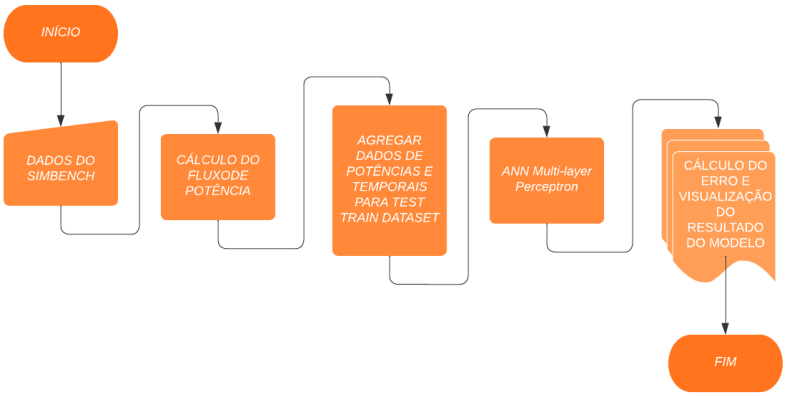
Na sequencia é usada outra rede elétrica padronizada, de distribuição de média tensão, desenvolvida pela Força-Tarefa CIGRE C6.04.02 para facilitar a análise e validação de novos métodos e técnicas que visam permitir a integração econômica, robusta e ambientalmente responsável dos DER (*Distributed Energy Resources*) (FRAUNHOFER, 2019a). As diversas redes CIGRE são um conjunto de sistemas de referência para permitir a análise da integração de DER's em alta tensão, média tensão e baixa tensão, de forma a se obter resultados replicáveis e homogêneos em análises e trabalhos de pesquisa. Nesta segunda rede é usado um processo de Aprendizado por Reforço considerando diversas condições de operação da rede e como favorecer as condições positivas e evitar as condições inadequadas de operação do sistema.

O Aprendizado Por Reforço é o treinamento de modelos de aprendizado de máquina para tomar uma sequência de decisões. O agente aprende a atingir uma meta em um ambiente incerto e complexo. No aprendizado por reforço, o modelo de inteligência artificial enfrenta uma situação. A máquina utiliza tentativa e erro para encontrar uma solução para o problema. Para que a máquina faça o que o programador deseja, a inteligência artificial recebe recompensas ou penalidades pelas ações que executa. Seu objetivo final é maximizar a recompensa total (DULHARE UMA N.; AHMAD, 2020).

3.2.1 pandapower e MLP

A primeira execução baseada em pandapower e métodos de AI trata de uma ANN do tipo Multi-Layer Perceptron modelada no sklearn, usando dados do SimBench processados no pandapower. A sequencia de procedimentos é ilustrada na Figura 21. Inicialmente, é criado um dataframe da rede HV2 do SimBench para análise e cálculo no pandapower.

Figura 21 – Execução da ANN com dados do pandapower



Fonte: Elaboração Própria

A rede HV1 (Figura 22) trata de uma rede com as seguintes características principais:

- 372 barramentos
- 79 cargas
- 98 gerações de armazenamento (Sgen, por exemplo eólico, fotovoltaico, etc.). No caso específico trata-se de gração eólica.
- 498 elementos de chaveamento
- 113 linhas
- 3 transformadores
- 14 subestações
- 79 cargas

Adicionalmente, a Figura 23 mostra a distribuição física destes componetes na rede elétrica.

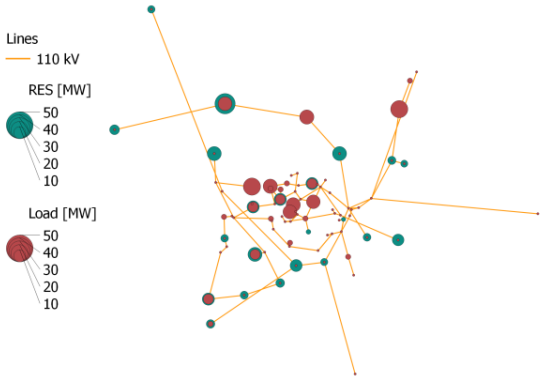
Figura 22 – Rede HV2 SimBench

(Exemplary) Subnet	SimBench code	Urbanzation character	Rated voltage [kV]	No. supply points	Transformer types	Generation types	Geo. Information with relation to reality
HV2	1-HV-urban--0-sw	urban	110	79	3x300MVA	Wind	yes

Fonte: Meinecke et al. (2020)

Os dados da rede HV2 são carregados em um dataframe e é executado o cálculo do fluxo de potência (Figura 21). Na sequência os dados de potência gerada e potência ativa e reativa na carga são agregados em um dataset com as informações temporais e selecionados para teste/treino através do scikit. Este dataset é aplicado a uma ANN do scikitlearn para gerar um modelo de previsão do comportamento da rede elétrica, visando detectar necessidades futuras e tomar medidas para prevenir colapso do sistema nos horários e temporadas onde a carga de pico do sistema irá aumentar (CHELLA, 2022b).

Figura 23 – Diagrama da Rede HV2 com as Gerações e Cargas do Modelo

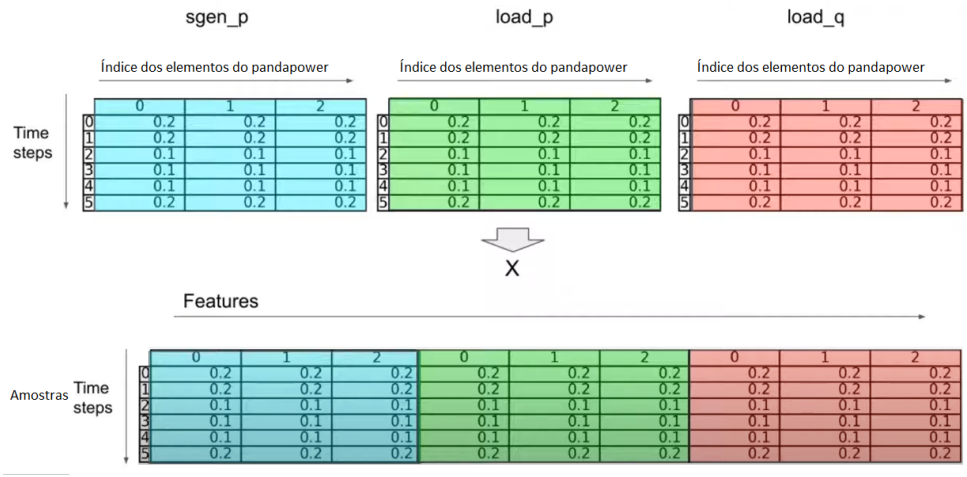


Fonte: Meinecke et al. (2020)

É feito uso da função ConstControl, empregada no pandapower para séries temporais, para ler dados de um dataframe e aplicá-lo em uma rede. Os dados de Potência P e Q da carga, e a Potência de geração Sgen são lidos das variáveis geradas sgen_p, load_p e load_q, extraídas dos dados da rede pandapower HV2 calculados para o fluxo de potência (FRAUNHOFER, 2019b).

O módulo de série temporal do pandapower é projetado para a simulação de operações baseadas no tempo. Dentro de uma simulação de série temporal, os controladores são usados para atualizar valores de diferentes elementos em cada passo de tempo em um loop. Um caso de uso comum é adicionar dois ConstControl ao grid para atualizar os valores de potência de cargas e geradores estáticos. Os valores de p_mw para cada passo de tempo são armazenados em uma fonte de dados, que é definida pela classe DataSource. Essa classe contém um dataframe pandas que armazena os valores para cada elemento e etapa de tempo. O ConstControl abre a fonte de dados, lê os valores p_mw dela e escreve os valores correspondentes para cada sgen/carga antes do cálculo do fluxo de potência. Depois de executar o cálculo do fluxo de potência, os resultados para cada etapa de tempo são gravados pela classe de gravador de saída OutputWriter em arquivos excel, csv, json ou pickle. Estes dados são concatenados no dataframe ds, conforme o esquema a seguir (Figura 24):

Figura 24 – DataFrame de Potências na Carga e Potência de Geração



Fonte: Elaboração Própria, baseado em Schaefer (2020)

Este conjunto de dados é aplicado à função train_test_split do sklearn, de modo a gerar o conjunto de treino e teste da rede neural, definida em 90% para treino e 10% para teste, neste caso específico. Os valores usados são sgen_p (Potência Aparente gerada, Figura 25), load_p e load_q (Potência Ativa e Reativa na carga, Figura 26) para a previsão dos carregamentos de linha em porcentagem (corrente máxima da linha). O carregamento da linha é calculado no fluxo de potência, em função da corrente máxima calculada que passa pela linha e da corrente máxima nominal do cabo usado na construção da linha. No caso da simulação da série temporal, para cada passo de tempo é calculado o valor do carregamento da linha.

Figura 25 – S_{gen} - 98 elementos geradores

6 sgen_p

	0	1	2	3	4	5	6	7	8	9	...	88	89	90	91	92	93	94	95	96	97
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.521221	3.450101	10.126047	21.623509	3.018839	8.237117	38.335381	8.136523	27.768684	13.942401
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.508898	3.444449	10.109457	21.588083	3.013893	8.223621	38.133541	8.093683	27.622479	13.868992
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.496576	3.438796	10.092867	21.552656	3.008947	8.210126	37.931701	8.050843	27.476273	13.795584
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.484254	3.433144	10.076278	21.517230	3.004001	8.196631	37.729861	8.008004	27.330068	13.722176
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.471932	3.427492	10.059688	21.481803	2.999055	8.183136	37.528021	7.965164	27.183863	13.648767
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.765590	0.809904	2.377067	5.076071	0.708666	1.933645	10.828114	2.298221	7.843471	3.938135
35132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
35133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
35134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
35135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

35136 rows × 98 columns

Fonte: Elaboração Própria

Figura 26 – $Load_p$ - 79 cargas

2 load_p

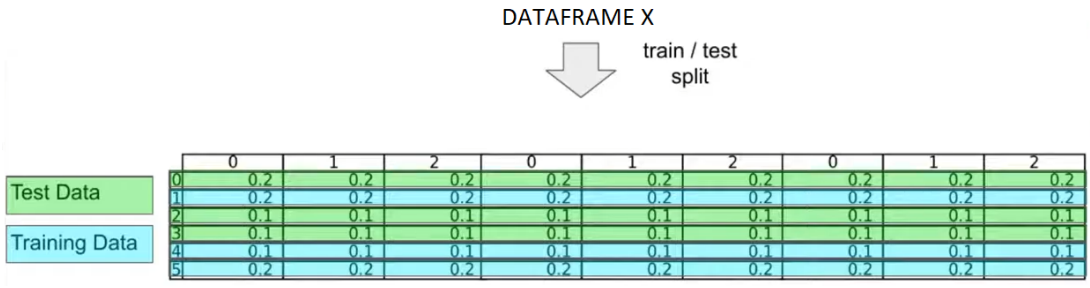
	0	1	2	3	4	5	6	7	8	9	...	69	70	71	72	73	74	75	76	77	78
0	0.166817	0.166817	0.166817	0.166817	0.166817	0.166817	0.166817	0.166817	0.166817	0.166817	...	3.584056	3.584056	3.584056	3.584056	3.584056	5.536304	5.536304	5.536304	8.466359	8.466359
1	0.151334	0.151334	0.151334	0.151334	0.151334	0.151334	0.151334	0.151334	0.151334	0.151334	...	3.325657	3.325657	3.325657	3.325657	3.325657	4.470922	4.470922	4.470922	6.881156	6.881156
2	0.138827	0.138827	0.138827	0.138827	0.138827	0.138827	0.138827	0.138827	0.138827	0.138827	...	3.770896	3.770896	3.770896	3.770896	3.770896	5.105051	5.105051	5.105051	7.772705	7.772705
3	0.122876	0.122876	0.122876	0.122876	0.122876	0.122876	0.122876	0.122876	0.122876	0.122876	...	3.507629	3.507629	3.507629	3.507629	3.507629	4.555496	4.555496	4.555496	6.951198	6.951198
4	0.137754	0.137754	0.137754	0.137754	0.137754	0.137754	0.137754	0.137754	0.137754	0.137754	...	3.649938	3.649938	3.649938	3.649938	3.649938	4.972258	4.972258	4.972258	7.672092	7.672092
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35131	0.136995	0.136995	0.136995	0.136995	0.136995	0.136995	0.136995	0.136995	0.136995	0.136995	...	2.991141	2.991141	2.991141	2.991141	2.991141	3.970504	3.970504	3.970504	5.906642	5.906642
35132	0.121983	0.121983	0.121983	0.121983	0.121983	0.121983	0.121983	0.121983	0.121983	0.121983	...	2.912332	2.912332	2.912332	2.912332	2.912332	3.900484	3.900484	3.900484	5.838191	5.838191
35133	0.101357	0.101357	0.101357	0.101357	0.101357	0.101357	0.101357	0.101357	0.101357	0.101357	...	2.611921	2.611921	2.611921	2.611921	2.611921	3.657236	3.657236	3.657236	5.495739	5.495739
35134	0.128687	0.128687	0.128687	0.128687	0.128687	0.128687	0.128687	0.128687	0.128687	0.128687	...	2.672003	2.672003	2.672003	2.672003	2.672003	3.682928	3.682928	3.682928	5.486095	5.486095
35135	0.116609	0.116609	0.116609	0.116609	0.116609	0.116609	0.116609	0.116609	0.116609	0.116609	...	2.767883	2.767883	2.767883	2.767883	2.767883	3.904692	3.904692	3.904692	5.899533	5.899533

35136 rows x 79 columns

Fonte: Elaboração Própria

Os arquivos S_{gen} 35135 x 98, $Load_p$ 35135 x 79 e $Load_q$ 35135 x 79 são concatenados para criar o dataframe X a ser aplicado no sklearn. O dataframe Y é obtido a partir do arquivo loading_percent.json criado pelo cálculo de fluxo de potência na série temporal do dataframe pandapower, composto de 113 elementos (linhas), cada um com 35135 valores de carregamento de linha (um valor em cada um dos 35135 passos de tempo). Cada combinação de valores do conjunto de S_{gen}, $Load_p$ e $Load_q$ gera uma resposta de carga nas 113 linhas, e este é a combinação de dados usado ao modelo de ANN do sklearn, para a construção da rede neural de previsão de carregamento das linhas do sistema (Figura 27).

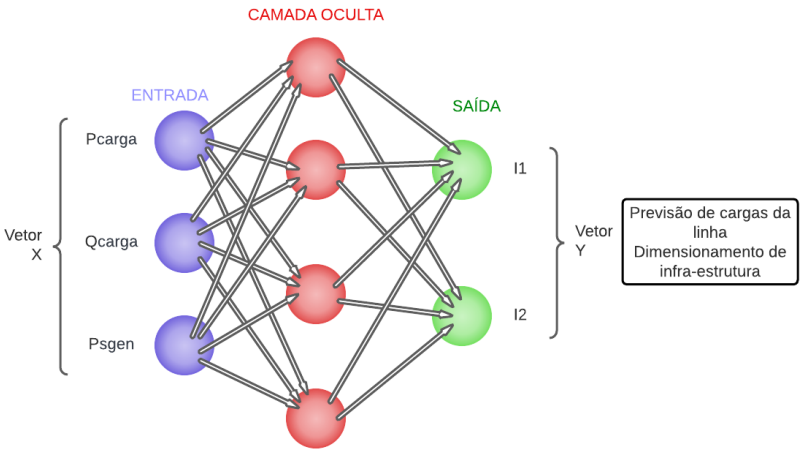
Figura 27 – train/test DataSet



Fonte: Elaboração Própria, baseado em Schaefer (2020)

Então, o conjunto treino/teste é submetido à uma ANN do tipo Multi-layer Perceptron Regressor do sklearn (Figura 28).

Figura 28 – Multi-layer Perceptron Regressor Executada no Exemplo



Fonte: Elaboração Própria, baseado em Schaefer (2020)

A apresentação dos resultados está no Capítulo 4, Seção 4.1.

3.2.2 pandapower, gym e stable-baselines com Aprendizado por Reforço

A segunda execução deste trabalho envolve a execução de cálculos de fluxo de potência em uma rede de energia e uma subsequente aplicação de um método de aprendizado por reforço neste conjunto de dados, avaliando o comportamento desta rede elétrica em função de mudanças no perfil de geração e de consumo nas cargas e selecionando os melhores resultados desta avaliação.

Em um sistema de energia elétrica, a produção de energia deve ser consumida em um curto espaço de tempo em algum lugar na grade. Excesso de disponibilidade de energia produzida por sistemas renováveis (fotovoltaico, hidráulico, eólico) deve ser consumido em algum lugar, o que pode extrapolar a capacidade das linhas elétricas na distribuição (se a corrente nestas linhas for muito elevada) e/ou prejudicar a qualidade da tensão em uma rede elétrica (elevando a tensão se há pouco consumo e diminuindo a tensão quando há muito consumo). Existe a opção de se desligar parte do sistema de geração se há excesso de produção de energia, porém nesta condição estará havendo um sub-aproveitamento dos recursos naturais, o que não é uma situação ideal. Com relação ao excesso de consumo, a única maneira de evitar é desonerando a utilização de energia em períodos de alta disponibilidade e elevar o custo de utilização de energia em horários onde a demanda de energia seja alta e a disponibilidade desta energia baixa (há a alternativa de se usar fontes extras não renováveis, porém isso acaba encarecendo o custo de energia de qualquer modo ou gerando consequências ambientais).

Uma solução que corresponde a este cenário, especificamente quando a fonte principal de energia é fotovoltaica solar, é fazer com que os consumidores mudem seu padrão de consumo de energia para que a grande disponibilidade de energia solar durante o dia seja aproveitada neste período.

Um método para alcançar essa de padrão de consumo, conforme demonstrado por Solberg (2019), é o chamado programa de resposta à demanda. Em princípio, seria possível fazer uma implementação em Python de um sistema automático de resposta à demanda simplificado, usando aprendizado por reforço e as ferramentas do pandapower para processamento dos dados de um sistema elétrico.

O algoritmo de aprendizado por reforço demonstrado por Solberg (2019) pode simular um aumento ou diminuição da potência consumida a cada hora em uma rede elétrica que tem uma grande quantidade de produção solar local e alta demanda de pico. Essa variação de consumo de energia pode, por exemplo, ser entendida como uma grande frota de veículos elétricos adiando ou antecipando seu carregamento no período de simulação considerado. Uma vez que o algoritmo faz a modificação do consumo de energia, as correntes e tensões de linha resultantes no sistema são calculados. O objetivo do algoritmo é aprender um comportamento que reduza o número de violações de corrente (corrente muito alta) e tensão (tensão muito alta ou muito baixa) na rede. O aprendizado por reforço é sobre tomar ações e obter recompensas ou custos com base em "o quanto foi boa ou ruim" a ação tomada pelo algoritmo. Portanto, a ação de controlar certos elementos de uma rede pandapower deve levar a um cálculo atualizado do fluxo de potência e as suas consequências devem ser avaliadas.

No processo demonstrado em Solberg (2019), a demanda na rede é modificada e os transformadores do sistema são atualizados para compensar as consequências da modificação executada. Os transformadores em um sistema de transmissão podem ter derivações controláveis que alteram a relação de enrolamento entre o lado de baixa e alta tensão do transformador. Ao fazer isso é possível controlar o valor da tensão nos barramentos conectados a um transformador compensando, com certas limitações, algumas situações de não-conformidade dos limites de tensão do sistema elétrico.

Existem também configurações de transformadores com deslocamento de fase que podem manipular o ângulo de tensão entre o lado de baixa e alta tensão. Os transformadores modelados no pandapower permitem o controle tanto da magnitude da tensão |U| como ângulo de tensão δ .

Por exemplo, no pandapower, um transformador padrão 25 MVA, 110/20 kV, e com a rede externa está conectado ao lado de alta tensão do transformador, os taps são colocados no lado de baixa tensão do transformador com o comando net.trafo['tp_side'] = 'lv'. Out[01] mostra a execução de um cálculo de fluxo de potência que fornece os resultados sem alterar a posição do tap no transformador (Figura 29). Depois, o ângulo de tensão é manipulado para 20 graus especificando 'net.trafo['shift_degree']. O valor da tensão é alterado especificando primeiro 'net.trafo[tp_st_percent' que é a mudança percentual na magnitude da tensão por

Figura 29 – Controlando Posição do Tap ($'tp\_pos'$) e o Ângulo de Fase ($'shift\_degree'$) de um Transformador no pandapower

```
In [1]:
import pandapower as pp
net = pp.create_empty_network()
b1 = pp.create_bus(net, vn_kv=110)
b2 = pp.create_bus(net, vn_kv=20)
t = pp.create_transformer(net, hv_bus=b1, lv_bus =b2,
                          std_type='25 MVA 110/20 kV')
net.trafo['tp_side'] = 'lv'
load = pp.create_load(net, bus=b2, p_kw=10000)
pp.create_ext_grid(net,bus=b1)
pp.runpp(net)

net.res_bus
```

Out[1]:

	vm_pu	va_degree	p_kw	q_kvar
0	1.00000	0.000000	-10030.468579	-493.595172
1	0.99717	-2.759365	10000.000000	0.000000

```
In [2]:
net.trafo['shift_degree'] = 20
net.trafo['tp_st_percent'] = 10

net.trafo['tp_pos'] = -1
pp.runpp(net,calculate_voltage_angles=True)
net.res_bus
```

Out[2]:

	vm_pu	va_degree	p_kw	q_kvar
0	1.000000	0.000000	-10030.468579	-493.595172
1	0.897453	-22.759365	10000.000000	0.000000

Fonte: Solberg (2019)

Figura 30 – Dobrando o consumo em cargas no pandapower

```
In [1]:
import pandapower as pp
import pandapower.networks as pn

net = pn.case4gs()
pp.runpp(net)
net.res_load
```

Out[1]:

	p_kw	q_kvar
0	50000.0	30990.0
1	170000.0	105350.0
2	200000.0	123940.0
3	80000.0	49580.0

```
In [2]:
net.load[['p_kw','q_kvar']] *= 2
pp.runpp(net)
net.res_load
```

Out[2]:

	p_kw	q_kvar
0	100000.0	61980.0
1	340000.0	210700.0
2	400000.0	247880.0
3	160000.0	99160.0

Fonte: Solberg (2019)

cada posição de tap (10% para cada posição de tap). A posição final do tap é definida usando 'net.trafo['tp_pos'] para -1, ou seja reduzindo em um ponto o tap de tensão.

Executando novamente o cálculo de fluxo de potência, através do comando net.res, é possível ver que o ângulo de tensão é deslocado 20 graus e que o valor da tensão reduzido em 10% em relação ao primeiro cálculo do fluxo de potência.

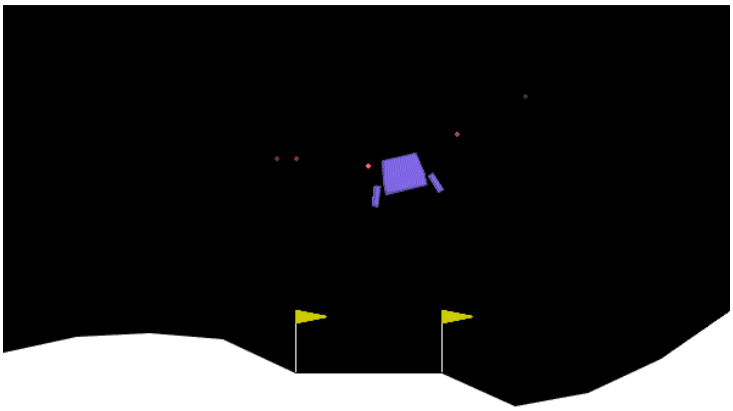
Pode haver várias cargas conectadas a um barramento no pandapower. Nesse caso, a injeção de potência da rede nesse barramento é a soma de todas as suas cargas. A Figura 30 mostra como dobrar a potência ativa e reativa consumida nas cargas.

O padrão geral de controle um elemento no pandapower é alterar a tabela de elementos, executar o cálculo do fluxo de energia, que por sua vez atualiza a tabela de resultados. A produção de energia solar em um barramento pode ser definida no mesma maneira, atualizando os valores na tabela 'net.sgen'.

A biblioteca Python gym é um conjunto de ferramentas usado para gerenciar ambientes de algoritmos de aprendizagem por reforço. Ela inclui milhares de ambientes de videogames clássicos e tarefas de teoria de controle, como o "*lunar lander*", o "pêndulo oscilante", etc.

"*Lunar lander*" é um problema clássico de otimização de trajetória de foguetes, e os detalhes estão na documentação do gym (<https://www.gymlibrary.dev/environments/box2d/lunar_lander/>). Este ambiente possui ações discretas: motor do foguete ligado ou desligado (esquerda, direita e abaixo). Entre outras características deste problema, a plataforma de pouso está sempre nas coordenadas (0,0), as coordenadas são os dois primeiros números no vetor de estado, a aterrissagem fora da plataforma de aterrissagem é possível. O combustível é infinito, então um agente pode aprender a voar e pousar na primeira tentativa.

Figura 31 – *Lunar lander*

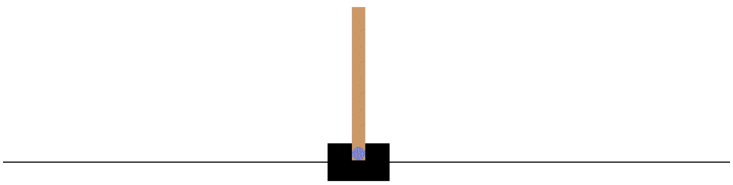


Fonte: documentação gym

Já o pêndulo oscilante ("*cart-pole*") também é descrito na documentação do gym (<https://www.gymlibrary.dev/environments/classic_control/cart_pole/>). Um poste é preso por uma junta a um carrinho, que se move ao longo de um trilho sem atrito.

O pêndulo é colocado verticalmente no carrinho e o objetivo é equilibrar o mastro aplicando forças na direção esquerda e direita no carrinho.

Figura 32 – *Pêndulo oscilante*



Fonte: documentação gym

Além destes exemplos, é possível construir ambientes próprios que facilmente podem ser usados em algoritmos de aprendizagem por reforço. Também existe um conjunto de ferramentas chamado stable_baselines, criado pela comunidade de AI, com uma interface do tipo scikit-learn. Este código também suporta o conjunto de ferramentas tensorboard, que pode ser usado para monitorar as recompensas e punições durante o treinamento. A implementação do algoritmo de reforço no trabalho de Solberg (2019) é feita usando gym e stable_baselines. Especificamente, uma classe de ambiente chamada ActiveEnv é implementada, que segue a estrutura padrão do ambiente gym.

Este ambiente pode ser totalmente testado usando a biblioteca pytest. O principal trabalho do ActiveEnv é realizar uma ação escolhida por um algoritmo de reforço, executar essa ação, encontrar o próximo estado resultante desse ação e calcular a recompensa/custo.

Neste caso, ActiveEnv recebe um vetor de ação 'a' onde cada componente determina a variação percentual no consumo de energia em cada carga flexível.

- Primeiro, o consumo e a geração de energia nos nós da rede são alterados de acordo com a previsão de demanda e previsão solar, respectivamente.
- O vetor de ação é processado e atualiza o consumo de energia em cada carga na rede pandapower.

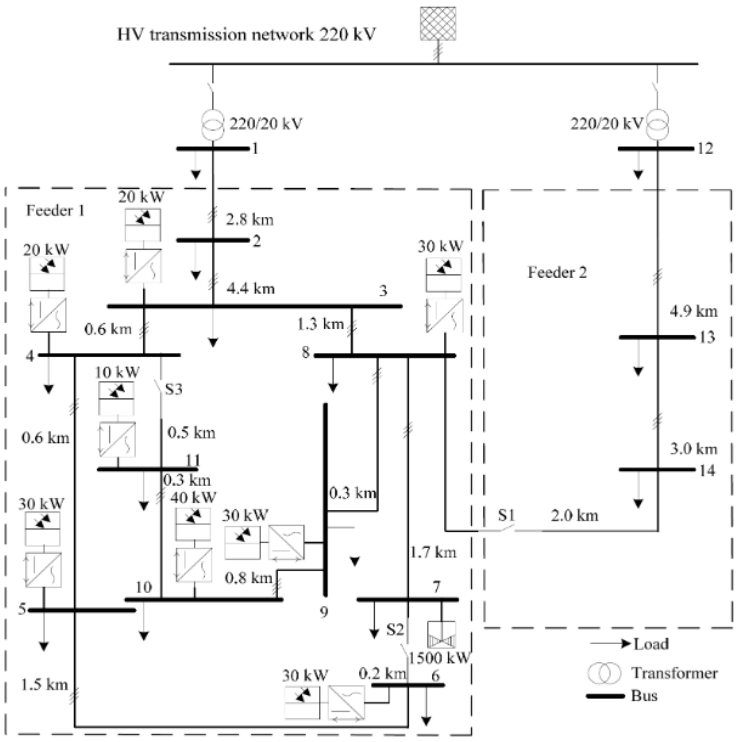
- As equações de fluxo de potência para a rede são resolvidas usando o pandapower.
- Depois que as equações de fluxo de potência foram resolvidas e os novos valores de tensão e corrente na rede é calculada, a recompensa/punição pode ser calculada.

Isso resume as etapas envolvidas em uma ação no algoritmo de aprendizagem por reforço.

As bibliotecas Python de AI sofreram alterações significativas desde a época que o trabalho em Solberg (2019) foi publicado. Este código é incompatível com as bibliotecas atuais e diversos recursos estão obsoletos e podem ser descontinuados. O downgrade das bibliotecas de AI e RL foi suficiente para rodar a demonstração do método de Aprendizado por Reforço, porém seria necessário um retrabalho nestas funções para as bibliotecas atuais do Python, o que vai demandar um tempo muito grande para revisão do código. A adaptação do programa rodando sobre as bibliotecas em downgrade está disponível em Chella (2022c) . Alguns arquivos adicionais precisam ser descompactados e carregados na área de Arquivos do Notebook para facilitar a reprodução das funções e subrotinas do exemplo (disponível em Chella (2022a)).

O projeto de Solberg (2019) é baseado em uma rede elétrica construída pelo Conselho Internacional de Grandes Sistemas Elétricos (CIGRE) europeu, como uma rede de referência que pode ser usada para analise e estudos de integração de recursos energéticos distribuídos. É uma rede predefinida no pandapower, e pode ser adaptada com diferentes fontes de energia renovável conectadas em diferentes barramentos.

Figura 33 – Rede CIGRE do estudo



Fonte: Solberg (2019)

Há uma unidade de geração de energia eólica conectada ao barramento 7 (Figura 33), mas, por simplicidade neste projeto, foi suposta que seja geração solar fotovoltaica. Isso foi feito a fim de facilitar a manipulação do dados dentro da simulação. A produção da planta solar é baseada em relatórios montados a partir de imagens de satélite na região central da Noruega (origem da rede de energia do estudo).

Cabe notar também que este ambiente poderia receber qualquer outra rede pandapower.

Resumindo, o agente (ActiveEnv) pode modificar o consumo de energia em nós do sistema de energia com alta demanda de pico e alta produção de energia solar. O objetivo do agente é reduzir o número de violações de corrente e tensão na rede aumentando/diminuindo o consumo em momentos apropriados.

O aumento/diminuição do consumo de energia pretende ser um programa simplificado de resposta à procura que explora a flexibilidade residencial na rede.

O ActiveEnv é uma função do stable_baselines que tem vários parâmetros que determinam o espaço de estado, ações, recompensas etc. na montagem do cenário de Aprendizado por Reforço. Os parâmetros são fornecidos como um atributo env.params. Especificamente neste estudo, é usada a previsão de geração solar e a previsão de demanda.

A previsão solar é gerada a partir de irradiação solar derivada de satélite na região central da Noruega (origem da rede de energia do estudo). A seguir é mostrado como definir o espaço de estado para incluir apenas a previsão solar.

O atributo env.observation_space fornece informações sobre o espaço de estado:

```
env = ActiveEnv()
env.set_parameters('state_space':['sun'])
env.set_parameters('forecast_horizon':24)
```

Este processo define que será usada a previsão solar nas próximas 24 horas, uma vez que o espaço de estado neste exemplo consiste apenas na irradiância solar. O estado atual do ambiente pode ser acessado pelo método _get_obs(). A irradiância solar é assumida igual em toda a rede para fins de simplificação do processo, portanto não há uma previsão única para cada barramento, o que seria necessário na implementação real.

Como nenhuma previsão é perfeita, os valores reais precisam se desviar dos valores previstos por um certo valor de incerteza, assim os valores reais precisam ser encontrados adicionando um termo de ruído à previsão. O termo de ruído segue uma distribuição gaussiana com média 0 e desvio padrão proporcional à previsão. O desvio padrão nas previsões é por padrão de 3%. A incerteza no precedimento de cálculo pode ser alterada com:

```
env.set_parameters('solar_std':0.1,'demand_std':0.3)
```

É necessário que o agente desloque o pico de consumo de energia no tempo, e não o altere em valor absoluto. O agente é penalizado se aumentar o desequilíbrio de energia no sistema. Para isso, o agente deve receber alguma informação que mostre se o sistema está em desequilíbrio de energia ou não. Este é o trabalho do estado de desequilíbrio. Este estado é um vetor onde cada componente é o desequilíbrio energético das cargas no sistema. Como existem 18 cargas, o tamanho do espaço de desequilíbrio é de 18 posições.

Forçar as cargas a consumir mais energia é possível através de dois procedimentos. o primeiro é simular um dia aumentando o consumo ao máximo para as cargas. O estado agora mostra que todas as cargas usaram mais energia do que o necessário nas últimas 24 horas. O desbalanceamento (em MW) é dimensionado pelo consumo nominal total das cargas na rede.

O segundo procedimento é considerar apenas o desequilíbrio total na rede. Isso é feito definindo o parâmetro total_imbalance igual a 'True'.

env.set_parameters('total_imbalance':True)

Sobre o Espaço de Ação, há uma ação para cada linha na tabela de carga da rede. O agente pode alterar independentemente o consumo em cada carga em um intervalo de flexibilidade, por exemplo, em $+/-$ 10%. Assume-se que as cargas têm um fator de potência constante. Em outras palavras, se a potência ativa aumenta em 10%, a potência reativa também aumenta em 10%.

Para a rede CIGRE, temos o espaço de ação $A = a_i | i = 1, 2, ...18, a_i \in [-1, 1]$.

A ação é então dimensionada pela flexibilidade (0-100%) e demanda prevista, que determina a variação do consumo em cada carga.

O espaço de ação é descrito em env.action_space. O método sample() pode ser usado para amostrar uma ação aleatória do espaço de ação. Baseado neste vetor, o ActiveEnv manipula o DataFrame net.load e resolve a equação do fluxo de energia. Neste exemplo, cada carga aumenta sua carga com flexibilidade de até 10%. Primeiro, é criada uma instância de ambiente e feita a previsão de demanda. A incerteza na previsão é definida como 0 para remover a estocasticidade. Em seguida, é criado o vetor de ação "ones"preenchido com o valor "1", o que significa que todas as cargas aumentam seu consumo o máximo possível.

a = np.ones(env.action_space.shape)
A ação interage com o ambiente usando o método step()
_,_,_,_ =env.step(a)

O comando env.step executará uma ação em cada etapa, retorna quatro parâmetros: observação, recompensa, feito (booleano informando se é hora de redefinir o ambiente) e informação (para debug, se necessário). Os valores de consumo e produção na rede são atualizados, e é feita a passagem uma hora no episódio.

O consumo em percentagem dos valores nominais é igual em cada carga.

consumption = env.powergrid.res_load['p_mw']/env.powergrid.load['sn_mva']

Porém, com 18 variáveis livres em um espaço de ação há uma quantidade bastante grande de parâmetros. É possível definir uma única ação global que modifica todas as cargas no intervalo de flexibilidade. A variação é um percentual no consumo e as cargas possuem diferentes níveis de consumo nominal e, desse modo, a mudança de potência absoluta varia de carga para carga.

env.set_parameters('one_action':True)

As recompensas são fundamentais para todos os algoritmos de aprendizado por reforço, mas, nesse caso específico, é mais simples falar de custos do que de recompensas. A recompensa é definida como o negativo do custo, portanto, maximizar a recompensa é o mesmo que minimizar o custo. Diferentes termos de recompensa podem ser especificados na classe ActiveEnv. O objetivo final do agente é reduzir o número de violações de corrente e tensão na rede.

O primeiro item desta simulação é o Custo de Tensão. Deve ser definido o que é uma violação neste caso: há uma violação de tensão na rede se o valor da tensão em algum ponto da rede sair do limite de segurança. Os limites de segurança são, por padrão, 0,95 a 1,05 pu.

Neste processo, o custo de tensão para um barramento é proporcional à violação de tensão, conforme mostrado na Figura 34. O custo total da tensão é encontrado pela soma de todos os barramentos.

Existe um peso que dimensiona o custo total da tensão, cujo valor padrão é 1. Para incluir apenas a tensão no cálculo da recompensa e alterar o peso da tensão:

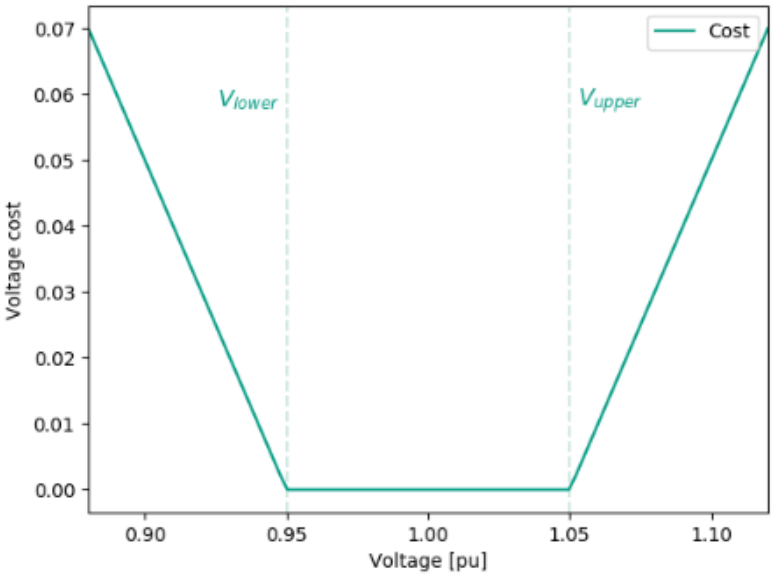
env.set_parameters('reward_terms':['voltage'], 'voltage_weight':3)

O limite inferior e superior pode ser alterado usando o método set_parameters():

env.set_parameters('v_lower':0.9,'v_upper':1.2)

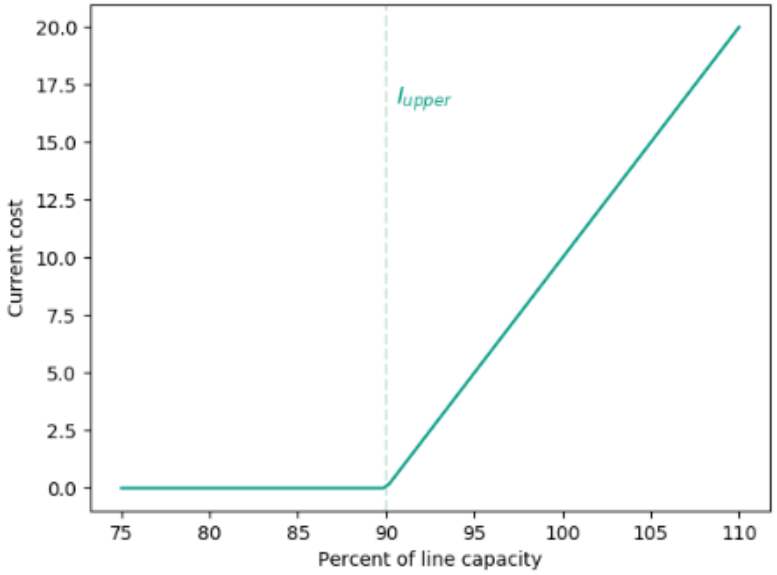
Da mesma forma, há uma violação de corrente em uma linha se a corrente exceder um limite de capacidade, que por padrão é definido como 90%. O custo é proporcional à violação na linha e o custo total é encontrado pela soma de todas as linhas. É mais correto que seja uma relação não linear, de modo que uma grande violação seja punida com mais rigor do que várias pequenas violações. O peso para o custo da corrente é 0,01 por padrão.

Figura 34 – Função do Custo de Tensão



Fonte: Solberg (2019)

Figura 35 – Função do Custo de Corrente



Fonte: Solberg (2019)

O custo do desequilíbrio penaliza um grande desequilíbrio energético. O agente deve, idealmente, aumentar o consumo de energia durante o pico de produção solar e diminuir durante o pico de demanda das cargas.

Esse custo do desequilíbrio existe para incentivar o agente a deslocar o consumo de energia.

Existem duas variações do custo de desequilíbrio:

- Desequilíbrio energético absoluto: O agente é penalizado proporcionalmente ao desequilíbrio energético das últimas 24 horas.
- Aumento do desequilíbrio energético: O agente é penalizado proporcionalmente ao aumento do desequilíbrio energético absoluto.

Por exemplo, considerando um cenário onde o desequilíbrio é de $+20$ MWh, e que aumenta para $+25$ MWh e após o agente ter realizado sua ação. O custo do desequilíbrio é então (25 - 20) * "*imbalance weight*" . Este custo torna-se negativo quando o desequilíbrio diminui e é a única vez que o agente pode obter uma recompensa positiva. A versão padrão deste estudo usa o desequilíbrio absoluto de energia. A próxima função mostra como alternar para o modo de alteração do desequilíbrio e como alterar o peso do desequilíbrio, que por padrão é $10 * e^{-5}$:

env.se_parameters('reward_terms':['imbalance'], 'imbalance_change':True, 'imbalance_weight':10e^{-3})

Finalmente, a flexibilidade não é um recurso gratuito. Neste procedimento, o custo de ativação é proporcional à variação de potência absoluta resultante da ação do agente, com preço constante, mas em um ambiente mais realista deveria ser incluído o custo no espaço de estados, para que o agente saiba que é caro alterar os consumos quando o preço é alto.

env.set_parameters('reward_terms':['activation'], 'activation_weight':10e^{-3})

O custo total de um agente em cada etapa é definido como uma combinação linear sobre o custo de tensão, corrente, ativação e desequilíbrio (Figura 36).

Figura 36 – Função do Custo Total

$$C = \kappa_1 \sum_{i=1}^{F} C_{\text{activation},i} + \kappa_2 \sum_{i=1}^{L} C_{\text{current},i} + \kappa_3 \sum_{i=1}^{N} C_{\text{voltage},i} + \kappa_4 \sum_{i=1}^{F} C_{\text{imbalance},i}$$

Fonte: Solberg (2019)

Finalmente, para treinar um agente de Aprendizado por Reforço, as stable-baselines simplificam o processo em Python, assim como o sklearn simplifica o aprendizado supervisionado. No caso de estudo específico, são usados os seguimtes processos:

from stable_baselines.common.vec_env.dummy_vec_env import DummyVecEnv

Cria um pacote vetorizado simples para vários ambientes, chamando cada ambiente em sequência no processo Python atual. Isso é útil para ambientes computacionalmente simples, como a simulação do "pêndulo oscilante", pois a sobrecarga de multithread supera o tempo de computação do ambiente. Isso também pode ser usado para métodos RL que exigem um ambiente vetorizado, mas com os quais você deseja treinar com um único ambiente.

from stable_baselines import PPO2

O algoritmo *Proximal Policy Optimization* PPO2 usa a ideia principal que, após uma atualização, a nova política não deva ficar muito distante da antiga. O PPO2 é uma implementação do OpenAI feita para GPU do PPO (OPENAI, 2022).

O PPO é um algoritmo proposto em 2017 e que apresentou um excelente desempenho quando foi implementado pela *OpenAI*. Para entender o algoritmo, primeiro é preciso entender o que é uma política. Uma política, na terminologia de Aprendizagem por Reforço, é um mapeamento do espaço de ação para o espaço de estado. Pode ser entendido como instruções para o agente do RL, em termos de quais ações ele deve tomar com base em qual estado do ambiente ele está atualmente. Quando é dito sobre avaliar um agente, geralmente significa avaliar a função de política para descobrir o desempenho do agente. É aqui que os métodos *Policy Gradient* desempenham a sua função. Quando um agente está ıaprendendoı e realmente não sabe quais ações produzem o melhor resultado nos estados correspondentes, ele faz o cálculo dos gradientes de política. Isto funciona como uma arquitetura de rede neural, em que o gradiente da saída, ou seja, o conjunto de probabilidades de ações naquele estado específico, é obtido em relação aos parâmetros do ambiente e a mudança é refletida na política, com base nos gradientes (SIDSEN99, 2022).

Embora esse método funcione bem, uma das principais desvantagens é sua extrema sensibilidade ao ajuste de hiperparâmetros (como escolha do tamanho do passo, taxa de aprendizado etc.). Ao contrário do Aprendizado Supervisionado, que tem uma rota garantida para o sucesso ou convergência com relativamente menos ajuste de hiperparâmetros, o Aprendizado por Reforço é muito mais complexo, com várias dinâmicas que precisam ser consideradas. O PPO visa encontrar um equilíbrio entre fatores importantes como facilidade de implementação, facilidade de ajuste, complexidade da amostra, eficiência da amostra e tentar calcular uma atualização em cada etapa que minimize a função de custo, garantindo que o desvio da política anterior seja relativamente pequeno. O PPO garante que a política atualizada não seja muito diferente da política antiga para manter uma baixa variação no treinamento. A implementação mais comum do PPO é por meio do modelo ator-crítico, que usa 2 redes neurais profundas, uma realizando a ação (ator) e a outra lidando com as recompensas (crítica) (SIDSEN99, 2022).

Resumindo, o PPO envolve o cálculo de probabilidades de saída, no incremento das etapas do algoritmo RL, com base em vários parâmetros e no cálculo dos gradientes para melhorar essas decisões ou probabilidades baseados nas passagens anteriores do algoritmo. Ele também garante que a política antiga e a nova política estejam pelo menos em uma certa proximidade e atualizações muito grandes não sejam implementadas.

from stable_baselines.common.policies import MlpPolicy

Este é uma política do stable_baselines que implementa objeto ator-crítico usando uma MLP (Multilayer Perceptron) de 2 camadas de 64 unidades em cada (OPENAI, 2022). Definir a política como "MlpPolicy" significa que estamos criando um vetor de estado como entrada para nosso modelo. Existem apenas outras duas opções de política, "CnnPolicy" para fornecer imagens como entrada e "MultiInputPolicy" para lidar com várias entradas (MUELLER, 2022).

A apresentação dos resultados obtidos está no Capítulo 4, Seção 4.2.

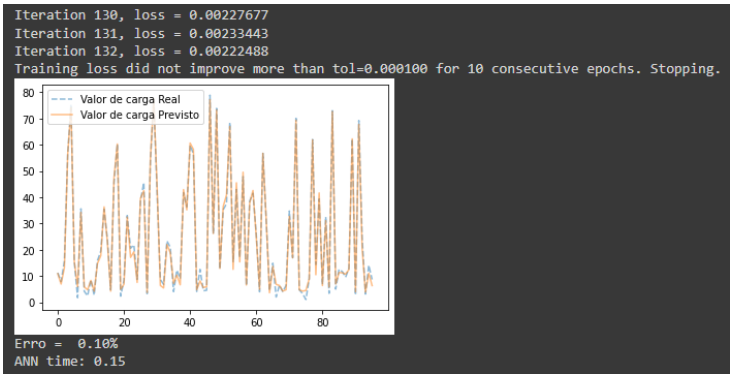
4 RESULTADOS

O objetivo de demonstrar o conjunto de funcionalidades da ferramenta pandapower em aplicações de Inteligência Artificial foi atingido, sendo demonstrados diversos usos de funções de modelamento de redes elétricas e cálculo de fluxo de potência em conjunto com as ferramentas de AI do tensorflow, gym e standard-baselines. Foram criados ambientes de simulação de diversas condições de sistemas elétricos de potência e feitas avaliações de condições de estado para estes sistemas, em busca de melhores opções de configuração e disposição de recursos.

4.1 Resultados para pandapower e MLP

O resultado da execução dos códigos da Subseção 3.2.1 foi obtido tendo sido observado um erro de 0,1%, com os valores de predição e valores reais ilustrados na Figura 37. Esse erro é medido pelo erro quadrático médio (MSE), estabelecido pela diferença entre o valor calculado e o valor previsto no conjunto de teste, e em seguida elevando ao quadrado esses resultado. Esses novos valores são somados e finalmente divididos pelo número de itens calculados.

Figura 37 – Multi-layer Perceptron Regressor Executada no Exemplo



Fonte: Elaboração Própria

Depois deste treinamento da rede, com o erro em um nível satisfatório, ela pode ser utilizada como uma ferramenta para classificação de novos dados. Para isto, a rede deverá ser utilizada apenas no modo progressivo (*feed-forward*), ou seja, novas entradas são apresentadas à camada de entrada, são processadas nas camadas intermediárias e os resultados são apresentados na camada de saída sem a retropropagação do erro. A saída apresentada é o modelo dos dados, na interpretação da rede. Uma rede MLP deste tipo é uma "caixa preta", na qual quase não se sabe porque a rede chega a um determinado resultado, uma vez que os modelos não apresentam justificativas para suas respostas. Para a extração de conhecimento de redes neurais artificiais e criação de procedimentos explicativos, é necessário o uso de outro tipo de algoritmo.

4.2 Resultados para pandapower, gym e stable-baselines com Aprendizado por Reforço

Mesmo com algumas limitações encontradas na situação de mudanças de funções e procedimentos nas bibliotecas tensorflow, gym e standard-baselines, foi possível reestabelecer a funcionalidade do conjunto de funções e métodos de Solberg (2019), as suas bases de dados originais e executar algumas funções de treinamento de Aprendizado por Reforço em sistemas elétricos de potência.

A utilização dos recursos das bibliotecas gym e stable_baselines para sistemas com grandes quantidades de elementos pôde ser estudada e simulada. Porém, ainda seria necessário um tempo considerável para reconstruir os processos nas bibliotecas Python atualizadas.

4.3 Objetivos do Trabalho

Objetivos alcançados:

- Estabelecer a técnica para uso do powerpandas;
- Definir a especificação de requisitos para diferentes projetos;
- Parcialmente alcançado: Avaliar por meio de estudo de caso a especificação de requisitos.

Foi possível demonstrar os resultados das ferramentas pandapower em cálculos e funções de manipulação de dados em sistemas elétricos de potência, bem como sklearn, gym e stable_baselines em aplicações de Inteligência Artificial, na previsão de comportamentos e tomada de decisões utilizando-se destes dados.

5 CONCLUSÃO

Conforme demonstrado neste estudo, a possibilidade de uso das ferramentas de Aprendizado por Reforço gym e stable_baselines, em conjunto com *datasets* de sistemas de potência tratados através do powerpandas, oferece uma ampla variedade de novas opções para estudo e controle de redes de energia. Neste contexto, o desenvolvimento de soluções que surgem de forma inesperada, onde um agente aprende a atingir uma meta em um ambiente incerto, é potencialmente vantajoso nestes cenários complexos. Como exemplo destes cenários complexos, podem ser previstas aplicações de melhoria de Eficiência Energética no posicionamento de estações de geração de energia fotovoltaica ou eólica, ou no uso de aprendizado de máquina para reduzir o consumo de energia em determinadas circunstâncias do sistema como descrito em Solberg (2019).

5.1 Limitações

A evolução das bibliotecas de AI do Python e a consequente substituição de métodos e funções levou a alguns problemas de compatibilidade com as versões mais novas destas bibliotecas. Porém, isso também representa uma oportunidade de projeto futuro, retrabalhando este material em função das novas funcionalidades e removendo inconsistências e funções depreciadas. A falta de redes modeladas em dados do Sistema Elétrico brasileiro também levou à necessidade de uso de bibliotecas de dados europeias para desenvolvimento do estudo. Igualmente, isso leva à outra possibilidade que seria modelar redes com dados do Sistema Elétrico nacional, ou meso dos campi da UTFPR, uma vez que os mesmos possuem redes com geração fotovoltaica.

5.2 Trabalhos Futuros

Como sugestão de trabalhos futuros baseados neste material:

- Modelagem das redes de energia mistas com fornecimento de energia externo e geração fotovoltaica própria da UTFPR, para fins didáticos. Todos os Campi, inclusive Dois Vizinhos, já possuem redes de energia fotovoltaicas que podem ser aproveitadas para gerar material de estudo para trabalhos acadêmicos. As localidades Dois Vizinhos e Francisco Beltrão, principalmente, seriam casos de interesse por possuírem diversos pontos de transformação e linhas aéreas de grande percurso, bem como geração fotovoltaica. Os estudos baseados nestas redes poderiam embasar análises de produção de energia fotovoltaica e economia de energia fornecida por concessionárias, entre outras possibilidades.
- Retrabalho dos processos mostrados no trabalho de Solberg (2019) para compatibilizar com as novas versões das bibliotecas de ML Python.

Referências

BHATIA, A. S. et al. Reinforcement learning. In: _____. **Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications**. [S.l.: s.n.], 2020. p. 281. ISBN 9781119654834. Citado na página 37.

BICHELS, A. **Sistemas Elétricos de Potência: Métodos de Análise e Solução**. EDUTFPR, 2018. 466 p. ISBN 978-85-7014-208-5. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/4610>. Citado 9 vezes nas páginas 14, 16, 22, 23, 24, 27, 28, 29 e 30.

CHELLA, M. **Dados de montagem do dataset para o algoritmo de RL**. [S.l.]: Google, 2022. <https://drive.google.com/file/d/1wtp0POuQAawglwonS2RuiencUHXiazrP/view?usp=sharing>. Citado na página 54.

CHELLA, M. **Execução da ANN com dados do simbench**. [S.l.]: Google, 2022. <https://colab.research.google.com/drive/1ikzLnXNRKhHZIKEC_ImxAb_hxpB1-2Zl?usp=sharing>. Citado na página 45.

CHELLA, M. **Execução do RL com dados do CIGRE/pandapower**. [S.l.]: Google, 2022. <https://colab.research.google.com/drive/1vBMYbBbqHi_nBpI4o-4grYBnd2Yzn-PK?usp=sharing>. Citado na página 54.

CORMEN, T. H. et al. **Introduction to Algorithms**. The MIT Press, 2001. ISBN 9780262531962. Disponível em: <https://mitpress.mit.edu/9780262531962/>. Citado na página 38.

DULHARE UMA N., A. K.; AHMAD, K. A. B. **Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications**. [S.l.]: John Wiley Sons, Inc, 2020. 512 p. ISBN 9781119654834. Citado 4 vezes nas páginas 36, 37, 38 e 44.

FRAUNHOFER. **CIGRE Networks**. [S.l.]: Fraunhofer IEE and University of Kassel, 2019. <https://pandapower.readthedocs.io/en/v2.0.1/networks/cigre.html>. Citado na página 44.

FRAUNHOFER. **ConstControl e Controles Pré Definidos no pandapower**. [S.l.]: Fraunhofer IEE and University of Kassel, 2019. <https://pandapower.readthedocs.io/en/v2.2.2/control/controller.html>. Citado na página 46.

FRAUNHOFER. **Power System Test Cases**. [S.l.]: Fraunhofer IEE and University of Kassel, 2019. <https://pandapower.readthedocs.io/en/v2.0.1/networks/power_system_test_cases.html>. Citado na página 41.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. [S.l.]: Bookman, 2001. Citado 2 vezes nas páginas 43 e 44.

MEINECKE, S. et al. Simbench—a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis. **Energies**, v. 13, n. 12, p. 3290, jun. 2020. Disponível em: <https://www.mdpi.com/1996-1073/13/12/3290>. Citado 4 vezes nas páginas 39, 40, 45 e 46.

MOHAMMED, M.; KHAN, M.; BASHIER, E. **Machine Learning: Algorithms and Applications**. [S.l.: s.n.], 2016. ISBN 9781498705387. Citado na página 36.

MUELLER, V. **Training RL agents in stable-baselines3 is easy**. [S.l.]: Towards Data Science Inc., 2022. <https://towardsdatascience.com/training-rl-agents-in-stable-baselines3-is-easy-9d01be04c9db>. Citado na página 61.

NIRUPAMA. Newton-raphson method to solve power flow problem | electrical engineering. 2018. Disponível em: <https://www.engineeringenotes.com/electrical-engineering/power-flow/newton-raphson-method-to-solve-power-flow-problem-electrical-engineering/25354>. Citado 4 vezes nas páginas 31, 32, 33 e 34.

OPENAI. **Stable Baselines docs**. 2022. <https://stable-baselines.readthedocs.io/en/master/index.html>. Citado 2 vezes nas páginas 60 e 61.

SCHAEFER, F. **pandapower basics and ML Tutorial**. [S.l.]: GitHub, 2020. ://github.com/FlorianShepherd/pandapower-youtube. Citado 4 vezes nas páginas 13, 47, 48 e 49.

SIDSEN99. **A Brief Introduction to Proximal Policy Optimization**. [S.l.]: GeeksforGeeks, 2022. <https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/>. Citado na página 60.

SOLBERG, V. Reinforcement learning for grid control in an electric distribution system. In: . [S.l.]: Norwegian University of Life Sciences, 2019. Citado 25 vezes nas páginas 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 34, 35, 50, 51, 53, 54, 58, 59, 63 e 64.

TEOH, T.; RONG, Z. **Artificial Intelligence with Python**. Springer, 2022. ISBN 978-981-16-8614-6. Disponível em: <https://doi.org/10.1007/978-981-16-8615-3>. Citado 2 vezes nas páginas 35 e 36.

THURNER, L. et al. Pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. **IEEE Transactions on Power Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 33, n. 6, p. 6510–6521, nov 2018. Disponível em: <https://doi.org/10.1109%2Ftpwrs.2018.2829021>. Citado na página 11.

TOURQUI, D. E.; BENAKCHA, M.; ALLAOUI, T. Improving the electrical stability by wind turbine and upfc. In: HATTI, M. (Ed.). **Artificial Intelligence in Renewable Energetic Systems**. Cham: Springer International Publishing, 2018. p. 121–132. ISBN 978-3-319-73192-6. Citado na página 12.

ZINAMAN, O.; SADAMOR, K. **Status of Power System Transformation 2019**. IEA Publications, 2019. Disponível em: <http://www.iea.org/>. Citado na página 11.