UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CAMPUS DE ILHA SOLTEIRA

CAROLINE BELTRAME SAKAE

LÓGICA FUZZY APLICADA NA ANÁLISE DE SOLOS DEGRADADOS UTILIZANDO LINGUAGEM DE PROGRAMAÇÃO PYTHON

Ilha Solteira
2025

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

CAROLINE BELTRAME SAKAE

LÓGICA FUZZY APLICADA NA ANÁLISE DE SOLOS DEGRADADOS UTILIZANDO LINGUAGEM DE PROGRAMAÇÃO PYTHON

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de **Engenheira Eletricista**

Profa. Dra. Suely Cunha Amaro Mantovani

Orientadora

Ilha Solteira

2025

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S158l

Sakae, Caroline Beltrame.
Lógica Fuzzy aplicada na análise de solos degradados utilizando linguagem de programação Python / Caroline Beltrame Sakae. -- Ilha Solteira: [s.n.], 2025
52 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2025

Orientador: Suely Cunha Amaro Mantovani

Inclui bibliografia

1. Lógica Fuzzy. 2. Linguagem de programação Python. 3. Solos degradados. 4. Análise de solos.

Sandra Montibeller - CRB-8/060

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte sete dias do mês de junho do ano de dois mil e vinte e cinco, a discente **CAROLINE BELTRAME SAKAE** matriculada sob o nº **172054923**, tendo como banca examinadora a sua orientadora, *a* Profa. Dra. Suely Cunha Amaro Mantovani, a Profa. Dra. Anna Diva Plasencia Lotufo e o Dr. Lucas do Carmo Yamaguti apresentou o Trabalho de Graduação intitulado "**LÓGICA FUZZY APLICADA NA ANÁLISE DE SOLOS DEGRADADOS UTILIZANDO LINGUAGEM DE PROGRAMAÇÃO PYTHON**", obtendo a nota 8,0 (oito) e conceito APROVADO.

Suely Cunha Amaro Mantovani
Profa. Dra. Suely Cunha Amaro Mantovani
- Orientadora -

Caroline Beltrame Sakae
Caroline Beltrame Sakae
- Discente -

Prof. DrA. Anna Diva Plasencia Lotufo
- Membro da Banca -

Lucas do Carmo Yamaguti
Dr. Lucas do Carmo Yamaguti
- Membro da Banca -

DEDICATÓRIA

Dedico este trabalho à minha família que sempre me incentivou no caminho dos estudos e em especial à minha mãe, que sempre me aplaudiu tão alto que eu nunca percebi quem não aplaudiu.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me conceder forças e sabedoria para trilhar esses últimos anos e concluir esse trabalho.

À minha mãe, Silvia Adriana Beltrame, por todo o apoio, amor e incentivo dedicado a mim em todas as etapas da minha vida, em especial na graduação. Obrigada por acreditar mais em mim do que eu mesma.

Aos meus tios, Silvana e Rogerio Gouveia e minha prima Rafaela Gouveia. Era em nossos momentos em família que eu encontrava descanso e forças para encarar mais uma semana, mais um semestre, mais um ano. Essa conquista também é de vocês.

À minha vó, Cleuza Nozela Beltrame que realiza esse sonho comigo lá do céu.

Ao Guilherme Castro por toda a paciência, companheirismo, amor, apoio e por trazer leveza para os meus dias. Obrigada por ter trilhado esses últimos anos comigo.

À família Castro (Roberto, Mônica, Letícia, Isabela, Victor, Helena e Aurora), pelo inestimável apoio, pela generosidade e por sempre me fazerem sentir em casa.

Aos amigos e colegas feitos ao longo dessa jornada, em especial à Bruna Moreira da Silva, minha irmã que a UNESP me deu, companheira em laboratórios, semanas de prova e estudos durante todo esse processo.

Aos meus professores e mestres dessa jornada, por todos os ensinamentos, em especial à minha orientadora, Profª. Drª. Suely Cunha Amaro Mantovani, meu eterno agradecimento pela paciência, competência e apoio, essenciais para a conclusão desse trabalho.

“Se você não tivesse capacidade, Deus não te daria a oportunidade. Seus medos você já conhece, experimente suas coragens”. Santa Terezinha

RESUMO

A proposta deste trabalho é o desenvolvimento de um algoritmo utilizando a lógica *Fuzzy* em linguagem de programação Python para análise de solos degradados, visto que a agricultura é um dos principais pilares da economia brasileira, sendo importante o entendimento e conhecimento do tipo de solo e sua análise para uma correção mais assertiva, colaborando para o aumento da produtividade agrícola no país. Para isso, são utilizados os parâmetros do solo profundidade e tratamento como entradas de dados do algoritmo de Lógica *Fuzzy* e macroporosidade, microporosidade e densidade do solo, como parâmetros de saídas. O algoritmo criado para Lógica *Fuzzy* utiliza a plataforma Google Colaboratory ou Colab, software gratuito disponível de forma online pelo Google (na nuvem), portanto, não dependendo da capacidade de processamento do computador do usuário. O programa desenvolvido no Google Colab na linguagem Python utiliza algumas bibliotecas, entre essas a scikit-fuzzy que contém todo o processo *Fuzzy*. A visualização dos resultados é realizada por meio de gráficos 3D obtidos pelo algoritmo criado que são comparados ao do Toolbox do MATLAB.

Palavras-chave: Solos degradados; Linguagem de programação Python; Lógica Fuzzy; Macroporosidade; Microporosidade; Densidade; Análise de solos.

ABSTRACT

The proposal of this work is the development of an algorithm using *Fuzzy* logic in Python programming language for the analysis of degraded soils, since agriculture is one of the main pillars of the Brazilian economy, and it is important to understand and know the type of soil and its analysis for a more assertive correction, contributing to the increase of agricultural productivity in the country. For this purpose, the soil parameters depth and treatment are used as input data for the *Fuzzy* logic algorithm, and macroporosity, microporosity and soil density are used as output parameters. The algorithm created for *Fuzzy* Logic uses the Google Colaboratory or Colab platform, free software available online by Google (in the cloud), therefore, it does not depend on the processing capacity of the user's computer. The program developed in Google Colab in the Python language uses some libraries, including scikit-fuzzy, which contains the entire *Fuzzy* process. The results are visualized through 3D graphs obtained by the algorithm created, which are compared to those of the MATLAB Toolbox.

Keywords: Degraded soils; Python Programming Language; *Fuzzy* logic; Macroporosity; Microporosity; Density; Soil analysis.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A vida na terra depende do solo, mas o aquecimento global, desmatamento das florestas tropicais e a poluição da água, são alguns dos fatores que impactam os ecossistemas terrestres e determinam a sua qualidade, a natureza dos ecossistemas das plantas e a capacidade da terra em sustentar a vida animal e a dos seres humanos (STEFFEN *et al.*, 2024).

Com a maioria da população vivendo nas cidades, perde-se de vista o quanto se depende do solo para o desenvolvimento da população e a sobrevivência. Além de suprirem com quase todo o alimento, o solo ainda é responsável por itens do nosso vestuário que se originam das plantas, fibras que são utilizadas na fabricação de papel. Portanto, o solo é fundamental para a agricultura, pois fornece o suporte físico, nutrientes e água necessários para o crescimento das plantas, atua na regulação do clima e na ciclagem de nutrientes, (CAVALCANTI, 2023).

Um solo fértil e saudável garante não somente a produção de alimentos, mas a exportação de produtos, principalmente produtos ligados à agropecuária. Além da exportação de produtos derivados do solo, outra fonte importante de renda no Brasil é o cultivo da cana-de-açúcar, que apresentou um crescimento elevado em termos de área plantada em 2023, 10 milhões de hectares no Brasil foram direcionados à plantação da cana de açúcar (IBGE, 2023). Devido a essa intensa exploração do solo, é importante que tecnologias atuais de tratamento e análise possam ser aliadas à agricultura para obter uma maior e melhor produtividade.

Um problema sério e recorrente na agricultura é a degradação do solo que ocorre por fatores diversos como a agricultura intensiva, a falta de cuidado com o solo e a falta de práticas sustentáveis. Isso acontece também, pela intensa procura por resultados rápidos, alta rentabilidade e crescimento econômico, levando a um uso indiscriminado do solo, acarretando na sua infertilidade. É necessário utilizar os recursos naturais de maneira adequada para gerar um sistema ecológico sustentável e que beneficie o meio ambiente, aumentando a produção de forma saudável.

Nas pesquisas que envolvem a área de ciências agrárias, normalmente, os dados obtidos em campo ou laboratório sobre o solo, são analisados por meio da estatística. Estes estudos consideram muitas indecisões, entre elas, profundidade do solo em função do tratamento usado, a técnica de manejo utilizada para cada tipo de

solo, entre outros. Por outro lado, outras ferramentas de análise dos solos, podem ser usadas, tais como os algoritmos de Inteligência Artificial, entre esses a Lógica *Fuzzy,* que trata muitas variáveis e incertezas, caso da análise dos solos (DE OLIVEIRA *et a*l, 2016).

A lógica *fuzzy* foi desenvolvida com o objetivo de modelar o raciocínio humano por meio do tratamento de informações, em um ambiente de incerteza e imprecisão, visto que a cada dia aparecem mais parâmetros e informações que não são de natureza objetiva, de forma que com a utilização da Lógica F*uzzy,* podem ser obtidas respostas aproximadas para uma questão baseada em um conhecimento que é inexato, incompleto ou não totalmente confiável (ZADEH, 1965).

Modelos de Inteligência Artificial têm sido utilizados com maior frequência na área da agronomia, por exemplo Redes Neurais visto em Bonini Neto (2014). Técnicas como sistema de Inferência F*uzzy* são encontradas nos estudos de Silva *et al* (2010) que realizam a análise da fertilidade do solo para plantações específicas do café Conilon, e também em Godinho, Canappele e Gasparoto (2021) que utilizaram Lógica *Fuzzy* para otimizar a aplicação de fertilizantes no rabanete.

Quando se estuda a degradação do solo, os parâmetros importantes são a macroporosidade e a microporosidade, visto que a relação de macroporos e microporos é diretamente proporcional à fertilidade e boa colheita. Nos macroporos ocorre o livre movimento do ar e água e nos microporos, o movimento do ar fica embaraçado e o da água fica restrito ao movimento capilar. Portanto, um solo ideal seria uma mistura dos dois (KIEHL, 1979).

1.1 Objetivos

Nesta pesquisa tem-se como objetivo o desenvolvimento de um algoritmo que utiliza a Lógica *Fuzzy* programada na linguagem Python, para a análise e avaliação da degradação do solo. Baseado nos parâmetros físicos do solo, são definidas as funções de pertinência das entradas (fuzzificador) como sendo Tratamento e Profundidade, um conjunto de regras que relacionam as entradas e saídas desejadas, um método de Inferência F*uzzy* e um modelo matemático de saída (defuzzificador) como sendo a macroporosidade, microporosidade e a densidade do solo, sendo utilizada a plataforma de software Google Colaboratory (Google Colab). Os resultados do algoritmo *Fuzzy* são comparados para a sua validação, aos

resultados do mesmo sistema desenvolvido no Toolbox do MATLAB (MATHWORKS, 2025).

1.2 Organização do Texto

Depois desta introdução o restante do texto foi estruturado em seis capítulos. É apresentado no segundo capítulo, a revisão de literatura, contendo os principais trabalhos, relacionados a Lógica *Fuzzy* e ao estudo sobre as variáveis do solo e sua relação com a fertilidade, os quais forneceram o embasamento para o desenvolvimento do algoritmo *Fuzzy.*

No capítulo 3 são apresentados os conceitos importantes para a criação do algoritmo da lógica *Fuzzy*, conceitos da própria lógica, linguagem de programação Python e o Google Colab plataforma escolhida para o desenvolvimento do algoritmo e os atributos para análise do solo.

O sistema *Fuzzy* que foi desenvolvido para o problema de solos degradados é apresentado no capítulo 4, iniciando pela escolha das funções de pertinência, em seguida as variáveis linguísticas, a base de regras, e o método de defuzzificação.

Apresentam-se no capítulo 5, os resultados obtidos e a comparação entre o método criado usando a linguagem de programação Python e os resultados do mesmo problema realizado no Toolbox do MATLAB.

No capítulo 6 tem-se a conclusão, seguido pelas referências bibliográficas e os apêndices.

2 REVISÃO DE LITERATURA

Alguns artigos que abordam a análise de solos degradados usando a Lógica *Fuzzy* foram estudados. Nesta revisão de literatura são apresentados três trabalhos que forneceram o conhecimento necessário para embasar esta pesquisa, tanto sobre o desenvolvimento do algoritmo da Lógica *Fuzzy*, quanto aos parâmetros a serem analisados.

Em De Oliveira *et al* (2016), descrevem em seu trabalho a utilização da Lógica *Fuzzy* para a análise de solos degradados. Para a análise e a obtenção dos resultados, utilizaram a Toolbox do MATLAB (Fuzzy Logic Designer), e como método de inferência o Mamdani. Com as regras definidas com o auxílio de um especialista, foram variadas as entradas, tratamento e profundidade, obtendo-se três variáveis de saída, macroporosidade, microporosidade e densidade do solo. Ao final, conclui-se que a metodologia utilizada fornece o comportamento de todos os dados analisados, em função dos dados de entrada.

No trabalho de Silva *et al* (2010) foi descrita a utilização da Lógica F*uzzy* para a análise de fertilidade de uma área experimental e sua relação com a produtividade do café conilon. Utilizou para isso, a geoestatística e o sistema de classificação *Fuzzy*, com base em atributos químicos do solo. O estudo foi realizado na fazenda experimental do INCAPER-ES, onde foram coletadas amostras do solo na profundidade de até 0,2 m, sendo analisados os atributos: fósforo, potássio, cálcio e magnésio, alumínio, soma de bases, capacidade de troca catiônica a pH 7 e saturação por bases. Ao final, os dados foram submetidos a uma análise descritiva, e a análise geoestatística, sendo que foi utilizado um sistema de classificação *Fuzzy* baseado nestes atributos, para inferir sobre a fertilidade do solo e sua relação com a produtividade da cultura. Como resultado, foi possível mapear de forma mais assertiva as mudanças das classes de fertilidade, visto que a Lógica *Fuzzy* utiliza os graus de pertinência para análise, ao invés de uma classificação exata.

No trabalho de Roveda e Lourenço (2011), a Lógica *Fuzzy* foi usada para analisar a permeabilidade de solos da baixada santista em uma região impactada, com coletas realizadas em solos das áreas urbanas e periféricas da baixada santista, sendo utilizados dois métodos de defuzzificação distintos para a verificação dos resultados obtidos, sendo eles o método centróide e a média dos máximos. Foram definidas 66 regras para este estudo, foram obtidos como resultados a

constatação de que a metodologia *Fuzzy* mostra-se viável para o estudo de permeabilidade, baseando-se na análise de que o método de defuzzificação, centróide

Com o auxílio dos trabalhos apresentados nesta revisão obteve-se os conhecimentos necessários para a criação do programa, a escolha da linguagem de programação Python e a utilização da Lógica *Fuzzy* para análise dos parâmetros de classificação do solo e a comparação com a Toolbox do MATLAB para validação dos resultados.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos teóricos sobre as ferramentas, como linguagem de programação, o Google Colab, conceitos utilizados na criação do algoritmo da Lógica *Fuzzy*, e os conceitos relacionados a degradação do solo.

3.1 Lógica *Fuzzy*

A Lógica *Fuzzy*, introduzida por Zadeh em 1965, é uma linguagem matemática que busca entendimento na incerteza. Partindo do próprio nome, *Fuzzy* é traduzido como sendo nebuloso (impreciso). O algoritmo é construído para sistemas ou problemas em computadores com muitas variáveis e incertezas, um exemplo, a informação do clima, que é uma informação imprecisa, pois o que pode ser frio para uma pessoa, pode não ser para outra pessoa, contrário a lógica Booleana tradicional, onde se tem apenas 0 ou 1, verdadeiro ou falso como resultados.

A aplicação da Lógica *Fuzzy* é cada vez mais comum em eletrodomésticos “Inteligentes” capazes de fazer uma escolha automática em processos com muitas variáveis (SAMSUNG, 2024), e navegação de robôs que usando sensores e Lógica *Fuzzy* obtém uma decisão mais assertiva, como no trabalho de Peixoto (2022).

Para melhor entendimento das diferenças entre a lógica *Fuzzy* e a lógica matemática convencional, supõe-se uma função de pertinência $\mu_A(x)$ do conjunto A dada pela Equação 1,

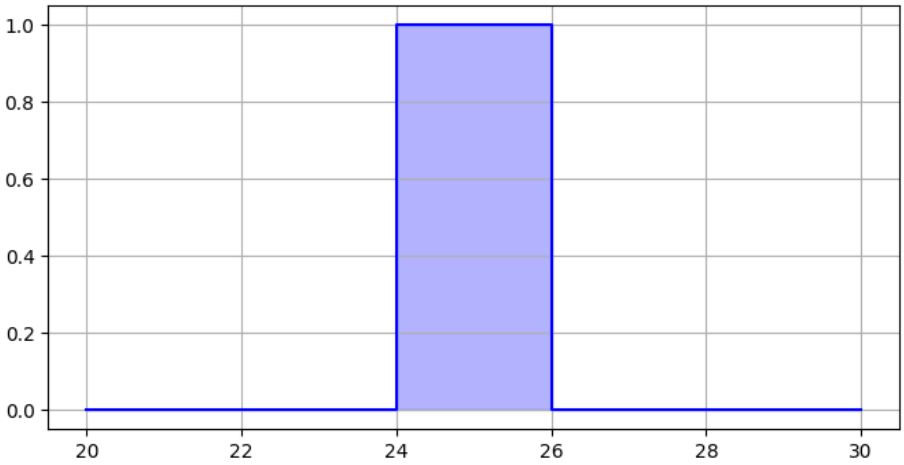
$$\mu_A(x) = \begin{cases} 1, & se\ x \in A \\ 0, & se\ x \notin A \end{cases} \quad (1)$$

Sendo x membro do conjunto A, quando $\mu_A(x) = 1$.

Pela definição do conjunto ele assume apenas valores com fronteiras bem definidas. Zadeh (1965) considera a limitação que esse tipo de conjunto apresenta para manusear problemas e situações em que a passagem de uma região para a outra não ocorrem de maneira abrupta, e sim em uma transição gradual.

Tomando como exemplo uma situação em que é analisada a temperatura, conforme mostrado no gráfico da Figura 1 e descrita pela função de pertinência, dada na Equação 2,

Figura 1 - Representação real - Função de Pertinência x Variável Temperatura, ^{0}C



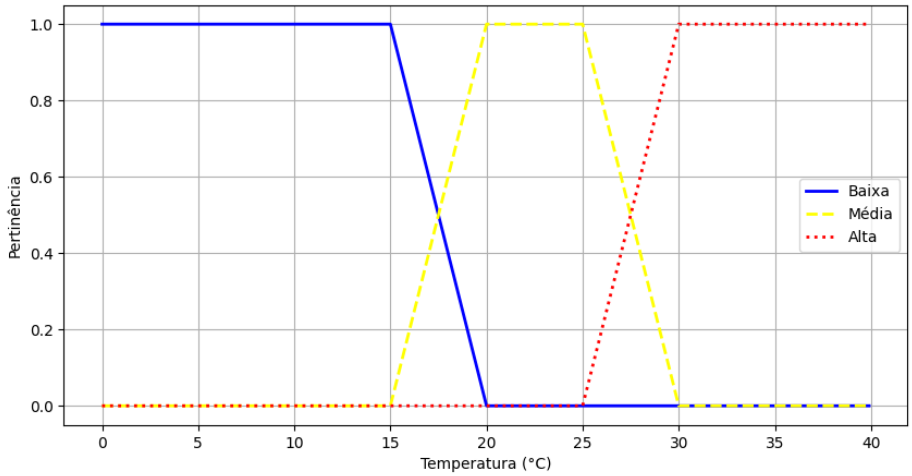
Fonte: Própria do autor.

$$\mu_N(x) = \begin{cases} 0, & x < 24 \\ 1, & 24 \leq x \leq 26 \\ 0, & x > 26 \end{cases} \quad (2)$$

Pela representação da Figura e da equação tem-se que o clima seria classificado de forma abrupta, ou seja, a mudança de quente para frio e de frio para quente seria de forma grosseira, sem fronteiras definidas, no método tradicional. Na Lógica *Fuzzy*, a análise de temperatura seria mais bem representada utilizando uma função trapezoidal ou triangular, por exemplo, como na Figura 2, descrita pela Equação 3,

$$T(temperatura) = baixa, média, alta \quad (3)$$

Figura 2 - Representação *Fuzzy* para a temperatura



Fonte: Própria do autor.

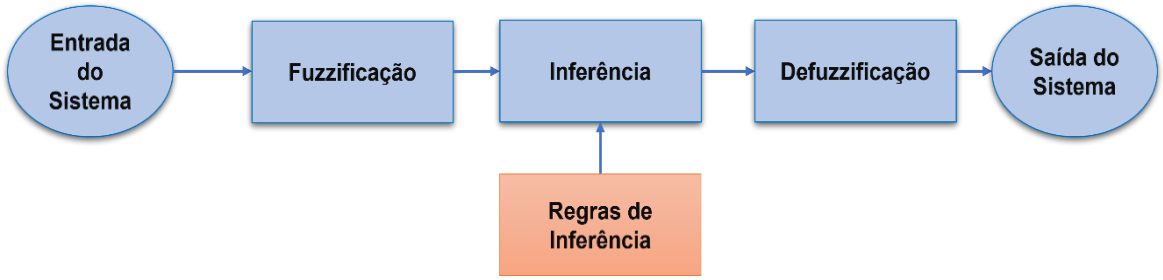
Portanto, para a análise de temperatura, os graus de pertinência, se assemelham mais ao pensamento humano, onde uma temperatura de 26 ^{0}C pode ou não ser quente dependendo do local e da pessoa.

Para este trabalho de pesquisa, essa transição gradual é importante, visto que para a análise de solos, existe uma faixa de valores para cada variável do solo que pode ser representada em transições graduais buscando uma solução mais eficaz.

3.1.1 Sistema de Inferência *Fuzzy*

O sistema *Fuzzy* é formado pelos blocos apresentados na Figura 3, mostrando cada etapa do processo.

Figura 3 - Diagrama de blocos de um Sistema *Fuzzy*



Fonte: Própria do autor.

Neste diagrama tem-se como entrada do sistema um valor real (ou crisp), que se deseja analisar. É nessa etapa que são definidos juntos a um especialista, as funções de pertinência, as variáveis linguísticas e o tipo de função de pertinência para as entradas e saídas, triangular, trapezoidal, gaussianas e as variáveis linguísticas, definidas por uma quíntupla:

N: nome da variável; T(N): conjunto de valores linguísticos; X: universo de discurso; G: regra sintática para gerar valores de N como uma composição de T(N); M: regra semântica para associar os valores (TANSCHEIT, 2003).

Para o exemplo apresentado nas Figuras (1) e (2), a quíntupla seria definida da seguinte forma:

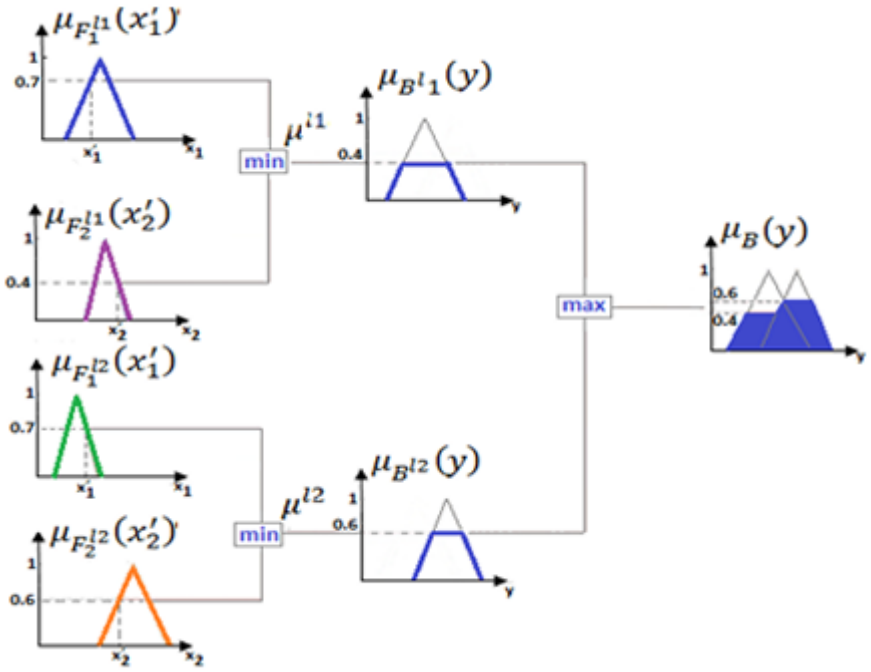
N: Temperatura; T(N): {baixa, média e alta}; X: 0° a 40°C (universo de discurso do exemplo); G: Temperatura fresca; M: Regra que associa o valor de G a um conjunto *Fuzzy* e sua função de pertinência.

.

Na etapa de fuzzificação, esse valor real se transforma em uma entrada *Fuzzy*, consiste da análise do problema para a definição de todas as variáveis, funções de pertinência e variáveis linguísticas. Ao passar pelo fuzzificador, um valor real se transforma em uma entrada *Fuzzy*, sendo então criadas as regras do tipo condicionais (RIZOL; DIAS, 2014).

A base de regras é construída de acordo com especialistas, combinando todas as entradas possíveis do sistema do tipo SE-ENTÃO. O controlador *Fuzzy* terá um bom desempenho se as regras forem consistentes. Os cálculos são realizados no bloco de **inferência**, onde é definido um sistema de inferência sendo que o mais utilizado é o método Mamdani ou método dos máximos e mínimos. O método consiste em obter o mínimo dos graus de ativação das regras ativadas. Essas funções mínimas obtidas são analisadas e o resultado utilizando é o máximo das funções resultantes, conforme na Figura 4.

Figura 4 – Exemplo método de inferência Mamdani



Fonte: Rizol e Dias, 2014.

Nesta Figura estão representadas as regras ativadas, o novo conjunto é formado pelo mínimo entre elas e a combinação *Fuzzy* final passará pela defuzzificação centróide (Rizol e Dias, 2014).

Além do método de inferência Mamdani, existe também o método Takagi-Sugeno, esse método produz como saída uma função matemática, não necessitando de defuzzificação.

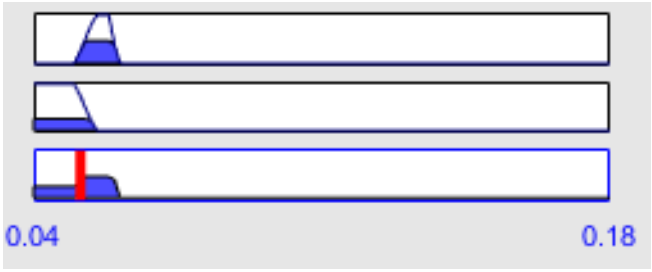
Por fim, a última etapa a defuzzificação, que consiste em transformar novamente os valores obtidos que são valores *Fuzzy* em valores reais. Nessa etapa, assim como o método de inferência, pode ser realizada de diversas formas. O método de defuzzificação centroide, ou centro de área, um dos mais utilizado, consiste em obter uma média de todas as áreas que representam o grau de pertinência do conjunto *Fuzzy* analisado. Para o domínio discreto, a expressão para o método de defuzzificação pela Equação 4,

$$G(B) = \frac{\sum_{i=0}^{n} u_i \varphi B(u_i)}{\sum_{i=0}^{n} \varphi B\,(u_i)} \tag{4}$$

sendo, u_i o valor de saída; $\varphi B(u_i)$ o grau de pertinência da saída no ponto u_i; n o número de pontos considerados e o valor final G(B) é um valor real.

Uma ilustração do método de defuzzificação centróide é apresentado na Figura 5, que ilustra a ativação de duas regras (primeira duas linhas), e na última linha dá o resultado, indicado pela linha vermelha, em torno de 0.0513:

Figura 5 – Exemplo defuzzificação centróide.



Fonte: Própria do autor.

3.2 Linguagem de Programação Python

Python é uma linguagem de programação criada em 1982 na cidade de Amsterdã na Holanda por Guido Van Hossum, que tinha como objetivo fazer uma linguagem de programação com um entendimento mais simples do que a linguagem C. Por ser uma linguagem de propósito geral, o Python se torna uma linguagem versátil para várias aplicações (SILVA; SILVA, 2019).

A linguagem de programação Python foi evoluindo ao longo do tempo, de forma que atualmente, a linguagem é uma das mais utilizadas no mundo todo, instalada de fábrica em alguns sistemas operacionais, utilizada para o desenvolvimento de jogos e até filmes com computação 3D, além de ser aplicada em grandes empresas como Yahoo, NASA, Intel, IBM, etc (SILVA; SILVA, 2019).

A popularidade e demanda pela linguagem de programação Python pode ser explicada pelo seu amplo uso em pesquisas e desenvolvimento de sistemas, tendo como principal vantagem suas bibliotecas, que podem ser utilizadas nos códigos de acordo com as necessidades, como a biblioteca NumPy para realização de cálculos numéricos, a biblioteca Matplotlib, utilizada para visualização gráfica, a biblioteca scikit-fuzzy, utilizada para o desenvolvimento de códigos em Lógica *Fuzzy*, entre outras bibliotecas que estão disponíveis, mas não são de interesse desta pesquisa.

As bibliotecas utilizadas nesse código e a maioria das bibliotecas desenvolvidas em Python podem ser encontradas no Github (GITHUBPAGES, 2024) que é uma plataforma colaborativa desenvolvida para o armazenamento e colaboração de desenvolvimento de códigos. É nela que se armazena o código fonte do sistema de Inferência *Fuzzy*, bem como exemplos e outros códigos desenvolvidos.

Por ser uma linguagem de altíssimo nível e fácil entendimento, a linguagem Python apresenta algumas diferenças básicas em sua implementação, como não usa ponto e vírgula para o encerramento de linhas de programação, sendo o final da linha indicado de forma implícita pelo caractere “\0” (SILVA; SILVA, 2019).

Para programar em Python em um PC é necessário fazer sua instalação, escolher um ambiente de desenvolvimento (como VS Code ou PyCharm), e então começar a aprender os conceitos básicos da linguagem, como variáveis, operadores, estruturas de controle e funções. No caso de trabalhar com o ambiente Google Colab não precisa de instalação. Comandos básicos do Python − Print, Input, If/Elif/Else e For, exemplo de programa, Figura 6.

Figura 6 – Exemplo de programa em Python

```
horario = "tarde"
if horario == "manhã":
 print("Bom dia!")
elif horario == "tarde": # Nesse exemplo, é aqui que a condição será verdade
 print("Boa tarde!") # Então essa é a mensagem que vai aparecer
else:
 print("Boa noite!")
```

Fonte: Própria do autor.

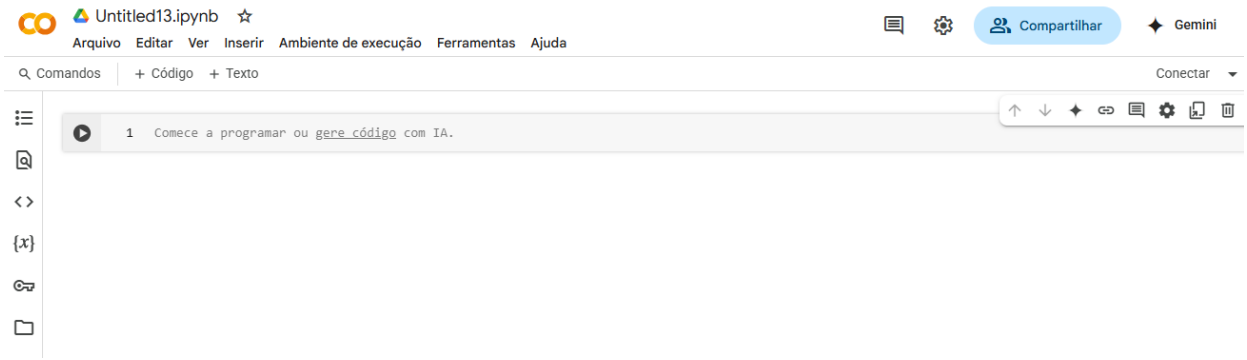
3.3 Google Colaboratory

O Google Colab é um ambiente de execução e criação de códigos na linguagem de programação Python. Foi desenvolvido de forma a fornecer acesso a pessoas de todas as classes sociais e intelectuais, sendo inicialmente gratuita e de fácil acesso.

Por ser um serviço do Jupyter Notebook hospedado online, o Google Colab não necessita configuração para uso, ou seja, não é necessário o *download* de nenhum aplicativo ou ambiente de execução na máquina. Os códigos são criados no próprio site do Google Colab, e armazenados automaticamente no Google Drive, de forma que ao executar o programa criado, o usuário está utilizando na verdade o hardware Google, sendo o processamento independente da capacidade da máquina do usuário, garantindo ainda mais, a sua acessibilidade e inclusão.

A programação no Google Colab é realizada dentro dos notebooks, que são ambientes de desenvolvimento que suportam textos, imagens e permitem a combinação de textos com a programação em si. Permite a importação de dados como planilhas do Google Drive e até mesmo códigos do Github. Por ser armazenado dentro do Drive, o compartilhamento com outras pessoas fica mais simples, sendo possível o desenvolvimento de códigos em colaboração com outras pessoas. A interface do Google Colab pode ser vista na Figura 7.

Figura 7 - Interface do Google Colaboratory.



Fonte: Própria do autor.

Pela sua facilidade e praticidade, o Google Colaboratory tem sido cada vez mais utilizado no meio científico e acadêmico, possuindo aplicações para Machine Learning, como desenvolvimento e treinamento de Redes Neurais.

Por utilizar GPUs (Unidades de Processamento Gráfico) do próprio Google, e por buscar uma acessibilidade maior a várias pessoas, o Google Colab pode apresentar certa limitação no tempo de uso gratuito, ou seja, o tempo disponível de uso do Google Colab pode variar de usuário para usuário. No plano gratuito, o uso costuma ser de 12 horas contínuas do notebook, porém, quanto mais tempo for utilizado, e maior a capacidade utilizada do GPU do Google, menor será o tempo de acesso gratuito ao Google Colab, sendo então disponibilizados planos de assinatura.

Além disso, outra limitação a ser citada no Google Colab pode ser o uso do Google Drive para armazenamento, visto que o mesmo também apresenta uma capacidade de armazenamento, sendo necessária a compra de mais espaço na nuvem para poder armazenar mais programas (GOOGLE, 2025).

3.4 Atributos para análise do solo

Como citado anteriormente, a vida na Terra depende completamente do solo. Desde a camada de ozônio, o aquecimento global e sua relação direta com o desmatamento e mau uso do solo, até o fato de comer um hambúrguer e lembrar-se de que o pão vem do trigo cultivado no solo e que a carne vem de bovinos que se alimentam de pastos cultivados no solo, tudo se relaciona com a capacidade produtiva do solo.

Um sistema ecológico não sustentável é gerado pela inadequada utilização dos recursos naturais. Solos utilizados de forma inadequada podem ser degradados. Na literatura são encontrados trabalhos de recuperação dos solos com adubos verdes estudados por Castro, Alves e Nascimento (2009), verificando os efeitos benéficos de coberturas vegetais modificam seus atributos físicos e químicos e também em Wang *et al* (2014) com o monitoramento desses atributos e seus efeitos na qualidade do solo.

Os atributos do solo são características que descrevem suas propriedades físicas, químicas e biológicas e são importantes para entender a qualidade do solo e seu potencial para diferentes usos, como agricultura e construção:

- Os atributos físicos do solo envolvem a Textura, Estrutura, Densidade, Porosidade, Capacidade de água disponível, Resistência à penetração e Percolação da água;

- Quanto aos atributos químicos tem-se o pH, Teor de matéria orgânica, Nutrientes, Capacidade de troca catiônica (CTC) e Teor de óxidos de ferro e alumínio;
- Atributos Biológicos do solo envolvem a Biomassa microbiana, Atividade enzimática e Diversidade e abundância de organismos do solo.

A avaliação e o monitoramento desses atributos do solo são fundamentais para a gestão sustentável dos recursos naturais, a produção agrícola eficiente e a proteção do meio ambiente.

Para a pesquisa em questão tem-se interesse nos atributos físicos, densidade e porosidade. A densidade é a massa do solo por unidade de volume, que pode afetar a disponibilidade de água e a resistência à penetração das raízes A porosidade é o volume de espaços vazios no solo, que são importantes para o armazenamento de água e a circulação de ar.

Um dos principais parâmetros físicos analisados no solo são os poros, visto que são eles os responsáveis pela aeração e a penetração da água. Os macroporos (maiores do que 0,08mm) são os poros que permitem a movimentação do ar e da água, enquanto os microporos (menores do que 0,08mm) são aqueles que estão ocupados por água, não permitindo uma circulação adequada do ar (BRADY; WEIL, 2013).

Quanto à densidade, está diretamente relacionada com os poros, sendo que quanto maior a densidade, menor o espaço poroso do solo, sendo que solos muito densos dificultam a penetração de raízes sendo que valores muito altos de densidade são indicativos de degradação no solo (BRADY; WEIL, 2013).

A equação que calcula a porosidade total do solo, Pt com base nos dados de densidade do solo, e de densidade de partículas D_p é dada pela Equação (5),

$$Pt(\%) = (1 - \left(\frac{D_s}{D_p}\right)) * 100 \quad (5)$$

sendo que, Pt significa a quantidade do espaço poroso.

A análise desses atributos é importante para a avaliação da qualidade do solo e as práticas de manejo aplicadas sobre o mesmo. Portanto, neste trabalho foi utilizada, além da densidade, a Microporosidade e a Macroporidade. De acordo com Bonini Neto (2014), para a determinação da Microporosidade usa-se o método da

mesa de tensão com coluna de água, a densidade pelo método do anel volumétrico e a macroporosidade foi calculada pela diferença entre a porosidade total do solo e a microporosidade, sendo a porosidade total obtida via saturação do solo. Todos os cálculos e análises foram feitos seguindo os critérios da EMBRAPA (1997).

Para calcular a microporosidade pelo método da mesa de tensão com coluna de água, de acordo com Teixeira (2017), consiste em determinar a massa de água retida em uma amostra de solo com volume conhecido, sendo as amostras não deformadas e não saturadas, colocadas em mesa de tensão, sendo então aplicada uma tensão de 0,6m de coluna de água e pesadas após o equilíbrio, e a expressão matemática que melhor representa essa metodologia é mostrada na Equação 6 (TEIXEIRA, 2017),

$$M_i = \frac{(a-b)}{c} \tag{6}$$

sendo, M_i a Microporosidade em $m^3.m^{-3}$; a é a massa do solo seco+água retida após o equilíbrio com um potencial de 60 cm de coluna de água em gramas; b é a massa do solo seco a 105°C em gramas; c é o volume total da amostra em *cm³*.

Para determinação da densidade, é usado o método do anel volumétrico (ou cilindro volumétrico), que consiste em realizar a coleta de uma amostra de solo em um cilindro metálico, sendo necessário evitar a compactação do solo no interior do cilindro, anotar, com a ajuda de um paquímetro as dimensões do cilindro para assim calcular seu volume, remover a amostra de solo para um recipiente identificado e com massa conhecida, secar a amostra de solo em uma estufa a 105°C por 48h e pesar novamente (TEIXEIRA, 2017). O cálculo para a obtenção da densidade é dado pela Equação 7,

$$D_s = \frac{m_a}{V} \tag{7}$$

sendo, D_s a densidade do solo; m_a é a massa da amostra de solo seco a 105°C; V é o volume do cilindro em *cm³*.

E para a obtenção da macroporosidade, o cálculo é dado pela Equação,

$$M_a = (Pt - M_i) \tag{8}$$

sendo, M_a a macroporosidade, Pt é a Porosidade total e M_i é a microporosidade.

Portanto, neste tópico foram apresentados os principais conceitos que fizeram parte do contexto deste trabalho auxiliando no entendimento do sistema *Fuzzy* desenvolvido para análise da degradação do solo.

4 SISTEMA FUZZY DESENVOLVIDO

Nesse capítulo é descrito o sistema *Fuzzy desenvolvido* para a aplicação no problema do cálculo de variáveis dos atributos físicos do solo, visando estimar a sua degradação, usando o ambiente Google Colab e a linguagem de programação Python.

4.1 Definição do Problema

Para desenvolver o sistema *Fuzzy* é necessário definir as variáveis de entradas, a base de regas, a defuzzificação e as variáveis de saída, relacionadas a um determinado tipo de solo. O solo utilizado nesta pesquisa foi baseado nos trabalhos de Bonini Neto (2014) e De Oliveira *et al* (2016), sendo do tipo Latossolo Vermelho-Escuro distrófico textura franco argilo-arenoso, muito profundo, rico em sesquióxidos, e os dados são provenientes de experimentos conduzidos na Fazenda de Ensino e Pesquisa, pertencente à Faculdade de Engenharia Campus de Ilha Solteira.

Definindo as variáveis de entrada reais (crisp) para o sistema, tem-se os atributos tratamento e a profundidade e como variáveis de saída, tem-se os atributos do solo macroporosidade, microporosidade e densidade.

4.2 Fuzzificação

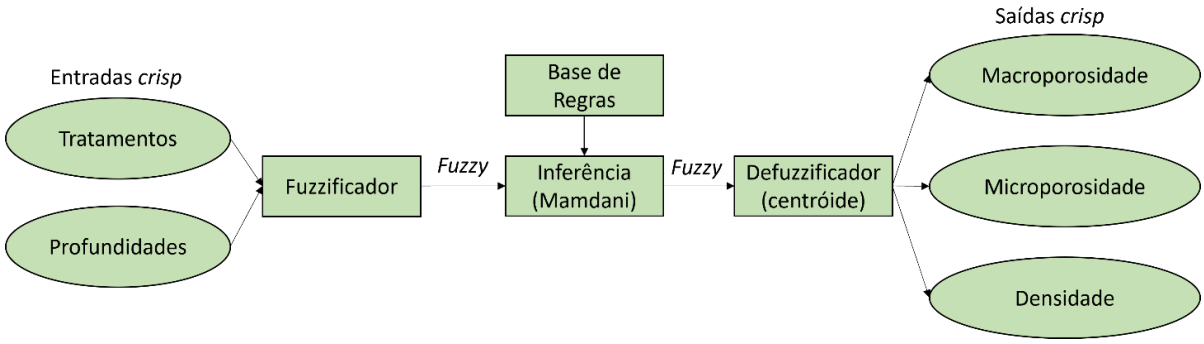
A fuzzificação engloba análise do problema, definição das variáveis de entrada e saída, definição do intervalo do universo de discurso, definição do número de funções de pertinência por variável, a definição do tipo de função de pertinência por variável de entrada e saída e por fim, a definição das variáveis linguísticas. As faixas de valores para cada atributo de entrada e saída são descritas a seguir:

- Profundidade: 0,10 a 0,40m;
- Tratamento: são considerados nove (9) tipos de tratamento (SE – solo exposto, sem técnica de recuperação, MA – vegetação nativa de Cerrado, SM – solo mobilizado, MP – Mucuna-preta, G – guandu, C/MP – calcário + mucuna-preta, C/G – calcário + guandu, C/Ge/MP – calcário + gesso + mucuna-preta e C/Ge/G – calcário + gesso + guandu).

- Macroporosidade: varia de MB (Muito baixa), M (média), A (alta) e MA (muito alta);
- Microporosidade: varia de MB (Muito baixa), B (Baixa), M (Média), A (Alta) e MA (Muito alta);
- Densidade: varia de MB (Muito baixa), B (Baixa), M (Média), A (Alta) e MA (Muito alta);

A partir desses dados, desenvolve-se as etapas de elaboração do restante do sistema F*uzzy,* utilizando como método de inferência o Mamdani. a base de regras e o defuzzificador. Para ilustrar apresenta-se um diagrama para o desenvolvimento do do algoritmo implementado na Figura 8.

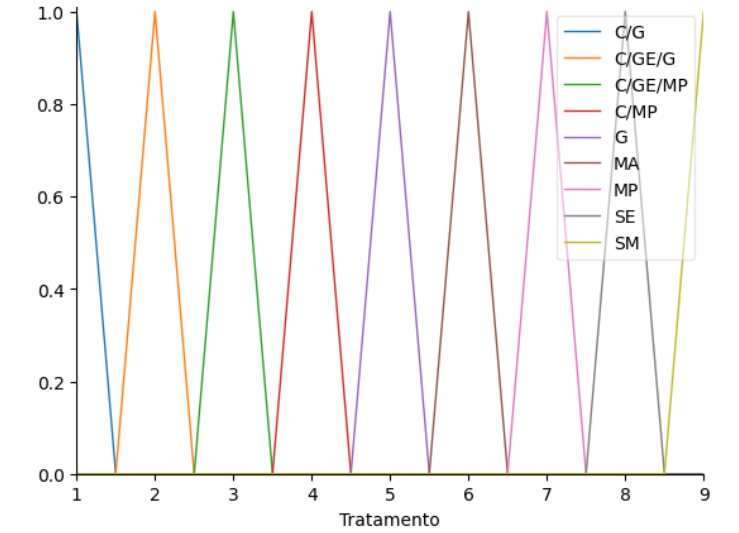
Figura 8 – Diagrama de blocos para o algoritmo *Fuzzy* para análise do solo



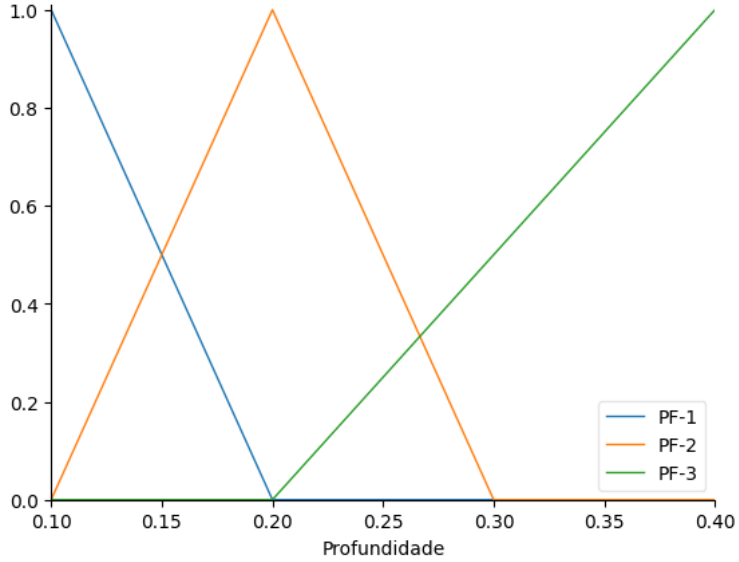
Fonte: Própria do autor.

Definidas as funções de pertinência para a entrada do fuzzificador como sendo profundidade (*p*) e tratamento (*t*), as variáveis linguísticas relacionadas a essas entradas, feita de forma subjetiva por especialistas, para a profundidade (*p*), 3 variáveis linguísticas, PF-1, PF-2 e PF-3, sendo utilizadas apenas funções triangulares. Para a função de pertinência do tratamento (*t*), foram definidas 9 variáveis linguísticas, já mencionadas, C/G, C/GE/G, C/GE/MP, C/MP, G, MA, MP, SE e SM, sendo utilizadas também apenas funções triangulares para representação. Com estas informações as funções de pertinência da entrada foram programadas em Python, cuja visualização gráfica é apresentada nas Figuras 9(a) e (b) e o programa na Figura 10.

Figura 9 –Funções de pertinência da entrada.a) Tratamento. b) Profundidade



(a)



(b)

Fonte: Própria do autor.

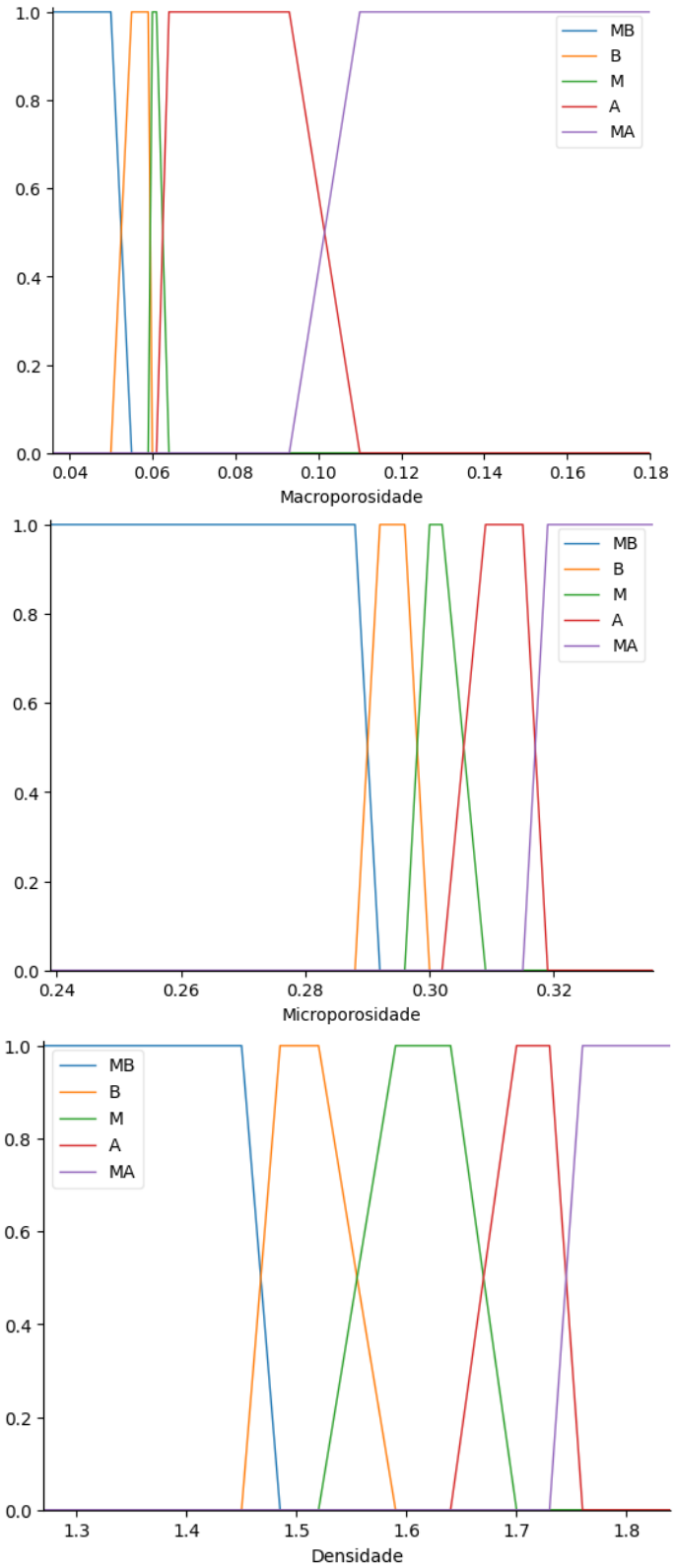
Figura 10 – Programa em Python para as funções de pertinência da entrada

```
# Criando os vetores das variáveis de entrada e saída
x_tratamento = np.arange(1,9.5,0.5)
x_profundidade = np.arange(0.1,0.4,0.02)
x_macroporosidade = np.arange(0.036,0.181,0.001)
x_microporosidade = np.arange(0.239,0.336,0.001)
x_densidade = np.arange(1.27,1.84,0.005)

# Criação das variáveis de entrada e saída
tratamento = ctrl.Antecedent(x_tratamento, 'Tratamento')
profundidade = ctrl. Antecedent(x_profundidade, 'Profundidade')
macroporosidade = ctrl.Consequent(x_macroporosidade, 'Macroporosidade')
microporosidade = ctrl.Consequent(x_microporosidade, 'Microporosidade')
densidade = ctrl.Consequent(x_densidade,'Densidade')

# Criação das variáveis linguísticas
tratamento['C/G'] = fuzz.trimf(x_tratamento,[0.5,1,1.5])
tratamento['C/GE/G'] = fuzz.trimf(x_tratamento, [1.5,2,2.5])
tratamento['C/GE/MP'] = fuzz.trimf(x_tratamento, [2.5,3,3.5])
tratamento['C/MP'] = fuzz.trimf(x_tratamento, [3.5,4,4.5])
tratamento['G'] = fuzz.trimf(x_tratamento, [4.5,5,5.5])
tratamento['MA'] = fuzz.trimf(x_tratamento, [5.5,6,6.5])
tratamento['MP'] = fuzz.trimf(x_tratamento, [6.5,7,7.5])
tratamento['SE'] = fuzz.trimf(x_tratamento, [7.5,8,8.5])
tratamento['SM'] = fuzz.trimf(x_tratamento, [8.5,9,9.5])

profundidade['PF-1'] = fuzz.trimf(x_profundidade, [0,0.1,0.2])
profundidade['PF-2'] = fuzz.trimf(x_profundidade, [0.1,0.2,0.3])
profundidade['PF-3'] = fuzz.trimf(x_profundidade, [0.2,0.4,0.6])
```

Fonte: Própria do autor.

Para a representação das três funções de pertinência de saída do modelo *fuzzy*, foram usadas variáveis linguísticas do tipo triangulares e trapezoidais, mudando somente a faixa de valores para cada uma delas, conforme definidas anteriormente. Com estas informações as funções de pertinência da saída foram programadas em Python, cuja visualização gráfica é apresentada nas Figura 11(a), (b) e (c) e o programa na Figura 12.

Figura 11 – Funções de pertinência da saída. a) Macroporosidade. b) Microporosidade.c) Densidade



Fonte: Própria do autor.

Figura 5 – Programa em Python para as funções de pertinência da saída

```
macroporosidade['MB'] = fuzz.trapmf(x_macroporosidade, [0.03,0.035,0.05,0.055])
macroporosidade['B'] = fuzz.trapmf(x_macroporosidade, [0.05,0.055,0.059,0.06])
macroporosidade['M'] = fuzz.trapmf(x_macroporosidade, [0.059,0.06,0.061,0.064])
macroporosidade['A'] = fuzz.trapmf(x_macroporosidade, [0.061,0.064,0.093,0.11])
macroporosidade['MA'] = fuzz.trapmf(x_macroporosidade, [0.093,0.11,0.19,0.21])

microporosidade['MB'] = fuzz.trapmf(x_microporosidade, [0.18,0.23,0.288,0.292])
microporosidade['B'] = fuzz.trapmf(x_microporosidade, [0.288,0.292,0.296,0.3])
microporosidade['M'] = fuzz.trapmf(x_microporosidade, [0.296,0.3,0.302,0.309])
microporosidade['A'] = fuzz.trapmf(x_microporosidade, [0.302,0.309,0.315,0.319])
microporosidade['MA'] = fuzz.trapmf(x_microporosidade, [0.315,0.319,0.37,0.42])

densidade['MB'] = fuzz.trapmf(x_densidade, [1.2,1.25,1.45,1.485])
densidade['B'] = fuzz.trapmf(x_densidade, [1.45,1.485,1.52,1.59])
densidade['M'] = fuzz.trapmf(x_densidade, [1.52,1.59,1.64,1.7])
densidade['A'] = fuzz.trapmf(x_densidade, [1.64,1.7,1.73,1.76])
densidade['MA'] = fuzz.trapmf(x_densidade, [1.73,1.76,1.9,2])
```

Fonte: Própria do autor.

4.3 Base de Regras

A base de regras para o sistema de inferência *Fuzzy* aplicado ao problema de solos degradados, é elaborada de acordo com a opinião de um especialista, combinando todas as entradas possíveis (9 funções de pertinência da entrada Tratamento x 3 funções de pertinência da entrada Profundidade, totalizando 27 regras. É composta por regras condicionais do tipo SE-ENTÃO; sendo as entradas $x1,\ldots,xp$ os antecedentes da regra e a saída y o consequente da regra. No Quadro 1 é mostrada as regras 1 e 2, mas a relação completa de 27 regras obtidas para o sistema da pesquisa podem ser encontrados no Quadro 5, no Apêndice A:

Quadro 1 – Amostras da base de regras

Regra 1: SE tratamento é C/G **E** profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é A e densidade é MA.
Regra 2: SE tratamento é C/GE/G **E** profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é B e densidade é A.
.....

Fonte: Própria do autor.

Ressalta-se que essa base de regras deve ser feita com o apoio de um especialista na área, visto que para a maioria dos solos essa definição pode ser subjetiva, ou seja, os tratamentos podem ser diferentes, a profundidade de

recolhimento das amostras também e os resultados podem ser outros, o que acarretaria em uma mudança nos intervalos das funções de pertinência e até na sua distribuição.

Com a base de regras definida, foi gerada a inferência, sendo utilizada por *default* a da biblioteca scikit-fuzzy, que é o método Mamdani. O sistema de controle foi criado da forma mostrada na Figura 13.

Figura 6 – Sistema de controle

```
sistema_controle=ctrl.ControlSystem([regra1,regra2,regra3,regra4,regra5,regra6,regra7,regra8,regra9,regra10,regra11,regra12,regra13,
regra14,regra15,regra16,regra17,regra18,regra19,regra20,regra21,regra22,regra23,regra24,regra25,regra26,regra27])
sistema=ctrl.ControlSystemSimulation(sistema_controle)
```

Fonte: Própria do autor.

4.4 Defuzzificação

Definida a inferência e a base de regras, é aplicada a etapa final que é a defuzzificação, ou seja, a conversão de todos os valores obtidos no domínio *Fuzzy* para o domínio real, utilizando o método centróide descrito em tópico anterior pela Equação 4.

Neste capítulo foi apresentado o desenvolvimento do Sistema *Fuzzy* para a análise de solos degradados, utilizando os atributos físicos.

5 RESULTADOS E DISCUSSÕES

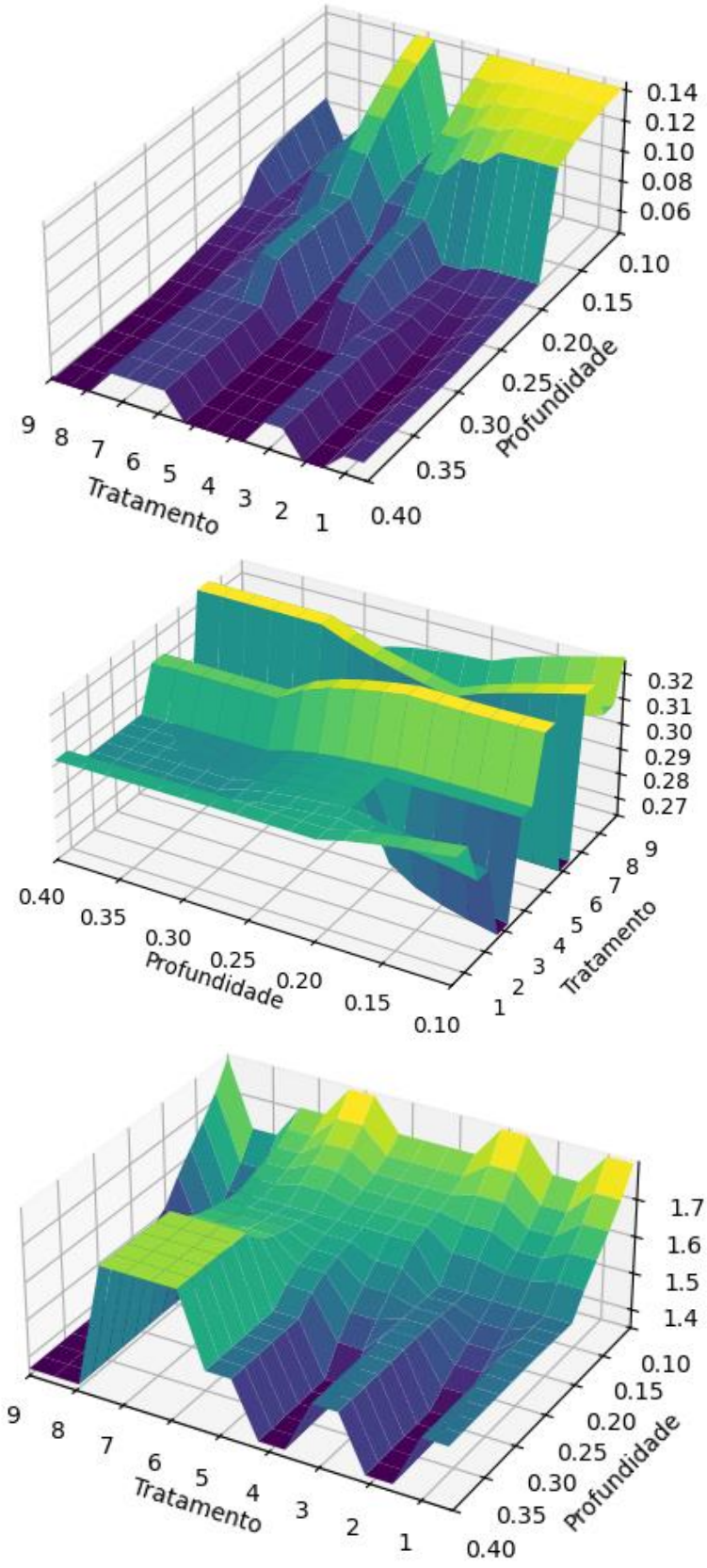
Nesse capítulo são apresentados os resultados obtidos no algoritmo criado após aplicação de todos os conceitos do sistema *fuzzy* e depois comparados ao Toolbox do MATLAB visando sua validação.

5.1 Resultados do algoritmo obtidos no Google Colab

No programa da Lógica *Fuzzy* aplicado a análise de solos degradados desenvolvido no Google Colab na linguagem Python, foram gerados os gráficos de superfície 3D para as três saídas (Macroporosidade, Microporosidade e Densidade) em função das entradas (Tratamento e Profundidade), que são apresentados nas Figuras 14 (a), (b) e (c).

Os gráficos gerados no Python apresentam as faixas de valores dadas no início da Fuzzificação, mas não tem rotatividade, nem apresentam valores ponto a ponto, dificultando a comparação com o Toobox do MATLAB.

Figura 7 – Gráficos de superfície para as 3 saídas no Python. a) Macroporosidade. b) Microporosidade. c) Densidade.

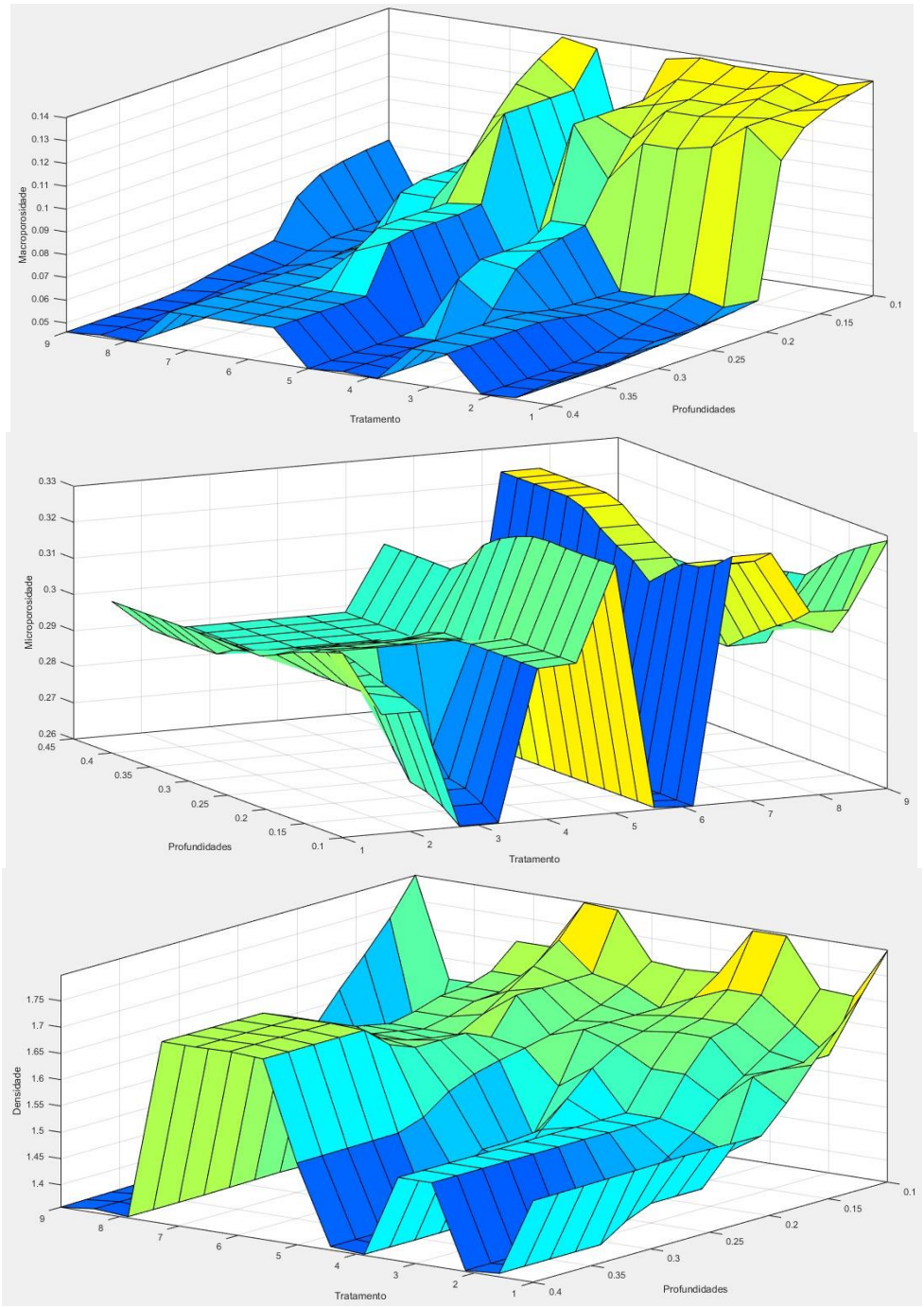


Fonte: Própria do autor.

5.2 Resultados Obtidos no Toolbox do MATLAB

Usando o *toolbox* do MATLAB, foi desenvolvido o sistema de inferência *Fuzzy* para o problema de degradação dos solos. Assim foram obtidos os resultados dos gráficos em superfície 3D no Toolbox do MATLAB, Figura 15.

Figura 8 – Gráficos de superfície para as 3 saídas no Toolbox. a) Macroporosidade. b) Microporosidade. c) Densidade.



Fonte: Própria do autor.

Com os gráficos gerados pelo Python e pelo Toolbox percebe-se alguma semelhança nas quantidades de elevações, mas não conclusiva entre cada saída nos dois métodos. Por outro lado, o Toolbox do MATLAB permite a movimentação dos gráficos em várias direções, possibilitando uma análise bem completa, diferente do programa criado em Python, que não permite (ou não foi programado). Visando uma comparação, um programa interativo no Google Colab foi criado, onde qualquer usuário digita o tratamento e a profundidade desejada, obtendo-se os valores para a Macro, Micro e a Densidade. Para comparação numérica dos resultados foram geradas para os dois métodos, duas tabelas com valores de saída para as mesmas combinações de entrada, conforme mostrado nas Quadro 2 e 3

Quadro 2 – Resultados para o programa em Python

	Profundidade								
Tratamento	PF-1			PF-2			PF-3		
C/G	0,141	0,311	1,792	0,061	0,302	1,513	0,056	0,302	1,513
C/GE/G	0,141	0,294	1,706	0,061	0,302	1,513	0,044	0,294	1,369
C/GE/MP	0,141	0,265	1,792	0,056	0,302	1,513	0,061	0,294	1,513
C/MP	0,141	0,302	1,706	0,082	0,302	1,612	0,044	0,294	1,369
G	0,082	0,326	1,706	0,056	0,326	1,612	0,044	0,311	1,513
MA	0,141	0,265	1,792	0,082	0,264	1,612	0,061	0,264	1,706
MP	0,082	0,326	1,706	0,061	0,311	1,612	0,061	0,326	1,706
SE	0,056	0,311	1,612	0,056	0,294	1,369	0,044	0,294	1,369
SM	0,082	0,326	1,792	0,056	0,311	1,513	0,044	0,302	1,369

Fonte: Própria do autor.

Quadro 3 – Resultados no Toolbox do MATLAB

	Profundidade								
Tratamento	PF-1			PF-2			PF-3		
C/G	0,14	0,311	1,8	0,062	0,302	1,51	0,056	0,302	1,51
C/GE/G	0,14	0,293	1,71	0,062	0,302	1,51	0,046	0,293	1,36
C/GE/MP	0,14	0,26	1,8	0,056	0,302	1,51	0,062	0,293	1,51
C/MP	0,14	0,302	1,71	0,082	0,302	1,61	0,046	0,293	1,36
G	0,082	0,328	1,706	0,056	0,329	1,61	0,046	0,311	1,51
MA	0,14	0,26	1,8	0,082	0,26	1,61	0,062	0,26	1,71
MP	0,082	0,329	1,71	0,062	0,311	1,61	0,062	0,329	1,71
SE	0,056	0,311	1,612	0,056	0,293	1,36	0,046	0,293	1,36
SM	0,082	0,329	1,8	0,056	0,311	1,51	0,046	0,302	1,36

Fonte: Própria do autor.

Nos dois quadros considera-se os valores em rosa referente à saída Macroporosidade, os valores em azul referentes a Microporosidade e em verde referentes a Densidade.

Comparando-se os resultados numéricos entre as duas tabelas, nota-se que os resultados são muito próximos, sendo que no Quadro 4 são apresentados os erros calculados entre os dois métodos.

Quadro 4 – Erro percentual entre o código em Python e o Toolbox do MATLAB

	Profundidade								
Tratamento	PF-1			PF-2			PF-3		
C/G	0,71%	0,00%	0,44%	1,61%	0,00%	0,20%	0,00%	0,00%	0,20%
C/GE/G	0,71%	0,34%	0,23%	1,61%	0,00%	0,20%	4,35%	0,34%	0,66%
C/GE/MP	0,71%	1,92%	0,44%	0,00%	0,00%	0,20%	1,61%	0,34%	0,20%
C/MP	0,71%	0,00%	0,23%	0,00%	0,00%	0,12%	4,35%	0,34%	0,66%
G	0,00%	0,61%	0,00%	0,00%	0,91%	0,12%	4,35%	0,00%	0,20%
MA	0,71%	1,92%	0,44%	0,00%	1,54%	0,12%	1,61%	1,54%	0,23%
MP	0,00%	0,91%	0,23%	1,61%	0,00%	0,12%	1,61%	0,91%	0,23%
SE	0,00%	0,00%	0,00%	0,00%	0,34%	0,66%	4,35%	0,34%	0,66%
SM	0,00%	0,91%	0,44%	0,00%	0,00%	0,20%	4,35%	0,00%	0,66%

Fonte: Próprio autor.

Analisando os resultados, têm-se que a média de erros da tabela é de 0,7%, demonstrado os valores muito próximos entre o Toolbox do MATLAB e o código desenvolvido em Python, validando o algoritmo criado no código em Python. Alguns valores, indicados no quadro, ficaram um pouco fora da média dos valores.

Além disso, é possível observar pelos quadros e pelos gráficos de superfície que os valores maiores de Macroporosidade se encontram na PF-1 (0,00m- 0,10m) o que indica que nessa profundidade há uma melhor aeração e infiltração de água. Para o Tratamento 8 (SE - solo exposto, sem técnica de recuperação), foram encontrados os menores valores, o que significa que os tratamentos aplicados neste solo foram úteis para a Macroporosidade.

Para a Microporosidade, valores constantes nos dois métodos, Toolbox e Python, em torno de (0,327) foi encontrado nos tratamentos G (guandu) e MP (mucuna preta), sendo que a alta Microporosidade, indica que tem uma alta concentração de água naquele solo, o que indica que esses Tratamentos podem ser úteis em climas mais secos.

Para a Densidade, quanto maior o valor obtido, mais denso está o solo, o que pode vir a prejudicar a penetração de raízes, sendo necessário evitar tratamentos que elevem muito a densidade do solo. Por fim, os gráficos de superfície e os quadros de resultados fornecem uma boa amostragem do comportamento do Tratamento x Profundidade.

6 CONCLUSÕES

Esta pesquisa teve como objetivo a aprendizagem da Lógica *Fuzzy* e a sua aplicação no problema de solos degradados. Para o desenvolvimento do algoritmo *Fuzzy* foi usado o Google Colab e a linguagem de programação Python.

A base de dados utilizada no desenvolvimento do programa foi retirada da literatura estudada, bem como as variáveis linguísticas e o entendimento das variáveis do solo, que auxiliaram nas comparações dos resultados obtidos

Constituiu um desafio importante o desenvolvimento da programação em linguagem Python e o uso do Google Colab, uma vez que ainda não existe muito material para consulta.

Comparando os resultados obtidos com o algoritmo criado na linguagem Python e ao Toolboox do Matlab, os erros mostrados foram baixos, considerando-se satisfatório o desempenho do algoritmo criado, apresentando o Google Colab como uma alternativa importante e de uso gratuito, com acesso mais amplo para o desenvolvimento de algoritmos mais específicos, enquanto que o uso do Toolbox do MATLAB é restrito para quem tem a licença.

Com relação ao algoritmo da Lógica *Fuzzy* criado, fornece a visualização das variações dos atributos analisados com relação aos dados de entrada, podendo ser incluídas outros atributos na análise. Cada tipo de solo pede uma nova definição de variáveis de entrada no sistema *Fuzzy* e da base de regras, visto que os tratamentos do solo analisados podem ser outros, as profundidades podem ser maiores e a faixa de valores da Macroporosidade, Microporosidade e Densidade também podem ser diferentes.

Para trabalhos futuros, sugere-se a criação de um programa com a possibilidade de conversão em um aplicativo para um smartphone no sistema Android, de forma a agilizar a análise dos parâmetros do solo em campo.

7 REFERÊNCIAS BIBLIOGRÁFICAS

BONINI NETO, A. **Projeto de Pesquisa: Sistemas computacionais Inteligentes para estimar Índice de recuperação de solo degradado**. Tupã, Universidade Estadual Paulista, 2014. 21p.

BRADY, N. C.; WEIL, Ray R. **Elementos da Natureza e Propriedades dos Solos**. 3. ed. Porto Alegre: Bookman, 2013. 715 p.

CASTRO, M. M. T.; ALVES, M. C.; NASCIMENTO, V.Castro. M.T.T. Revegetation on a Removed Topsoil: Effect on Aggregate Stability. Communications in Soil Science and Plant Analysis, v. 40, p. 771-786, 2009.

CAVALCANTI, M. C. **Atividades Econômicas do Brasil: entenda as principais do país**. 2023. Disponível em: https://querobolsa.com.br/enem/historia-brasil/economia-brasileira. Acesso em: 15 abril 2025.

COLAB.GOOGLE. Disponível em: https://colab.research.google.com/#. Acesso em: 15 julho 2024

DE OLIVEIRA, M. G. M. et al. **Aplicação da Lógica Difusa para Análise de Solos Degradados**. Disponível em: https://www.ime.unicamp.br/~cbsf4/Papers_IVCBSF/CBSF_2016_paper_51. Acesso em: 15 julho 2024.

EMBRAPA – Empresa Brasileira de Pesquisa Agropecuária. Manual de métodos de análise de solo. 2.ed. Rio de Janeiro: EMBRAPA/CNPSO, 212p. 1997.

GODINHO, E. Z.; GASPAROTTO, H. V.; CANEPPELE, F. de L. Lógica Fuzzy na Agricultura. **Cadernos de Educação Tecnologia e Sociedade**, Inhumas, v. 15, n. 1, p. 126-139, 20 mar. 2022. Brazilian Journal of Education, Technology and Society (BRAJETS). http://dx.doi.org/10.14571/brajets.v15.n1.126-139. Acesso em : 13 maio 2025.

GITHUB PAGES. Disponível em: https://pages.github.com/. Acesso em: 20 julho 2024.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA**.** Disponível em: https://www.ibge.gov.br/. Acesso em: 23 abril 2025.

KIEHL., E. J. Manual de Edafologia. São Paulo, SP: Editora Agronômica “Ceres”, LTDA, 1979.

MAMDANI, E. H., ASSILIAN, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies,* v. 7, n. 1, 1–13. https://doi.org/10.1016/S0020-7373(75)80002-2. Acesso em: 20 junho 2025.

MATHWORKS. Disponível em: https://www.mathworks.com/. Acesso em 05 mai. 2024

PEIXOTO, Thiago Aquino. **CONTROLE DE ROBÔ MÓVEL USANDO LÓGICA FUZZY**. 2022. 52 f. TCC (Graduação) - Curso de Engenharia Elétrica, Universidade Federal Rural do Semi Árido, Caraúbas, 2022. Disponível em: https://repositorio.ufersa.edu.br/handle/prefix/9945. Acesso em: 01 jul. 2025.

RIZOL, P. M. S.; DIAS, R. A. **Desmistificando a Lógica Fuzzy**. 2014. Disponível em: https://www.abenge.org.br/cobenge/legado/arquivos/5/Artigos/129231. Acesso em 01. julho 2024.

ROVEDA, S. R. M.; ROVEDA, J. A. F.; LOURENÇO, R. W. **Aplicação da Lógica Fuzzy para estudo de permeabilidade de solos de região impactada da baixada Santista**. **Holos Environment**, [S.L.], v. 11, n. 2, p. 180, 16 abr. 2011. Lepidus Tecnologia. Disponível em: http://dx.doi.org/10.14295/holos.v11i2.5634. Acesso em: 01 janeiro 2025

SAMSUNG. What is Fuzzy Logic in a Washing Machine? Samsung Support, 2024. Disponível em: https://www.samsung.com/in/support/home-appliances/what-is-fuzzy-logic-in-a-washing-machine/?srsltid=AfmBOoqYJP4XteLv_ccOO3QM8gf6bu5rRePX8Q5_5iCqZIlReKZKxjCY. Acesso em: 1. jul. 2025.

SILVA, I. R. S.; SILVA, R. O. da. Linguagem de Programação Python. Tecnologias em Projeção, [S.l], v. 10, n. 1, p. 55-71, 19 ago. 2019. Disponível em: https://projecaociencia.com.br/index.php/Projecao4/article/view/1359. Acesso em: 27 abril 2025.

SILVA, S de A. *et al*. **Lógica fuzzy na avaliação da fertilidade do solo e produtividade do café conilon**.2010. Disponível em: https://www.scielo.br/j/rca/a/6yQ6b5xcRgmdJYvfXSLKFcv/?lang=pt#. Acesso em: 13 jul. 2024.

STEFFEN, G. P. K.; STEFFEN, R. B.; BOENI, M.; SCHÚ, A.L.; MALDANER, J.; CONTERATO, I. F. A importância do solo para a sustentação da vida no planeta Terra. Porto Alegre: Seapi/Ddpa, 2024. 24 p.

TAKAGI, T.; SUGENO, M. **Fuzzy identification of systems and its applications to modeling and control**. IEEE Trans. Syst., v. SMC-15, n. 1, p. 116-132, jan./fev. 1985.

TANSCHEIT, R. Sistemas Fuzzy. In: VI Simpósio Brasileiro de Automação Inteligente, 2003, Bauru, SP. Anais de Minicursos do VI SBAI, 2003. p. 35 páginas.

TEIXEIRA, P. C. **Manual de Métodos de Análise de Solo**. 3. ed. Brasília: Embrapa Solos, 2017. 577 p. Disponível em: https://portalgepes.com/wp-content/uploads/2023/. Acesso em: 03 junho 2025.

WANG, X.; CAMMERAAT, E. L. H.; CERLI, C.; KALBITZ, K. Soil aggregation and stabilization of organic carbono as affected by erosion and deposition. Soil Biology & Biochemistry, 72, 55-65. 2014.

ZADEH, L. A. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338–353, jun. 1965.

APÊNDICE A – Base de Regras

Quadro 5 – Base de regras *Fuzzy*

Regra 1: SE tratamento é C/G E profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é A e densidade é MA.
Regra 2: SE tratamento é C/GE/G E profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é B e densidade é A.
Regra 3: SE tratamento é C/GE/MP E profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é MB e densidade é MA.
Regra 4: SE tratamento é C/MP E profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é M e densidade é A.
Regra 5: SE tratamento é G E profundidade é PF-1 ENTÃO macroporosidade é A, microporosidade é MA e densidade é A.
Regra 6: SE tratamento é MA E profundidade é PF-1 ENTÃO macroporosidade é MA, microporosidade é MB e densidade é MA.
Regra 7: SE tratamento é MP E profundidade é PF-1 ENTÃO macroporosidade é A, microporosidade é MA e densidade é A.
Regra 8: SE tratamento é SE E profundidade é PF-1 ENTÃO macroporosidade é B, microporosidade é A e densidade é M.
Regra 9: SE tratamento é SM E profundidade é PF-1 ENTÃO macroporosidade é A, microporosidade é MA e densidade é MA.
Regra 10: SE tratamento é C/G E profundidade é PF-2 ENTÃO macroporosidade é M, microporosidade é M e densidade é B.
Regra 11: SE tratamento é C/GE/G E profundidade é PF-2 ENTÃO macroporosidade é M, microporosidade é M e densidade é B.
Regra 12: SE tratamento é C/GE/MP E profundidade é PF-2 ENTÃO macroporosidade é B, microporosidade é M e densidade é B.
Regra 13: SE tratamento é C/MP E profundidade é PF-2 ENTÃO macroporosidade é A, microporosidade é M e densidade é M.
Regra 14: SE tratamento é G E profundidade é PF-2 ENTÃO macroporosidade é B, microporosidade é MA e densidade é M.
Regra 15: SE tratamento é MA E profundidade é PF-2 ENTÃO macroporosidade é A, microporosidade é MB e densidade é M.
Regra 16: SE tratamento é MP E profundidade é PF-2 ENTÃO macroporosidade é M, microporosidade é A e densidade é M.
Regra 17: SE tratamento é SE E profundidade é PF-2 ENTÃO macroporosidade é B, microporosidade é B e densidade é MB.
Regra 18: SE tratamento é SM E profundidade é PF-2 ENTÃO macroporosidade é B, microporosidade é A e densidade é B.

Regra 19: SE tratamento é C/G E profundidade é PF-3 ENTÃO macroporosidade é B, microporosidade é M e densidade é B.
Regra 20: SE tratamento é C/GE/G E profundidade é PF-3 ENTÃO macroporosidade é MB, microporosidade é B e densidade é MB.
Regra 21: SE tratamento é C/GE/MP E profundidade é PF-3 ENTÃO macroporosidade é M, microporosidade é B e densidade é B.
Regra 22: SE tratamento é C/MP E profundidade é PF-3 ENTÃO macroporosidade é MB, microporosidade é A e densidade é B.
Regra 23: SE tratamento é G E profundidade é PF-3 ENTÃO macroporosidade é MB, microporosidade é A e densidade é B.
Regra 24: SE tratamento é MA E profundidade é PF-3 ENTÃO macroporosidade é M, microporosidade é MB e densidade é A.
Regra 25: SE tratamento é MP E profundidade é PF-3 ENTÃO macroporosidade é M, microporosidade é MA e densidade é A.
Regra 26: SE tratamento é SE E profundidade é PF-3 ENTÃO macroporosidade é MB, microporosidade é B e densidade é MB.
Regra 27: SE tratamento é SM E profundidade é PF-3 ENTÃO macroporosidade é MB, microporosidade é M e densidade é MB.

Fonte: Próprio autor.

APÊNDICE B – CÓDIGO PYTHON

```
pip install scikit-fuzzy
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Criando os vetores das variáveis de entrada e saída

```
x_tratamento = np.arange(1,9.5,0.5)
x_profundidade = np.arange(0.1,0.4,0.02)
x_macroporosidade = np.arange(0.036,0.181,0.001)
x_microporosidade = np.arange(0.239,0.336,0.001)
x_densidade = np.arange(1.27,1.84,0.005)
# Criação das variáveis de entrada e saída
tratamento = ctrl.Antecedent(x_tratamento, 'Tratamento')
profundidade = ctrl. Antecedent(x_profundidade, 'Profundidade')
macroporosidade = ctrl.Consequent(x_macroporosidade, 'Macroporosidade')
microporosidade = ctrl.Consequent(x_microporosidade, 'Microporosidade')
densidade = ctrl.Consequent(x_densidade,'Densidade')
```

Criando as funções de pertinência

```
tratamento['C/G'] = fuzz.trimf(x_tratamento,[0.5,1,1.5])
tratamento['C/GE/G'] = fuzz.trimf(x_tratamento, [1.5,2,2.5])
tratamento['C/GE/MP'] = fuzz.trimf(x_tratamento, [2.5,3,3.5])
tratamento['C/MP'] = fuzz.trimf(x_tratamento, [3.5,4,4.5])
tratamento['G'] = fuzz.trimf(x_tratamento, [4.5,5,5.5])
tratamento['MA'] = fuzz.trimf(x_tratamento, [5.5,6,6.5])
tratamento['MP'] = fuzz.trimf(x_tratamento, [6.5,7,7.5])
tratamento['SE'] = fuzz.trimf(x_tratamento, [7.5,8,8.5])
tratamento['SM'] = fuzz.trimf(x_tratamento, [8.5,9,9.5])

profundidade['PF-1'] = fuzz.trimf(x_profundidade, [0,0.1,0.2])
profundidade['PF-2'] = fuzz.trimf(x_profundidade, [0.1,0.2,0.3])
profundidade['PF-3'] = fuzz.trimf(x_profundidade, [0.2,0.4,0.6])

macroporosidade['MB'] = fuzz.trapmf(x_macroporosidade,
[0.03,0.035,0.05,0.055])
macroporosidade['B'] = fuzz.trapmf(x_macroporosidade,
[0.05,0.055,0.059,0.06])
macroporosidade['M'] = fuzz.trapmf(x_macroporosidade,
[0.059,0.06,0.061,0.064])
macroporosidade['A'] = fuzz.trapmf(x_macroporosidade,
[0.061,0.064,0.093,0.11])
macroporosidade['MA'] = fuzz.trapmf(x_macroporosidade,
[0.093,0.11,0.19,0.21])
```

```
        microporosidade['MB'] = fuzz.trapmf(x_microporosidade,
[0.18,0.23,0.288,0.292])
        microporosidade['B'] = fuzz.trapmf(x_microporosidade,
[0.288,0.292,0.296,0.3])
        microporosidade['M'] = fuzz.trapmf(x_microporosidade,
[0.296,0.3,0.302,0.309])
        microporosidade['A'] = fuzz.trapmf(x_microporosidade,
[0.302,0.309,0.315,0.319])
        microporosidade['MA'] = fuzz.trapmf(x_microporosidade,
[0.315,0.319,0.37,0.42])

        densidade['MB'] = fuzz.trapmf(x_densidade, [1.2,1.25,1.45,1.485])
        densidade['B'] = fuzz.trapmf(x_densidade, [1.45,1.485,1.52,1.59])
        densidade['M'] = fuzz.trapmf(x_densidade, [1.52,1.59,1.64,1.7])
        densidade['A'] = fuzz.trapmf(x_densidade, [1.64,1.7,1.73,1.76])
        densidade['MA'] = fuzz.trapmf(x_densidade, [1.73,1.76,1.9,2])
```

#**Criando as 27 regras**

```
        regra1=ctrl.Rule((tratamento['C/G'] & profundidade['PF-1']),
consequent=[macroporosidade['MA'], microporosidade['A'], densidade['MA']])
        regra2=ctrl.Rule((tratamento['C/GE/G'] & profundidade['PF-1']),
consequent=[macroporosidade['MA'], microporosidade['B'], densidade['A']])
        regra3=ctrl.Rule((tratamento['C/GE/MP'] & profundidade['PF-1']),
consequent=[macroporosidade['MA'], microporosidade['MB'], densidade['MA']])
        regra4=ctrl.Rule((tratamento['C/MP'] & profundidade['PF-1']),
consequent=[macroporosidade['MA'], microporosidade['M'], densidade['A']])
        regra5=ctrl.Rule((tratamento['G'] & profundidade['PF-1']),
consequent=[macroporosidade['A'], microporosidade['MA'], densidade['A']])
        regra6=ctrl.Rule((tratamento['MA'] & profundidade['PF-1']),
consequent=[macroporosidade['MA'], microporosidade['MB'], densidade['MA']])
        regra7=ctrl.Rule((tratamento['MP'] & profundidade['PF-1']),
consequent=[macroporosidade['A'], microporosidade['MA'], densidade['A']])
        regra8=ctrl.Rule((tratamento['SE'] & profundidade['PF-1']),
consequent=[macroporosidade['B'], microporosidade['A'], densidade['M']])
        regra9=ctrl.Rule((tratamento['SM'] & profundidade['PF-1']),
consequent=[macroporosidade['A'], microporosidade['MA'], densidade['MA']])
        regra10=ctrl.Rule((tratamento['C/G'] & profundidade['PF-2']),
consequent=[macroporosidade['M'], microporosidade['M'], densidade['B']])
        regra11=ctrl.Rule((tratamento['C/GE/G'] & profundidade['PF-2']),
consequent=[macroporosidade['M'], microporosidade['M'], densidade['B']])
        regra12=ctrl.Rule((tratamento['C/GE/MP'] & profundidade['PF-2']),
consequent=[macroporosidade['B'], microporosidade['M'], densidade['B']])
        regra13=ctrl.Rule((tratamento['C/MP'] & profundidade['PF-2']),
consequent=[macroporosidade['A'], microporosidade['M'], densidade['M']])
        regra14=ctrl.Rule((tratamento['G'] & profundidade['PF-2']),
consequent=[macroporosidade['B'], microporosidade['MA'], densidade['M']])
        regra15=ctrl.Rule((tratamento['MA'] & profundidade['PF-2']),
consequent=[macroporosidade['A'], microporosidade['MB'], densidade['M']])
```

```
        regra16=ctrl.Rule((tratamento['MP'] & profundidade['PF-2']),
consequent=[macroporosidade['M'], microporosidade['A'], densidade['M']])
        regra17=ctrl.Rule((tratamento['SE'] & profundidade['PF-2']),
consequent=[macroporosidade['B'], microporosidade['B'], densidade['MB']])
        regra18=ctrl.Rule((tratamento['SM'] & profundidade['PF-2']),
consequent=[macroporosidade['B'], microporosidade['A'], densidade['B']])
        regra19=ctrl.Rule((tratamento['C/G'] & profundidade['PF-3']),
consequent=[macroporosidade['B'], microporosidade['M'], densidade['B']])
        regra20=ctrl.Rule((tratamento['C/GE/G'] & profundidade['PF-3']),
consequent=[macroporosidade['MB'], microporosidade['B'], densidade['MB']])
        regra21=ctrl.Rule((tratamento['C/GE/MP'] & profundidade['PF-3']),
consequent=[macroporosidade['M'], microporosidade['B'], densidade['B']])
        regra22=ctrl.Rule((tratamento['C/MP'] & profundidade['PF-3']),
consequent=[macroporosidade['MB'], microporosidade['B'], densidade['MB']])
        regra23=ctrl.Rule((tratamento['G'] & profundidade['PF-3']),
consequent=[macroporosidade['MB'], microporosidade['A'], densidade['B']])
        regra24=ctrl.Rule((tratamento['MA'] & profundidade['PF-3']),
consequent=[macroporosidade['M'], microporosidade['MB'], densidade['A']])
        regra25=ctrl.Rule((tratamento['MP'] & profundidade['PF-3']),
consequent=[macroporosidade['M'], microporosidade['MA'], densidade['A']])
        regra26=ctrl.Rule((tratamento['SE'] & profundidade['PF-3']),
consequent=[macroporosidade['MB'], microporosidade['B'], densidade['MB']])
        regra27=ctrl.Rule((tratamento['SM'] & profundidade['PF-3']),
consequent=[macroporosidade['MB'], microporosidade['M'], densidade['MB']])
        sistema_controle=ctrl.ControlSystem([regra1,regra2,regra3,regra4,regra5,regr
a6,regra7,regra8,regra9,regra10,regra11,regra12,regra13,regra14,regra15,regra16,r
egra17,regra18,regra19,regra20,regra21,regra22,regra23,regra24,regra25,regra26,r
egra27])
        sistema=ctrl.ControlSystemSimulation (sistema_controle)
        tratamento.view()
        profundidade.view()
        macroporosidade.view()
        microporosidade.view()
        densidade.view()
        sistema.input ['Tratamento'] = float(input('Qual é o tratamento utilizado?'))
        sistema.input ['Profundidade'] = float(input('Qual é a profundidade utilizada?'))
        sistema.compute()
        print('A macroporosidade é:', sistema.output['Macroporosidade'])
        print('A microporosidade é:',sistema.output['Microporosidade'])
        print('A densidade é:',sistema.output['Densidade'])
```

Criando as superfícies para visualização

```
        X_tratamento, X_profundidade = np.meshgrid(x_tratamento, x_profundidade)
        activation_macroporosidade = np.zeros_like(X_tratamento, dtype=float)
        activation_microporosidade = np.zeros_like(X_tratamento, dtype=float)
        activation_densidade = np.zeros_like(X_tratamento, dtype=float)

        for i in range(X_tratamento.shape[0]):
          for j in range(X_tratamento.shape[1]):
```

```
            tratamento_atual = X_tratamento[i, j]
            if 1 <= tratamento_atual <= 9:
             sistema.input['Tratamento'] = tratamento_atual
             sistema.input['Profundidade'] = X_profundidade[i, j]
             sistema.compute()
             activation_macroporosidade[i, j] = sistema.output.get('Macroporosidade',
np.nan)
             activation_microporosidade[i, j] = sistema.output.get('Microporosidade',
np.nan)
             activation_densidade[i, j] = sistema.output.get('Densidade', np.nan)
            else:
             activation_macroporosidade[i, j] = np.nan
             activation_microporosidade[i, j] = np.nan
             activation_densidade[i, j] = np.nan
        for i in range(X_tratamento.shape[0]):
          for j in range(X_tratamento.shape[1]):
            tratamento_atual = X_tratamento[i, j]
            if 1 <= tratamento_atual <= 9:
             sistema.input['Tratamento'] = tratamento_atual
             sistema.input['Profundidade'] = X_profundidade[i, j]
             sistema.compute()
             activation_macroporosidade[i, j] = sistema.output['Macroporosidade']
             activation_microporosidade[i, j] = sistema.output['Microporosidade']
             activation_densidade[i, j] = sistema.output['Densidade']

        fig = plt.figure(figsize= (24,10), dpi=(10000))
        X_tratamento, X_profundidade = np.meshgrid(np.linspace(0.5, 9,
activation_macroporosidade.shape[1]),
                           np.linspace(0.1, 0.4, activation_macroporosidade.shape[0]))

        fig1 = plt.figure()
        ax1 = fig1.add_subplot(111, projection='3d')
        ax1.plot_surface(X_tratamento, X_profundidade, activation_macroporosidade,
cmap='viridis')
        ax1.set_title('Macroporosidade')
        ax1.set_xlabel('Tratamento')
        ax1.set_ylabel('Profundidade')
        ax1.set_xlim(0.4,9)
        ax1.set_ylim(0.1,0.4)
        ax1.set_zlim(0.045,0.145)
        ax1.invert_xaxis()
        ax1.invert_yaxis()
        ax1.set_xticks(np.arange(1, 10, 1))
        ax1.set_yticks(np.arange(0.1, 0.4, 0.05))
        ax1.set_zticks(np.arange(0.06, 0.14, 0.02))
        ax1.set_box_aspect((2, 3, 1))
        plt.show()

        fig2 = plt.figure()
        ax2 = fig2.add_subplot(111, projection='3d')
```

```
        ax2.plot_surface(X_profundidade, X_tratamento, activation_microporosidade,
cmap='viridis')
        ax2.set_title('Microporosidade')
        ax2.set_xlabel('Profundidade')
        ax2.set_ylabel('Tratamento')
        ax2.set_xlim(0.1,0.4)
        ax2.set_ylim(0.4,9)
        ax2.set_zlim(0.263,0.325)
        ax2.invert_xaxis()
        ax2.set_xticks(np.arange(0.1, 0.4, 0.05))
        ax2.set_yticks(np.arange(1, 10, 1))
        ax2.set_zticks(np.arange(0.27, 0.33, 0.01))
        ax2.set_box_aspect((5, 4, 2))
        plt.show()

        fig3 = plt.figure()
        ax3 = fig3.add_subplot(111, projection='3d')
        ax3.plot_surface(X_tratamento, X_profundidade, activation_densidade,
cmap='viridis')
        ax3.set_title('Densidade')
        ax3.set_xlabel('Tratamento')
        ax3.set_ylabel('Profundidade')
        ax3.set_xlim(0.4,9)
        ax3.set_ylim(0.1,0.4)
        ax3.set_zlim(1.35,1.8)
        ax3.invert_xaxis()
        ax3.invert_yaxis()
        ax3.set_xticks(np.arange(1, 10, 1))
        ax3.set_yticks(np.arange(0.1, 0.4, 0.05))
        ax3.set_zticks(np.arange(1.4, 1.7, 0.1))
        ax3.set_box_aspect((5, 4, 2))
        plt.show()
```